Advanced Privacy Notions for Electronic Voting and Digital Signatures

Ashley Fraser

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

Information Security Group
School of Electrical Engineering, Physical and Mathematical Sciences
Royal Holloway, University of London

2021

Declaration

These doctoral studies were conducted under the supervision of Dr. Elizabeth A. Quaglia.

The work presented in this thesis is the result of original research I conducted, in collaboration with others, whilst enrolled in the School of Electrical Engineering, Physical and Mathematical Sciences as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

Ashley Fraser

14 August 2021

Abstract

In the context of electronic voting, the term privacy is used to describe several security goals that aim to protect a voter's fundamental right to a secret ballot. This thesis adopts this naming convention, using the term privacy to refer to any security goal of a cryptographic protocol that is concerned with keeping information secret. Accordingly, privacy can refer to *data privacy*, the property that data is secret to all but the intended recipient, and *data source privacy*, the property that the data source is secret. In this thesis, we consider advanced notions of data privacy in the context of e-voting and digital signatures and consider data source privacy for ring signature variants. We use these settings to introduce new definitions of privacy and security and to contextualise our new definitions within the wider literature.

In the setting of e-voting, we explore ballot secrecy, a fundamental privacy property for e-voting that intuitively states that a voter's vote remains secret throughout an election. We introduce new definitions of ballot secrecy for e-voting schemes, wherein we consider an attacker that can corrupt various election officials. By contrast, existing definitions generally capture an attacker with limited corruption powers. We then consider the receipt-freeness property, which strengthens ballot secrecy with additional protection against vote-buying. In the literature, there is no rigorous analysis of existing definitions of receipt-freeness. Accordingly, we analyse three definitions of receipt-freeness, uncovering weaknesses and limitations in all three definitions. We address these drawbacks by introducing new definitions of receipt-freeness. Additionally, we analyse our new definitions, highlighting the difficulties of defining advanced notions of privacy for complex cryptographic protocols.

Our second setting considers digital signature schemes. We introduce incoercible signature schemes, a variant of a standard digital signature. Incoercible signatures enable signers to generate fake signatures that are indistinguishable from real signatures when coerced to produce a signature for a message chosen by an attacker. We present a security model for our new primitive that captures notions of receipt-freeness and coercion-resistance, as defined for e-voting, within this setting. We also present two incoercible signature scheme constructions, demonstrating that our security model is achievable.

Finally, we examine data source privacy and how to balance it with other desirable security properties in the context of ring signatures. Standard ring signatures allow signers to produce publicly verifiable signatures on behalf of a group of users, known as the ring. The signer's identity within the ring is not revealed, providing a degree of anonymity to signers. We introduce report and trace ring signature schemes and an accompanying security model. Our new primitive balances the desire for signer anonymity with the ability to report malicious behaviour and subsequently revoke anonymity. Moreover, our security model ensures anonymity of the reporter *and* the signer (unless their identity is traced). We

introduce a security model for our new primitive and formally relate report and trace ring signatures to other ring signature variants in the literature. We conclude by presenting a construction of a report and trace ring signature scheme.

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. Elizabeth A. Quaglia, for her support throughout my Ph.D. Thank you for guiding me in the right direction and for your advice over the past four years. I am also very grateful to have had a supervisor who has acted as an excellent role model and who has encouraged and supported me to pursue a career in academia.

I am thankful to the EPSRC and the CDT at Royal Holloway for funding my Ph.D. and for all the opportunities to travel and attend events. I would also like to thank Professor Keith Martin for the valuable discussions and the advice given at various stages of my Ph.D. I was lucky to spend three months as an intern at Fetch.AI, and I would like to thank Fetch.AI for funding my internship and David Galindo for his supervision during my time as an intern.

During my Ph.D. I have had the opportunity to collaborate with a number of great co-authors. Many thanks to Ben Smyth, Fernando Virdia, Illaria Chilloti, and Lydia Garms for our fruitful collaborations and the many helpful discussions we have had. I have worked alongside a fantastic cohort of students at Royal Holloway who have been a great source of support throughout my Ph.D. In particular, I would like to thank Pallavi for putting up with my complaining, always being a source of common sense, and grounding me when things don't go to plan.

I want to thank my friends and family who have been there for me during this journey. To my parents, for their support and always understanding. Last, but definitely not least, thank you to Stuart. For listening to presentations and proofreading work, despite not knowing what I'm talking or writing about. For understanding when life gets put on hold and generally for always being supportive and having my back.

Chapter 1

Introduction

Contents

This chapter provides the motivation for the content of this thesis, and outlines its structure.

1.1 Motivation

Privacy, which is a term we use to refer to any security goal of a cryptographic protocol that is concerned with keeping information secret, is an essential requirement of many cryptographic protocols, from the fundamental symmetric and asymmetric (public-key) encryption primitives to the complex cryptographic protocols that realise services such as instant messaging and electronic voting (e-voting). Indeed, cryptographic protocols often communicate sensitive data and, without privacy, anyone can potentially access this data. Moreover, malicious actors can abuse it. For example, in the context of e-voting, a lack of privacy can reveal a voter's voting preferences, violating the secret ballot requirement of democratic elections. Therefore, cryptographic protocols must satisfy some notion of privacy when privacy is necessary or desirable. Informally, privacy can be defined as *data* privacy and *data source* privacy. Data privacy is the property that a piece of data is secret, and data source privacy requires that the identity of a protocol user that creates/communicates data is secret. In this thesis, we are concerned with defining and

understanding notions of data privacy and data source privacy for several cryptographic protocols and primitives.

For core cryptographic primitives such as public-key encryption, data privacy is a basic requirement and various notions exist, e.g., [70, 103, 110]. These notions all capture the following intuition: knowledge of a ciphertext reveals no more information about the plaintext message than can be deduced without knowledge of the ciphertext. They differ with respect to the capabilities of a potential attacker, ranging from an honest-but-curious attacker that can only view ciphertexts [70] to malicious attackers that can additionally decrypt ciphertexts [103, 110]. These notions are well-established, and rigorous formal definitions exist in the literature. Moreover, the relationship between these notions is well-understood [9].

On the other hand, for complex cryptographic protocols, privacy is more difficult to define. For example, in the e-voting literature, basic privacy intuitively requires that a voter's vote is secret. Formally capturing this intuition is complex for the following reasons. Firstly, the tally of an election is (usually) public and may reveal information about voting preferences. Thus, definitions of privacy for e-voting must consider the 'privacy leakage' of the election tally. Secondly, e-voting schemes require several entities, including voters and numerous election officials. As a result, it is difficult to reason about the capabilities of an attacker. By contrast, public-key encryption is traditionally concerned with a sender who encrypts plaintexts and sends the resulting ciphertext to a receiver, who can decrypt the ciphertext. As such, the attacker's capabilities, for example, to view ciphertexts transmitted via a communication channel and to additionally corrupt a receiver and decrypt ciphertexts, can be clearly defined.

In addition to these challenges, it is often desirable that privacy holds in the setting where a protocol user is coerced. For instance, *deniable* public-key encryption [34] allows a sender to convince an attacker that a ciphertext encrypts a different plaintext message from the one encrypted by the sender. Deniable encryption requires that established notions of privacy for public-key encryption [114] hold even if the sender of a ciphertext is coerced to prove which plaintext message is encrypted. Furthermore, for e-voting protocols, notions of receipt-freeness [17] and coercion-resistance [88] exist, capturing the requirement that a voter's vote is secret even if the voter is coerced to prove their vote or follow the coercer's instructions. These advanced notions of privacy for e-voting are a topic of current

research [22, 29, 48, 75] on which the literature has not yet reached a consensus.

For the cryptographic primitives and protocols previously discussed, the need for data privacy is rather intuitive. For other primitives, the need for privacy is not immediately apparent. For instance, digital signature schemes allow signers to produce publicly verifiable signatures for messages. They aim to guarantee that the signed message has not been altered and that the signer did, in fact, produce the signature. However, if a signer is coerced to sign a message, the signer may wish to indicate coercion to a trusted party and keep this act of notification private from the coercer. In this way, data privacy issues can arise for cryptographic primitives and protocols that are not traditionally concerned with privacy.

In many settings, privacy must be delicately balanced with other aims of cryptographic protocols. For example, e-voting protocols, in addition to privacy, generally require that the election result is verifiable. In this context, privacy and verifiability are often competing and must be carefully balanced. In other contexts, it may be desirable to revoke privacy. For example, consider ring signatures, which provide a guarantee of data source privacy, or anonymity. Ring signatures allow signers to produce signatures and remain private within a set of users known as the ring. In the event of malicious behaviour, it may be desirable to revoke anonymity and uncover the signer's identity. Thus, a signer should be anonymous by default, yet there must exist a mechanism to revoke anonymity. These competing aims present a challenge for security and privacy definitions and protocol design.

One way in which privacy (and, more generally, security) of cryptographic protocols can be reasoned about is through the provable security paradigm [69]. Provable security is the branch of cryptography that formally and rigorously defines the security of a protocol. In doing so, it is possible to prove formally that a protocol meets its stated security goals. In this thesis, we examine the challenges related to privacy outlined above through the lens of the provable security paradigm. We use several different settings, namely, e-voting, digital signature schemes, and ring signature schemes, to present and explore results related to formal definitions of privacy.

1.2 Thesis Structure

Here, we detail the structure of this thesis and outline the main results.

Preliminaries. In **Chapter 2**, we present the preliminaries for this thesis. We provide an overview of provable security, focusing on the techniques that we use in this thesis. We then present formal definitions of the cryptographic primitives and related security notions that we use in the subsequent chapters of this thesis. Additionally, in Chapters 3 – 6, we present background material and provide motivation specific to each chapter.

Electronic Voting. Privacy for e-voting is often presented as a hierarchy of security properties [51], encompassing notions of ballot secrecy, receipt-freeness, and coercion-resistance. Firstly, *ballot secrecy* intuitively states that a voter's vote remains secret throughout the election, except when the result of the election reveals the vote. Next, *receipt-freeness* strengthens ballot secrecy with additional protection against vote-buying. Finally, *coercion-resistance*, whereby a voter can cast their vote as they intended, even if they are under the control of an attacker for some time during the election, is the strongest privacy property. With respect to formal definitions, rigorous definitions of ballot secrecy have been proposed in the literature [22]. However, these definitions generally position themselves in the so-called honest model. That is, they consider all election officials to be trusted. Moreover, formal definitions of receipt-freeness and coercion-resistance have not been the subject of careful analysis, and precise definitions of these notions remain an open problem. We address these gaps in the literature in Chapters 3 and 4.

In **Chapter 3**, we direct our attention to ballot secrecy, contributing new formal definitions. Our definitions are based on well-established techniques used to define privacy within the cryptographic literature. As a starting point, we present an intuitive definition of ballot secrecy in the honest model. This definition provides a basis for extension to the malicious model, and it also presents an advantage over existing definitions. For e-voting schemes that register and issue credentials to voters, our new definition captures adaptive voter corruption, which is not captured in previous definitions. We elaborate on adaptive corruption strategies and this particular feature of our definition in Chapters 2.2 and 3 respectively. We then show that our ballot secrecy definition can be extended to

the malicious setting, presenting two further definitions of ballot secrecy that capture a malicious election tallier and a malicious ballot box manager (who controls the ballots submitted by voters before the election tallying process). As with our definition in the honest model, our definitions in the malicious setting present advantages over existing work. That being said, existing definitions from the literature have their own advantages. To fully understand the many benefits and limitations of any definition, we compare and contrast our new definitions with those that exist within the e-voting literature throughout Chapter 3. We conclude that our new definitions offer advantages, and, more generally, the work presented in this chapter complements and clarifies the rich and complex landscape of ballot secrecy definitions.

We then consider the stronger notion of receipt-freeness in **Chapter 4**. Specifically, we systematically analyse three state-of-the-art definitions of receipt-freeness, finding weaknesses and/or limitations with all three definitions. That is, we uncover technical weaknesses in the definitions we consider, meaning that e-voting schemes that are not intuitively receipt-free can satisfy a formal definition of receipt-freeness. Fortunately, we show that these weaknesses can be addressed; we present suitable adjustments to each definition. We also expose a common limitation of existing definitions: there exist e-voting schemes that are intuitively receipt-free that do not satisfy the definitions. As previously noted in this introduction, defining advanced notions of privacy is difficult. In Chapter 4, we find that this is certainly the case for the receipt-freeness property. Indeed, not only do definitions have weaknesses and limitations, we also find that each definition considers an attacker with different capabilities. Consequently, existing definitions do not agree on the precise attack model for receipt-freeness.

In the remainder of Chapter 4, we introduce new definitions of receipt-freeness. Our new definitions avoid the weaknesses of existing definitions. Moreover, we draw upon our discussions surrounding the receipt-freeness attacker captured by existing definitions and, more generally, the wider receipt-freeness literature to capture a widely accepted attack model for receipt-freeness. We present two definitions of receipt-freeness to avoid the common limitation in existing definitions and show that e-voting schemes, which are not in the scope of existing definitions, fall within the scope of our definitions. We demonstrate that one of our definitions of receipt-freeness is stronger than the other. Furthermore, we prove that e-voting schemes that are intuitively coercion-resistant only satisfy our weaker receipt-freeness notion, yet there exist receipt-free (and not coercion-resistant) schemes

that can satisfy our stronger notion. Our results in Chapter 4 show that, though our definitions address multiple issues from the literature, it is challenging to get advanced notions of privacy 'right'. However, we believe that our contributions in this chapter can act as a springboard for future research on this topic.

Incoercible Digital Signatures. Inspired by work that defines notions of receipt-freeness and coercion-resistance for multi-party protocols [127], and to further our understanding of these notions more generally, we consider receipt-freeness and coercion-resistance within a more fundamental setting. In the context of digital signature schemes, we consider the scenario in which an attacker instructs a signer to produce signatures for messages chosen by the attacker. It may be desirable that a signature scheme is resistant to such an attack. However, standard digital signatures are publicly verifiable, and an attacker can therefore determine whether the signer produced a requested signature or not. Therefore, there is a need for resistance from coercive attacks for digital signatures. As such, in Chapter 5, we consider the adoption of receipt-freeness and coercion-resistance to the setting of digital signature schemes.

In **Chapter 5** we define a new primitive: *incoercible digital signatures*. Like standard digital signatures, incoercible signatures allow users to produce publicly verifiable signatures for messages chosen by the user. Additionally, incoercible signatures define an entity to whom the signer can indicate coercion, which we call the designated authenticator. In this way, if a signer is coerced to produce a signature for a message chosen by the attacker, the signer can produce a publicly verifiable signature (thereby convincing the attacker that the coercion attempt was successful) and indicate coercion to the authenticator without detection by the coercer. We define a security model for our new primitive that captures notions of receipt-freeness and coercion-resistance. We define these properties analogously to privacy definitions within the e-voting setting. Given the challenges defining privacy for e-voting, which we discuss in Chapter 4, one of the motivations for defining incoercible signatures is to explore whether it is more straightforward to capture these advanced privacy notions in a different setting. Our work in this chapter demonstrates that it is possible to reason clearly about these notions in the context of digital signatures. Furthermore, we are hopeful that some insights and results from the incoercible signature work will influence progress on this topic in the e-voting setting.

As incoercible signature schemes are a new primitive, we show that it is possible to achieve them. Consequently, we present two generic incoercible signature constructions and show that both constructions provably satisfy our security model. Our constructions are built using established cryptographic primitives from the literature, namely, a standard digital signature scheme and a deniable encryption scheme. Finally, we reflect on the primitives used in our construction. In [34], it was noted that deniable encryption can be used to construct receipt-free e-voting schemes, alluding to a potential relationship between receipt-freeness and deniable encryption. As one of the main goals of this thesis is to understand advanced privacy notions, we explore this relationship further. We show that a variant of deniable encryption is necessary to construct receipt-free incoercible signatures.

Report and Trace Ring Signatures. Finally, we investigate the topic of balancing privacy with other security goals. We consider this problem in the context of ring signatures [112] because ring signatures provide signers with anonymity within a set of users, known as the ring. *Accountable ring signatures* [31, 130] balance anonymity with *traceability*, the property that a signer's anonymity can be revoked, and, to this end, define a trusted designated tracer. In practice, to begin the tracing process, the designated tracer in an accountable ring signature may receive a report of malicious behaviour from a reporter. However, the reporter is outside the scope of the syntax and security model of accountable ring signatures. Consequently, it is implicit that the tracer must be trusted not to revoke anonymity without first receiving a report. Moreover, by omitting the role of the reporter from the security model, it is not possible to make any formal statements about the reporter's privacy. In Chapter 6, we explore a rebalancing of the competing interests of anonymity and traceability.

In **Chapter 6**, we introduce *report and trace ring signatures*, a new primitive that formally models the role of the reporter. Specifically, report and trace ring signatures require that a report is submitted before tracing can commence. We also present a security model for our new primitive that captures anonymity, traceability and additionally captures reporter anonymity. Thus, we ensure that our new primitive incorporates the ability to trace malicious users while protecting the anonymity of honest users. We then present a generic construction and concrete instantiation of a report and trace ring signature and prove its security. We base our construction on the accountable ring signature of [31], which allows us to demonstrate the additional costs of reporting. Our efficiency findings lead

us to conclude that it is possible to design report and trace ring signatures that balance anonymity and traceability at a reasonable cost.

Within the ring signature literature, there exist several ring signature variants that provide some ability to trace signers, including linkable [98, 99], link and traceable [68, 74], and accountable ring signatures [31, 130]. We aim to clarify the relationship between these existing ring signature variants and their relationship to report and trace ring signatures. Thus, in Chapter 6, we also define syntax and a security model that captures these variants and then formally prove relations amongst these variants.

Conclusions. We summarise the findings of this thesis and provide concluding remarks in **Chapter 7**. In particular, we reflect on our results, highlighting that the security definitions and models that we present address gaps in the literature, capturing new attacks and corruption strategies. Additionally, throughout this thesis, we contextualise our work into the broader literature. We hope that the results in this thesis aid understanding of complex notions of privacy and act as inspiration for future work on privacy definitions.

1.3 Publications

The content of this thesis is based on several co-authored papers. Here, I introduce the relevant publications and outline my contributions.

Chapter 3 is based on the paper 'Protecting the Privacy of Voters: New Definitions of Ballot Secrecy for E-Voting' [65]. This work was completed under the supervision of Elizabeth A. Quaglia and was presented at SAC 2020.

Chapter 4.1 – 4.5 is based on the paper 'A Critique of Game-Based Definitions of Receipt-Freeness for Voting' [66], which is joint work with Elizabeth A. Quaglia and Ben Smyth. This work was presented at ProvSec 2019 and won the Best Student Paper award. I was the lead author on this paper and completed this work under the supervision and guidance of Ben and Liz. In particular, with respect to the weaknesses and limitations of existing receipt-freeness definitions that we identified, I developed and formalised our ideas and contributed all proofs contained in [66] and the versions that appear in this

thesis. Following the publication of [66], I developed the ideas in Chapter 4.6 under the supervision of Liz.

Chapter 5 is based on the paper 'On the Incoercibility of Digital Signatures', which is joint work with Lydia Garms and Elizabeth A. Quaglia. This paper is currently awaiting submission at an appropriate venue. In this paper, we define two types of incoercible digital signatures: publicly verifiable incoercible signatures and strong designated verifier incoercible signatures. Under the supervision of Liz, Lydia and I co-led this project. Specifically, I focused on the publicly verifiable setting, and Lydia focused on the strong designated verifier setting. In Chapter 5, I include only the work on publicly verifiable incoercible signatures, for which I was the lead author. As such, I contributed the security model, constructions and security proofs presented in Chapter 5.

Chapter 6 is based on the paper 'Trace Me For A Reason: Report and Trace Ring Signatures'. This work was completed under the supervision of Elizabeth A. Quaglia and was presented at CANS 2021.

Chapter 2

Preliminaries

Contents

In this chapter we introduce the definitions and ideas that we use in this thesis. We first introduce notation. Then, we provide an overview of provable security and present definitions and security models for the cryptographic primitives that are used throughout this thesis.

2.1 Notation

We first introduce some notation that will be used throughout this thesis.

Sets, lists and vectors. We denote the set of natural numbers as $\mathbb{N}$, the set of integers as $\mathbb{Z}$ and the set of real numbers as $\mathbb{R}$. We let $\mathbb{Z}_p$ denote the set $\{1, \ldots, p-1\}$ and

write $[n]$ to denote the set $\{1, \ldots, n\}$ for $n > 1$. The empty set is denoted as $\emptyset$. We write $\{\{x_1, \ldots, x_n\}\}$ to denote a multiset consisting of elements $x_1, \ldots, x_n$, writing $\{\{\}\}$ to denote the empty multiset. We write $L = (x_1, \ldots, x_n)$ to denote a list L with entries $x_1, \ldots, x_n$, and write $L \leftarrow L||y$ to denote appending y to the list L. We write $\mathbf{v}$ to denote a vector and write $\mathbf{v}[i]$ to denote the value of vector $\mathbf{v}$ at position i.

Variables and strings. We write $x \leftarrow X$ to denote assignment of X to x. We write $y \leftarrow\!\!\$\, Y$ to denote choosing an element from set Y uniformly at random and assigning it to the variable y. We let ϵ denote the empty string.

Algorithms. We let $y \leftarrow\!\!\$\, A(x_1, \ldots, x_n; c)$ denote the result of running probabilistic algorithm A on inputs $x_1, \ldots, x_n$ and coins c. Generally, we omit coins and simply write $y \leftarrow\!\!\$\, A(x_1, \ldots, x_n)$. We write PPT to denote a probabilistic algorithm that runs in polynomial time. Moreover, we denote $y \leftarrow A(x_1, \ldots, x_n)$ as the result of running deterministic algorithm A on inputs $x_1, \ldots, x_n$. We denote failure of an algorithm as $\perp$. We write $A^{\mathcal{O}}$ to denote algorithm A with access to an oracle $\mathcal{O}$, and write $\mathcal{O}\mathsf{x}_{(y_1,\ldots,y_n)}(z_1, \ldots, z_n)$ to denote oracle $\mathcal{O}\mathsf{x}$ with access to $y_1, \ldots, y_n$ that takes as input $z_1, \ldots, z_n$. For brevity, we sometimes simply write $\mathcal{O}\mathsf{x}(z_1, \ldots, z_n)$.

Functions. We write $\mathsf{poly}(\lambda)$ to denote any polynomial function of λ. We say that a function $f\colon \mathbb{N} \to \mathbb{R}$ is negligible if, for every positive polynomial p, there exists an N such that for all integers $n > N$, $f(n) = 1/p(n)$. Throughout this thesis, we denote an arbitrary negligible function as $\mathsf{negl}(\lambda)$.

2.2 Provable Security

The provable security paradigm first appeared in [69], in which Goldwasser and Micali introduced a public-key encryption scheme and proposed a formalisation of the property now commonly known as IND-CPA security. The authors then presented a formal security proof that their scheme satisfied their security definition. In doing so, Goldwasser and Micali departed from the previous approach to security analysis of cryptographic protocols, which

relied on intuition and exhaustively ruling out any known attacks. Since its introduction, the provable security paradigm has risen in popularity. It has been used to define basic notions of security for digital signature schemes [71], and for more complex cryptographic protocols, for example, see [22, 38, 88].

Provable security can lead to good guarantees of security. For example, suppose a cryptographic protocol is formally shown to satisfy a given security definition. Then, it is guaranteed that the protocol is secure against all attacks captured by the security definition. However, the security guarantees of a formal proof are only as good as the definition. If a definition is not well-crafted, it may miss some attacks and, consequently, a formal security proof provides no guarantee that a protocol is free from these missed attacks. Therefore, careful consideration must be given to a definition to ensure it resists relevant attacks.

In this thesis, we present new definitions of privacy and security for various cryptographic primitives and protocols. To do so, we rely on techniques from the provable security paradigm that we outline here, namely, formal security definitions and a proof technique known as game hopping.

2.2.1 Security Definitions

A formal definition of security describes an experiment, denoted Exp, that is played between a challenger and an adversary, denoted $\mathcal{A}$. We describe the role of the challenger here but note that this player is implicit in the experiments within this thesis. The challenger performs setup operations for a cryptographic scheme, for example, generating public parameters and key pairs for cryptographic primitives, and provides inputs to $\mathcal{A}$. The experiment defines the actions that $\mathcal{A}$ can take and how they interact with the scheme. Usually, interaction with the scheme is modelled by providing $\mathcal{A}$ with access to oracles, which are black boxes that $\mathcal{A}$ can query and receive input from. Eventually, $\mathcal{A}$ terminates, and the experiment outputs a bit β.

Adversarial success. We define the probability that $\mathcal{A}$ succeeds in the experiment, which we call the advantage of the adversary. In the experiments described in this thesis, $\mathcal{A}$ can succeed in two ways. Firstly, $\mathcal{A}$ may be tasked with solving a problem (for example,

outputting a valid forgery of a digital signature, e.g., Definition 18). In this event, the experiment returns 1 if $\mathcal{A}$ solves the problem. Formally, we write the advantage of an adversary $\mathcal{A}$ in an experiment X against a protocol Π as

$$\mathsf{Adv}_{\Pi,\mathcal{A}} = \Pr\left[\mathsf{Exp}^{\mathsf{X}}_{\Pi,\mathcal{A}}(\lambda) = 1\right]$$

where the probability is taken over the randomness of the experiment and the adversary. In this thesis, we usually require that advantage $\mathsf{Adv}_{\Pi,\mathcal{A}}$ is negligible.

Sometimes, $\mathcal{A}$ is asked to distinguish two different experiments. The experiment is determined by selecting a bit $\beta \leftarrow_{\$} \{0,1\}$. $\mathcal{A}$ is tasked with guessing the bit β, and the experiment returns the bit β' that is guessed by $\mathcal{A}$. Here, we write the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}_{\Pi,\mathcal{A}} = \left| \Pr\left[\mathsf{Exp}^{\mathsf{X},\beta=0}_{\Pi,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{X},\beta=1}_{\Pi,\mathcal{A}}(\lambda) = 1\right] \right|$$

where the probability is taken over the randomness of the experiment and the adversary. We require that the advantage of $\mathcal{A}$ is negligible. This is equivalent to writing that

$$\mathsf{Adv}_{\Pi,\mathcal{A}} = \left| \Pr\left[\mathsf{Exp}^{\mathsf{X},\beta}_{\Pi,\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right|.$$

Simulation-based definitions. Security definitions can be formulated as simulation-based definitions. In contrast to the security experiment outlined above, the simulation-based approach tasks the adversary with distinguishing a run of a real protocol from a run of an ideal protocol. Simulation-based definitions are commonly used to present universally composable proofs of security [33]. In this thesis, we follow the security definition outlined above but note that alternative methods for defining security are available.

Corruption strategies. Adversary $\mathcal{A}$ may be allowed to corrupt some protocol participants. Looking forward to Chapter 3, in an e-voting scheme, $\mathcal{A}$ may be able to corrupt voters. In our experiments, this is usually modelled by providing $\mathcal{A}$ with access to an oracle that returns the secrets of the entity if $\mathcal{A}$ is permitted to corrupt the queried entity. Corruption strategies may be *static* or *adaptive*. Static corruption strategies mean that $\mathcal{A}$ can corrupt entities at the beginning of the experiment only, and *before* performing any other actions. Adaptive corruption strategies, on the other hand, allow $\mathcal{A}$ to corrupt

entities at any point during the experiment.

2.2.2 Game Hopping

Once a formal definition of privacy is established, the next step is to prove formally that a cryptographic protocol satisfies the definition. One technique that can be used to perform this step is the game hopping proof technique. Here, we provide an outline of game hopping proofs and refer the reader to [13] for a complete description.

Game hopping defines a finite sequence of games $G_0, \ldots, G_n$ where game G_0 is typically defined to be the experiment $\mathsf{Exp}^{\mathsf{X}}_{\Pi,\mathcal{A}}(\lambda)$. Then, each subsequent game is defined as a modification of the previous game. That is, Game G_i is a modification of game G_{i-1} for $i \in \{1, \ldots, n\}$. After each game hop, we write the chance of $\mathcal{A}$ succeeding in game G_i as S_i, and we require that the value

$$\Big|\Pr[S_{i-1}] - \Pr[S_i]\Big|$$

is reasonably bounded, i.e., negligible. We choose modifications such that any two subsequent games are minimally different. In doing so, the value above can be bounded, typically by showing a reduction to the security of a cryptographic primitive within the construction. Then, by the triangle inequality,

$$\Big|\Pr[S_0] - \Pr[S_n]\Big| \leq \Big|\Pr[S_0] - \Pr[S_1]\Big| + \cdots + \Big|\Pr[S_{n-1}] - \Pr[S_n]\Big|$$

and, if all these probabilities are negligible, we conclude that the success of the adversary in the experiment is negligible as required.

2.3 Number Theory and Cryptographic Hardness Assumptions

Let $\mathcal{G}$ be a group generation algorithm that outputs a tuple $(\mathbb{G}, p, g)$ where $\mathbb{G}$ is the description of a cyclic group with prime order p and $g \in \mathbb{G}$ is a generator of the group. Here, we recall some cryptographic hardness assumptions related to cyclic groups.

Discrete logarithms. In a cyclic group, for every element $y \in \mathbb{G}$, there exists a unique $x \in \mathbb{Z}_p$ such that $y = g^x$. Element x is known as the discrete logarithm of y with respect to $\mathbb{G}$. The *discrete logarithm problem* asks whether, given y, it is possible to compute the discrete logarithm x. The *discrete logarithm assumption* states that the discrete logarithm problem is hard for a group $\mathbb{G}$.

Definition 1 (Discrete logarithm assumption)**.** *The* discrete logarithm assumption *holds in group* $(\mathbb{G}, p, g)$ *if, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr\left[y \leftarrow_{\$} \mathbb{G};\ x \leftarrow_{\$} \mathcal{A}(\mathbb{G}, p, g, y)\ :\ g^x \coloneqq y\right] \leq \mathsf{negl}(\lambda)\,.$$

Decisional Diffie-Hellman. In a cyclic group, the Decisional Diffie-Hellman assumption states that, given elements g^x and g^y for $x, y \leftarrow_{\$} \mathbb{Z}_p$, the value g^{xy} is indistinguishable from a random element from $\mathbb{G}$.

Definition 2 (Decisional Diffie-Hellman assumption)**.** *The* decisional Diffie-Hellman assumption *holds in a group* $(\mathbb{G}, p, g)$ *if, for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$|\Pr[x, y \leftarrow_{\$} \mathbb{Z}_p\ :\ \mathcal{A}(g^x, g^y, g^{xy}) = 1] - \Pr[x, y, z \leftarrow_{\$} \mathbb{Z}_p\ :\ \mathcal{A}(g^x, g^y, g^z) = 1]| \leq \mathsf{negl}(\lambda)\,.$$

2.4 Fundamental Building Blocks

Here, we define the basic cryptographic building blocks used throughout this thesis.

2.4.1 One-Way Functions

Definition 3 (One-way function)**.** *Let* f*,* X *and* Y *be a function, domain and range respectively.* $f : X \to Y$ *is a* one-way function *if the following conditions hold*

- ***Easy to compute****: there exists a polynomial-time algorithm that, on input* $x \in X$*, outputs* $f(x)$*.*

- ***Hard to invert**: for all PPT adversaries $\mathcal{A}$ there exists a negligible function* negl *such that*

$$\Pr\left[x \leftarrow_{\$} X;\ y \leftarrow f(x);\ x' \leftarrow_{\$} \mathcal{A}(y)\ :\ \ f(x') \coloneqq y\right] \leq \mathsf{negl}(\lambda)\,.$$

2.4.2 Hash Functions

A hash function H is a function that takes input from a domain X and outputs a string of fixed length ℓ. We write $H\colon X \to \{0,1\}^{\ell}$. Hash functions are frequently used as building blocks within cryptographic protocols and, to prove security of a protocol, it is often necessary that the hash function satisfies certain properties. We recall these properties here.

Collision-resistance A hash function is said to be collision-resistant if it is hard to find two elements x and x' from the domain of the hash function such that $H(x) = H(x')$. More formally, this can be stated as follows.

Definition 4 (Collision-resistance)**.** *A hash function H is* collision-resistant *if, for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\Pr\left[(x,x') \leftarrow_{\$} \mathcal{A}()\ :\ \ H(x) = H(x')\ \wedge\ x \neq x'\right] \leq \mathsf{negl}(\lambda)\,.$$

Random Oracle Model Collision-resistance is not always sufficient to prove the security of a protocol. That is to say, within the provable security paradigm, it may be the case that a cryptographic scheme cannot be declared to satisfy a given security notion unless a stronger security assumption holds. In such cases, security proofs assume that hash function H is a *random oracle*. In other words, it is assumed that H is a truly random function and can be modelled as a black-box that, on input of some $x \in X$, returns the value of $H(x)$. In practice, the random oracle is instantiated with a concrete hash function for which an attacker can compute the hash of any element from the domain.

The random oracle model was first introduced in [12] and is a useful proof technique in cryptography. In particular, it has three useful properties. Firstly, **uniformity** means that, if x has not been queried to H, the value of $H(x)$ is uniform. Conversely, if x has

been previously queried to H, the oracle always returns the same value $H(x)$. Secondly, in a proof by reduction, if an adversary queries x to H, the reduction can see the query and learn x. This property is known as **extractability**. Finally, the reduction can set the value of $H(x)$ as long as it is uniform. This property is known as **programmability**.

The usefulness of the random oracle model is debated within the cryptographic community (see [89] for a discussion). On the one hand, there exist schemes that are secure in the random oracle model but for which there is no way to instantiate the scheme securely [36]. On the other hand, these schemes are considered to be contrived, and, as of yet, there have been no successful real-world attacks on schemes proven secure in the random oracle model if the scheme is instantiated with a suitable hash function. For our purposes, we believe that a proof in the random oracle model is better than no proof at all. This comes with the caveat that a suitable hash function should always be carefully selected to ensure that any real-world instantiation is secure.

2.4.3 Public-Key Encryption

We present the definition of a public-key encryption scheme and define its security.

Definition 5 (PKE scheme)**.** *A* public-key encryption scheme PKE *is a tuple of polynomial time algorithms* $(\mathsf{PKE.Setup}, \mathsf{PKE.KGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *such that*

$\mathsf{PKE.Setup}(1^\lambda)$ On input of security parameter 1^λ, algorithm $\mathsf{PKE.Setup}$ outputs public parameters pp_{PKE}. We assume that pp_{PKE} defines the message space $\mathsf{M}_{\mathsf{PKE}}$, randomness space $\mathsf{Rand}_{\mathsf{PKE}}$, and key spaces PK_{PKE} and SK_{PKE}.

$\mathsf{PKE.KGen}(pp_{\mathsf{PKE}})$ On input of public parameters pp_{PKE}, algorithm $\mathsf{PKE.KGen}$ outputs a key pair $(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}})$ where pk_{PKE} is the public encryption key and sk_{PKE} is the secret decryption key.

$\mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, m)$ On input of public parameters pp_{PKE}, public encryption key pk_{PKE} and message $m \in \mathsf{M}_{\mathsf{PKE}}$, algorithm $\mathsf{PKE.Enc}$ outputs a ciphertext c.

$\mathsf{PKE.Dec}(pp_{\mathsf{PKE}}, sk_{\mathsf{PKE}}, c)$ On input of public parameters pp_{PKE}, secret decryption key sk_{PKE} and ciphertext c, algorithm $\mathsf{PKE.Dec}$ outputs a message m.

Correctness. A public-key encryption scheme must satisfy *correctness*, which we define in Definition 6.

Definition 6 (Correctness)**.** *A public-key encryption scheme* PKE *satisfies* correctness *if, for any message* $m \in \mathsf{M}_{\mathsf{PKE}}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} pp_{\mathsf{PKE}} \leftarrow\!\$\, \mathsf{PKE.Setup}(1^\lambda); \\ (pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}}) \leftarrow\!\$\, \mathsf{PKE.KGen}(pp_{\mathsf{PKE}}); \\ c \leftarrow\!\$\, \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, m) \end{array} : \mathsf{PKE.Dec}(pp_{\mathsf{PKE}}, sk_{\mathsf{PKE}}, c) \coloneqq m\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Security against chosen-plaintext attacks. An attacker should learn no more information from a ciphertext about the plaintext message than can be learned without the ciphertext. This is captured as the *indistinguishability under a chosen-plaintext attack* or, more simply, $\mathsf{IND\text{-}CPA}$ security. We formally capture this property in Definition 7, which describes an experiment in which an adversary can query an oracle $\mathcal{O}\mathsf{enc}$ once to obtain a challenge ciphertext that contains an encryption of a message m_β, where m_0 and m_1 are input to the oracle by the adversary. The adversary attempts to guess bit β. We say that a public-key encryption scheme satisfies $\mathsf{IND\text{-}CPA}$ security if the adversary correctly guesses β with probability only negligibly more than $1/2$.

Definition 7 ($\mathsf{IND\text{-}CPA}$ [9])**.** *A public-key encryption scheme* PKE *satisfies* $\mathsf{IND\text{-}CPA}$ *if, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}CPA},0}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}CPA},1}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{IND\text{-}CPA},\beta}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ *is the experiment defined in Figure 2.1 for* $\beta \in \{0,1\}$*.*

$\mathsf{Exp}^{\mathsf{IND\text{-}CPA},\beta}_{\mathsf{PKE},\mathcal{A}}(\lambda)$	$\mathcal{O}\mathsf{enc}_{(\mathsf{flag},pp_{\mathsf{PKE}},pk_{\mathsf{PKE}})}(m_0,m_1)$
$\mathsf{flag} \leftarrow 0$	**if** $\mathsf{flag} = 1$ **return** $\perp$
$pp_{\mathsf{PKE}} \leftarrow\!\$\, \mathsf{PKE.Setup}(1^\lambda)$	$c^* \leftarrow\!\$\, \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, m_\beta)$
$(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}}) \leftarrow\!\$\, \mathsf{PKE.KGen}(pp_{\mathsf{PKE}})$	$\mathsf{flag} \leftarrow 1$
$\beta' \leftarrow\!\$\, \mathcal{A}^{\mathcal{O}\mathsf{enc}}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}})$	**return** c^*
return β'	

Figure 2.1: The $\mathsf{IND\text{-}CPA}$ experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CPA},\beta}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ for public-key encryption scheme PKE.

$\mathsf{IND\text{-}CPA}$ security can be defined for *multiple encryptions*. To model this, the adversary is not restricted to a single query to oracle $\mathcal{O}\mathsf{enc}$, and can make multiple queries of the form $\mathcal{O}\mathsf{enc}(m_0, m_1)$. We note that stronger indistinguishability notions exist for public-key encryption schemes, namely $\mathsf{IND\text{-}CCA}$ [9], but we do not require this in this thesis.

Non-malleable security against chosen-plaintext attacks. *Non-malleability under a chosen-plaintext attack* (NM-CPA security) strengthens IND-CPA security by capturing an attacker that can additionally obtain decryptions of a vector of ciphertexts. As in our IND-CPA experiment, an adversary in the NM-CPA experiment can query oracle $\mathcal{O}\mathsf{enc}$ once. Furthermore, the adversary can query a vector of ciphertexts to an oracle $\mathcal{O}\mathsf{dec}$, which returns a vector of plaintexts that correspond to the decryption of a vector of ciphertexts. NM-CPA security can be defined for multiple encryptions analogously to IND-CPA for multiple encryptions.

Definition 8 (NM-CPA [14])**.** *A public-key encryption scheme* PKE *satisfies* NM-CPA *if, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{Exp}^{\mathsf{NM\text{-}CPA},0}_{\mathsf{PKE},\mathcal{A}}(\lambda)=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{NM\text{-}CPA},1}_{\mathsf{PKE},\mathcal{A}}(\lambda)=1\right]\leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{NM\text{-}CPA},\beta}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ *is the experiment defined in Figure 2.2 for* $\beta \in \{0,1\}$*.*

$\underline{\mathsf{Exp}^{\mathsf{NM\text{-}CPA},\beta}_{\mathsf{PKE},\mathcal{A}}(\lambda)}$

$\mathsf{flag} \leftarrow 0$
$\mathsf{flag}' \leftarrow 0$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}})$
$\beta' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{enc},\mathcal{O}\mathsf{dec}}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}})$
if $\forall i = 1,\ldots,|\mathbf{c}| \ :\ c^* \neq \mathbf{c}[i]$
  return β'

$\underline{\mathcal{O}\mathsf{enc}_{(\mathsf{flag},\mathsf{flag}',pp_{\mathsf{PKE}},pk_{\mathsf{PKE}})}(m_0,m_1)}$

if $\mathsf{flag} = 1 \ \vee \mathsf{flag}' = 1$ **return** $\perp$
$c^* \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, m_\beta)$
$\mathsf{flag} \leftarrow 1$
return c^*

$\underline{\mathcal{O}\mathsf{dec}_{(\mathsf{flag}',pp_{\mathsf{PKE}},sk_{\mathsf{PKE}})}(\mathbf{c})}$

if $\mathsf{flag}' = 1$ **return** $\perp$
for $i = 1,\ldots,|\mathbf{c}| \ :\ m_i \leftarrow \mathsf{PKE.Dec}(pp_{\mathsf{PKE}}, sk_{\mathsf{PKE}}, \mathbf{c}[i])$
$\mathbf{m} \leftarrow (m_1,\ldots,m_{|\mathbf{c}|})$
$\mathsf{flag}' \leftarrow 1$
return $\mathbf{m}$

Figure 2.2: The NM-CPA experiment $\mathsf{Exp}^{\mathsf{NM\text{-}CPA},\beta}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ for public-key encryption scheme PKE.

Homomorphic encryption. A public-key encryption scheme can be *homomorphic*, which means that computations can be performed on ciphertexts, resulting in a ciphertext that is an encryption of the result of the computation. Looking forward, we use a *multiplicatively* homomorphic public-key encryption scheme to build an e-voting scheme in Chapter 3 of this thesis. In doing so, we follow an established approach to building e-voting scheme constructions. In fact, the Helios e-voting scheme [2, 77] and related e-voting scheme constructions [22, 23] are built using a multiplicatively homomorphic public-key encryption scheme.

We say that a public-key encryption scheme is multiplicatively homomorphic (which we henceforth refer to as simply homomorphic for brevity) if, for all pp_{PKE} output by algorithm $\mathsf{PKE.Setup}$ and all $(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}})$ output by algorithm $\mathsf{PKE.KGen}$, if $c_1 \leftarrow\!\!\$\, \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, m_1)$ and $c_2 \leftarrow\!\!\$\, \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, m_2)$ for $m_1, m_2 \in \mathsf{M}_{\mathsf{PKE}}$, then $\mathsf{PKE.Dec}(pp_{\mathsf{PKE}}, sk_{\mathsf{PKE}}, c_1 \cdot c_2) \coloneqq m_1 \cdot m_2$.

ElGamal encryption. Within this thesis, we rely on the ElGamal encryption scheme. Specifically, we use ElGamal encryption for our concrete instantiation of a report and trace ring signature scheme in Chapter 6. ElGamal encryption was first introduced in [61], and we recall its formal definition in Figure 2.3.

$\underline{\mathsf{PKE.Setup}(1^\lambda)}$
$(\mathbb{G}, p, g) \leftarrow\!\!\$\, \mathcal{G}(1^\lambda)$
return $pp_{\mathsf{PKE}} = (\mathbb{G}, p, g)$

$\underline{\mathsf{PKE.KGen}(pp_{\mathsf{PKE}})}$
parse pp_{PKE} as $(\mathbb{G}, p, g)$
$sk_{\mathsf{PKE}} \leftarrow\!\!\$\, \mathbb{Z}_p$
$pk_{\mathsf{PKE}} \leftarrow g^{sk_{\mathsf{PKE}}}$
return $(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}})$

$\underline{\mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, m)}$
parse pp_{PKE} as $(\mathbb{G}, p, g)$
$r \leftarrow\!\!\$\, \mathbb{Z}_p$
return $c \leftarrow (c_1 = g^r, c_2 = pk_{\mathsf{PKE}}^r \cdot m)$

$\underline{\mathsf{PKE.Dec}(pp_{\mathsf{PKE}}, sk_{\mathsf{PKE}}, c)}$
parse c as (c_1, c_2)
return $\dfrac{c_2}{c_1^{sk_{\mathsf{PKE}}}}$

Figure 2.3: The ElGamal public-key encryption scheme [61].

The ElGamal public-key encryption scheme is homomorphic. Indeed, for any public parameters pp_{PKE} and key pair $(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}})$ generated by algorithms $\mathsf{PKE.Setup}$ and $\mathsf{PKE.KGen}$ respectively, for $m_1, m_2 \in \mathsf{M}_{\mathsf{PKE}}$ and $r_1, r_2 \in \mathsf{Rand}_{\mathsf{PKE}}$, we have that

$$\mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, m_1; r_1) \cdot \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, m_2; r_2) = (g^{r_1+r_2}, pk_{\mathsf{PKE}}^{r_1+r_2} \cdot m_1 \cdot m_2)$$

and, as required,

$$\mathsf{PKE.Dec}(pp_{\mathsf{PKE}}, sk_{\mathsf{PKE}}, (g^{r_1+r_2}, pk_{\mathsf{PKE}}^{r_1+r_2} \cdot m_1 \cdot m_2)) = \frac{pk_{\mathsf{PKE}}^{r_1+r_2} \cdot m_1 \cdot m_2}{(g^{r_1+r_2})^{sk_{\mathsf{PKE}}}} = m_1 \cdot m_2.$$

2.4.4 Deniable Encryption

Deniable encryption was introduced in [34], and we recall the syntax and security model for a public-key sender-deniable encryption scheme presented in [114], which we refer to as a deniable encryption scheme for brevity. A deniable encryption scheme is a standard

public-key encryption scheme, equipped with an additional algorithm $\mathsf{DEN.Exp}$ that, for a ciphertext c that encrypts message m and is output by encryption algorithm $\mathsf{DEN.Enc}$, generates randomness such that c appears to encrypt an alternative message m'.

Definition 9 (Deniable Encryption Scheme)**.** *A* deniable encryption scheme (DEN) *is a tuple of polynomial time algorithms* ($\mathsf{DEN.Setup}, \mathsf{DEN.KGen}, \mathsf{DEN.Enc}, \mathsf{DEN.Dec}, \mathsf{DEN.}\,\mathrm{exp}$) *such that*

$\mathsf{DEN.Setup}(1^\lambda)$ On input of security parameter 1^λ, algorithm $\mathsf{DEN.Setup}$ outputs public parameters pp_{DEN}. We assume that pp_{DEN} defines the message space $\mathsf{M}_{\mathsf{DEN}}$ and the randomness space $\mathsf{Rand}_{\mathsf{DEN}}$.

$\mathsf{DEN.KGen}(pp_{\mathsf{DEN}})$ On input of public parameters pp_{DEN}, algorithm $\mathsf{DEN.KGen}$ outputs a key pair $(pk_{\mathsf{DEN}}, sk_{\mathsf{DEN}})$ where pk_{DEN} is the public encryption key and sk_{DEN} is the secret decryption key.

$\mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, m)$ On input of public parameters pp_{DEN}, public key pk_{DEN} and message $m \in \mathsf{M}_{\mathsf{DEN}}$, algorithm $\mathsf{DEN.Enc}$ outputs a ciphertext c.

$\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, c)$ On input of public parameters pp_{DEN}, secret key sk_{DEN} and ciphertext c, algorithm $\mathsf{DEN.Dec}$ outputs a message m.

$\mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, c, m)$ On input of public parameters pp_{DEN}, public key pk_{DEN}, ciphertext c and message m, algorithm $\mathsf{DEN.Dec}$ outputs a string r.

Correctness. Similarly to a standard public-key encryption scheme, a deniable encryption scheme must satisfy *correctness*.

Definition 10 (Correctness)**.** *A deniable encryption scheme* DEN *satisfies* correctness *if, for any message* $m \in \mathsf{M}_{\mathsf{DEN}}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} pp_{\mathsf{DEN}} \leftarrow\!\$\, \mathsf{DEN.Setup}(1^\lambda); \\ (pk_{\mathsf{DEN}}, sk_{\mathsf{DEN}}) \leftarrow\!\$\, \mathsf{DEN.KGen}(pp_{\mathsf{DEN}}); \\ c \leftarrow\!\$\, \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, m) \end{array} : \mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_{\mathsf{DEN}}, c) \coloneqq m\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Security against chosen-plaintext attacks. We recall a definition of $\mathsf{IND\text{-}CPA}$ security for a deniable encryption scheme, which is defined analogously to $\mathsf{IND\text{-}CPA}$ security of a public-key encryption scheme.

Definition 11 (IND-CPA [114])**.** *A deniable encryption scheme* DEN *satisfies* IND-CPA *if, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}CPA},0}_{\mathsf{DEN},\mathcal{A}}(\lambda)=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}CPA},1}_{\mathsf{DEN},\mathcal{A}}(\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{IND\text{-}CPA},\beta}_{\mathsf{DEN},\mathcal{A}}(\lambda)$ *is the experiment defined in Figure 2.4 for* $\beta\in\{0,1\}$*.*

Indistinguishability of explanations. A deniable encryption scheme should satisfy *indistinguishability of explanations* (IND-EXP security). This captures the intuition that an attacker cannot distinguish randomness used to encrypt and randomness output by algorithm DEN.Exp. We therefore define an experiment in which the adversary is provided with a ciphertext and randomness that is either used by algorithm DEN.Enc to produce the ciphertext, or is the output of algorithm DEN.Exp, depending on a bit β. If the adversary can guess β with probability only negligibly more than $1/2$, we say that a deniable encryption scheme satisfies IND-EXP security.

Definition 12 (IND-EXP [114])**.** *A deniable encryption scheme* DEN *satisfies* IND-EXP *if, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}EXP},0}_{\mathsf{DEN},\mathcal{A}}(\lambda)=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}EXP},1}_{\mathsf{DEN},\mathcal{A}}(\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{IND\text{-}EXP},\beta}_{\mathsf{DEN},\mathcal{A}}(\lambda)$ *is the experiment defined in Figure 2.4 for* $\beta\in\{0,1\}$*.*

2.4.5 Commitment Schemes

Commitment schemes allow a sender to commit to a message. The message is hidden to the receiver of the commitment, and the sender can reveal the message at any point by revealing the randomness used to commit. A receiver can then recompute the commitment and check whether it matches the one generated by the sender.

Definition 13 (Commitment Scheme)**.** *A* commitment scheme COM *is a tuple of polynomial time algorithms* (COM.Setup, COM.KGen, COM.Commit) *such that*

$\mathsf{COM.Setup}(1^\lambda)$ On input of security parameter 1^λ, algorithm COM.Setup outputs public parameters pp_{COM}. We assume that pp_{COM} defines the message space $\mathsf{M}_{\mathsf{COM}}$ and the

$\mathsf{Exp}^{\mathsf{IND\text{-}CPA},\beta}_{\mathsf{DEN},\mathcal{A}}(\lambda)$

$\mathsf{flag} \leftarrow 0$
$pp_{\mathsf{DEN}} \leftarrow\!\!\$\ \mathsf{DEN.Setup}(1^\lambda)$
$(pk_{\mathsf{DEN}}, sk_{\mathsf{DEN}}) \leftarrow\!\!\$\ \mathsf{DEN.KGen}(pp_{\mathsf{DEN}})$
$\beta' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{enc}}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}})$
return β'

$\mathcal{O}\mathsf{enc}_{(\mathsf{flag},pp_{\mathsf{DEN}},pk_{\mathsf{DEN}})}(m_0, m_1)$

if $\mathsf{flag} = 1$ **return** $\perp$
$c^* \leftarrow\!\!\$\ \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, m_\beta)$
$\mathsf{flag} \leftarrow 1$
return c^*

$\mathsf{Exp}^{\mathsf{IND\text{-}EXP},\beta}_{\mathsf{DEN},\mathcal{A}}(\lambda)$

$\mathsf{flag} \leftarrow 0$
$pp_{\mathsf{DEN}} \leftarrow\!\!\$\ \mathsf{DEN.Setup}(1^\lambda)$
$(pk_{\mathsf{DEN}}, sk_{\mathsf{DEN}}) \leftarrow\!\!\$\ \mathsf{DEN.KGen}(pp_{\mathsf{DEN}})$
$\beta' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{exp}}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}})$
return β'

$\mathcal{O}\mathsf{exp}_{(\mathsf{flag},pp_{\mathsf{DEN}},pk_{\mathsf{DEN}})}(m^*)$

if $\mathsf{flag} = 1$ **return** $\perp$
$r_0 \leftarrow\!\!\$\ \mathsf{Rand}_{\mathsf{DEN}}$
$c^* \leftarrow\!\!\$\ \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, m^*; r_0)$;
$r_1 \leftarrow\!\!\$\ \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, c^*, m^*)$;
$\mathsf{flag} \leftarrow 1$
return (c^*, r_β)

Figure 2.4: The IND-CPA experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CPA},\beta}_{\mathsf{DEN},\mathcal{A}}(\lambda)$ and IND-EXP experiment $\mathsf{Exp}^{\mathsf{IND\text{-}EXP},\beta}_{\mathsf{DEN},\mathcal{A}}(\lambda)$ for deniable encryption scheme DEN.

randomness space $\mathsf{Rand}_{\mathsf{COM}}$.

$\mathsf{COM.KGen}(pp_{\mathsf{COM}})$ On input of public parameters pp_{COM}, algorithm COM.KGen outputs a commitment key $\mathsf{k}_{\mathsf{COM}}$.

$\mathsf{COM.Commit}(pp_{\mathsf{COM}}, \mathsf{k}_{\mathsf{COM}}, m; r)$ On input of public parameters pp_{COM}, commitment key $\mathsf{k}_{\mathsf{COM}}$, message $m \in \mathsf{M}_{\mathsf{COM}}$ and randomness $r \in \mathsf{Rand}_{\mathsf{COM}}$, algorithm COM.Commit outputs a commitment cm.

Commitment schemes should be *hiding* and *binding*. We provide definitions of computational hiding and binding, taken from [89].

Hiding. The hiding property captures the intuition that an attacker cannot distinguish between a commitment to two possible messages chosen by the attacker. We capture this in Definition 14 that describes an experiment in which an adversary is provided with a commitment for a message m_β for a bit β, where messages m_0 and m_1 are output by the adversary. We say that a commitment scheme is hiding if the adversary guesses β with probability only negligibly more than 1/2.

Definition 14 (Hiding [89])**.** *A commitment scheme* COM *is* hiding *if, for any PPT*

adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* negl *such that*

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{Hiding},0}_{\mathsf{COM},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{Hiding},1}_{\mathsf{COM},\mathcal{A}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{Hiding},\beta}_{\mathsf{COM},\mathcal{A}}(\lambda)$ *is the experiment defined in Figure 2.5 for* $\beta \in \{0,1\}$*.*

Binding. A binding commitment scheme has the property that a commitment can only be opened to one message. We formally define binding in Definition 15, which requires that an adversary can output two messages and randomness that commit to the same commitment with negligible probability.

Definition 15 (Binding [89])**.** *A commitment scheme* COM *is* binding *if, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{Exp}^{\mathsf{Binding}}_{\mathsf{COM},\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{Binding}}_{\mathsf{COM},\mathcal{A}}(\lambda)$ *is the experiment defined in Figure 2.5.*

$\mathsf{Exp}^{\mathsf{hiding},\beta}_{\mathsf{COM},\mathcal{A}}(\lambda)$	$\mathsf{Exp}^{\mathsf{binding}}_{\mathsf{COM},\mathcal{A}}(\lambda)$
$pp_{\mathsf{COM}} \leftarrow\!\!\$\ \mathsf{COM.Setup}(1^\lambda)$	$pp_{\mathsf{COM}} \leftarrow\!\!\$\ \mathsf{COM.Setup}(1^\lambda)$
$\mathsf{k}_{\mathsf{COM}} \leftarrow\!\!\$\ \mathsf{COM.KGen}(pp_{\mathsf{COM}})$	$\mathsf{k}_{\mathsf{COM}} \leftarrow\!\!\$\ \mathsf{COM.KGen}(pp_{\mathsf{COM}})$
$(m_0, m_1, st) \leftarrow\!\!\$\ \mathcal{A}_1(pp_{\mathsf{COM}}, \mathsf{k}_{\mathsf{COM}})$	$(m_0, m_1, r_0, r_1) \leftarrow\!\!\$\ \mathcal{A}(pp_{\mathsf{COM}}, \mathsf{k}_{\mathsf{COM}})$
$cm^* \leftarrow\!\!\$\ \mathsf{COM.Commit}(pp_{\mathsf{COM}}, \mathsf{k}_{\mathsf{COM}}, m_\beta)$	**if** $\mathsf{COM.Commit}(pp_{\mathsf{COM}}, \mathsf{k}_{\mathsf{COM}}, m_0; r_0) = \mathsf{COM.Commit}(pp_{\mathsf{COM}}, \mathsf{k}_{\mathsf{COM}}, m_1; r_1)$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2(cm^*, st)$	$\wedge\ m_0 \neq m_1$ **return** 1
return β'	**else return** 0

Figure 2.5: The hiding experiment $\mathsf{Exp}^{\mathsf{hiding},\beta}_{\mathsf{COM},\mathcal{A}}(\lambda)$ and the binding experiment $\mathsf{Exp}^{\mathsf{binding}}_{\mathsf{COM},\mathcal{A}}(\lambda)$ for commitment scheme COM.

We say that a commitment scheme is *perfectly* hiding or binding if the advantage in the above experiments is defined to be 0, rather than negligible.

Pedersen commitments. We define Pedersen commitments [107], which we use in our report and trace ring signature instantiation in Chapter 6, in Figure 2.6.

Pedersen commitments are perfectly hiding and computationally binding if the discrete logarithm assumption holds with respect to the cyclic group generated by algorithm $\mathsf{COM.Setup}$. Additionally, Pedersen commitments are homomorphic, meaning that for any $m_0, m_1, r_0, r_1 \in \mathbb{Z}_p$ and pp_{COM} and $\mathsf{k}_{\mathsf{COM}}$ output by algorithms $\mathsf{COM.Setup}$ and $\mathsf{COM.KGen}$

$\mathsf{COM.Setup}(1^\lambda)$	$\mathsf{COM.KGen}(pp_{\mathsf{COM}})$	$\mathsf{COM.Commit}(pp_{\mathsf{COM}}, \mathsf{k}_{\mathsf{COM}}, m)$
$(\mathbb{G}, p, g) \leftarrow\!\!\$\ \mathcal{G}(1^\lambda)$	**parse** pp_{COM} as $(\mathbb{G}, p, g)$	**parse** pp_{COM} as $(\mathbb{G}, p, g)$
return $pp_{\mathsf{COM}} = (\mathbb{G}, p, g)$	$h \leftarrow\!\!\$\ \mathbb{G}$	$r \leftarrow\!\!\$\ \mathbb{Z}_p$
	return $\mathsf{k}_{\mathsf{COM}} = h$	**return** $cm := g^r {\mathsf{k}_{\mathsf{COM}}}^m$

Figure 2.6: The Pedersen commitment scheme [107].

respectively, it holds that

$$\mathsf{COM.Commit}(pp_{\mathsf{COM}}, \mathsf{k}_{\mathsf{COM}}, m_0; r_0) \cdot \mathsf{COM.Commit}(pp_{\mathsf{COM}}, \mathsf{k}_{\mathsf{COM}}, m_1; r_1) = \mathsf{COM.Commit}(pp_{\mathsf{COM}}, \mathsf{k}_{\mathsf{COM}}, m_0 + m_1; r_0 + r_1).$$

2.4.6 Signature Schemes

We define the syntax of a standard digital signature scheme and the existential unforgeability property here.

Definition 16 (Signature Scheme)**.** *A* signature scheme SIG *is a tuple of polynomial time algorithms* $(\mathsf{SIG.Setup}, \mathsf{SIG.KGen}, \mathsf{SIG.Sign}, \mathsf{SIG.Verify})$ *such that*

$\mathsf{SIG.Setup}(1^\lambda)$ On input of security parameter 1^λ, algorithm $\mathsf{SIG.Setup}$ outputs public parameters pp_{SIG}. We assume that pp_{SIG} defines the message space $\mathsf{M}_{\mathsf{SIG}}$ and the randomness space $\mathsf{Rand}_{\mathsf{SIG}}$.

$\mathsf{SIG.KGen}(pp_{\mathsf{SIG}})$ On input of public parameters pp_{SIG}, algorithm $\mathsf{SIG.KGen}$ outputs a key pair $(pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}})$ where pk_{SIG} is the public verification key and sk_{SIG} is the secret signing key.

$\mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, m)$ On input of public parameters pp_{SIG}, signing key sk_{SIG} and message $m \in \mathsf{M}_{\mathsf{SIG}}$, algorithm $\mathsf{SIG.Sign}$ outputs a signature σ.

$\mathsf{SIG.Verify}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, m, \sigma)$ On input of public parameters pp_{SIG}, verification key pk_{SIG}, message $m \in \mathsf{M}_{\mathsf{SIG}}$ and signature σ, algorithm $\mathsf{SIG.Verify}$ outputs 1 if σ verifies, and 0 otherwise.

Correctness. A signature scheme SIG should satisfy correctness.

Definition 17 (Correctness)**.** *A signature scheme* SIG *satisfies* correctness *if, for any message* $m \in \mathsf{M}_{\mathsf{SIG}}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} pp_{\mathsf{SIG}} \leftarrow\!\$\, \mathsf{SIG.Setup}(1^\lambda); \\ (pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}) \leftarrow\!\$\, \mathsf{SIG.KGen}(pp_{\mathsf{SIG}}); \\ \sigma \leftarrow\!\$\, \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, m) \end{array} : \mathsf{SIG.Verify}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, m, \sigma) \coloneqq 1\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Existential unforgeability against chosen message attacks. An attacker should be unable to generate signatures on behalf of signers such that algorithm SIG.Verify returns 1. This is captured in the existential unforgeability against a chosen message attack, or the EUF-CMA, property. In the EUF-CMA experiment, an adversary attempts to output a valid message/signature pair for an honest signer (i.e., a forgery) where the adversary can obtain signatures on behalf of the signer via an signing oracle. A signature scheme is EUF-CMA secure if the adversary succeeds in this task and does not query the output message to the signing oracle. A stronger version of this property exists [3], which requires that the tuple (m^*, σ^*) is not queried to the oracle. However, for our purposes, the weaker notion is sufficient.

Definition 18 (EUF-CMA [71])**.** *A signature scheme* SIG *satisfies* EUF-CMA *if, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} \mathcal{Q} \leftarrow \emptyset; \\ pp_{\mathsf{SIG}} \leftarrow\!\$\, \mathsf{SIG.Setup}(1^\lambda); \\ (pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}) \leftarrow\!\$\, \mathsf{SIG.KGen}(pp_{\mathsf{SIG}}); \\ (m^*, \sigma^*) \leftarrow\!\$\, \mathcal{A}^{\mathcal{O}\mathsf{sign}}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}) \end{array} : \mathsf{SIG.Verify}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, m^*, \sigma^*) \coloneqq 1 \wedge m^* \notin \mathcal{Q}\right] \leq \mathsf{negl}(\lambda)$$

where oracle $\mathcal{O}\mathsf{sign}_{(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}})}(m)$ *computes and outputs* $\sigma \leftarrow \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, m)$*, and updates set* $\mathcal{Q}$ *to include message* m*.*

2.4.7 Ring Signature Schemes

We define ring signatures, which allow signers to produce verifiable signatures. Furthermore, the signer is anonymous within a set of users, chosen by the signer, that is known as the ring.

Definition 19 (Ring signature)**.** *A* ring signature scheme RS *is a tuple of polynomial time algorithms* (RS.Setup, RS.KGen, RS.Sign, RS.Verify) *such that*

$\mathsf{RS.Setup}(1^\lambda)$ *On input of security parameter* 1^λ*, algorithm* RS.Setup *outputs public parameters* pp_{RS}*. We assume that* pp_{RS} *defines the message space* M_{RS} *and the randomness space* $\mathsf{Rand}_{\mathsf{RS}}$*.*

$\mathsf{RS.KGen}(pp_{\mathsf{RS}})$ *On input of public parameters* pp_{RS}*, algorithm* RS.KGen *outputs a key pair* $(pk_{\mathsf{RS}}, sk_{\mathsf{RS}})$ *where* pk_{RS} *is the public verification key and* sk_{RS} *is the secret signing key.*

$\mathsf{RS.Sign}(pp_{\mathsf{RS}}, sk_{\mathsf{RS}}, m, R)$ *On input of public parameters* pp_{RS}*, secret signing key* sk_{RS}*, message* $m \in \mathsf{M}_{\mathsf{RS}}$ *and ring* R*, algorithm* RS.Sign *outputs a signature* σ*.*

$\mathsf{RS.Verify}(pp_{\mathsf{RS}}, m, R, \sigma)$ *On input public parameters* pp_{RS}*, message* $m \in \mathsf{M}_{\mathsf{RS}}$*, ring* R *and signature* σ*, algorithm* RS.Verify *outputs 1 if* σ *is a valid signature on* m *with respect to* R*, and 0 otherwise.*

Correctness. We define correctness for a ring signature scheme in an intuitive way.

Definition 20 (Correctness)**.** *A ring signature* RS *satisfies* correctness *if, for any* $n = \mathsf{poly}(\lambda)$*,* $j \in [n]$ *and message* $m \in \mathsf{M}_{\mathsf{RS}}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} pp_{\mathsf{RS}} \leftarrow\!\$\ \mathsf{RS.Setup}(1^\lambda); \\ \textit{for } i = 1,\ldots,n \ : \ (pk_{\mathsf{RS}_i}, sk_{\mathsf{RS}_i}) \leftarrow\!\$\ \mathsf{RS.KGen}(pp_{\mathsf{RS}}); \\ R \leftarrow \{pk_{\mathsf{RS}_1}, \ldots, pk_{\mathsf{RS}_n}\}; \\ \sigma \leftarrow\!\$\ \mathsf{RS.Sign}(pp_{\mathsf{RS}}, sk_{\mathsf{RS}_j}, m, R) \end{array} : \ \mathsf{RS.Verify}(pp_{\mathsf{RS}}, m, R, \sigma) \coloneqq 1\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Anonymity. Ring signatures aim to 'hide' the identity of a signer within a ring and, as such, anonymity is a basic requirement of any ring signature scheme. In [19], Bender, Katz and Morselli define three anonymity experiments for ring signature schemes. These experiments require that an adversary, when provided with a challenge signature, cannot determine which of two potential signers generated the signature. In the experiments the adversary chooses the two potential signers, and is given access to a signing oracle. The three experiments differ with respect to the corruption strategy of the adversary. In the first, called *basic anonymity*, the adversary cannot corrupt or control any signers. The second experiment, *anonymity against adversarially chosen keys*, requires that the two signers outputs by the adversary are honest; all other signers may be corrupt or controlled by the adversary. Finally, *anonymity against full key exposure* allows the adversary to corrupt *all* signers. The notions of anonymity presented in [19] have since been extended to the setting of accountable ring signatures [31]. We define anonymity with respect to

adversarially generated keys, which is required for the security of our report and trace ring signature scheme that we introduce in Chapter 6.

Definition 21 (Anonymity [19])**.** *A ring signature scheme* RS *is* anonymous with respect to adversarially generated keys *if, for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* negl *such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{RSanon},0}_{\mathsf{RS},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{RSanon},1}_{\mathsf{RS},\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{RSanon},\beta}_{\mathsf{RS},\mathcal{A}}(\lambda)$ *is the experiment defined in Figure 2.7 for* $\beta \in \{0,1\}$*.*

$\underline{\mathsf{Exp}^{\mathsf{RSanon},\beta}_{\mathsf{RS},\mathcal{A}}(\lambda)}$

$pp_{\mathsf{RS}} \leftarrow\!\!\$\ \mathsf{RS.Setup}(1^\lambda)$
$\mathcal{Q}\mathsf{reg} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{corr} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset$
$(m^*, R^*, pk_{\mathsf{RS}_0}, pk_{\mathsf{RS}_1}, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign}}(pp_{\mathsf{RS}})$
$\sigma^* \leftarrow\!\!\$\ \mathsf{RS.Sign}(pp_{\mathsf{RS}}, sk_{\mathsf{RS}_\beta}, m^*, R^* \cup \{pk_{\mathsf{RS}_0}, pk_{\mathsf{RS}_1}\})$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign}}(\sigma^*, st)$
if $\{(pk_{\mathsf{RS}_0}, \cdot), (pk_{\mathsf{RS}_1}, \cdot)\} \subseteq \mathcal{Q}\mathsf{reg} \backslash \mathcal{Q}\mathsf{corr}$
  return β'

$\underline{\mathcal{O}\mathsf{reg}_{(pp_{\mathsf{RS}},\mathcal{Q}\mathsf{reg})}()}$

$(pk_{\mathsf{RS}}, sk_{\mathsf{RS}}) \leftarrow\!\!\$\ \mathsf{RS.KGen}(pp_{\mathsf{RS}})$
$\mathcal{Q}\mathsf{reg} \leftarrow \mathcal{Q}\mathsf{reg} \cup \{(pk_{\mathsf{RS}}, sk_{\mathsf{RS}})\}$
return pk_{RS}

$\underline{\mathcal{O}\mathsf{corrupt}_{(\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corr})}(pk_{\mathsf{RS}})}$

if $(pk_{\mathsf{RS}}, \cdot) \notin \mathcal{Q}\mathsf{reg}$ **return** $\perp$
$\mathcal{Q}\mathsf{corr} \leftarrow \mathcal{Q}\mathsf{corr} \cup \{(pk_{\mathsf{RS}}, sk_{\mathsf{RS}})\}$
return sk_{RS}

$\underline{\mathcal{O}\mathsf{sign}_{(pp_{\mathsf{RS}},\mathcal{Q}\mathsf{reg})}(pk_{\mathsf{RS}}, m, R)}$

$\sigma \leftarrow\!\!\$\ \mathsf{RS.Sign}(pp_{\mathsf{RS}}, sk_{\mathsf{RS}}, m, R \cup \{pk_{\mathsf{RS}}\})$
$\mathcal{Q}\mathsf{sign} \leftarrow \mathcal{Q}\mathsf{sign} \cup \{(pk_{\mathsf{RS}}, m, R, \sigma)\}$
return σ

Figure 2.7: The anonymity experiment $\mathsf{Exp}^{\mathsf{RSanon},\beta}_{\mathsf{RS},\mathcal{A}}(\lambda)$ for ring signature scheme RS.

Unforgeability. As for standard digital signatures, ring signatures must satisfy existential unforgeability against a chosen-message attack (EUF-CMA). This means that an attacker, for a message and ring of their choosing, cannot output a valid signature (i.e., a forgery) on behalf of an honest signer. This is defined in [19] analogously to EUF-CMA security for a standard digital signatures, but additionally requires that the signature output by the adversary must be generated for a ring of honest signers. That is, the adversary can corrupt signers, but the ring must not contain any corrupt signers. Otherwise, even if the ring contains only a single corrupt signer, the adversary can trivially output a valid signature on behalf of the ring.

Definition 22 (Unforgeability [19])**.** *A ring signature scheme* RS *is* unforgeable *if, for*

any PPT adversary $\mathcal{A}$, there exists a negligible function negl *such that*

$$\Pr\left[\begin{array}{l} pp_{\mathsf{RS}} \leftarrow\!\$\, \mathsf{Setup}(1^\lambda); \\ \mathcal{Q}\mathsf{reg}\leftarrow\emptyset;\ \mathcal{Q}\mathsf{corr}\leftarrow\emptyset;\ \mathcal{Q}\mathsf{sign}\leftarrow\emptyset \\ (m^*,R^*,\sigma^*) \leftarrow\!\$\, \mathcal{A}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign}}(pp_{\mathsf{RS}}) \end{array} : \begin{array}{l} \mathsf{RS.Verify}(pp_{\mathsf{RS}},m^*,R^*,\sigma^*)=1 \\ \wedge\ (\cdot,m^*,R^*,\sigma^*)\notin \mathcal{Q}\mathsf{sign} \\ \wedge\ \left\{\{(pk_{\mathsf{RS}_1},\cdot),\ldots,(pk_{\mathsf{RS}_{|R|}},\cdot)\}\subseteq \mathcal{Q}\mathsf{reg}\backslash\mathcal{Q}\mathsf{corr}\right. \\ \left.\text{for } R=\{pk_{\mathsf{RS}_1},\ldots,pk_{\mathsf{RS}_{|R|}}\}\right\} \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where oracles $\mathcal{O}\mathsf{reg}$, $\mathcal{O}\mathsf{corrupt}$ and $\mathcal{O}\mathsf{sign}$ are defined in Figure 2.7.

2.5 Proof Systems

In this section, we recall several proof systems that are used in this thesis. A proof system is a protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. The prover attempts to prove that a statement s is contained in some language $\mathcal{L}$. To do this, the prover demonstrates that they know a witness w such that the tuple (s, w) is contained in some binary relation $\mathcal{R}$. That is, $s \in \mathcal{L}$ if and only if there exists a witness such that $(s, w) \in \mathcal{R}$.

2.5.1 Sigma Protocols

In this thesis, we use Σ-protocols, which are proof systems with a three-move structure. Specifically, the prover outputs an initial message that is input to the verifier. The verifier then generates a challenge and the prover produces a response. We formally define a Σ-protocol in Definition 23.

Definition 23 (Σ-protocol)**.** *A Σ-protocol is a tuple of PPT algorithms $(\mathcal{P}, \mathcal{V})$ defined relative to a binary relation $\mathcal{R}$ that defines a language $\mathcal{L}_\mathcal{R} := \{s \mid \exists w : (s, w) \in \mathcal{R}\}$ such that*

$\mathcal{P}(s, w)$ On input of statement s and witness w, prover $\mathcal{P}$ outputs an initial message a.

$\mathcal{V}(a)$ On input of initial message a, verifier $\mathcal{V}$ outputs a challenge x.

$\mathcal{P}(x)$ On input of challenge x, prover $\mathcal{P}$ outputs a response z.

Following this three-move protocol, the verifier, on input of a transcript (s, a, x, z), can verify that there exists a witness w such that $(s, w) \in \mathcal{R}$. We define *completeness*, *n-special*

soundness and *special honest verifier zero-knowledge* for a Σ-protocol. These security properties are drawn from [72].

Completeness. A Σ-protocol must be *complete*, meaning that, if a prover knows a witness for a statement that is contained in a language, they should be able to convince the verifier of this fact. We formally define completeness as the property that, if a prover $\mathcal{P}$ that knows a witness w for a statement s such that $(s, w) \in \mathcal{R}$, $\mathcal{P}$ can always generate a transcript that verifies.

Definition 24 (Completeness [72])**.** *A Σ-protocol* $(\mathcal{P}, \mathcal{V})$ *is* complete *if, for any* $(s, w) \in \mathcal{R}$*, there exists a negligible function* negl *such that*

$$\Pr\left[a \leftarrow\!\!\$\, \mathcal{P}(s, w);\ x \leftarrow\!\!\$\, \{0,1\}^{\lambda};\ z \leftarrow\!\!\$\, \mathcal{P}(x)\ :\ \mathcal{V}(s, a, x, z)\ :=\ 1\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

n-special soundness. A Σ-protocol should satisfy a notion of *soundness*. This requires that, if a prover does not know a witness for a statement, they should be unable to output a transcript that convinces a verifier. In this thesis, we require n-special soundness, the property that, if an adversary outputs n transcripts for the same initial message a and each transcript verifies, then an extractor can output a witness with overwhelming probability.

Definition 25 (n-special soundness [72])**.** *A Σ-protocol* $(\mathcal{P}, \mathcal{V})$ *satisfies* n-special soundness *if there exists an algorithm* $\mathsf{Extract}$ *such that*

$\mathsf{Extract}(s, a, x_1, z_1, \ldots, x_n, z_n)$ *On input of statement* s*, initial message* a*, challenges* $x_1, \ldots, x_n$*, and responses* $z_1, \ldots, z_n$*, algorithm* $\mathsf{Extract}$ *outputs a witness* w*.*

Then, for all PPT adversaries $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{matrix}(s,a,x_1,z_1,\ldots,x_n,z_n) \leftarrow\!\!\$\, \mathcal{A}(1^{\lambda}); \\ w \leftarrow\!\!\$\, \mathsf{Extract}(s,a,x_1,z_1,\ldots,x_n,z_n)\end{matrix}\ :\ \begin{matrix}\left(\textbf{for } i = 1,\ldots,n\ :\ \mathcal{V}(s,a,x_i,z_i) := 1\right) \\ \wedge\ (s,w) \in \mathcal{R}\end{matrix}\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Special honest verifier zero-knowledge. A Σ-protocol should not reveal anything about the witness, other than what can be deduced from the transcript. We define special honest verifier zero-knowledge, which requires that an adversary cannot distinguish between a transcript generated by a prover and a simulated transcript.

Definition 26 (Special honest verifier zero-knowledge [72])**.** *A Σ-protocol $(\mathcal{P}, \mathcal{V})$ satisfies* special honest verifier zero-knowledge *if there exists a PPT algorithm* Sim *such that*

$\mathsf{Sim}(s, x)$ *On input of statement s and challenge x, algorithm* Sim *outputs an initial message a and response z.*

Then, for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl *such that,*

$$\left|\Pr\left[\begin{smallmatrix}(s,w,x) \leftarrow\$ \mathcal{A}(1^\lambda);\\ a \leftarrow\$ \mathcal{P}(s,w);\\ z \leftarrow\$ \mathcal{P}(x)\end{smallmatrix} \;:\; \mathcal{A}(a,z)=1\right] - \Pr\left[\begin{smallmatrix}(s,w,x) \leftarrow\$ \mathcal{A}(1^\lambda);\\ (a,z) \leftarrow\$ \mathsf{Sim}(s,x)\end{smallmatrix} \;:\; \mathcal{A}(a,z)=1\right]\right| \leq \mathsf{negl}(\lambda)\,.$$

Quasi-unique responses. In this thesis, we require the property that a Σ-protocol has *quasi-unique responses*. This means that, for a transcript (s, a, x, z) such that $\mathcal{V}(s, a, x, z) = 1$, it is difficult to find some $z' \neq z$ such that $\mathcal{V}(s, a, x, z') = 1$.

2.5.2 Non-Interactive Zero-Knowledge Proofs

Non-interactive zero-knowledge (NIZK) proof systems are proof systems for which the prover does not need to interact with the verifier to create a transcript. That is, the prover can generate a transcript without receiving a challenge from the verifier, and the transcript can be verified by the verifier. NIZK proof systems can be constructed from Σ-protocols by applying the Fiat-Shamir transform [64], a heuristic that replaces the challenge generated by the verifier with the output of a random oracle that takes as input the transcript of the protocol. In practice, the random oracle is instantiated by applying a hash function to the protocol transcript. We define a NIZK proof system NIZK in Definition 27.

Definition 27 (NIZK proof system)**.** *A* NIZK proof system NIZK *is a tuple of polynomial time algorithms* $(\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ *defined relative to a binary relation $\mathcal{R}$ that defines a language $\mathcal{L}_\mathcal{R} := \{s \mid \exists w : (s, w) \in \mathcal{R}\}$ such that*

$\mathsf{NIZK.Setup}(1^\lambda)$ On input of security parameter 1^λ, algorithm $\mathsf{NIZK.Setup}$ outputs public parameters pp_{NIZK}.

$\mathsf{NIZK.Prove}(pp_{\mathsf{NIZK}}, s, w)$ On input of public parameters pp_{NIZK}, statement s and witness w, algorithm $\mathsf{NIZK.Prove}$ outputs a proof ρ that $(s, w) \in \mathcal{R}$.

$\mathsf{NIZK.Verify}(pp_{\mathsf{NIZK}}, s, \rho)$ On input of public parameters pp_{NIZK}, statement s and proof ρ, algorithm NIZK.Verify outputs 1 if the proof verifies and 0 otherwise.

We define *completeness*, *soundness*, *knowledge extractability* and *zero-knowledge* for a NIZK proof system. The definitions below are drawn from [73] and are intuitively similar to the security properties of a Σ-protocol.

Completeness. A NIZK proof system is *complete* if a prover can always output a verifiable proof for $(s, w) \in \mathcal{R}$.

Definition 28 (Completeness [73])**.** *A NIZK proof system* NIZK *is* complete *if, for any* $(s, w) \in \mathcal{R}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{matrix} pp_{\mathsf{NIZK}} \leftarrow\!\$\, \mathsf{NIZK.Setup}(1^\lambda); \\ \rho \leftarrow\!\$\, \mathsf{NIZK.Prove}(pp_{\mathsf{NIZK}}, s, w) \end{matrix} \;:\; \mathsf{NIZK.Verify}(pp_{\mathsf{NIZK}}, s, \rho) \coloneqq 1\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Soundness. If a prover does not know a witness, the prover should be unable to output a verifiable proof. This intuition is captured by the *soundness* property, which requires that an adversary cannot output a verifiable proof for a statement s that is not in the language $\mathcal{L}_{\mathcal{R}}$.

Definition 29 (Soundness [73])**.** *A NIZK proof system* NIZK *satisfies* soundness *if, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[pp_{\mathsf{NIZK}} \leftarrow\!\$\, \mathsf{NIZK.Setup}(1^\lambda);\ (s, \rho) \leftarrow\!\$\, \mathcal{A}(pp_{\mathsf{NIZK}}) \;:\; s \notin \mathcal{L}_{\mathcal{R}} \wedge \mathsf{NIZK.Verify}(pp_{\mathsf{NIZK}}, s, \rho) \coloneqq 1\right] \leq \mathsf{negl}(\lambda)\,.$$

Knowledge extractability. Knowledge extractability is stronger than soundness [73], requiring that an adversary cannot output a verifiable proof unless there exists a witness w that can be extracted by an algorithm NIZK.Extract that we define below.

Definition 30 (Knowledge Extractability [73])**.** *A NIZK proof system* NIZK *satisfies* knowledge extractability *if there exists a polynomial time algorithms* NIZK.SimSetup *and* NIZK.Extract *such that*

$\mathsf{NIZK.SimSetup}(1^\lambda)$ *On input of security parameter* 1^λ*, algorithm* NIZK.SimSetup *outputs public parameters* pp_{NIZK} *and trapdoor* τ*.*

$\mathsf{NIZK.Extract}(pp_{\mathsf{NIZK}}, \tau, s, \rho)$ *On input of public parameters* pp_{NIZK}*, trapdoor* τ*, statement* s *and proof* ρ*, algorithm* NIZK.Extract *outputs a witness* w*.*

Then, for all PPT adversaries $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{matrix}(pp_{\mathsf{NIZK}},\tau) \leftarrow\!\$\ \mathsf{NIZK.SimSetup}(1^\lambda) \\ (s,\rho) \leftarrow\!\$\ \mathcal{A}(pp_{\mathsf{NIZK}}) \\ w \leftarrow\!\$\ \mathsf{NIZK.Extract}(pp_{\mathsf{NIZK}},\tau,s,\rho)\end{matrix} : \ (s,w) \in \mathcal{R} \ \vee \ \mathsf{NIZK.Verify}(pp_{\mathsf{NIZK}},s,\rho) \coloneqq 0\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

In this thesis, we assume that the extractor NIZK.Extract is *straightline*. Recall that a NIZK proof system can be constructed from a Σ-protocol and the Fiat-Shamir transform. In the event that the Σ-protocol has quasi-unique responses, a (non-straightline) extractor must rewind the prover to output witnesses for each transcript submitted by the prover. In this event, it is difficult to determine how many times the extractor must rewind. A solution to this is to make the extractor straightline, which means that the extractor only sees a single transcript from the prover. We refer the reader to [26] for a discussion on different extractors. We also note that we assume the extractor for a signature of knowledge (which we define below) is also straightline.

Zero-knowledge. A NIZK proof should reveal nothing about the witness. This is captured formally in an experiment in which an adversary can obtain proofs from a proving oracle that takes as input statement/witness pairs. A NIZK proof is zero-knowledge if the adversary cannot distinguish, with more than negligible probability, real proofs (i.e., generated by running algorithm NIZK.Prove with input of a witness) and simulated proofs that are generated without input of a witness.

Definition 31 (Zero-knowledge)**.** *A NIZK proof system* NIZK *satisfies* zero-knowledge *if there exists PPT algorithm* NIZK.SimSetup *(as defined for knowledge extractability) and* NIZK.SimProve *such that*

$\mathsf{NIZK.SimProve}(pp_{\mathsf{NIZK}}, \tau, s)$ *On input of public parameters* pp_{NIZK}*, trapdoor* τ *and statement* s*, algorithm* NIZK.SimProve *outputs a proof* ρ*.*

Then, for all PPT adversaries $\mathcal{A}$*, there exists a negligible function* negl *such that,*

$$\left|\Pr\left[\begin{matrix}pp_{\mathsf{NIZK}} \leftarrow\!\$\ \mathsf{NIZK.Setup}(1^\lambda): \\ \mathcal{A}^{\mathcal{O}\mathsf{NIZK.Prove}}(pp_{\mathsf{NIZK}}) = 1\end{matrix}\right] - \Pr\left[\begin{matrix}(pp_{\mathsf{NIZK}},\tau) \leftarrow\!\$\ \mathsf{NIZK.SimSetup}(1^\lambda): \\ \mathcal{A}^{\mathcal{O}\mathsf{NIZK.SimProve}}(pp_{\mathsf{NIZK}}) = 1\end{matrix}\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathcal{O}\mathsf{NIZK.Prove}_{(pp_{\mathsf{NIZK}})}(s, w)$ *returns* $\mathsf{NIZK.Prove}(pp_{\mathsf{NIZK}}, s, w)$ *and* $\mathcal{O}\mathsf{NIZK.SimProve}_{(pp_{\mathsf{NIZK}},\tau)}(s, w)$ *returns* $\mathsf{NIZK.SimProve}(pp_{\mathsf{NIZK}}, \tau, s)$ *on input of query* $(s, w) \in \mathcal{R}$.

2.5.3 Signatures of Knowledge

A signature of knowledge [39] transforms a zero-knowledge proof into a signature. That is to say, a signature of knowledge is a statement that the signer who knows a witness for some statement $s \in \mathcal{L}$ has signed a message m. As with NIZK proofs, signatures of knowledge can be constructed by applying the Fiat-Shamir transform to a Σ-protocol. Our syntax and security definitions for a signature of knowledge are drawn from [39], with the exception of our extractability definition, which is drawn from [31].

Definition 32 (Signature of Knowledge)**.** *A* signature of knowledge SOK *is a tuple of polynomial time algorithms* $(\mathsf{SoK.Setup}, \mathsf{SoK.Sign}, \mathsf{SoK.Verify})$ *defined relative to a binary relation* $\mathcal{R}$ *that defines a language* $\mathcal{L}_{\mathcal{R}} := \{s \mid \exists w : (s, w) \in \mathcal{R}\}$ *such that*

$\mathsf{SoK.Setup}(1^\lambda)$ On input of security parameter 1^λ, algorithm $\mathsf{SoK.Setup}$ outputs public parameters pp_{SOK}.

$\mathsf{SoK.Sign}(pp_{\mathsf{SOK}}, s, w, m)$ On input of public parameters pp_{SOK}, statement s, witness w and message m, algorithm $\mathsf{SoK.Sign}$ outputs a signature σ if $(s, w) \in \mathcal{R}$.

$\mathsf{SoK.Verify}(pp_{\mathsf{SOK}}, s, m, \sigma)$ On input of public parameters pp_{SOK}, statement s, message m and signature σ, algorithm $\mathsf{SoK.Verify}$ outputs 1 if the signature verifies and 0 otherwise.

Additionally, the security experiments for a signature of knowledge rely on algorithms $\mathsf{SoK.SimSetup}$ and $\mathsf{SoK.SimSign}$ that are defined as follows.

$\mathsf{SoK.SimSetup}(1^\lambda)$ On input of security parameter 1^λ, algorithm $\mathsf{SoK.SimSetup}$ outputs public parameters pp_{SOK} and trapdoor τ.

$\mathsf{SoK.SimSign}(pp_{\mathsf{SOK}}, \tau, s, m)$ On input of public parameters pp_{SOK}, trapdoor τ, statement s and message m, algorithm $\mathsf{SoK.SimSign}$ outputs a signature σ.

Correctness. A signature of knowledge that is generated via algorithm SoK.Sign should verify. This is captured by the correctness property.

Definition 33 (Correctness [39])**.** *A signature of knowledge* SOK *satisfies* correctness *if, for any* $(s, w) \in \mathcal{R}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\begin{matrix} pp_{\mathsf{SOK}} \leftarrow\!\$\, \mathsf{SoK.Setup}(1^\lambda); \\ \sigma_{\mathsf{SOK}} \leftarrow\!\$\, \mathsf{SoK.Sign}(pp_{\mathsf{SOK}}, s, w, m) \end{matrix} \;:\; \mathsf{SoK.Verify}(pp_{\mathsf{SOK}}, s, m, \sigma_{\mathsf{SOK}}) \coloneqq 1\right] \geq 1 - \mathsf{negl}(\lambda)$$

Simulatability. Simulatability is similar in spirit to the zero-knowledge property of a NIZK proof system. That is, a signature of knowledge should reveal nothing about the witness used to construct it. As such, simulatability requires that an adversary cannot distinguish between real signatures output by algorithm SoK.Sign and simulated signatures output by algorithm SoK.SimSign.

Definition 34 (Simulatability [39])**.** *A signature of knowledge* SOK *satisfies* simulatability *if there exists algorithms* SoK.SimSetup *and* SoK.SimSign *and a negligible function* negl*, such that, for all PPT adversaries* $\mathcal{A}$*,*

$$\left|\Pr\left[\begin{matrix} pp_{\mathsf{SOK}} \leftarrow\!\$\, \mathsf{SoK.Setup}(1^\lambda): \\ \mathcal{A}^{\mathcal{O}\mathsf{SOKSign}}(pp_{\mathsf{SOK}}) = 1 \end{matrix}\right] - \Pr\left[\begin{matrix} (pp_{\mathsf{SOK}}, \tau) \leftarrow\!\$\, \mathsf{SoK.SimSetup}(1^\lambda): \\ \mathcal{A}^{\mathcal{O}\mathsf{SimSign}}(pp_{\mathsf{SOK}}) = 1 \end{matrix}\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathcal{O}\mathsf{SOKSign}_{(pp_{\mathsf{SOK}})}(s, w, m)$ *returns* $\mathsf{SoK.Sign}(pp_{\mathsf{SOK}}, s, w, m)$ *and* $\mathcal{O}\mathsf{SimSign}_{(pp_{\mathsf{SOK}}, \tau)}(s, w, m)$ *returns* $\mathsf{SoK.SimSign}(pp_{\mathsf{SOK}}, \tau, s, m)$ *on input of query* $(s, w) \in \mathcal{R}$ *and message* m*.*

Extractability. Extractability is similar to the soundness property of a NIZK proof system, requiring that a signer cannot output a verifiable signature without knowledge of a witness. To this end, extractability requires that, if an adversary can output a signature of knowledge for a statement s and message m that verifies, then it must be the case that there exists a witness w such that $(s, w) \in \mathcal{R}$.

Definition 35 (Extractability [31])**.** *A signature of knowledge* SOK *satisfies* extractability *if there exists a PPT algorithm* Extract *such that*

$\mathsf{SOK.Extract}(pp_{\mathsf{SOK}}, \tau, s, m, \sigma)$ *On input of public parameters* pp_{SOK}*, trapdoor* τ*, statement* s*, message* m *and signature* σ*, algorithm* SOK.Extract *outputs a witness* w*.*

Then, for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl *such that*

$$\Pr\left[\mathsf{Exp}^{\mathsf{Ext}}_{\mathsf{SOK},\mathcal{A},\mathsf{Extract}}(\lambda) = 1\right] \geq 1 - \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{Ext}}_{\mathsf{SOK},\mathcal{A},\mathsf{Extract}}(\lambda)$ *is the experiment defined in Figure 2.8.*

$\mathsf{Exp}^{\mathsf{Ext}}_{\mathsf{SOK},\mathcal{A},\mathsf{Extract}}(\lambda)$	$\mathcal{O}_{(pp_{\mathsf{SOK}},\tau,\mathcal{Q})}(s,w,m)$
$\mathcal{Q} \leftarrow \emptyset$	$\sigma \leftarrow\!\!\$\ \mathsf{SOK.SimSign}(pp,\tau,s,m)$
$(pp_{\mathsf{SOK}},\tau) \leftarrow\!\!\$\ \mathsf{SOK.SimSetup}(1^\lambda)$	$\mathcal{Q} \leftarrow \mathcal{Q} \cup (s,m,\sigma)$
$(s^*,m^*,\sigma^*) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}}(pp_{\mathsf{SOK}})$	**return** σ
$y^* \leftarrow\!\!\$\ \mathsf{SOK.Extract}(pp_{\mathsf{SOK}},\tau,s^*,m^*,\sigma^*)$	
if $(\exists\ w$ s.t. $(s^*,w) \in \mathcal{R}\ \wedge\ y^* = f(w)) \vee (s^*,m^*,\sigma^*) \in \mathcal{Q}$	
$\vee\ \mathsf{SoK.Verify}(pp_{\mathsf{SOK}},s^*,m^*,\sigma^*) = 0$ **return** 1	
else return 0	

Figure 2.8: The extractability experiment $\mathsf{Exp}^{\mathsf{Ext}}_{\mathsf{SOK},\mathcal{A},\mathsf{Extract}}(\lambda)$ for signature of knowledge SOK.

2.6 Threshold Cryptography

In this thesis, we utilise some threshold cryptography techniques, which requires that an operation is performed by at least t of n total protocol participants in order to output a result. In particular, we use threshold public-key encryption to construct an e-voting scheme in Chapter 3. Indeed, threshold encryption can be used to facilitate distribution of the election tallier role, ensuring that no individual tallier can decrypt ballots and henceforth break the ballot secrecy property. Moreover, we use publicly verifiable secret sharing to construct a report and trace ring signature scheme for multiple reporters in Chapter 6. Here, we recall the syntax and required security properties for these primitives.

2.6.1 Threshold Public-Key Encryption

Threshold public-key encryption is defined analogously to standard public-key encryption. We extend definitions of public-key encryption to the setting where a threshold of protocol participants are required to decrypt a ciphertext.

Definition 36 (Threshold PKE scheme)**.** *A* (t,n)-threshold public-key encryption scheme THR *is a tuple of polynomial time algorithms* (THR.Setup, THR.KGen, THR.Enc, THR.Dec,

$\mathsf{THR.Combine}$) *such that*

$\mathsf{THR.Setup}(1^\lambda)$ On input of security parameter 1^λ, algorithm $\mathsf{THR.Setup}$ outputs public parameters pp_{THR}. We assume that pp_{THR} defines the message space $\mathsf{M}_{\mathsf{THR}}$ and the randomness space $\mathsf{Rand}_{\mathsf{THR}}$.

$\mathsf{THR.KGen}(pp_{\mathsf{THR}}, t, n)$ On input of public parameters pp_{THR}, threshold t and number of protocol participants n, algorithm $\mathsf{THR.KGen}$ outputs a tuple $(pk_{\mathsf{THR}}, (sk_{\mathsf{THR},1}, \dots sk_{\mathsf{THR},n}))$ where pk_{THR} is the public encryption key and $(sk_{\mathsf{THR},1}, \dots sk_{\mathsf{THR},n})$ is the set of n secret decryption keys.

$\mathsf{THR.Enc}(pp_{\mathsf{THR}}, pk_{\mathsf{THR}}, m)$ On input of public parameters pp_{THR}, public encryption key pk_{THR} and message $m \in \mathsf{M}_{\mathsf{THR}}$, algorithm $\mathsf{THR.Enc}$ outputs a ciphertext c.

$\mathsf{THR.Dec}(pp_{\mathsf{THR}}, i, sk_{\mathsf{THR},i}, c)$ On input of public parameters pp_{THR}, an index $i \in \{1, \dots, n\}$, secret decryption key $sk_{\mathsf{THR},i}$ and ciphertext c, algorithm $\mathsf{THR.Dec}$ outputs a decryption share c'_i.

$\mathsf{THR.Combine}(pp_{\mathsf{THR}}, c, (c'_1, \dots, c'_t))$ On input of public parameters pp_{THR}, ciphertext c and t decryption shares $c_1, \dots, c_t$, algorithm $\mathsf{THR.Combine}$ outputs a message m.

We adjust definitions of correctness and $\mathsf{IND\text{-}CPA}$ for a public-key encryption scheme to a (t, n)-threshold public-key encryption scheme in a straightforward way.

Definition 37 (Correctness)**.** *A (t, n)-threshold public-key encryption scheme* THR *satisfies* correctness *if, for any message $m \in \mathsf{M}_{\mathsf{THR}}$, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} pp_{\mathsf{THR}} \leftarrow\$ \mathsf{THR.Setup}(1^\lambda); \\ (pk_{\mathsf{THR}}, (sk_{\mathsf{THR},1}, \dots sk_{\mathsf{THR},n})) \leftarrow\$ \mathsf{THR.KGen}(pp_{\mathsf{THR}}); \\ c \leftarrow\$ \mathsf{THR.Enc}(pp_{\mathsf{THR}}, pk_{\mathsf{THR}}, m); \\ \{i_1, \dots, i_t\} \leftarrow\$ [n]; \\ \textbf{for } j = 1, \dots, t: \ c'_{i_j} \leftarrow \mathsf{THR.Dec}(pp_{\mathsf{THR}}, i_j, sk_{\mathsf{THR},i_j}, c) \end{array} : \mathsf{THR.Combine}(pp_{\mathsf{THR}}, c, (c'_{i_1}, \dots, c'_{i_t})) \coloneqq m \right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Definition 38 ($\mathsf{tIND\text{-}CPA}$)**.** *A (t, n)-threshold public key encryption scheme* THR *satisfies* $\mathsf{tIND\text{-}CPA}$ *if, for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* negl *such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{tIND\text{-}CPA},0}_{\mathsf{THR},\mathcal{A}}(\lambda, t, n) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{tIND\text{-}CPA},1}_{\mathsf{THR},\mathcal{A}}(\lambda, t, n) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{tIND\text{-}CPA},\beta}_{\mathsf{THR},\mathcal{A}}(\lambda, t, n)$ is the experiment defined in Figure 2.9 for $\beta \in \{0, 1\}$.

$\mathsf{Exp}^{\mathsf{tIND\text{-}CPA},\beta}_{\mathsf{THR},\mathcal{A}}(\lambda, t, n)$

$\mathsf{flag} \leftarrow 0$
$pp_{\mathsf{THR}} \leftarrow\!\!\$\ \mathsf{THR.Setup}(1^\lambda)$
$(pk_{\mathsf{THR}}, (sk_{\mathsf{THR},1}, \ldots sk_{\mathsf{THR},n})) \leftarrow\!\!\$\ \mathsf{THR.KGen}(pp_{\mathsf{THR}})$
$(st, \{i_1, \ldots, i_{t-1}\}) \leftarrow\!\!\$\ \mathcal{A}_1(pp_{\mathsf{THR}}, pk_{\mathsf{THR}})$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{enc}}(st, \{sk_{\mathsf{THR},i_1}, \ldots, sk_{\mathsf{THR},i_{t-1}}\})$
return β'

$\mathcal{O}\mathsf{enc}_{(\mathsf{flag},pp_{\mathsf{THR}},pk_{\mathsf{THR}})}(m_0, m_1)$

if $\mathsf{flag} = 1$ **return** $\perp$
$c^* \leftarrow\!\!\$\ \mathsf{THR.Enc}(pp_{\mathsf{THR}}, pk_{\mathsf{THR}}, m_\beta)$
$\mathsf{flag} \leftarrow 1$
return c^*

Figure 2.9: The tIND-CPA experiment $\mathsf{Exp}^{\mathsf{tIND\text{-}CPA},\beta}_{\mathsf{THR},\mathcal{A}}(\lambda, t, n)$ for threshold public-key encryption scheme THR.

2.6.2 Threshold Publicly Verifiable Secret Sharing

We recall the definition and preliminaries for a publicly verifiable secret sharing scheme PVSS here, where definitions are drawn from [117].

Definition 39 (Monotone access structure)**.** *Let $p = \{p_1, \ldots, p_n\}$ be a set of parties. We say that $\mathbb{A} \subseteq 2^{\{1,\ldots,n\}}$ is monotone if, for every B and C, if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure (resp., monotone access structure) is a collection (resp., montone collection) $\mathbb{A}$ of non-empty subsets of $2^{\{1,\ldots,n\}}$.*

Definition 40 ((t, n)-Threshold PVSS)**.** *A (t, n)-threshold publicly verifiable secret sharing scheme ((t, n)-PVSS) is a tuple of polynomial time algorithms* (SS.Gen, SS.Verify, SS.Combine) *relative to a monotone access structure $\mathbb{A} = \{A \in 2^{\{1,\ldots,n\}} : |A| \geq t\}$ such that any coalition of at least t participants can recover the secret, with algorithms defined as follows.*

$\mathsf{SS.Gen}(s)$ On input of a secret s, algorithm SS.Gen outputs secret shares $\{s_i : i \in \mathbb{A}\}$, public shares $\{S_i : i \in \mathbb{A}\}$ and a public version of the secret S.

$\mathsf{SS.Verify}(S, \{S_i : i \in \mathbb{A}\})$ On input of public secret S and public shares $\{S_i : i \in \mathbb{A}\}$, algorithm SS.Verify outputs 1 if all secret shares verify, and outputs 0 otherwise.

$\mathsf{SS.Combine}(\{s_i : i \in \mathbb{A}\})$ On input of secret shares $|\{s_i : i \in \mathbb{A}\}$, such that $\{s_i : i \in \mathbb{A}\}| \geq t$, algorithm SS.Combine outputs the secret s.

2.7 Mixnets

We conclude this chapter by briefly recalling mixnets, which, within e-voting, are occasionally used to shuffle ballots submitted by voters before the election tally is computed. Mixnets sever the link between voter and vote and, therefore, help to preserve ballot secrecy. Mixnets are complex protocols and we refer the reader to [1] for a thorough review. In this thesis, we denote a mixnet as an algorithm $\mathsf{MixNet}(pk, L, \phi)$ that takes as input a public key pk (which is usually the election public key), a list of ciphertexts L and a permutation ϕ, and outputs a shuffled set of ciphertexts. Commonly, algorithm MixNet partially decrypts and re-encrypts the list of ciphertexts L, and then applies permutation ϕ to the re-encrypted ciphertexts.

Chapter 3

Ballot Secrecy for E-Voting

Contents

In this chapter, we present new definitions of ballot secrecy for e-voting schemes. First, we propose an intuitive definition in the honest model, that is, a model in which all election officials are honest. Then, we show that this definition can be easily extended to the malicious ballot box setting and a setting that allows for a distributed tallier. We demonstrate that our definitions of ballot secrecy are satisfiable, defining e-voting scheme constructions which we prove satisfy our definitions. The work in this chapter appears in [65], which is joint work with Elizabeth A. Quaglia.

3.1 Introduction

Voting, whether traditional paper-based or electronic, presents challenges. In particular, voters must be able to cast their vote in private and should be assured that the result of the

election is computed correctly[1]. In a paper-based election, voters are (usually)[2] required to vote in person, for example, by attending a polling station and casting a ballot from the privacy of a voting booth. Ballots are then hand-counted to produce the election result. The counting process is often subject to strict rules and regulations, guaranteeing that the final result is accurate. Therefore, there exist well-established procedures that ensure paper-based elections are capable of overcoming the challenges associated with voting.

Electronic voting, or e-voting, uses electronic aids at some stage in the voting process. In this thesis, we use the term e-voting to refer to *remote* e-voting that does not require paper at any point in the process. Under this definition, and in comparison to traditional paper-based voting, e-voting can, in theory, be accomplished anywhere in the world and can automate the process of tallying elections. However, the introduction of technology and remote voting brings about new challenges. Firstly, automating the task of tallying means that voters (and others) can no longer rely on the stringent rules and procedures of the tallying process to guarantee that the election result is correct. To address this, the e-voting community has developed definitions of *verifiability* for e-voting schemes and designed verifiable e-voting schemes. Specifically, verifiability is typically defined as individual verifiability (any voter can check that their ballot is counted), universal verifiability (anyone can check that the published tally is correct), and eligibility verifiability (only eligible voters voted). In doing so, e-voting has introduced new challenges; the e-voting community requires rigorous definitions of verifiability and the design of verifiable e-voting schemes. We refer the interested reader to [46, 120, 122] for a discussion on the subject of verifiability. Secondly, voters can no longer rely on the privacy afforded by the voting booth, which leads to a challenge in designing privacy-preserving systems and demonstrating that they are secure. A step towards overcoming this challenge is to provide rigorous privacy definitions for e-voting schemes. In this way, it is possible to make formal statements about whether or not a scheme satisfies a given definition and, hence, preserves voters' privacy.

In Chapter 1 we stated a hierarchy of privacy properties for e-voting that includes ballot secrecy, receipt-freeness and coercion-resistance, as defined in [51]. Here, we recall those properties and briefly expand on their intuitive definitions. In this chapter, we focus on ballot secrecy and consider the stronger notion of receipt-freeness in Chapter 4.

[1] This is *not* an exhaustive list of challenges associated with voting. The interested reader can consult material available from the Electoral Integrity Project (`https://www.electoralintegrityproject.com/`), a body that conducts research on the integrity of elections around the world.

[2] Of course, modern paper-based elections may offer alternative means of voting, for example, the ability to vote by proxy or cast a mail-in ballot(see, e.g., [126])

Ballot secrecy Throughout the election, a voter's vote remains secret, except when the election result reveals the vote, for example, in the event of a unanimous result.

Receipt-freeness A voter cannot prove their vote to anyone. Receipt-freeness strengthens ballot secrecy with additional protection against vote-buying, ensuring that potential attackers have no incentive to buy votes. That is to say, a voter cannot prove how they voted, and therefore cannot prove that their vote was truly 'bought'.

Coercion-resistance A voter can cast their vote as intended, even if they are under the control of an attacker for some time during the election. In particular, coercion-resistance strengthens receipt-freeness by additionally protecting against randomisation, abstention and simulation attacks [88].

Rigorous definitions of ballot secrecy have been proposed in the literature, and, most commonly, definitions describe an experiment as outlined in Chapter 2.2. Early definitions [16, 18] follow the well-established route of indistinguishability experiments (such as, for example, IND-CPA for public-key encryption [9], as defined in Definition 7). Several ballot secrecy definitions adopt this approach [18, 28, 29, 47, 91]. However, to address the fact that this approach limits the class of voting result functions that can be considered, a separate line of research departed from this approach, focusing instead on definitions that provide an adversary with a view of a 'real' election or a 'fake' election [22, 23, 27, 44, 48]. In particular, BPRIV [22], a highly-regarded definition of ballot secrecy, favours this approach. More generally, the majority of ballot secrecy definitions position themselves in the so-called *honest* model. That is, they consider all election officials to be trusted. With respect to formal definitions, the consideration of the malicious setting is a young area of research and has focused on a malicious ballot box [29, 47, 48]. In this chapter, we present a more detailed explanation of these approaches and present new definitions of ballot secrecy that build upon and extend existing work in this area.

3.1.1 Our Contributions

We revisit the approach taken in [18] and present new definitions of ballot secrecy. We choose this approach due to the well-established, intuitive nature of indistinguishability experiments. Moreover, though we recognise that this approach limits the class of result functions considered, definitions that adopt this approach, and our definitions specifically,

provide a number of advantages over existing definitions, which we discuss throughout this chapter. Here, we summarise the contributions of this chapter.

Defining ballot secrecy. We define BSec, a definition of ballot secrecy in the honest model. BSec builds upon [18], capturing several additional functionalities. First of all, BSec models e-voting schemes with a registration phase. That is, eligible voters are provided with a credential that is required to cast a ballot. Voter registration is not modelled in [18] or subsequent, related definitions [28, 29] and, hence, BSec is the first definition that follows this approach to model voter registration. Thus, BSec reflects how many advanced e-voting schemes are modelled, for example, Belenios [8, 45] and Civitas [42, 43]. Secondly, BSec allows for *adaptive* corruption of voters. Previous definitions that model registration of voters are limited to static corruption of voters only [44, 47] or allow for voters to cast only a single ballot [91]. Therefore, BSec improves upon existing definitions in the honest model by capturing voter registration and adaptive corruption of voters.

Our second definition, mbbBS, extends BSec to the malicious ballot box setting. mbbBS is similar to the definition presented in [29], which also adopts the approach of [18]. However, contrary to [29], and similarly to BSec, mbbBS models registration of voters. mbbBS captures static corruption of voters only, though we note that previous ballot secrecy definitions that model voter registration in the malicious ballot box setting [47, 48] are also limited in this respect.

We then define dtBS, which, to the best of our knowledge, is the first ballot secrecy definition that models a malicious ballot box and a distributed tallier. dtBS extends mbbBS to model an adversary that corrupts a subset of talliers. Like mbbBS, dtBS considers static corruption of voters.

Achieving our notions of ballot secrecy. We prove that all our definitions can be satisfied. Specifically, we define an e-voting scheme Γ_{mini} and prove it satisfies BSec and mbbBS. We then extend Γ_{mini} to a setting with a distributed tallier, in a construction that we call $\Gamma^{*}_{\mathsf{mini}}$, and prove that it satisfies dtBS.

Comparing our notions and existing literature. Throughout this chapter, we place our definitions in the context of the existing literature. It emerges that our definitions provide an advantage with respect to extendibility to the malicious setting, when compared to BPRIV. Moreover, our definitions improve upon existing definitions that follow the same approach, which we demonstrate throughout this chapter. More generally, given that the malicious setting has received significantly less attention than the honest model, we believe that our definitions in the malicious model are valuable and desirable. In particular, it is not always possible to trust election officials in practice, and the role of the tallier can be distributed, e.g., in the Helios e-voting scheme [77]. Therefore, it is desirable to prove security under realistic trust assumptions; the definitions presented in this chapter are a step towards this goal.

3.2 Chapter Preliminaries

In this section, we present a high-level overview of an e-voting scheme and introduce the syntax of an e-voting scheme.

3.2.1 Overview of an E-Voting Scheme

An e-voting scheme proceeds in four phases. We describe these phases below, but first we outline the election officials required for an e-voting scheme as follows.

- **Election administrator**: Computes and publishes public parameters for an election.
- **Registrar**: Registers eligible voters and maintains a list of registered voters, denoted $\mathcal{L}$. That is, the registrar provides eligible voters with a public and a secret credential and stores the public credential in $\mathcal{L}$. We note that not all e-voting schemes register voters. In particular, the Helios e-voting scheme [2, 77] requires that voters log in to the Helios system using credentials generated for an external system, for example, Facebook or Google. Other e-voting schemes such as Belenios [8, 45] and Civitas [42, 43], on the other hand, present voters with a credential pair, which is required to submit a ballot. We refer to these types of e-voting schemes as schemes with external and internal credentials, respectively.

- **Ballot box manager**: Collects ballots from voters and stores them in a ballot box bb prior to tallying. We assume that the contents of the ballot box are private but that the ballot box manager may additionally provide a public view of the ballot box, known as the bulletin board, denoted $\mathcal{BB}$.
- **Tallier**: Computes and publishes the result of the election with a proof of correct tallying that anyone can verify.

With these entities, an election proceeds in four phases, as illustrated in Figure 3.1. Firstly, in the *setup* phase, an election administrator performs the setup operations for the election. The registrar then registers voters during the (optional) *registration* phase of the election. The scheme then proceeds to the *voting* phase, wherein voters cast ballots (that are linked to their public credential in the event of internal credentials) by submitting a message to the ballot box bb. In this chapter, we assume that the e-voting scheme is non-interactive, meaning that voters need only submit a single message to the ballot box to cast their ballot. Finally, in the *tally* phase, the tallier computes and publishes the result of the election.

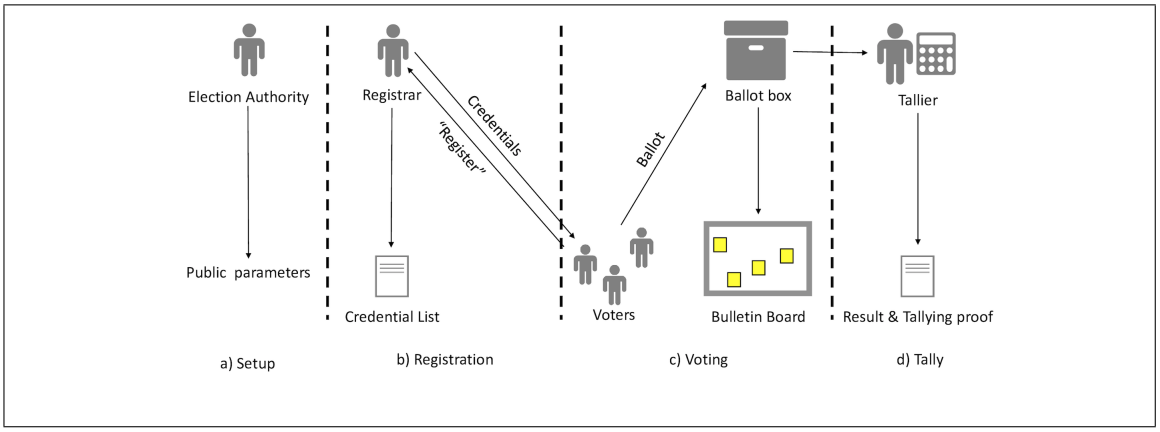

Figure 3.1: An overview of the four phases of a generic e-voting protocol: setup, registration, voting and tallying. Dashed lines indicate separation of the four phases. Note that registration is a phase that is specific to e-voting schemes with internal credentials.

A note on re-voting policies. Re-voting is a common feature of many e-voting schemes, e.g., [8, 42, 77]. If re-voting is permitted, voters may submit more than one ballot to the ballot box. As a result, ballots must be filtered to retain (and include in the final result) a single ballot per voter during tallying. This filtering process implements a re-voting policy. A common re-voting policy is last-vote-counts, which is implemented, for example, in the Estonian e-voting scheme [62]. As expected, this policy ensures that the election result counts only the final ballot cast by each voter. In this chapter, we present new ballot

secrecy definitions that capture a last-vote-counts policy. However, we recognise that some elections require a first-vote-counts policy, equivalent to saying that a voter cannot re-vote. Therefore, we describe how our definitions can be adapted straightforwardly to capture this re-vote policy.

We now introduce the syntax for an e-voting scheme. Our new definitions consider e-voting schemes with internal credentials. However, we also discuss comparisons between existing ballot secrecy definitions, many of which consider e-voting schemes with external credentials. To facilitate this, we formally define the syntax of an e-voting scheme, adapted from the syntax of [22, 38], for schemes with external and internal credentials. For brevity, we refer to e-voting schemes with internal credentials as, simply, e-voting schemes and refer explicitly to e-voting schemes with external credentials.

3.2.2 E-voting Schemes With External Credentials

Notation. In Chapters 3 and 4, we use r to denote the result of an election. To distinguish this from the randomness taken as input by algorithms, we always use a subscript, e.g., r_i, or the notation coins, to denote randomness in Chapters 3 and 4. We use the notation r without a subscript to denote the election result.

We define syntax for an e-voting scheme relative to a result function $f : (\mathcal{V} \times \mathcal{C})^* \to R$, where $\mathcal{V} = \{id_1, \ldots, id_{|\mathcal{V}|}\}$ is the set of eligible voters, $\mathcal{C}$ is the set of candidates (e.g., vote choices available to a voter), and R is the result space of the election.

Definition 41 (E-voting scheme with external credentials)**.** *An* e-voting scheme with external credentials Γ-EXT *for a result function f, set of voters $\mathcal{V}$ and set of candidates $\mathcal{C}$, is a tuple of polynomial time algorithms* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Valid}, \mathsf{Append}, \mathsf{Publish}, \mathsf{Tally}, \mathsf{Verify})$ *such that*

$\mathsf{Setup}(1^\lambda)$ On input of security parameter 1^λ, algorithm Setup outputs an election key pair (pk, sk) where pk is the election public key and sk is the election secret key.

$\mathsf{Vote}(v, id, pk)$ On input of vote v, voter identity id and election public key pk, algorithm Vote outputs a ballot b.

$\mathsf{Valid}(b, \mathsf{bb}, pk)$ On input of ballot b, ballot box bb and election public key pk, algorithm

Valid outputs 1 if ballot b is valid (e.g., well-formed, not a duplicate ballot), and 0 otherwise.

$\mathsf{Append}(b, \mathsf{bb}, pk)$ On input of ballot b, ballot box bb and election public key pk, algorithm Append outputs the updated ballot box to include ballot b.

$\mathsf{Publish}(\mathsf{bb})$ On input of ballot box bb, algorithm $\mathsf{Publish}$ outputs bulletin board $\mathcal{BB}$.

$\mathsf{Tally}(\mathsf{bb}, sk)$ On input of ballot box bb and election secret key sk, algorithm Tally computes and outputs the election result r with a proof ρ that the result is correct.

$\mathsf{Verify}(\mathcal{BB}, r, \rho, pk)$ On input of bulletin board $\mathcal{BB}$, result r, proof ρ and election public key pk, algorithm Verify outputs 1 if the election result verifies, and 0 otherwise.

E-voting schemes must satisfy *correctness*, which requires that the result output by algorithm Tally is equivalent to result function f applied to all votes and voter identities input to algorithm Vote.

Definition 42 (Correctness)**.** *An e-voting scheme* Γ-EXT *defined with respect to a result function* f*, set of voters* $\mathcal{V}$ *and set of candidates* $\mathcal{C}$*, satisfies* correctness *if, for any set of voters* $\mathcal{V} = \{id_1, \ldots, id_{|\mathcal{V}|}\}$ *and votes* $v_1, \ldots, v_{|\mathcal{V}|} \in \mathcal{C}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l}\mathsf{bb}\leftarrow();\\ (pk,sk) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda);\\ \textbf{for } i = 1,\ldots,|\mathcal{V}|\\ \quad b_i \leftarrow\!\$\ \mathsf{Vote}(v_i,id_i,pk);\\ \quad \textbf{if } \mathsf{Valid}(b_i,\mathsf{bb},pk) = 1 : \mathsf{bb} \leftarrow\!\$\ \mathsf{Append}(b_i,\mathsf{bb},pk);\\ (r,\rho) \leftarrow\!\$\ \mathsf{Tally}(\mathsf{bb},sk)\end{array} : \ r = f((id_1,v_1),\ldots,(id_{|\mathcal{V}|},v_{|\mathcal{V}|}))\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

3.2.3 E-Voting Schemes With Internal Credentials

We now define syntax for an e-voting scheme with internal credentials, which is very similar to Definition 41, but is equipped with an additional algorithm $\mathsf{Register}$ that generates credential pairs for voters. Additionally, voters are represented by their public credential when voting, rather than their identity, and the list of public credentials $\mathcal{L}$ is included as input to several algorithms. Though the syntax is similar, we choose to present it in full as it is used extensively in this chapter.

Definition 43 (E-voting scheme)**.** *An* e-voting scheme Γ *for a result function* f*, set of voters* $\mathcal{V}$ *and set of candidates* $\mathcal{C}$*, is a tuple of polynomial time algorithms* $(\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Valid}, \mathsf{Append}, \mathsf{Publish}, \mathsf{Tally}, \mathsf{Verify})$ *such that*

$\mathsf{Setup}(1^\lambda)$ On input of security parameter 1^λ, algorithm Setup outputs an election key pair (pk, sk) where pk is the election public key and sk is the election secret key.

$\mathsf{Register}(id, pk)$ On input of voter identity id and election public key pk, algorithm $\mathsf{Register}$ outputs a credential pair (pk_{id}, sk_{id}) where pk_{id} is voter id's public credential and sk_{id} is id's secret credential. We implicitly assume that set $\mathcal{L}$ is updated with pk_{id} such that $\mathcal{L} \leftarrow \mathcal{L} \cup \{pk_{id}\}$.

$\mathsf{Vote}(v, pk_{id}, sk_{id}, pk)$ On input of vote v, public credential pk_{id}, secret credential sk_{id} and election public key pk, algorithm Vote outputs a ballot b. We assume that b includes the voter's public credential.

$\mathsf{Valid}(b, \mathsf{bb}, \mathcal{L}, pk)$ On input of ballot b, ballot box bb, set $\mathcal{L}$ and election public key pk, algorithm Valid outputs 1 if ballot b is valid, and 0 otherwise.

$\mathsf{Append}(b, \mathsf{bb}, \mathcal{L}, pk)$ On input of ballot b, ballot box bb, set $\mathcal{L}$ and election public key pk, algorithm Append outputs the updated ballot box to include ballot b.

$\mathsf{Publish}(\mathsf{bb})$ On input of ballot box bb, algorithm $\mathsf{Publish}$ outputs bulletin board $\mathcal{BB}$.

$\mathsf{Tally}(\mathsf{bb}, \mathcal{L}, sk)$ On input of ballot box bb, set $\mathcal{L}$ and election secret key sk, algorithm Tally computes and outputs the election result r with a proof ρ that the result is correct.

$\mathsf{Verify}(\mathcal{BB}, \mathcal{L}, r, \rho, pk)$ On input of bulletin board $\mathcal{BB}$, set $\mathcal{L}$, result r, proof ρ and election public key pk, algorithm Verify outputs 1 if the election result verifies, and 0 otherwise.

We define correctness for this syntax, which naturally extends Definition 42 to schemes with internal credentials.

Definition 44 (Correctness)**.** *An e-voting scheme* Γ *defined with respect to a result function* f*, set of voters* $\mathcal{V}$ *and set of candidates* $\mathcal{C}$*, satisfies* correctness *if, for any set of voters* $\mathcal{V} = \{id_1, \ldots, id_{|\mathcal{V}|}\}$ *and votes* $v_1, \ldots, v_{|\mathcal{V}|} \in \mathcal{C}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l}\mathsf{bb}\leftarrow(); \ \mathcal{L}\leftarrow\emptyset; \\ (pk,sk) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda); \\ \textbf{for } i = 1,\ldots,|\mathcal{V}| \\ \quad (pk_{id_i},sk_{id_i}) \leftarrow\!\$\ \mathsf{Register}(id_i,pk); \\ \quad \mathcal{L} \leftarrow \mathcal{L} \cup \{pk_{id_i}\}; \\ \quad b_i \leftarrow\!\$\ \mathsf{Vote}(v_i,pk_{id_i},sk_{id_i},pk); \\ \quad \textbf{if } \mathsf{Valid}(b_i,\mathsf{bb},\mathcal{L},pk) = 1 \\ \quad\quad \mathsf{bb} \leftarrow\!\$\ \mathsf{Append}(b_i,\mathsf{bb},\mathcal{L},pk); \\ (r,\rho) \leftarrow\!\$\ \mathsf{Tally}(\mathsf{bb},\mathcal{L},sk)\end{array} : \ r = f((id_1,v_1),\ldots,(id_{|\mathcal{V}|},v_{|\mathcal{V}|}))\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

3.3 A Comparison of Ballot Secrecy Definitions

Ballot secrecy is a fundamental privacy property for e-voting, and the literature has devoted much space to presenting and analysing formal definitions of ballot secrecy. All definitions come with strengths and weaknesses, many of which are shared by multiple definitions. In this section, we review existing definitions by grouping them according to their intuition, which allows us to identify common strengths and weaknesses. In particular, we identify two types of definitions. Firstly, some provide an adversary with a view of a 'real' or a 'fake' election. The adversary must distinguish these views when provided with a result corresponding to the 'real' election. Secondly, some definitions describe an experiment in which an adversary must distinguish two different election views when presented with a result computed for the viewed election. We consider these two approaches in turn.

3.3.1 Tally the 'Real' Election

The approach of tallying the 'real' election was introduced in [23] and refined in [27]. Subsequently, in [22], Cortier *et al.* reviewed ballot secrecy definitions from the literature and defined BPRIV as an alteration of [23, 27], avoiding weaknesses found in both previous, related, definitions. Arguably, BPRIV has since become the most well-known and widely used definition of ballot secrecy for e-voting schemes with external credentials in the literature. In fact, in [21], it was used to prove the security of Helios and has been extended to capture receipt-freeness [38]. Additionally, it has been extended to model e-voting schemes with internal credentials [44] and to capture a malicious ballot box [48]. Moreover, BPRIV is shown to imply a simulation-based notion of ballot secrecy [22], reinforcing the soundness of BPRIV.

As BPRIV is the state-of-the-art definition in this category, we focus on BPRIV and its extensions from [44] and [48] to discuss the strengths and limitations associated with this approach. Accordingly, we now cast BPRIV into our syntax for e-voting schemes with external credentials (Definition 41). Moreover, in Chapter 4, we require a formal definition of BPRIV in our syntax to prove a number of results.

The BPRIV experiment. In the BPRIV experiment, depending on the value of a bit β, an adversary $\mathcal{A}$ is provided with a view of a 'real' ($\beta = 0$) or a 'fake' election ($\beta = 1$) and outputs a guess at bit β. The experiment can be described as follows. At the beginning of the experiment, two ballot boxes bb_0 and bb_1, where bb_0 represents the real election and bb_1 the fake election, are initialised as empty. An election key pair (and auxiliary information τ if $\beta = 1$) is generated and the election public key pk is input to $\mathcal{A}$. Then, $\mathcal{A}$ can query a number of oracles defined in Figure 3.2, under the constraint that oracle $\mathcal{O}\mathsf{tally}$ appears as the final oracle call. Oracle $\mathcal{O}\mathsf{vote}$ allows the adversary to submit two votes (the left- and right-hand vote) on behalf of a voter, and two ballots (the left- and right-hand ballot) are computed. The left-hand ballot is appended to bb_0 and the right-hand ballot is appended to bb_1. Additionally, $\mathcal{A}$ can submit ballots via oracle $\mathcal{O}\mathsf{cast}$, which are appended to ballot boxes bb_0 and bb_1. At any time, the adversary can view the election by calling oracle $\mathcal{O}\mathsf{board}$, which returns bulletin board $\mathcal{BB}_\beta$. In this way, $\mathcal{A}$ is provided with a view of a 'real' or a 'fake' election. Finally, $\mathcal{A}$ obtains the result of the real election (i.e., the result computed over the contents of bb_0) by calling oracle $\mathcal{O}\mathsf{tally}$ and terminates by outputting a guess of bit β. An e-voting scheme with external credentials is said to satisfy BPRIV if the adversary can guess β with probability only negligibly greater than 1/2.

As defined in [22], BPRIV relies on algorithms Sim and Sim_ρ, which facilitate the ability to simulate the tallying proof ρ such that the outcome computed over the contents of bb_0 appears to be computed over the contents of bb_1, when $\beta = 1$. Algorithms Sim and Sim_ρ are defined as follows.

$\mathsf{Sim}(1^\lambda)$ On input of security parameter 1^λ, algorithm Sim outputs a tuple (pk, sk, τ) where pk is the election public key, sk is the election secret key and τ is auxiliary information that is used to output a simulated proof during the tally phase of the election.

$\mathsf{Sim}_\rho(\tau, \mathsf{bb}, r)$ On input of auxiliary information τ, ballot box bb and election result r, algorithm Sim_ρ outputs a proof ρ that r is the election result computed for the contents of bb.

With these algorithms, we cast the definition of BPRIV from [22] into our syntax as follows.

Definition 45 (BPRIV [22])**.** *An e-voting scheme with external credentials* Γ-EXT *for a result function* f*, set of voters* $\mathcal{V}$*, and set of candidates* $\mathcal{C}$ *satisfies* BPRIV *if there exists*

algorithms Sim *and* Sim_ρ *such that, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{BPRIV},0}_{\Gamma\text{-}\mathsf{EXT},\mathcal{A},\mathsf{Sim},\mathsf{Sim}_\rho,\mathcal{V},\mathcal{C}}(\lambda)=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{BPRIV},1}_{\Gamma\text{-}\mathsf{EXT},\mathcal{A},\mathsf{Sim},\mathsf{Sim}_\rho,\mathcal{V},\mathcal{C}}(\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{BPRIV},\beta}_{\Gamma\text{-}\mathsf{EXT},\mathcal{A},\mathsf{Sim},\mathsf{Sim}_\rho,\mathcal{V},\mathcal{C}}(\lambda)$ *is the experiment defined in Figure 3.2 for* $\beta\in\{0,1\}$*.*

$\underline{\mathsf{Exp}^{\mathsf{BPRIV},\beta}_{\Gamma\text{-}\mathsf{EXT},\mathcal{A},\mathsf{Sim},\mathsf{Sim}_\rho,\mathcal{V},\mathcal{C}}(\lambda)}$

$\mathsf{bb}_0\leftarrow();\ \mathsf{bb}_1\leftarrow()$
if $\beta=0:\ (pk,sk)\leftarrow_\$\mathsf{Setup}(1^\lambda)$
if $\beta=1:\ (pk,sk,\tau)\leftarrow_\$\mathsf{Sim}(1^\lambda)$
$\beta'\leftarrow_\$\mathcal{A}^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}(pk)$
return β'

$\underline{\mathcal{O}\mathsf{tally}_{(sk,\tau,\mathsf{bb}_0,\mathsf{bb}_1)}()}$

if $\beta=0:\ (r,\rho)\leftarrow_\$\mathsf{Tally}(\mathsf{bb}_0,sk)$
if $\beta=1$
  $(r,\rho')\leftarrow_\$\mathsf{Tally}(\mathsf{bb}_0,sk)$
  $\rho\leftarrow_\$\mathsf{Sim}_\rho(\tau,\mathsf{bb}_1,r)$
return (r,ρ)

$\underline{\mathcal{O}\mathsf{board}_{(\mathsf{bb}_\beta)}()}$

return $\mathsf{Publish}(\mathsf{bb}_\beta)$

$\underline{\mathcal{O}\mathsf{vote}_{(pk,\mathsf{bb}_0,\mathsf{bb}_1)}(id,v_0,v_1)}$

$b_0\leftarrow_\$\mathsf{Vote}(v_0,id,pk)$
$b_1\leftarrow_\$\mathsf{Vote}(v_1,id,pk)$
if $\mathsf{Valid}(b_\beta,\mathsf{bb}_\beta,pk)=0$ **return** $\bot$
$\mathsf{bb}_0\leftarrow_\$\mathsf{Append}(b_0,\mathsf{bb}_0,pk)$
$\mathsf{bb}_1\leftarrow_\$\mathsf{Append}(b_1,\mathsf{bb}_1,pk)$
return $\top$

$\underline{\mathcal{O}\mathsf{cast}_{(pk,\mathsf{bb}_0,\mathsf{bb}_1)}(b)}$

if $\mathsf{Valid}(b,\mathsf{bb}_\beta,pk)=0$ **return** $\bot$
$\mathsf{bb}_0\leftarrow_\$\mathsf{Append}(b,\mathsf{bb}_0,pk)$
$\mathsf{bb}_1\leftarrow_\$\mathsf{Append}(b,\mathsf{bb}_1,pk)$
return $\top$

Figure 3.2: The ballot secrecy experiment $\mathsf{Exp}^{\mathsf{BPRIV},\beta}_{\Gamma\text{-}\mathsf{EXT},\mathcal{A},\mathsf{Sim},\mathsf{Sim}_\rho,\mathcal{V},\mathcal{C}}(\lambda)$ for $\beta\in\{0,1\}$ from [22] and cast into our syntax.

BPRIV avoids many limitations of existing definitions, including those definitions that follow the 'tally the viewed election' approach (see Chapter 3.3.2). However, we highlight two drawbacks of BPRIV.

Need for additional properties. As a stand-alone definition, BPRIV is subject to attacks, as highlighted by Cortier *et al.*, the authors of BPRIV [22]. Specifically, BPRIV does not capture an attacker that can cause the rejection of honestly created ballots, which can lead to a violation of ballot secrecy. To demonstrate this, Cortier *et al.* define an e-voting scheme such that ballots are appended with a bit and algorithm Vote always appends '0'. Then, if there exists a ballot in bb that is appended with '1', all subsequent ballots are rejected. A real-life attacker against this scheme can ensure that a majority of honestly created ballots are rejected, potentially revealing the votes of a small subset

of honest voters. However, as BPRIV always computes the result of the 'real' election, an adversary in the BPRIV experiment that follows this attack strategy learns nothing more than if they had not followed the strategy. Therefore, the scheme satisfies BPRIV despite the fact that ballot secrecy can be broken.

To capture this attack, Cortier *et al.* define an additional property that they call strong correctness. Strong correctness describes an experiment in which an adversary $\mathcal{A}$ outputs a ballot box bb, a voter identity id and a vote v. A scheme is said to satisfy strong correctness if a ballot constructed for voter id and vote v is not rejected (i.e., algorithm Valid returns 1 and the ballot is added to the ballot box). We cast strong correctness into our syntax.

Definition 46 (Strong correctness [22])**.** *An e-voting scheme with external credentials* Γ-EXT *for a result function f, set of voters $\mathcal{V}$, and set of candidates $\mathcal{C}$ satisfies* strong correctness *if, for any PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l}(pk,sk) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda);\\ (id,v,\mathsf{bb}) \leftarrow\!\$\ \mathcal{A}(pk);\\ b \leftarrow\!\$\ \mathsf{Vote}(v,id,pk)\end{array} : \ id \in \mathcal{V} \wedge \mathsf{Valid}(b,\mathsf{bb},pk) \coloneqq 0\right] \leq \mathsf{negl}(\lambda)\,.$$

Cortier *et al.* highlight that BPRIV is subject to a further attack in which an attacker can cause well-formed ballots to be excluded from the election result. They describe an e-voting scheme for a referendum (i.e., $\mathcal{C} = \{0,1\}$) that rejects the ballot of the first voter if they vote '1'. An attacker against this scheme can determine how the first voter voted; if the result does not contain the first voter's ballot, it is clear that the voter cast a ballot for '1'. As with strong consistency, this scheme satisfies BPRIV because the result is always computed for the real election. To capture this attack, BPRIV must be accompanied by a second additional property, strong consistency, which is also defined in [22]. Strong consistency describes an experiment in which an adversary $\mathcal{A}$ outputs a ballot box bb. A scheme satisfies strong consistency if the result output by algorithm Tally for bb is equal to the result computed by function f on input of the votes encoded in the ballots in bb, and all ballots meet some minimal validity requirement. Strong consistency requires two algorithms Extract and ValidInd such that,

$\mathsf{Extract}(b, sk)$ On input of ballot b and election secret key sk, algorithm Extract outputs a tuple (id, v) where v is the vote encoded in ballot b that is produced by a voter id.

$\mathsf{ValidInd}(b, pk)$ On input of ballot b and election public key pk, algorithm ValidInd outputs

1 if the ballot is well-formed, and 0 otherwise.

With these algorithms, we cast the definition of strong consistency from [22] into our syntax.

Definition 47 (Strong consistency [22])**.** *An e-voting scheme with external credentials* Γ-EXT *for a result function* f*, set of voters* $\mathcal{V}$*, and set of candidates* $\mathcal{C}$ *satisfies* strong consistency *if there exists algorithms* Extract *and* ValidInd *such that the following conditions hold.*

- *For any* $v \in \mathcal{C}$*,* $id \in \mathcal{V}$ *and any* (pk, sk) *output by algorithm* Setup*, there exists a negligible function* negl *such that,*

$$\Pr\left[\begin{array}{l} b \leftarrow\!\$\ \mathsf{Vote}(v,id,pk); \\ \mathsf{bb} \leftarrow\!\$\ \mathsf{Append}(b,\mathsf{bb},pk) \end{array} : \ \mathsf{Extract}(\mathsf{bb}[1],sk) \coloneqq (id,v)\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

- *For any* (pk, sk) *output by algorithm* Setup *and any adversary* $\mathcal{A}(pk, sk)$ *that outputs tuple* (bb, b)*, if* $\mathsf{Valid}(b, \mathsf{bb}, pk) = 1$*, then* $\mathsf{ValidInd}(b, pk) = 1$*.*

- *For all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\begin{array}{l} \mathsf{bb}' \leftarrow (); \\ (pk,sk) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda); \\ \mathsf{bb} \leftarrow\!\$\ \mathcal{A}(pk); \\ (r,\rho) \leftarrow\!\$\ \mathsf{Tally}(\mathsf{bb},sk); \\ \textbf{for } i = 1,\ldots,|\mathsf{bb}| \\ \quad \mathsf{bb}'[i] \leftarrow\!\$\ \mathsf{Extract}(\mathsf{bb}[i],sk); \\ r' \leftarrow f(\mathsf{bb}'[1],\ldots,\mathsf{bb}'[|\mathsf{bb}'|]) \end{array} : \begin{array}{l} r \neq r' \ \wedge \\ \textbf{for } i = 1,\ldots,|\mathsf{bb}| \ : \ \mathsf{ValidInd}(\mathsf{bb}[i],pk) \coloneqq 1 \end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

We also note that definitions derived from BPRIV [25, 38, 44, 48] also require strong consistency and strong correctness to capture the two attacks outlined above.

Extendibility. Generally, BPRIV does not lend itself well to extensions. Though it is possible to extend BPRIV to capture stronger notions of privacy for e-voting schemes, namely, receipt-freeness (see Chapter 4.4), we show that extending BPRIV to other scenarios has proven difficult. We demonstrate this by recalling existing definitions that extend BPRIV in various ways and highlighting limitations in these definitions.

In [44], BPRIV was extended to e-voting schemes with internal credentials, but allows for *static* voter corruption only. Briefly, the BPRIV extension defines an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

$\mathcal{A}_1$ is given access to an oracle that, on input a voter id and a bit d, registers voter id and returns the voter's public credential (if $d = 0$) or the voter's credential pair (if $d = 1$). As such, the oracle models registration (if $d = 0$) and corruption (if $d = 1$) of voters. Then, $\mathcal{A}_2$ is input the state of $\mathcal{A}_1$ and is defined equivalently to adversary $\mathcal{A}$ in the original BPRIV experiment of Figure 3.2 but for the following exception: $\mathcal{A}_2$ can only query oracle $\mathcal{O}$vote (resp., $\mathcal{O}$cast) on behalf of honest (resp., corrupt) voters. In this way, the BPRIV experiment models an attacker that can only corrupt voters *before* the voting phase of the election.

BPRIV has also been extended to the malicious ballot box setting [48]. Recall that, in the BPRIV experiment, the adversary is presented with a tally computed for the 'real' election, which includes the left-hand ballots computed by oracle $\mathcal{O}$vote. The modelling of a 'real' and a 'fake' election presents a challenge in the malicious ballot box setting. If the adversary controls the ballot box, oracle $\mathcal{O}$vote returns the left- (if $\beta = 0$) or right-hand (if $\beta = 1$) ballot to the adversary, rather than appending the ballots to the corresponding ballot boxes. The adversary can then choose whether or not to append the ballot to the ballot box (possibly after making modifications to the ballot). If $\beta = 0$, the election result is computed for the contents of the ballot box constructed by the adversary. However, if $\beta = 1$, the adversary constructs a ballot box for the 'fake' election; there is no ballot box for the real election. Thus, extending BPRIV to the malicious ballot box setting presents the following challenge: how can a tally for the real election be computed if there is no corresponding ballot box? This challenge was addressed in [48] in the following way. An algorithm is defined that determines how an attacker can modify the ballots of honest voters (e.g., the algorithm can be defined to include one or more of the following actions: delete, modify, re-order ballots). Then, if $\beta = 1$, a ballot box for the 'real' election is constructed by transforming every right-hand ballot produced by oracle $\mathcal{O}$vote according to the transformation that the adversary applied to the corresponding left-hand ballot, if that ballot is included in the adversary's ballot box. The transformed right-hand ballots replace the transformed left-hand ballots in the adversary's ballot box to produce a ballot box for the 'real' election. In this way, the extension of BPRIV to the malicious ballot box setting defined in [48] is flexible, capable of capturing various ballot modifications. However, this means that, before applying the definition, it is necessary first to define an algorithm, presenting an opportunity for flawed security proofs if the algorithm is not defined correctly.

BPRIV cannot be easily extended to the distributer tallier setting where a subset of talliers can be corrupted. Indeed, in [50], del Pino *et al.* state that it is challenging to adapt BPRIV to this setting because corrupted talliers (i.e., talliers controlled by the adversary) must participate in the tallying stage of the election for *both* ballot boxes. However, BPRIV only ever returns the result for the ballot box corresponding to the 'real' election. As such, if $\beta = 1$, the adversary can only compute the tally for bb_1, not bb_0 as required in the BPRIV experiment. To overcome this, del Pino *et al.* depart from BPRIV's approach and pivot towards the approach that we discuss next: tallying the viewed election, which, as we shall see, lends itself well to extensions into the malicious setting.

3.3.2 Tally the 'Viewed' Election

The second approach we consider includes early definitions of ballot secrecy that pre-date the BPRIV approach. This approach was introduced in [16, 18] for voting schemes with external credentials and has been adopted, in modified forms, in [28, 29, 47]. Unlike the BPRIV approach, there is no single definition in this category that is regarded as state-of-the-art. For this reason, we do not recall a formal definition of ballot secrecy from the literature that falls within this category. Rather, we describe this approach, highlighting the features, strengths, and limitations shared by the definitions presented in [16, 18, 28, 29, 47].

Overview of this approach. Each definition in this category describes an experiment in which an adversary $\mathcal{A}$, depending on a bit β, is presented with a view of a left- (if $\beta = 0$) or right-hand (if $\beta = 1$) election and the corresponding election result. At the beginning of the experiment, a ballot box bb is initialised as empty, and an election key pair is generated. The election public key is input to $\mathcal{A}$, which can then query oracles as follows.

- $\mathcal{O}\mathsf{vote}(id, v_0, v_1)$ runs $b \leftarrow\!\!\$\, \mathsf{Vote}(v_\beta, id, pk)$ and appends ballot b to ballot box bb.
- $\mathcal{O}\mathsf{cast}(b)$ appends ballot b to ballot box bb.

Moreover, the adversary can view bulletin board $\mathcal{BB}$ throughout the experiment. Finally, the result computed over the contents of bb and the corresponding tallying proof is input

to $\mathcal{A}$, which outputs a guess at bit β. If $\mathcal{A}$ can guess β with probability only negligibly more than 1/2, an e-voting scheme is said to satisfy ballot secrecy.

Balancing conditions. Definitions that follow this approach require a balancing condition such that the left- and right-hand votes submitted to oracle $\mathcal{O}\mathsf{vote}$ are balanced. Otherwise, an adversary can trivially guess β as follows. An adversary can query '0' as the left-hand vote and '1' as the right-hand vote to oracle $\mathcal{O}\mathsf{vote}$ on behalf of *all* voters and can choose never to query oracle $\mathcal{O}\mathsf{cast}$. Then, the election result is unanimous. If the result contains only votes for '0' (resp.,'1'), the adversary outputs '0' (resp., '1'), correctly guessing β. Therefore, the scheme is declared not to satisfy ballot secrecy, even if it intuitively does.

To capture a balancing condition, oracle $\mathcal{O}\mathsf{vote}$ must maintain multisets V_0 and V_1 that, respectively, track the left- and right-hand votes submitted to the oracle. Two popular balancing conditions have been proposed in the literature, which we now describe.

1. $V_0 = V_1$ [28, 29]: This requires that V_0 and V_1 are equal when viewed as multisets. It ensures that, for every left-hand vote of a voter, there exists a voter for whom the same vote is submitted as their right-hand vote, and vice versa.
2. $f(V_0) = f(V_1)$ [16, 47]: The condition requires that the result function applied to each multiset produces an identical election tally.

As a result of these balancing conditions, definitions that tally the viewed election share a common limitation: the class of result functions captured by these definitions is limited.

Restricted result functions. Both balancing conditions described above restrict the class of result functions, but each balancing condition results in a different restriction. Balancing condition 2 ($f(V_0) = f(V_1)$) requires a *partial tally assumption* [22, 47]. That is to say, if a set of votes is written as $V = V' \cup V''$, the partial tally assumption states that $f(V) = f(V') * f(V'')$. This assumption is necessary because the ballot secrecy experiment computes the election result for the contents of V_0 *and* the ballots inputs to oracle $\mathcal{O}\mathsf{cast}$. If the partial tally assumption does not hold, the balancing condition may hold but the election result reveals the bit β. On the other hand, balancing condition 1 ($V_0 = V_1$) does

not require the partial tally assumption, but does not capture result functions that allow different vote assignments that lead to the same result. Indeed, the balancing condition does not permit left-hand votes and right-hand votes that lead to the same result, yet are not identical when viewed as multisets. Therefore, a definition with this balancing condition provides no guarantee of ballot secrecy for all potential voting patterns if an intuitively ballot secret scheme implements such a result function.

We now demonstrate these restrictions by adopting an example of an e-voting scheme presented in [22]. Consider an e-voting scheme that is intuitively ballot secret for which voters can select a vote from the set $\mathcal{C} = \{0, 1\}$. Define the result function to output the vote choice for which the most ballots are submitted, e.g., if three votes are submitted for '0' and 2 votes are submitted for '1', the result is defined as $r = 0$. Moreover, in the event of a draw, the result is defined to be '0'. Thus, this result function allows different vote assignments that lead to the same result and does not permit partial tallying. We now consider an adversary in the 'tally the viewed election' experiment against this e-voting scheme for each balancing condition.

1. $V_0 = V_1$. Suppose an adversary submits the following queries: $\mathcal{O}\mathsf{vote}(id_1,0,1)$; $\mathcal{O}\mathsf{vote}(id_2,1,0)$; $\mathcal{O}\mathsf{vote}(id_3,0,0)$; $\mathcal{O}\mathsf{vote}(id_4,0,1)$. In this event, despite the fact that the scheme is intuitively ballot secret and the election result is identitcal, the adversary cannot succeed as $V_0 \neq V_1$.

2. $f(V_0) = f(V_1)$. Suppose an adversary submits the following queries: $\mathcal{O}\mathsf{vote}(id_1,0,1)$; $\mathcal{O}\mathsf{vote}(id_2,1,0)$; $\mathcal{O}\mathsf{vote}(id_3,0,0)$; $\mathcal{O}\mathsf{vote}(id_4,0,1)$; $\mathcal{O}\mathsf{cast}(\mathsf{Vote}(1,id_5,\mathrm{pk}))$. In this case, the balancing condition is satisfied. However, if $\beta = 0$ (resp., $\beta = 1$), algorithm Tally outputs result $r = 0$ (resp., $r = 1$). As a result, the adversary can trivially guess β and the scheme does not satisfy ballot secrecy.

Therefore, we see that the class of result functions must be restricted in order to make meaningful statements about the ballot secrecy of an e-voting scheme. Despite this, common result functions, such as plurality voting (i.e., first-past-the-post), are within the scope of definitions in this category. In our definitions, we choose to avoid the partial tally assumption and follow the approach of [28, 29]. In particular, we adjust the balancing condition to the setting of internal credentials and model revoting policies (neither of which is considered in previous definitions with this balancing condition).

Extendibility. Unlike BPRIV, it has been shown that it is straightforward to extend definitions in this category to the malicious setting. In particular, Bernhard and Smyth [29] and Cortier and Lallemand [47] extend this approach to the malicious ballot box setting in an intuitive way. Furthermore, in Chapter 3.5, we also demonstrate how this approach can be extended to the malicious ballot box and distributed tallier settings.

On the other hand, extending to model e-voting schemes with internal credentials is not as well understood in this setting. The only definition in this category that models voter registration allows static corruption of voters only [47]. Next, we show that it is possible to model adaptive corruption of voters, presenting a formal definition of ballot secrecy that captures this strategy.

Concluding remarks. Definitions in this category provide the advantages of being easy to extend to the malicious setting but restrict the class of result function. In comparison, BPRIV does not restrict the class of result functions but is not as straightforward to extend to the malicious setting or e-voting schemes with internal credentials.

While BPRIV has been widely discussed in recent literature, the 'tally the viewed election' approach presents benefits over BPRIV that we believe are worth exploring. Moreover, though this setting does restrict the class of result functions, many elections rely on basic result functions (e.g., plurality voting), which is captured by definitions in this category. We also note that, although many public office elections require more complex result functions that are out of scope of this category, e-voting is not yet recommended for such high-stakes elections [78]. Therefore, we now turn our attention to presenting new definitions of ballot secrecy in the 'tally the viewed election' approach.

3.4 Ballot Secrecy in the Honest Model (BSec)

We introduce a new definition of ballot secrecy that we call BSec. Our new definition builds upon the approach defined in [18] and is most similar to the definition presented in [28]. However, BSec differs from existing literature, which we elaborate on in Chapter 3.4.4. BSec captures an adversary that can *adaptively* corrupt voters, submitting ballots on their behalf, and can submit two votes (the left and right vote) on behalf of honest voters. Thus,

BSec describes an experiment in which an adversary is provided with access to a bulletin board and the corresponding election result. The result consists of ballots for the left or right vote submitted on behalf of honest voters, in addition to ballots submitted on behalf of corrupted voters. If the adversary cannot determine whether the bulletin board and result contains the left or right votes submitted by honest voters, a scheme is said to satisfy BSec. In this section, we describe the BSec experiment and our balancing condition, and then show that BSec is satisfiable.

3.4.1 The BSec Experiment

The BSec experiment for adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, formally defined in Figure 3.3, proceeds as follows. A challenger initialises lists $\mathbf{V_0}$ and $\mathbf{V_1}$ to model the balancing condition, and lists **pk** and **sk** that track voter public and secret credentials respectively, as empty and generates the election key pair (pk, sk). Additionally, ballot box bb is set as empty. Adversary $\mathcal{A}_1$ is input the public key pk and proceeds to query a number of oracles, formally defined in Figure 3.3 and described as follows.

$\mathcal{O}\mathsf{reg}_{(pk,\mathcal{L}\mathcal{Q}\mathsf{reg},\mathbf{pk},\mathbf{sk})}(id)$ registers an eligible voter. If id is in the set of eligible voters $\mathcal{V}$, oracle $\mathcal{O}$reg runs algorithm Register on behalf of id and returns the voter's public credential pk_{id} to $\mathcal{A}_1$. Oracle $\mathcal{O}$reg additionally updates a set of queries $\mathcal{Q}$reg to include the voter's identity id.

$\mathcal{O}\mathsf{corrupt}_{(\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{sk})}(id)$ corrupts a voter. If id is a registered voter, oracle $\mathcal{O}$corrupt returns the voter's private credential sk_{id} to $\mathcal{A}_1$, and updates a set of queries $\mathcal{Q}$corrupt to include id.

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb},\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathcal{Q}\mathsf{vote},\mathbf{pk},\mathbf{sk},\mathbf{V_0},\mathbf{V_1})}(id, v_0, v_1)$ produces and submits a ballot for vote v_β on behalf of an uncorrupted voter. If voter pk_{id} is registered but not corrupt, and v_0, v_1 are valid vote choices, oracle $\mathcal{O}$vote runs algorithms Vote and Append on behalf of voter pk_{id} and vote v_β. Oracle $\mathcal{O}$vote also updates lists $\mathbf{V_0}$ and $\mathbf{V_1}$ to include votes v_0 and v_1 respectively, and removes any previous entries for pk_{id}, modelling an e-voting scheme with a last-vote-counts revote policy.

$\mathcal{O}\mathsf{cast}_{(pk,\mathcal{L},\mathsf{bb},\mathcal{Q}\mathsf{corrupt},\mathcal{Q}\mathsf{vote},\mathbf{pk},\mathbf{V_0},\mathbf{V_1})}(id, b)$ submits a ballot on behalf of a voter. If ballot b is valid and created for voter pk_{id}, and ballot b does not exist in $\mathcal{Q}$vote, oracle

$\mathcal{O}\mathsf{cast}$ runs algorithm Append to append ballot b to ballot box bb. Oracle $\mathcal{O}\mathsf{cast}$ also removes any entries in lists $\mathbf{V_0}$ and $\mathbf{V_1}$ for pk_{id}.

$\mathcal{O}\mathsf{board}_{\mathsf{bb}}()$ returns bulletin board $\mathcal{BB}$ to $\mathcal{A}_1$.

Adversary $\mathcal{A}_1$ outputs state material that is input to adversary $\mathcal{A}_2$ along with the election result (and accompanying tallying proof) computed over the contents of bb. Then, $\mathcal{A}_2$ outputs a bit that is returned by the experiment if the balancing condition holds.

Definition 48 (BSec)**.** *An e-voting scheme Γ for a result function f, set of voters $\mathcal{V}$, and set of candidates $\mathcal{C}$ satisfies* BSec *if, for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* negl *such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{BSec},0}_{\Gamma,\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{BSec},1}_{\Gamma,\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{BSec},\beta}_{\Gamma,\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda)$ is the experiment defined in Figure 3.3 for $\beta \in \{0,1\}$.

$\mathsf{Exp}^{\mathsf{BSec},\beta}_{\Gamma,\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda)$

$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow (); \ \mathbf{V_0} \leftarrow (); \ \mathbf{V_1} \leftarrow (); \ \mathsf{bb} \leftarrow ()$
$\mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{vote} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset$
$(pk, sk) \leftarrow_{\$} \mathsf{Setup}(1^\lambda)$
$st \leftarrow_{\$} \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board}}(pk)$
$(r, \rho) \leftarrow_{\$} \mathsf{Tally}(\mathsf{bb}, \mathcal{L}, sk)$
for $i \in \{0,1\}$: $V'_i \coloneqq \{\{v_i | v_i \in \mathcal{C} \wedge v_i = \mathbf{V_i}[id] \text{ for some } id \in \mathcal{V}\}\}$
$\beta' \leftarrow_{\$} \mathcal{A}_2^{\mathcal{O}\mathsf{board}}(r, \rho, st)$
if $V'_0 = V'_1$ **return** β'

$\mathcal{O}\mathsf{reg}_{(pk,\mathcal{L},\mathcal{Q}\mathsf{reg},\mathbf{pk},\mathbf{sk})}(id)$

if $id \notin \mathcal{V} \ \vee \ id \in \mathcal{Q}\mathsf{reg}$ **return** $\perp$
$(\mathbf{pk}[id], \mathbf{sk}[id]) \leftarrow_{\$} \mathsf{Register}(id, pk)$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathbf{pk}[id]\}$
$\mathcal{Q}\mathsf{reg} \leftarrow \mathcal{Q}\mathsf{reg} \cup \{id\}$
return $\mathbf{pk}[id]$

$\mathcal{O}\mathsf{board}_{(\mathsf{bb})}()$

return $\mathsf{Publish}(\mathsf{bb})$

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb},\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathcal{Q}\mathsf{vote},\mathbf{pk},\mathbf{sk},\mathbf{V_0},\mathbf{V_1})}(id, v_0, v_1)$

if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt} \ \vee \ v_0, v_1 \notin \mathcal{C}$ **return** $\perp$
$b \leftarrow_{\$} \mathsf{Vote}(v_\beta, \mathbf{pk}[id], \mathbf{sk}[id], pk)$
$\mathsf{bb} \leftarrow_{\$} \mathsf{Append}(b, \mathsf{bb}, \mathcal{L}, pk)$
for $i \in \{0,1\}$: $\mathbf{V_i}[id] \leftarrow v_i$
$\mathcal{Q}\mathsf{vote} \leftarrow \mathcal{Q}\mathsf{vote} \cup \{(id, b)\}$
return $\top$

$\mathcal{O}\mathsf{cast}_{(pk,\mathcal{L},\mathsf{bb},\mathcal{Q}\mathsf{corrupt},\mathcal{Q}\mathsf{vote},\mathbf{pk},\mathbf{V_0},\mathbf{V_1})}(id, b)$

if $id \notin \mathcal{Q}\mathsf{corrupt} \ \vee \ \mathbf{pk}[id] \notin b \ \vee \ (id, b) \in \mathcal{Q}\mathsf{vote}$ **return** $\perp$
if $\mathsf{Valid}(b, \mathsf{bb}, \mathcal{L}, pk) = 0$ **return** $\perp$
$\mathsf{bb} \leftarrow_{\$} \mathsf{Append}(b, \mathsf{bb}, \mathcal{L}, pk)$
for $i \in \{0,1\}$: $\mathbf{V_i}[id] \leftarrow \epsilon$
return $\top$

$\mathcal{O}\mathsf{corrupt}_{(\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{sk})}(id)$

if $id \notin \mathcal{Q}\mathsf{reg}$ **return** $\perp$
$\mathcal{Q}\mathsf{corrupt} \leftarrow \mathcal{Q}\mathsf{corrupt} \cup \{id\}$
return $\mathbf{sk}[id]$

Figure 3.3: The ballot secrecy experiment $\mathsf{Exp}^{\mathsf{BSec},\beta}_{\Gamma,\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda)$ for $\beta \in \{0,1\}$.

3.4.2 Our Balancing Condition

We incorporate a balancing condition into our definition via sets V_0 and V_1 and multisets V'_0 and V'_1. Following a query to oracle $\mathcal{O}\mathsf{vote}$ on behalf of voter id, lists $\mathbf{V_0}$ and $\mathbf{V_1}$ are updated to contain v_0 and v_1 at positions $\mathbf{V_0}[id]$ and $\mathbf{V_1}[id]$. After the result of the

election is computed, multisets V_0' and V_1' are defined to contain the votes v_0 and v_1 from lists V_0 and V_1 respectively. Our balancing condition, $V_0' = V_1'$, ensures that, for every left-hand vote of an honest voter, there exists an honest voter for whom the same vote is submitted as their right-hand vote.

This notion of balance is inspired by Benaloh and Yung's early definition of ballot secrecy [18] and is used by Bernhard and Smyth in their ballot secrecy definition IND-SEC [28]. However, in comparison to BSec, neither IND-SEC nor Benaloh and Yung's definitions and syntax model internal credentials. It transpires that, to capture schemes with internal credentials, and, additionally, to capture revoting and adaptive corruption of voters, the balancing condition described above must include more complex features. We describe these subtleties and demonstrate their necessity through examples.

Updating entries in $\mathcal{O}$vote. We require that, following a query to $\mathcal{O}$vote for a tuple (id, v_0, v_1), lists $\mathbf{V_0}$ and $\mathbf{V_1}$ are updated to remove votes submitted previously to $\mathcal{O}$vote on behalf of voter id, and to replace these entries in lists $\mathbf{V_0}$ and $\mathbf{V_1}$ with the new votes v_0 and v_1 submitted in the current query to oracle $\mathcal{O}$vote. Else, an adversary can submit oracle queries that allow trivial distinctions, even if a scheme is intuitively ballot secret. For example, consider an e-voting scheme that employs a last-vote-counts revote policy and allows voters to cast a ballot for '0' or '1'. Let the election result be a vector of size two that indicates the number of votes cast for each candidate. Assume that $\mathcal{O}$vote does *not* remove any entries from lists $\mathbf{V_0}$ or $\mathbf{V_1}$. That is, the lists are updated with tuples of the form (id, v_i), rather than updating list $\mathbf{V_i}$ at position id for $i \in \{0, 1\}$. Then, an adversary in the BSec experiment can make the following queries: $\mathcal{O}\mathsf{vote}(id_1, 0, 1)$; $\mathcal{O}\mathsf{vote}(id_2, 1, 0)$; $\mathcal{O}\mathsf{vote}(id_1, 1, 0)$; $\mathcal{O}\mathsf{vote}(id_3, 0, 1)$. Lists $\mathbf{V_0} = ((id_1, 0), (id_2, 1), (id_1, 1), (id_3, 0))$ and $\mathbf{V_1} = ((id_1, 1), (id_2, 0), (id_1, 0), (id_3, 1))$. Subsequently, $V_0' = V_1'$ and the balancing condition is satisfied. However, if $\beta = 0$, $r = (1, 2)$ and, if $\beta = 1$, $r = (2, 1)$. Then the adversary trivially distinguishes the two views and succeeds in the BSec experiment. Therefore, it is essential that $\mathcal{O}$vote removes previous entries that contain id from lists $\mathbf{V_0}$ and $\mathbf{V_1}$. We ensure this by updating lists at position id. In this way, only the final query to $\mathcal{O}$vote for each voter is tracked. In our example above, this means that the balancing condition is not satisfied and the adversary cannot succeed in the BSec experiment.

Updating entries in $\mathcal{O}$cast. Similarly, following a query to $\mathcal{O}\mathsf{cast}$ on behalf of voter id, it is essential that entries for id are removed from lists $\mathbf{V_0}$ and $\mathbf{V_1}$. We implement this by updating the entries at $\mathbf{V_0}[id]$ and $\mathbf{V_1}[id]$ with the empty string ϵ. Otherwise, an adversary can trivially succeed in the BSec experiment. Consider the example above but, rather than the second two queries to $\mathcal{O}\mathsf{vote}$, the adversary queries $\mathcal{O}\mathsf{cast}(id_1, b)$ where b encodes a vote for '1'. Then, if $\mathcal{O}\mathsf{cast}$ does not remove the previous entry for id_1, the balancing condition is satisfied. Yet, the result $r = (0, 2)$ (if $\beta = 0$) or $r = (1, 1)$ (if $\beta = 1$). Under static corruption of voters, removal of entries is not necessary as an adversary cannot make a query to $\mathcal{O}\mathsf{vote}$ on behalf of a corrupted voter, i.e., voter id_1 in the example. Thus, removing previous entries is key to ensure that our balancing condition allows for adaptive corruption of voters.

Re-voting policies. Our balancing condition models a last-vote-counts revote policy. We choose to implement this policy to demonstrate how our definition can capture revoting policies. Moreover, last-vote counts is implemented in deployed e-voting schemes, for example, the Estonian i-Voting scheme [62]. On the other hand, it is common to implement an e-voting scheme with a no revote policy. For instance, Helios [2, 77] allows for both a last-vote-counts and a no revote policy. For this reason, we briefly describe how our balancing condition can be modified in a straightforward way to account for a no revote policy as follows. We define $\mathcal{O}\mathsf{vote}$ such that, on input a tuples (id, v_0, v_1), previous entries for id are not replaced from lists $\mathbf{V_0}$ and $\mathbf{V_1}$. That is, if $\mathbf{V_0}[id]$ and $\mathbf{V_1}[id]$ are not equal to the empty string, $\mathcal{O}\mathsf{vote}$ does not add v_0 and v_1 to $\mathbf{V_0}$ and $\mathbf{V_1}$ respectively. Finally, we redefine $\mathcal{O}\mathsf{cast}$ such that lists $\mathbf{V_0}$ and $\mathbf{V_1}$ are not updated.

3.4.3 Satisfiability of BSec

We now demonstrate satisfiability of BSec by introducing an e-voting scheme that we call Γ_{mini}, and showing that it satisfies BSec.

Description of Γ_{mini}. Our Γ_{mini} construction requires a homomorphic public-key encryption scheme PKE and a signature of knowledge SOK. In particular, we leverage the homomorphic property of PKE to compute the election result (i.e., as in the Helios e-voting scheme [2, 77] and related constructions [22, 23], we compute the product of all ballots

and then decrypt the resulting ciphertext to obtain the final result).

We present an intuition of Γ_{mini} here and formally define it in Figure 3.4. A voter with credential pair (pk_{id}, sk_{id}), generated by a one-way function g, casts a ballot by producing an encryption of $v \in \mathcal{C} = \{0, 1\}$, denoted c, and generating a signature of knowledge, denoted σ. The signature of knowledge proves that the ciphertext c encrypts v and is signed with the voter's credential pair. The form of a ballot b is (pk_{id}, c, σ). If ciphertext c does not appear in a ballot on the ballot box, and the signature of knowledge σ verifies, the ballot is appended to the ballot box. The ballot box and the bulletin board are identical in this scheme. To compute the result, the ciphertexts from the final ballot cast by each voter are homomorphically tallied. Then, the homomorphic tally is decrypted, giving the result of the election.

We define algorithm Tally to output $\perp$ in place of a tallying proof, and algorithm Verify to always return 1. As a result, Γ_{mini} is not a verifiable e-voting scheme. However, as shown in [23], it is possible to construct fully verifiable e-voting schemes from schemes such as Γ_{mini}. Indeed, in [23], Bernhard *et al.* introduce minivoting, a simple e-voting scheme in which voters simply encrypt their vote and a tallier decrypts each ciphertext and computes the result from the resulting plaintext votes. Like Γ_{mini}, minivoting is not verifiable. However, Bernhard *et al.* prove that minivoting satisfies a notion of ballot secrecy and subsequently build a generic, verifiable, e-voting scheme with homomorphic tallying that also satisfies ballot secrecy.

Ballot Secrecy of Γ_{mini}. To satisfy BSec, we require that PKE satisfies IND-CPA, and SOK satisfies simulatability and extractability. We obtain the result in Theorem 1.

Theorem 1. *E-voting scheme Γ_{mini} (Figure 3.4) satisfies* BSec *if public key encryption scheme* PKE *satisfies* IND-CPA *and signature of knowledge* SOK *satisfies simulatability and extractability.*

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary in the BSec experiment that makes at most k_1 queries to oracle $\mathcal{O}$vote and at most k_2 queries to oracle $\mathcal{O}$cast. We proceed through a series of game hops that we show are indistinguishable to the adversary. In the final game, the view of the adversary will be identical for $\beta = 0$ and $\beta = 1$. We define Game 0 as the BSec experiment with β chosen randomly. Let S_i be the event that adversary $\mathcal{A}$ correctly

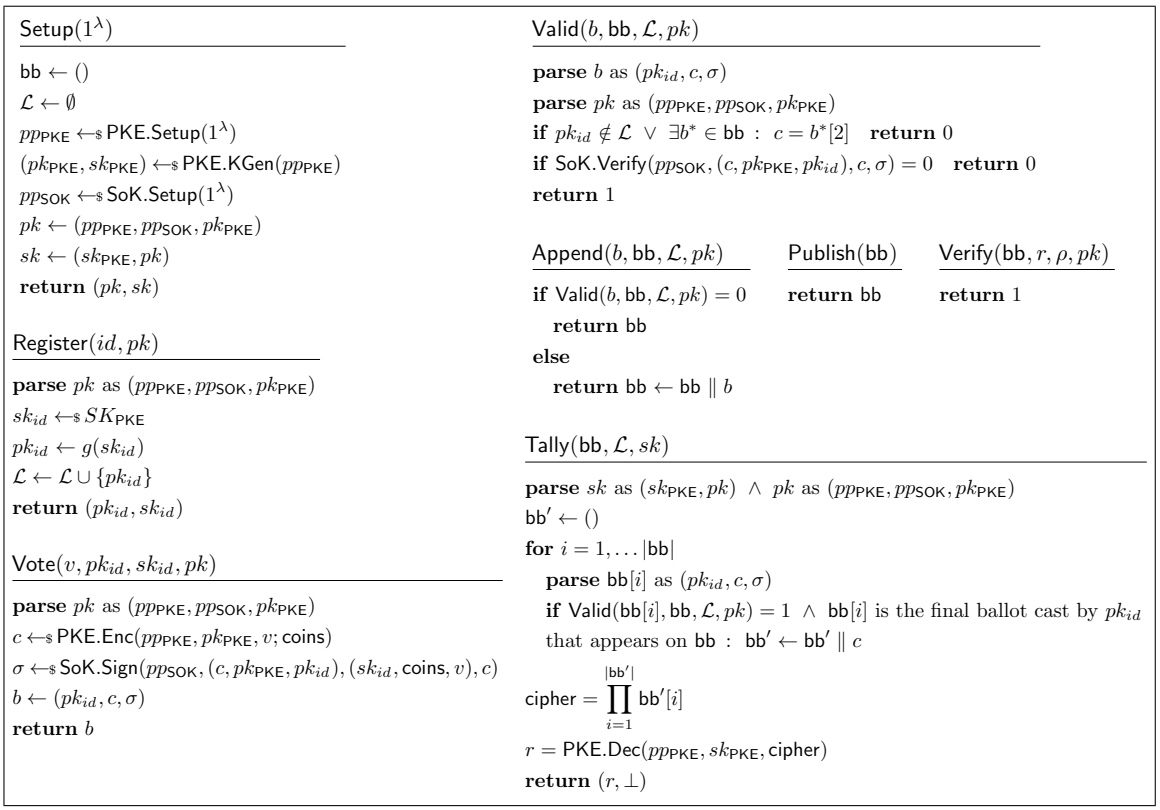

Figure 3.4: The e-voting scheme Γ_{mini} constructed from public-key encryption scheme PKE and signature of knowledge SOK.

guesses β in Game i.

Game 1 is identical to Game 0 except that, when generating the public parameters for SOK, we run algorithm SoK.SimSetup to generate a trapdoor τ in addition to public parameters pp_{SOK}. We define the modified experiment for Game 1 in Figure 3.5. As this change is superficial and does not change the view of adversary $\mathcal{A}$, we have,

$$\Big|\Pr[S_0] - \Pr[S_1]\Big| = 0.$$

Game 2 is identical to Game 1 except for a change to oracle $\mathcal{O}$vote. Rather than generating a real signature of knowledge, $\mathcal{O}$vote simulates the signature by running algorithm SoK.SimSign. We describe oracle $\mathcal{O}$vote for Game 2 in Figure 3.6.

We show that Game 1 and Game 2 are indistinguishable if signature of knowledge σ satisfies simulatability. We give a distinguishing algorithm in Figure 3.7. $\mathcal{D}_1$ aims to guess a bit β^* in the simulatability experiment for SOK and is given access to a signing oracle $\mathcal{O}$SoKSign, which returns a real (i.e., the output of algorithm SoK.Sign) or simulated signature (i.e.,

$\underline{\mathsf{Exp}^{\mathsf{BSec},\beta}_{\Gamma_{\mathsf{mini}},\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda)}$

$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow (); \ \mathbf{V_0} \leftarrow (); \ \mathbf{V_1} \leftarrow (); \ \mathsf{bb} \leftarrow ()$
$\mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{vote} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}})$
$(pp_{\mathsf{SOK}}, \tau) \leftarrow\!\!\$\ \mathsf{SoK.SimSetup}(1^\lambda)$
$pk \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}}, pk_{\mathsf{PKE}})$
$sk \leftarrow (sk_{\mathsf{PKE}}, pk)$
$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board}}(pk)$
$(r, \rho) \leftarrow\!\!\$\ \mathsf{Tally}(\mathsf{bb}, \mathcal{L}, sk)$
for $i \in \{0,1\}$: $V_i' \coloneqq \{\{v_i | v_i \in \mathcal{C} \wedge v_i = \mathbf{V_i}[id] \text{ for some } id \in \mathcal{V}\}\}$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{board}}(r, \rho, st)$
if $V_0' = V_1'$ **return** β'

Figure 3.5: The modified BSec experiment for Game 1 in the proof of Theorem 1.

$\underline{\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb},\tau,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathcal{Q}\mathsf{vote},\mathbf{pk},\mathbf{sk},\mathbf{V_0},\mathbf{V_1})}(id, v_0, v_1)}$

if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt} \ \vee \ v_0, v_1 \notin \mathcal{C}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}}, pk_{\mathsf{PKE}})$
$c \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_\beta; \mathsf{coins})$
$\sigma \leftarrow\!\!\$\ \mathsf{SoK.SimSign}(pp_{\mathsf{SOK}}, \tau, (c, pk_{\mathsf{PKE}}, \mathbf{pk}[id]), c)$
$b \leftarrow (\mathbf{pk}[id], c, \sigma)$
$\mathsf{bb} \leftarrow\!\!\$\ \mathsf{Append}(b, \mathsf{bb}, \mathcal{L}, pk)$
for $i \in \{0,1\}$: $\mathbf{V_i}[id] \leftarrow v_i$
$\mathcal{Q}\mathsf{vote} \leftarrow \mathcal{Q}\mathsf{vote} \cup \{(id, b)\}$
return $\top$

Figure 3.6: The oracle $\mathcal{O}\mathsf{vote}$ for Game 2 in the proof of Theorem 1.

the output of algorithm SoK.SimSign), depending on bit β^*. We show that, when $\beta^* = 0$, inputs to $\mathcal{A}$ are identical to Game 1 and, when $\beta^* = 1$, inputs to $\mathcal{A}$ are identical to Game 2. Note that Game 1 and Game 2 are identical with the exception of oracle $\mathcal{O}\mathsf{vote}$. If $\beta^* = 0$, $\mathcal{D}_1$ is input a real signature of knowledge as in Game 1. If $\beta^* = 1$, $\mathcal{D}_1$ is input a simulated signature as in Game 2. Therefore,

$$\big|\Pr[S_1] - \Pr[S_2]\big| \leq \mathsf{negl}_{\mathsf{SOK\text{-}sim}}$$

where $\mathsf{negl}_{\mathsf{SOK\text{-}sim}}$ is the advantage in breaking the simulatability property of SOK.

Game 3 is identical to Game 2 except for a change to the tallying process. Rather than decrypting the list of ballots bb', we run algorithm SOK.Extract to recover the plaintext message in each ballot, where the ballot is queried to oracle $\mathcal{O}\mathsf{cast}$. Then, the election result is computed by applying result function f to the outputs of algorithm

$\mathcal{D}_1^{\mathcal{O}\mathsf{SoKSign}}(pp_{\mathsf{SOK}})$

$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow (); \ \mathbf{V_0} \leftarrow (); \ \mathbf{V_1} \leftarrow (); \ \mathsf{bb} \leftarrow ()$
$\mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{vote} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}})$
$pk \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}}, pk_{\mathsf{PKE}})$
$sk \leftarrow (sk_{\mathsf{PKE}}, pk)$
$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board}}(pk)$
$(r, \rho) \leftarrow\!\!\$\ \mathsf{Tally}(\mathsf{bb}, \mathcal{L}, sk)$
for $i \in \{0,1\}$: $V_i' \coloneqq \{\{v_i | v_i \in \mathcal{C} \wedge v_i = \mathbf{V_i}[id] \text{ for some } id \in \mathcal{V}\}$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{board}}(r, \rho, st)$
if $V_0' = V_1'$ **return** β'

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb},\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathcal{Q}\mathsf{vote},\mathbf{pk},\mathbf{sk},\mathbf{V_0},\mathbf{V_1})}(id, v_0, v_1)$

if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt} \ \vee \ v_0, v_1 \notin \mathcal{C}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}}, pk_{\mathsf{PKE}})$
$c \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_\beta; \mathsf{coins})$
$\sigma \leftarrow\!\!\$\ \mathcal{O}\mathsf{SoKSign}((c, pk_{\mathsf{PKE}}, \mathbf{pk}[id]), (\mathsf{coins}, \mathbf{sk}[id], v_\beta), c)$
$b \leftarrow (\mathbf{pk}[id], c, \sigma)$
$\mathsf{bb} \leftarrow\!\!\$\ \mathsf{Append}(b, \mathsf{bb}, \mathcal{L}, pk)$
for $i \in \{0,1\}$: $\mathbf{V_i}[id] \leftarrow v_i$
$\mathcal{Q}\mathsf{vote} \leftarrow \mathcal{Q}\mathsf{vote} \cup \{(id, b)\}$
return $\top$

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{cast}/\mathcal{O}\mathsf{board}$

As in Game 0.

Figure 3.7: Distinguisher $\mathcal{D}_1$ that distinguishes Game 1 and Game 2 in the proof of Theorem 1.

SOK.Extract, and the voter/vote tuples (id, v_β) that are input to oracle $\mathcal{O}$vote. We describe the modified experiment in Figure 3.8. If the SOK scheme satisfies extractability then, with overwhelming probability, the witness output by algorithm SoK.Extract is identical to the input to algorithm PKE.Enc to generate ciphertext c in the ballot. As such, the view of the adversary is identical following this game hop, unless the SOK scheme does not satisfy extractability. Therefore,

$$|\Pr[S_2] - \Pr[S_3]| \leq k_2 \cdot \mathsf{negl}_{\mathsf{SOK\text{-}Ext}}$$

where $\mathsf{negl}_{\mathsf{SOK\text{-}Ext}}$ is the probability that the SOK scheme does not satisfy extractability and k_2 is the maximum number of queries to oracle $\mathcal{O}$cast.

We define Game 4 to be identical to Game 3 but with the following change to oracle $\mathcal{O}$vote. When $\beta = 0$, the oracle encrypts vote v_1 rather than vote v_0. Else, oracle $\mathcal{O}$vote proceeds as usual. We define the modified oracle $\mathcal{O}$vote in Figure 3.9. Moreover, the tally is computed as in Game 3 except that, for queries to oracle $\mathcal{O}$vote, the result function takes as input v_1 rather than v_β.

We show that Game 3 and Game 4 are indistinguishable if public-key encryption scheme PKE satisfies IND-CPA security. We give a distinguishing algorithm $\mathcal{D}_2$ in Figure 3.10 that guesses a bit β^* in the IND-CPA experiment for multiple encryptions against scheme PKE, and is given access to oracle $\mathcal{O}$enc. In this experiment, instead of receiving a single ciphertext from oracle $\mathcal{O}$enc, the adversary can make several queries of the form $\mathcal{O}\mathsf{enc}(m_0, m_1)$ and receives a ciphertext for each query. We show that, when $\beta^* = 0$, inputs

$\mathsf{Exp}^{\mathsf{BSec},\beta}_{\Gamma_{\mathsf{mini}},\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda)$

$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow (); \ \mathbf{V_0} \leftarrow (); \ \mathbf{V_1} \leftarrow (); \ \mathsf{bb} \leftarrow ()$
$\mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{vote} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}})$
$(pp_{\mathsf{SOK}}, \tau) \leftarrow\!\!\$\ \mathsf{SoK.SimSetup}(1^\lambda)$
$pk \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}}, pk_{\mathsf{PKE}})$
$sk \leftarrow (sk_{\mathsf{PKE}}, pk)$
$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board}}(pk)$
Run algorithm Tally to obtain $\mathsf{bb}' = ((pk_{id_1}, b_1), \ldots, (pk_{id_{|bb'|}}, b_{|\mathsf{bb}'|}))$.
$r = f((id_1, v^1), \ldots, (id_{|\mathsf{bb}'|}, v^{|\mathsf{bb}'|}))$
where $(\mathsf{coins}, \mathbf{sk}[id_i], v^i) \leftarrow\!\!\$\ \mathsf{SOK.Extract}(pp_{\mathsf{SOK}}, \tau, (b_i, pk_{\mathsf{PKE}}, \mathbf{pk}[id_i]), c_i)$ if $(id_i, b_i = (\mathbf{pk}[id_i], c_i, \sigma_i))$ is queried to $\mathcal{O}\mathsf{cast}$ and $v^i = v_\beta$ if $(id_i, b_i) \in \mathcal{Q}\mathsf{vote}$ for a query of (id_i, v_0, v_1) to $\mathcal{O}\mathsf{vote}$
$\rho \leftarrow \perp$
for $i \in \{0,1\}$: $V_i' \coloneqq \{\{v_i | v_i \in \mathcal{C} \wedge v_i = \mathbf{V_i}[id] \text{ for some } id \in \mathcal{V}\}\}$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{board}}(r, \rho, st)$
if $V_0' = V_1'$ **return** β'

Figure 3.8: The modified BSec experiment for Game 3 in the proof of Theorem 1.

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb},\tau,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathcal{Q}\mathsf{vote},\mathbf{pk},\mathbf{sk},\mathbf{V_0},\mathbf{V_1})}(id, v_0, v_1)$

if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt} \ \vee \ v_0, v_1 \notin \mathcal{C}$ **return** $\perp$
parse parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}}, pk_{\mathsf{PKE}})$
if $\beta = 0$
  $c \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_1; \mathsf{coins})$
  $\sigma \leftarrow\!\!\$\ \mathsf{SoK.SimSign}(pp_{\mathsf{SOK}}, \tau, (c, pk_{\mathsf{PKE}}, \mathbf{pk}[id]), c)$
  $b \leftarrow (\mathbf{pk}[id], c, \sigma)$
  $\mathsf{bb} \leftarrow\!\!\$\ \mathsf{Append}(b, \mathsf{bb}, \mathcal{L}, pk)$
  for $i \in \{0,1\}$: $\mathbf{V_i}[id] \leftarrow v_i$
  $\mathcal{Q}\mathsf{vote} \leftarrow \mathcal{Q}\mathsf{vote} \cup \{(id, b)\}$
  return $\top$
else Proceed as usual

Figure 3.9: The oracle $\mathcal{O}\mathsf{vote}$ for Game 4 in the proof of Theorem 1.

to $\mathcal{A}$ are identical to Game 3 and, when $\beta^* = 1$, inputs to $\mathcal{A}$ are identical to Game 4. Game 3 and Game 4 are identical with the exception of oracle $\mathcal{O}\mathsf{vote}$ when $\beta = 0$. When $\beta = 0$ and $\beta^* = 0$, $\mathcal{D}_2$ is input an encryption of v_0 and the inputs to $\mathcal{A}$ are identical to Game 3. If $\beta = 0$ and $\beta^* = 1$, $\mathcal{D}_2$ is input an encryption of v_1 and the inputs to $\mathcal{A}$ are identical to Game 4. Therefore,

$$\left| \Pr[S_3] - \Pr[S_4] \right| \leq \mathsf{negl}_{\mathsf{IND\text{-}CPA\text{-}mult}}$$

where $\mathsf{negl}_{\mathsf{IND\text{-}CPA\text{-}mult}}$ is the advantage in breaking the IND-CPA for multiple encryptions property of PKE. By a standard argument,

$$\big|\Pr[S_3] - \Pr[S_4]\big| \leq k_1 \cdot \mathsf{negl}_{\mathsf{IND\text{-}CPA}}$$

where $\mathsf{negl}_{\mathsf{IND\text{-}CPA}}$ is the advantage in breaking the IND-CPA property of PKE and k_1 is the maximum number of queries to oracle $\mathcal{O}$vote.

$\mathcal{D}_2^{\mathcal{O}\mathsf{enc}}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}})$

$\mathbf{pk} \leftarrow ()$; $\mathbf{sk} \leftarrow ()$; $\mathbf{V_0} \leftarrow ()$; $\mathbf{V_1} \leftarrow ()$; $\mathsf{bb} \leftarrow ()$
$\mathcal{Q}\mathsf{reg} \leftarrow \emptyset$; $\mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset$; $\mathcal{Q}\mathsf{vote} \leftarrow \emptyset$; $\mathcal{L} \leftarrow \emptyset$
$(pp_{\mathsf{SOK}}, \tau) \leftarrow\!\!\$\ \mathsf{SoK.SimSetup}(1^\lambda)$
$pk \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}}, pk_{\mathsf{PKE}})$
$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board}}(pk)$
Run algorithm Tally to obtain $\mathsf{bb}' = ((pk_{id_1}, b_1), \ldots, (pk_{id_{|\mathsf{bb}'|}}, b_{|\mathsf{bb}'|}))$.
$r = f((id_1, v^1), \ldots, (id_{|\mathsf{bb}'|}, v^{|\mathsf{bb}'|}))$
where $(\mathsf{coins}, \mathbf{sk}[id_i], v^i) \leftarrow\!\!\$\ \mathsf{SOK.Extract}(pp_{\mathsf{SOK}}, \tau, (b_i, pk_{\mathsf{PKE}}, \mathbf{pk}[id_i]), c_i)$ if $(id_i, b_i = (\mathbf{pk}[id_i], c_i, \sigma_i))$ is queried to $\mathcal{O}\mathsf{cast}$ and $v^i = v_1$ if $(id_i, b_i) \in \mathcal{Q}\mathsf{vote}$ for a query of (id_i, v_0, v_1) to $\mathcal{O}\mathsf{vote}$
$\rho \leftarrow \perp$
for $i \in \{0,1\}$: $V_i' \coloneqq \{\{v_i | v_i \in \mathcal{C} \wedge v_i = \mathbf{V_i}[id] \text{ for some } id \in \mathcal{V}\}\}$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{board}}(r, \rho, st)$
if $V_0' = V_1'$ **return** β'

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb},\tau,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathcal{Q}\mathsf{vote},\mathbf{pk},\mathbf{sk},\mathbf{V_0},\mathbf{V_1})}(id, v_0, v_1)$

if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt} \vee v_0, v_1 \notin \mathcal{C}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}}, pk_{\mathsf{PKE}})$
if $\beta = 0$
  $c \leftarrow\!\!\$\ \mathcal{O}\mathsf{enc}(v_0, v_1)$
  $\sigma \leftarrow\!\!\$\ \mathsf{SoK.SimSign}(pp_{\mathsf{SOK}}, \tau, (c, pk_{\mathsf{PKE}}, \mathbf{pk}[id]), c)$
  $b \leftarrow (\mathbf{pk}[id], c, \sigma)$
  $\mathsf{bb} \leftarrow\!\!\$\ \mathsf{Append}(b, \mathsf{bb}, \mathcal{L}, pk)$
  for $i \in \{0,1\}$: $\mathbf{V_i}[id] \leftarrow v_i$
  $\mathcal{Q}\mathsf{vote} \leftarrow \mathcal{Q}\mathsf{vote} \cup \{(id, b)\}$
  return $\top$
else Proceed as usual

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{cast}/\mathcal{O}\mathsf{board}$

As in Game 0.

Figure 3.10: Distinguisher $\mathcal{D}_2$ that distinguishes Game 3 and Game 4 in the proof of Theorem 1.

In Game 4, the inputs to $\mathcal{A}$ are identical for $\beta = 0$ and $\beta = 1$. In fact, the bulletin board and election result returned to $\mathcal{A}$ contains ballots submitted by $\mathcal{A}$ to oracle $\mathcal{O}$cast and ballots computed for v_1 that are submitted to oracle $\mathcal{O}$vote, regardless of the bit β chosen in the experiment. Therefore, $\Pr[S_4] = 1/2$. We now have that $|\Pr[S_0] - 1/2| \leq \mathsf{negl}_{\mathsf{SOK\text{-}sim}} + k_2 \cdot \mathsf{negl}_{\mathsf{SOK\text{-}Ext}} + k_1 \cdot \mathsf{negl}_{\mathsf{IND\text{-}CPA}}$ and conclude that the advantage of the adversary in Game 0 is negligible as required. □

3.4.4 Comparing BSec and Existing Definitions

We conclude this section by reflecting on the benefits and limitations of BSec in the context of the wider literature.

In comparison to the BPRIV approach, BSec (and other definitions in the same category [16, 18, 28, 29, 47]) limit the class of result functions that can be captured. However, BSec and [16, 18, 28, 29, 47] are more straightforward to extend to the malicious setting, as we shall see in Chapter 3.5. Moreover, BSec and [16, 18, 28, 29, 47] do not require the additional properties known as strong correctness and strong consistency that are defined for BPRIV. In particular, with respect to the attack captured by strong consistency, the balancing condition of BSec ensures that votes of honest voters are added to multisets V_0' and V_1', even if the ballot is rejected by algorithm Valid, yet the votes contained in these

multisets will not necessarily be included in the result. As a consequence, the adversary can output a ballot box such that the balancing condition is satisfied, and determine β from the result computed over bb. Similarly, BSec captures the attack prevented by strong correctness because, following a query to $\mathcal{O}$vote, if the first voter's ballot is rejected, the vote choices are still added to the multisets V_0' and V_1'. The election result computed over the viewed ballot box differs but the balancing condition is satisfied. Therefore, the adversary can trivially guess β and schemes that allow such attacks do not satisfy BSec.

Within the 'tally the viewed election' category, BSec is most similar to the definition presented in [28], which we refer to as IND-SEC. In particular, BSec uses the same balancing condition. BSec differs in that it models e-voting schemes with internal credentials, whereas IND-SEC is defined for e-voting schemes with external credentials. As we saw in Chapter 3.4.1, modelling internal credentials requires that the balancing condition must be carefully crafted to avoid trivial attacks. We conclude by discussing a further difference between BSec and IND-SEC: IND-SEC does not fully capture verifiable e-voting schemes.

Like BSec, IND-SEC requires that the multisets of left- and right-hand votes submitted on behalf of honest voters are equal. However, if these two multisets are not equal, IND-SEC returns the result and accompanying tallying proof computed over the ballot box corresponding to the IND-SEC experiment when $\beta = 0$. In [22], Cortier *et al.* prove that an e-voting scheme cannot simultaneously satisfy IND-SEC *and* tally uniqueness, a minimal property required to ensure verifiability of an e-voting scheme, as a result of the actions performed when the multisets are not equal. We refer the reader to [22] for full details of this result. Therefore, IND-SEC and verifiability are not compatible. To overcome this limitation, a fix to IND-SEC was put forth in [121]. The fix proposed that, if the multisets are not equal, the IND-SEC experiment returns the result computed over bb when $\beta = 0$, but does not return the tallying proof. However, a definition that does not return a tallying proof does not capture verifiable voting schemes. BSec, on the other hand, avoid the verifiability compatibility issue of IND-SEC by restricting the adversary and *requiring* that the two multisets are equal, i.e., the experiment returns 0 otherwise.

3.5 Extending Ballot Secrecy to the Malicious Setting

Corrupt election officials can break ballot secrecy. In particular, a malicious *ballot box manager* can 'stuff' the ballot box with ballots, which can lead to attacks against ballot secrecy [49]. Furthermore, a malicious *tallier* can potentially reveal the votes of all honest voters, for example, if ballots consist of a vote encrypted under the tallier's public key, such as in our construction Γ_{mini}. To mitigate this attack, many e-voting schemes distribute the role of the tallier [88, 108] and assume that a proportion of talliers are honest. In this section, we define mbbBS, a definition that extends BSec to the malicious ballot box setting, and dtBS, an extension of mbbBS in which the adversary can further corrupt a subset of talliers where the role of the tallier is distributed. In doing so, we provide definitions that allow a scheme designer to prove ballot secrecy in the event of an attacker that can corrupt the ballot box *and* a proportion of talliers.

3.5.1 Malicious Ballot Box Manager (mbbBS)

We extend BSec to mbbBS, defining an adversary that arbitrarily constructs the ballot box, obtaining ballots for honest voters that correspond to either a left or right vote submitted to an oracle $\mathcal{O}$vote. Hence, mbbBS models a corrupt ballot box manager, but considers all other election entities to be honest. We note that, in this setting, the adversary can only *statically* corrupt voters, a common restriction [44, 48]. This arises as a consequence of the balancing condition, which we elaborate on following the formal definition.

The mbbBS experiment. The mbbBS experiment for adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, formally defined in Figure 3.11 proceeds as follows. On input of election public key pk, adversary $\mathcal{A}_1$ can query oracle $\mathcal{O}$reg to register eligible voters. Oracle $\mathcal{O}$reg returns either the voter's public credential or the voter's credential pair, depending on whether the adversary requests the voter to be corrupt or not. Adversary $\mathcal{A}_2$ is provided with access to an oracle $\mathcal{O}$vote that, on input of a tuple (id, v_0, v_1), returns a ballot for v_β on behalf of voter id. Adversary $\mathcal{A}_2$ constructs a ballot box bb, which may include both honestly (i.e., the output of oracle $\mathcal{O}$vote) and maliciously generated ballots. Upon receiving the result r computed over bb and a proof of correct tallying ρ, $\mathcal{A}_3$ outputs a bit β'. If the balancing condition is satisfied, the experiment returns β. We observe that the adversary

does not require access to oracle $\mathcal{O}\mathsf{corrupt}$, as defined for BSec, because voters are statically corrupted through access to oracle $\mathcal{O}\mathsf{reg}$. Moreover, the adversary does not require access to an oracle $\mathcal{O}\mathsf{cast}$ because $\mathcal{A}_2$ constructs the ballot box, which includes ballots generated for corrupt voters.

Definition 49 (mbbBS). *An e-voting scheme Γ for a result function f, set of voters $\mathcal{V}$, and set of candidates $\mathcal{C}$ satisfies* mbbBS *if, for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there exists a negligible function* negl *such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{mbbBS},0}_{\Gamma,\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{mbbBS},1}_{\Gamma,\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{mbbBS},\beta}_{\Gamma,\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda)$ is the experiment defined in Figure 3.11 for $\beta \in \{0,1\}$.

$\mathsf{Exp}^{\mathsf{mbbBS},\beta}_{\Gamma,\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda)$

$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow ()$
$V \leftarrow \emptyset; \ \mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset; \ V_0 \leftarrow \{\{\}\}; \ V_1 \leftarrow \{\{\}\}$
$(pk, sk) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$st_1 \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
$(\mathsf{bb}, st_2) \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote}}(st_1)$
$(r, \rho) \leftarrow\!\!\$\ \mathsf{Tally}(\mathsf{bb}, \mathcal{L}, sk)$
for $i = 1, \ldots, |\mathsf{bb}|$
  if $\mathsf{bb}[i]$ appears in a tuple in set V and is the final tuple in set V that contains a voter public credential pk_{id}
    $V_0 \leftarrow V_0 \cup \{\{v_0\}\}; \ V_1 \leftarrow V_1 \cup \{\{v_1\}\}$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_3(r, \rho, st_2)$
if $V_0 = V_1$ **return** β'

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},V)}(id, v_0, v_1)$

if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt} \ \vee \ v_0, v_1 \notin \mathcal{C}$ **return** $\perp$
$b \leftarrow\!\!\$\ \mathsf{Vote}(v_\beta, \mathbf{pk}[id], \mathbf{sk}[id], pk)$
$V \leftarrow V \cup \{(\mathbf{pk}[id], v_0, v_1, b)\}$
return b

$\mathcal{O}\mathsf{reg}_{(pk,\mathcal{L},\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk})}(id, d)$

if $id \notin \mathcal{V} \ \vee \ id \in \mathcal{Q}\mathsf{reg}$ **return** $\perp$
$(\mathbf{pk}[id], \mathbf{sk}[id]) \leftarrow\!\!\$\ \mathsf{Register}(id, pk)$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathbf{pk}[id]\}$
$\mathcal{Q}\mathsf{reg} \leftarrow \mathcal{Q}\mathsf{reg} \cup \{id\}$
if $d = 0$ **return** $\mathbf{pk}[id]$
if $d = 1$
  $\mathcal{Q}\mathsf{corrupt} \leftarrow \mathcal{Q}\mathsf{corrupt} \cup \{id\}$
  return $(\mathbf{pk}[id], \mathbf{sk}[id])$
else return $\perp$

Figure 3.11: The malicious ballot box ballot secrecy experiment $\mathsf{Exp}^{\mathsf{mbbBS},\beta}_{\Gamma,\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda)$ for $\beta \in \{0,1\}$.

Our balancing condition. The mbbBS experiment maintains a list of queries to $\mathcal{O}\mathsf{vote}$ in a set V such that each entry in V consists of a tuple (pk_{id}, v_0, v_1, b). Then, if ballot b

appears on ballot box bb and the tuple contains the final ballot that appears on bb with respect to voter pk_{id}, the vote v_0 (resp., v_1) is added to a multiset V_0 (resp., V_1). In other words, multisets V_0 and V_1 contain only the final vote for every honest voter such that a corresponding ballot is appended to bb. As with BSec, our balancing condition models a last-vote-counts revote policy but can be modified to model a no-revote policy.

As a result of our balancing condition, mbbBS considers *static* corruption of voters. Indeed, if mbbBS allows *adaptive* corruption of voters, a trivial distinguishing attack is possible. To demonstrate this, we consider an e-voting scheme for a set of candidates $\mathcal{C} = \{0, 1\}$ with a last-vote-counts revote policy. We assume that the tally outputs a vector of size two that indicates the number of votes cast for each candidate. Let the adversary in the mbbBS experiment adaptively corrupt voters. That is, the adversary has access to oracles $\mathcal{O}\mathsf{reg}$ and $\mathcal{O}\mathsf{corrupt}$ as defined in Figure 3.3. An adversary in the mbbBS experiment can make the following queries: $b_1 \leftarrow\!\!\$\, \mathcal{O}\mathsf{vote}(pk_{id_1}, 0, 1)$; $b_2 \leftarrow \mathcal{O}\mathsf{vote}(pk_{id_2}, 1, 0)$ for pk_{id_1} and pk_{id_2} obtained via queries to $\mathcal{O}\mathsf{reg}$. The adversary can then corrupt pk_{id_1} via a query to $\mathcal{O}\mathsf{corrupt}$ and outputs a ballot box $\mathsf{bb} = (b_1, b_2, b_3)$ where b_3 is a ballot for '1' on behalf of pk_{id_1}. The balancing condition is satisfied but, if $\beta = 0$ (resp., $\beta = 1$), $r = (0, 2)$ (resp., $r = (1, 1)$), and the adversary can trivially distinguish the two views. Consequently, we restrict mbbBS to allow only static corruption of voters. Then, if the adversary wishes to corrupt pk_{id_1}, they cannot make queries to $\mathcal{O}\mathsf{vote}$ on behalf of pk_{id_1} and, in the example above, the balancing condition is not satisfied.

Satisfiability of mbbBS. We show that our e-voting construction Γ_{mini} (Figure 3.4) satisfies mbbBS under the same conditions that Γ_{mini} satisfies BSec. Indeed,the ballots in Γ_{mini} are *non-malleable*, which is required to satisfy mbbBS, a fact that we elaborate on following our formal result. We require that public-key encryption scheme PKE satisfies $\mathsf{IND\text{-}CPA}$ security and signature of knowledge SOK satisfies simulatability and extractability and obtain the result in Theorem 2.

Theorem 2. *E-voting scheme Γ_{mini} (Figure 3.4) satisfies mbbBS if public-key encryption scheme PKE satisfies $\mathsf{IND\text{-}CPA}$ security and signature of knowledge SOK satisfies simulatability and extractability.*

Proof. The proof of Theorem 2 is very similar to the proof that Γ_{mini} satisfies BSec (Theorem 1). That is, we let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ be an adversary in the mbbBS experiment

that makes at most k queries to oracle $\mathcal{O}\mathsf{vote}$. We proceed through a series of game hops that we show are indistinguishable to the adversary. In the final game, the view of the adversary will be identical for $\beta = 0$ and $\beta = 1$. We define Game 0 as the mbbBS experiment with β chosen randomly. Let S_i be the event that adversary $\mathcal{A}$ correctly guesses β in Game i.

Game 1 is identical to Game 0 except that, when generating the public parameters for SOK, we run algorithm $\mathsf{SoK.SimSetup}$ to generate a trapdoor τ in addition to public parameters pp_{SOK}. We define the modified experiment for Game 1 in Figure 3.12. As this change is superficial and does not change the view of adversary $\mathcal{A}$, we have,

$$\Big|\Pr[S_0] - \Pr[S_1]\Big| = 0.$$

$\mathsf{Exp}^{\mathsf{mbbBS},\beta}_{\Gamma_{\mathsf{mini}},\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda)$

$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow (); V \leftarrow \emptyset; \ \mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset; \ V_0 \leftarrow \{\{\}\}; \ V_1 \leftarrow \{\{\}\}$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}})$
$(pp_{\mathsf{SOK}}, \tau) \leftarrow\!\!\$\ \mathsf{SoK.SimSetup}(1^\lambda)$
$pk \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}}, pk_{\mathsf{PKE}})$
$sk \leftarrow (sk_{\mathsf{PKE}}, pk)$
$st_1 \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
$(\mathsf{bb}, st_2) \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote}}(st_1)$
$(r, \rho) \leftarrow\!\!\$\ \mathsf{Tally}(\mathsf{bb}, \mathcal{L}, sk)$
for $i = 1, \ldots, |\mathsf{bb}|$
  if $\mathsf{bb}[i]$ appears in a tuple in set V and is the final tuple in
  set V that contains a voter public credential pk_{id}
    $V_0 \leftarrow V_0 \cup \{\{v_0\}\}; \ V_1 \leftarrow V_1 \cup \{\{v_1\}\}$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_3(r, \rho, st_2)$
if $V_0 = V_1$ **return** β'

Figure 3.12: The modified mbbBS experiment for Game 1 in the proof of Theorem 2.

Game 2 is identical to Game 1 except for a change to oracle $\mathcal{O}\mathsf{vote}$. Rather than generating a real signature of knowledge, $\mathcal{O}\mathsf{vote}$ simulates the signature by running algorithm $\mathsf{SoK.SimSign}$. We describe oracle $\mathcal{O}\mathsf{vote}$ for Game 2 in Figure 3.13.

We show that Game 1 and Game 2 are indistinguishable if signature of knowledge σ satisfies simulatability. We give a distinguishing algorithm in Figure 3.14. $\mathcal{D}_1$ aims to guess a bit β^* in the simulatability experiment for SOK and is given access to a signing oracle $\mathcal{O}\mathsf{SoKSign}$, which returns a real (i.e., the output of algorithm $\mathsf{SoK.Sign}$) or simulated signature (i.e.,

$\underline{\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb},\tau,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},V)}(id, v_0, v_1)}$

if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt} \ \vee \ v_0, v_1 \notin \mathcal{C}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}}, pk_{\mathsf{PKE}})$
$c \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_\beta; \mathsf{coins})$
$\sigma \leftarrow\!\!\$\ \mathsf{SoK.SimSign}(pp_{\mathsf{SOK}}, \tau, (c, pk_{\mathsf{PKE}}, \mathbf{pk}[id]), c)$
$b \leftarrow (\mathbf{pk}[id], c, \sigma)$
$V \leftarrow V \cup \{(\mathbf{pk}[id], v_0, v_1, b)\}$
return b

Figure 3.13: The oracle $\mathcal{O}\mathsf{vote}$ for Game 2 in the proof of Theorem 2.

the output of algorithm SoK.SimSign), depending on bit β^*. We show that, when $\beta^* = 0$, inputs to $\mathcal{A}$ are identical to Game 1 and, when $\beta^* = 1$, inputs to $\mathcal{A}$ are identical to Game 2. Note that Game 1 and Game 2 are identical with the exception of oracle $\mathcal{O}\mathsf{vote}$. If $\beta^* = 0$, $\mathcal{D}_1$ is input a real signature of knowledge as in Game 1. If $\beta^* = 1$, $\mathcal{D}_1$ is input a simulated signature as in Game 2. Therefore,

$$\big|\Pr[S_1] - \Pr[S_2]\big| \leq \mathsf{negl}_{\mathsf{SOK\text{-}sim}}$$

where $\mathsf{negl}_{\mathsf{SOK\text{-}sim}}$ is the advantage in breaking the simulatability property of SOK.

$\underline{\mathcal{D}_1^{\mathcal{O}\mathsf{SoKSign}}(pp_{\mathsf{SOK}})}$

$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow (); V \leftarrow \emptyset; \ \mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset; \ V_0 \leftarrow \{\{\}\}; \ V_1 \leftarrow \{\{\}\}$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}})$
$pk \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}}, pk_{\mathsf{PKE}})$
$sk \leftarrow (sk_{\mathsf{PKE}}, pk)$
$st_1 \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
$(\mathsf{bb}, st_2) \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote}}(st_1)$
$(r, \rho) \leftarrow\!\!\$\ \mathsf{Tally}(\mathsf{bb}, \mathcal{L}, sk)$
for $i = 1, \ldots, |\mathsf{bb}|$
  if $\mathsf{bb}[i]$ appears in a tuple in set V and is the final tuple in set V that contains a voter public credential pk_{id}
    $V_0 \leftarrow V_0 \cup \{\{v_0\}\}; \ V_1 \leftarrow V_1 \cup \{\{v_1\}\}$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_3(r, \rho, st_2)$
if $V_0 = V_1$ **return** β'

$\underline{\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb},\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathcal{Q}\mathsf{vote},\mathbf{pk},\mathbf{sk},V)}(id, v_0, v_1)}$

if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt} \ \vee \ v_0, v_1 \notin \mathcal{C}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}}, pk_{\mathsf{PKE}})$
$c \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_\beta; \mathsf{coins})$
$\sigma \leftarrow\!\!\$\ \mathcal{O}\mathsf{SoKSign}((c, pk_{\mathsf{PKE}}, \mathbf{pk}[id]), (\mathsf{coins}, \mathbf{sk}[id], v_\beta), c)$
$b \leftarrow (\mathbf{pk}[id], c, \sigma)$
$V \leftarrow V \cup \{(\mathbf{pk}[id], v_0, v_1, b)\}$
return b

$\underline{\mathcal{O}\mathsf{reg}}$

As in Game 0.

Figure 3.14: Distinguisher $\mathcal{D}_1$ that distinguishes Game 1 and Game 2 in the proof of Theorem 2.

Game 3 is identical to Game 2 except for a change to the tallying process. Rather than decrypting the list of ballots bb', we run algorithm SOK.Extract to recover the plaintext message in each ballot, where the ballot is not queried to oracle $\mathcal{O}\mathsf{vote}$ (i.e., the ballot does not appear in a tuple in set V). Then, the election result is computed by applying result function f to the outputs of algorithm SOK.Extract, and the voter/vote tuples (id, v_β) that are input to oracle $\mathcal{O}\mathsf{vote}$. We describe the modified experiment in Figure 3.15. If the SOK scheme satisfies extractability then, with overwhelming probability, the witness output by algorithm SoK.Extract is identical to the input to algorithm PKE.Enc to generate ciphertext

$\mathsf{Exp}^{\mathsf{mbbBS},\beta}_{\Gamma_{\mathsf{mini}},\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda)$

$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow (); V \leftarrow \emptyset; \ \mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset; \ V_0 \leftarrow \{\{\}\}; \ V_1 \leftarrow \{\{\}\}$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}})$
$(pp_{\mathsf{SOK}}, \tau) \leftarrow\!\!\$\ \mathsf{SoK.SimSetup}(1^\lambda)$
$pk \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}}, pk_{\mathsf{PKE}})$
$sk \leftarrow (sk_{\mathsf{PKE}}, pk)$
$st_1 \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
$(\mathsf{bb}, st_2) \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote}}(st_1)$
Run algorithm Tally to obtain $\mathsf{bb}' = ((pk_{id_1}, b_1), \ldots, (pk_{id_{|bb'|}}, b_{|\mathsf{bb}'|}))$.
$r = f((id_1, v^1), \ldots, (id_{|\mathsf{bb}'|}, v^{|\mathsf{bb}'|}))$
where $v^i = v_\beta$ if $(\mathbf{pk}[id_i], v_0, v_1, b_i) \in V$
and $(\mathsf{coins}, \mathbf{sk}[id_i], v^i) \leftarrow\!\!\$\ \mathsf{SOK.Extract}(pp_{\mathsf{SOK}}, \tau, (b_i, pk_{\mathsf{PKE}}, \mathbf{pk}[id_i]), c_i)$ for $b_i = (\mathbf{pk}[id_i], c_i, \sigma_i)$ otherwise
$\rho \leftarrow \perp$
for $i = 1, \ldots, |\mathsf{bb}|$
  if $\mathsf{bb}[i]$ appears in a tuple in set V and is the final tuple in
  set V that contains a voter public credential pk_{id}
    $V_0 \leftarrow V_0 \cup \{\{v_0\}\}; \ V_1 \leftarrow V_1 \cup \{\{v_1\}\}$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_3(r, \rho, st_2)$
if $V_0 = V_1$ **return** β'

Figure 3.15: The modified mbbBS experiment for Game 3 in the proof of Theorem 2.

c in the ballot. As such, the view of the adversary is identical following this game hop, unless the SOK scheme does not satisfy extractability. Therefore,

$$|\Pr[S_2] - \Pr[S_3]| \leq k' \cdot \mathsf{negl}_{\mathsf{SOK\text{-}Ext}}$$

where $\mathsf{negl}_{\mathsf{SOK\text{-}Ext}}$ is the probability that the SOK scheme does not satisfy extractability and k' is the maximum number of ballots in ballot box bb' that are not the output of oracle $\mathcal{O}\mathsf{vote}$.

We define Game 4 to be identical to Game 3 but with the following change to oracle $\mathcal{O}\mathsf{vote}$. When $\beta = 0$, the oracle encrypts vote v_1 rather than vote v_0. Else, oracle $\mathcal{O}\mathsf{vote}$ proceeds as usual. We define the modified oracle $\mathcal{O}\mathsf{vote}$ in Figure 3.16. Moreover, the tally is computed as in Game 3 except that, for queries to oracle $\mathcal{O}\mathsf{vote}$, the result function takes as input v_1 rather than v_β.

We show that Game 3 and Game 4 are indistinguishable if public-key encryption scheme PKE satisfies IND-CPA security. We give a distinguishing algorithm $\mathcal{D}_2$ in Figure 3.17 that guesses a bit β^* in the IND-CPA experiment for multiple encryptions against scheme

$\underline{\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb},\tau,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},V)}(id, v_0, v_1)}$

if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt} \ \vee \ v_0, v_1 \notin \mathcal{C}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}}, pk_{\mathsf{PKE}})$
if $\beta = 0$
  $c \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_1; \mathsf{coins})$
  $\sigma \leftarrow\!\!\$\ \mathsf{SoK.SimSign}(pp_{\mathsf{SOK}}, \tau, (c, pk_{\mathsf{PKE}}, \mathbf{pk}[id]), c)$
  $b \leftarrow (\mathbf{pk}[id], c, \sigma)$
  $V \leftarrow V \cup \{(\mathbf{pk}[id], v_0, v_1, b)\}$
  return b
else Proceed as usual

Figure 3.16: The oracle $\mathcal{O}\mathsf{vote}$ for Game 4 in the proof of Theorem 2.

PKE, and is given access to oracle $\mathcal{O}\mathsf{enc}$. In this experiment, instead of receiving a single ciphertext from oracle $\mathcal{O}\mathsf{enc}$, the adversary can make several queries of the form $\mathcal{O}\mathsf{enc}(m_0, m_1)$ and receives a ciphertext for each query. We show that, when $\beta^* = 0$, inputs to $\mathcal{A}$ are identical to Game 3 and, when $\beta^* = 1$, inputs to $\mathcal{A}$ are identical to Game 4. Game 3 and Game 4 are identical with the exception of oracle $\mathcal{O}\mathsf{vote}$ when $\beta = 0$. When $\beta = 0$ and $\beta^* = 0$, $\mathcal{D}_2$ is input an encryption of v_0 and the inputs to $\mathcal{A}$ are identical to Game 3. If $\beta = 0$ and $\beta^* = 1$, $\mathcal{D}_2$ is input an encryption of v_1 and the inputs to $\mathcal{A}$ are identical to Game 4. Therefore,

$$\left| \Pr[S_3] - \Pr[S_4] \right| \leq \mathsf{negl}_{\mathsf{IND\text{-}CPA\text{-}mult}}$$

where $\mathsf{negl}_{\mathsf{IND\text{-}CPA\text{-}mult}}$ is the advantage in breaking the IND-CPA for multiple encryptions property of PKE. By a standard argument,

$$\left| \Pr[S_3] - \Pr[S_4] \right| \leq k \cdot \mathsf{negl}_{\mathsf{IND\text{-}CPA}}$$

where $\mathsf{negl}_{\mathsf{IND\text{-}CPA}}$ is the advantage in breaking the IND-CPA property of PKE and k is the maximum number of queries to oracle $\mathcal{O}\mathsf{vote}$.

In Game 4, the inputs to $\mathcal{A}$ are identical for $\beta = 0$ and $\beta = 1$. In fact, the output of oracle $\mathcal{O}\mathsf{vote}$ and the election result returned to $\mathcal{A}$ contains ballots submitted by $\mathcal{A}$ to the bulletin board and ballots computed for v_1 that are submitted to oracle $\mathcal{O}\mathsf{vote}$, regardless of the bit β chosen in the experiment. Therefore, $\Pr[S_4] = 1/2$. We now have that $|\Pr[S_0] - 1/2| \leq \mathsf{negl}_{\mathsf{SOK\text{-}sim}} + k' \cdot \mathsf{negl}_{\mathsf{SOK\text{-}Ext}} + k \cdot \mathsf{negl}_{\mathsf{IND\text{-}CPA}}$ and conclude that the advantage of the adversary in Game 0 is negligible as required. □

$\mathcal{D}_2^{\mathcal{O}\mathsf{enc}}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}})$

$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow (); V \leftarrow \emptyset; \ \mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset; \ V_0 \leftarrow \{\{\}\}; \ V_1 \leftarrow \{\{\}\}$
$(pp_{\mathsf{SOK}}, \tau) \leftarrow\!\!\$\ \mathsf{SoK.SimSetup}(1^\lambda)$
$pk \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}}, pk_{\mathsf{PKE}})$
$st_1 \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
Run algorithm Tally to obtain $\mathsf{bb}' = ((pk_{id_1}, b_1), \ldots, (pk_{id_{|\mathsf{bb}'|}}, b_{|\mathsf{bb}'|}))$.
$r = f((id_1, v^1), \ldots, (id_{|\mathsf{bb}'|}, v^{|\mathsf{bb}'|}))$
where $v^i = v_1$ if $(\mathbf{pk}[id_i], v_0, v_1, b_i) \in V$
and $(\mathsf{coins}, \mathbf{sk}[id_i], v^i) \leftarrow\!\!\$\ \mathsf{SOK.Extract}(pp_{\mathsf{SOK}}, \tau, (b_i, pk_{\mathsf{PKE}}, \mathbf{pk}[id_i]), c_i)$ for $b_i = (\mathbf{pk}[id_i], c_i, \sigma_i)$ otherwise
$\rho \leftarrow \perp$
for $i = 1, \ldots, |\mathsf{bb}|$
  if $\mathsf{bb}[i]$ appears in a tuple in set V and is the final tuple in
  set V that contains a voter public credential pk_{id}
    $V_0 \leftarrow V_0 \cup \{\{v_0\}\}; \ V_1 \leftarrow V_1 \cup \{\{v_1\}\}$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_3(r, \rho, st_2)$
if $V_0' = V_1'$ **return** β'

$\mathcal{O}\mathsf{vote}_{(pk, \mathcal{L}, \mathsf{bb}, \tau, \mathcal{Q}\mathsf{reg}, \mathcal{Q}\mathsf{corrupt}, \mathcal{Q}\mathsf{vote}, \mathbf{pk}, \mathbf{sk}, V)}(id, v_0, v_1)$

if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt} \ \vee \ v_0, v_1 \notin \mathcal{C}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}}, pk_{\mathsf{PKE}})$
if $\beta = 0$
  $c \leftarrow\!\!\$\ \mathcal{O}\mathsf{enc}(v_0, v_1)$
  $\sigma \leftarrow\!\!\$\ \mathsf{SoK.SimSign}(pp_{\mathsf{SOK}}, \tau, (c, pk_{\mathsf{PKE}}, \mathbf{pk}[id]), c)$
  $b \leftarrow (\mathbf{pk}[id], c, \sigma)$
  $V \leftarrow V \cup \{(\mathbf{pk}[id], v_0, v_1, b)\}$
  return b
else Proceed as usual

$\mathcal{O}\mathsf{reg}$

As in Game 0.

Figure 3.17: Distinguisher $\mathcal{D}_2$ that distinguishes Game 3 and Game 4 in the proof of Theorem 2.

Requirement for non-malleable ballots. Ballots must be non-malleable to satisfy mbbBS. Intuitively, suppose an attacker with control of the ballot box can transform ballots. In this case, they can append ballots to the ballot box that are meaningfully related to the vote of an honest voter such that the result reveals the honest voter's vote. We demonstrate this through the following example. Consider an e-voting scheme for a set of candidates $\mathcal{C} = \{0, 1\}$ with a last-vote-counts revote policy. We assume that the tally outputs a vector of size two that indicates the number of votes cast for each candidate. Let ballots be of the form $b = (pk_{id}, c, \rho, \sigma)$ where c is an encryption of a vote for a public-key encryption scheme PKE, ρ is proof of correct encryption generated via a non-interactive zero-knowledge proof scheme NIZK, and σ is a digital signature generated by a standard digital signature scheme SIG. As adversary in the mbbBS experiment can make the following queries: $b_1 \leftarrow\!\!\$\ \mathcal{O}\mathsf{vote}(pk_{id_1}, 0, 1)$; $b_2 \leftarrow\!\!\$\ \mathcal{O}\mathsf{vote}(pk_{id_2}, 1, 0)$; $b_3 \leftarrow\!\!\$\ \mathcal{O}\mathsf{vote}(pk_{id_3}, 0, 1)$ for pk_{id_1}, pk_{id_2} and pk_{id_3} obtained via queries to $\mathcal{O}\mathsf{reg}$. Then, the adversary parses b_3 as $(pk_{id_3}, c_3, \rho_3, \sigma_3)$ and generates a modified ballot $b_4 = (pk_{id_4}, c_3, \rho_3, \sigma_4)$ where pk_{id_4} is the public credential of corrupt voter id_4 and $\sigma_4 \leftarrow\!\!\$\ \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{id_4}, (c, \rho_3))$. The adversary outputs a ballot box $\mathsf{bb} = (b_1, b_2, b_4)$. The balancing condition is satisfied, yet, if $\beta = 0$, $r = (2, 1)$ and, if $\beta = 1$, $r = (1, 2)$. As such, the adversary can trivially guess β. This attack is possible because the ballot is malleable. A non-malleable public-key encryption scheme can be constructed from public-key encryption scheme PKE and non-interactive zero-knowledge proof system NIZK [27]. However, the addition of the digital signature results in a malleable ballot. Our construction Γ_{mini} avoids this by using a signature of knowledge in place of a separate non-interactive zero-knowledge proof and digital signature.

We recognise that there are e-voting schemes for which ballots are malleable, such as e-voting schemes that include a timestamp in the ballot. These schemes cannot satisfy

mbbBS. However, these schemes may intuitively satisfy ballot secrecy in the malicious ballot box setting. Despite this, we believe that non-malleable ballots are desirable. For example, if a ballot includes a malleable timestamp, an attacker with control of the ballot box can modify the timestamp, potentially ensuring that the election result does not include the ballot. Furthermore, ballots with malleable elements can be modified to ensure non-malleability. For instance, a ballot can include a signature of knowledge or proof of knowledge that is tied to the malleable part of the ballot, ensuring that the malleable part of the ballot cannot be modified without detection.

3.5.2 Distributed and Malicious Tallier (dtBS)

We now consider an extension of mbbBS for e-voting schemes with a distributed tallier, that is, we write tallier T as $\mathsf{T} = (\mathsf{T}_1, \ldots, \mathsf{T}_n)$. In this case, we consider an election private key that is distributed amongst n talliers such that $sk = (sk_1, \ldots, sk_n)$ and at least t shares are required to reconstruct sk where $t \leq n$. We note that, specifically, $t = n$ is possible. That is, all n shares are required to reconstruct the private key. We extend mbbBS to a definition dtBS that models a corrupt ballot box *and* a subset of corrupt talliers. In particular, we model an attacker that obtains the private key share of up to $t - 1$ talliers. As with mbbBS, we consider other election entities to be honest and only allow the static corruption of voters. dtBS is a preliminary exploration of ballot secrecy with a malicious ballot box and tallier. Therefore, we assume that all key shares are generated honestly. In other words, the attacker corrupts talliers *after* key generation and cannot generate key shares for corrupt talliers.

The dtBS experiment. We define the dtBS experiment parametrised by the number of talliers n and the number of shares required to reconstruct the election secret key t, as the mbbBS experiment, defined in Figure 3.11, but with the following modifications. In addition to statically corrupting a subset of voters, adversary $\mathcal{A}$ corrupts $t - 1$ talliers. That is, $\mathcal{A}$ submits $t - 1$ unique indices $\{i_1, \ldots, i_{t-1}\}$ and obtains the set of private key shares $\{sk_{i_1}, \ldots, sk_{i_{t-1}}\}$. Additionally, algorithm Tally takes as input t private key shares that include the $t - 1$ shares returned to $\mathcal{A}$. In all other respects, the dtBS experiment is identical to the mbbBS experiment.

Definition 50 (dtBS)**.** *An e-voting scheme* Γ *for result function* f*, set of voters* $\mathcal{V}$*, set*

of candidates $\mathcal{C}$, n talliers and threshold t, where election private key $sk = (sk_1, \ldots, sk_n)$, satisfies dtBS *if, for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there exists a negligible function* negl *such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{dtBS},0}_{\Gamma,\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda,t,n)=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{dtBS},1}_{\Gamma,\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda,t,n)=1\right]\right|\leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{dtBS},\beta}_{\Gamma,\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda,t,n)$ *is the experiment defined in Figure 3.18 for $\beta \in \{0,1\}$.*

$\underline{\mathsf{Exp}^{\mathsf{dtBS},\beta}_{\Gamma,\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda,t,n)}$

$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow (); \ V \leftarrow ()$
$V \leftarrow \emptyset; \ \mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ V_0 \leftarrow \{\{\}\}; \ V_1 \leftarrow \{\{\}\}$
$(pk, sk = (sk_1, \ldots, sk_n)) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(st_1, \{i_1, \ldots, i_{t-1}\}) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
$(\mathsf{bb}, st_2) \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote}}(st_1, \{sk_{i_1}, \ldots, sk_{i_{t-1}}\})$
$(r, \rho) \leftarrow\!\!\$\ \mathsf{Tally}(\mathsf{bb}, \mathcal{L}, (sk_{i_1}, \ldots, sk_{i_{t-1}}, sk_{j,(j\in\{1,\ldots,n\}\setminus\{i_1,\ldots,i_{t-1}\})}))$
for $i = 1, \ldots, |\mathsf{bb}|$
  if $\mathsf{bb}[i]$ appears in a tuple in set V and is the final tuple in set V that contains a voter public credential pk_{id}
    $V_0 \leftarrow V_0 \cup \{\{v_0\}\}; \ V_1 \leftarrow V_1 \cup \{\{v_1\}\}$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_3(r, \rho, st_2)$
if $V_0 = V_1$ **return** β'

$\underline{\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},V)}(id, v_0, v_1)}$

if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt} \ \vee \ v_0, v_1 \notin \mathcal{C}$ **return** $\perp$
$b \leftarrow\!\!\$\ \mathsf{Vote}(v_\beta, \mathbf{pk}[id], \mathbf{sk}[id], pk)$
$V \leftarrow V \cup \{(\mathbf{pk}[id], v_0, v_1, b)\}$
return b

$\underline{\mathcal{O}\mathsf{reg}_{(pk,\mathcal{L},\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk})}(id, d)}$

if $id \notin \mathcal{V} \ \vee \ id \in \mathcal{Q}\mathsf{reg}$ **return** $\perp$
$(\mathbf{pk}[id], \mathbf{sk}[id]) \leftarrow\!\!\$\ \mathsf{Register}(id, pk)$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathbf{pk}[id]\}$
$\mathcal{Q}\mathsf{reg} \leftarrow \mathcal{Q}\mathsf{reg} \cup \{id\}$
if $d = 0$ **return** $\mathbf{pk}[id]$
if $d = 1$
  $\mathcal{Q}\mathsf{corrupt} \leftarrow \mathcal{Q}\mathsf{corrupt} \cup \{id\}$
  return $(\mathbf{pk}[id], \mathbf{sk}[id])$
else return $\perp$

Figure 3.18: The distributed tallier ballot secrecy experiment $\mathsf{Exp}^{\mathsf{dtBS},\beta}_{\Gamma,\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda,t,n)$ for $\beta \in \{0,1\}$.

Satisfiability of dtBS**.** We briefly illustrate that dtBS is satisfiable. We consider a modification to Γ_{mini} (Figure 3.4) that uses a (t,n)-threshold public-key encryption scheme THR. We call our modified construction Γ^*_{mini}. Briefly, Γ^*_{mini} is identical to Γ_{mini} but is built upon a threshold public-key encryption scheme THR rather than a standard public-key encryption scheme PKE. Specifically, during key generation, rather than generating a single decryption key, n secret decryption keys are generated. Secondly, algorithm Tally requires

interaction between t talliers. In detail, any tallier can produce the homomorphic ciphertext cipher, and, then, t talliers each produce a partial decryption of cipher by running algorithm $\mathsf{THR.Dec}$. The final result r is computed by running algorithm $\mathsf{THR.Combine}$ on input of the t partial decryptions. Following the argument presented in Theorem 1, to satisfy dtBS, we require that THR satisfies $\mathsf{tIND\text{-}CPA}$ security and signature of knowledge SOK satisfies simulatability and extractability. We obtain the result in Theorem 3.

Theorem 3. *E-voting scheme* Γ^*_{mini} *satisfies* dtBS *if* (t, n)*-threshold public-key encryption scheme* THR *satisfies* $\mathsf{tIND\text{-}CPA}$ *security and signature of knowledge* SOK *satisfies simulatability and extractability.*

Proof. The proof of Theorem 3 is very similar to the proof that Γ_{mini} satisfies BSec (Theorem 1) and mbbBS (Theorem 2). That is, we let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ be an adversary in the dtBS experiment that makes at most k queries to oracle $\mathcal{O}\mathsf{vote}$. We proceed through a series of game hops that we show are indistinguishable to the adversary. In the final game, the view of the adversary will be identical for $\beta = 0$ and $\beta = 1$. We define Game 0 as the dtBS experiment with β chosen randomly. Let S_i be the event that adversary $\mathcal{A}$ correctly guesses β in Game i.

Game 1 is identical to Game 0 except that, when generating the public parameters for SOK, we run algorithm $\mathsf{SoK.SimSetup}$ to generate a trapdoor τ in addition to public parameters pp_{SOK}. We define the modified experiment for Game 1 in Figure 3.19. As this change is superficial and does not change the view of adversary $\mathcal{A}$, we have,

$$\Big|\Pr[S_0] - \Pr[S_1]\Big| = 0.$$

Game 2 is identical to Game 1 except for a change to oracle $\mathcal{O}\mathsf{vote}$. Rather than generating a real signature of knowledge, $\mathcal{O}\mathsf{vote}$ simulates the signature by running algorithm $\mathsf{SoK.SimSign}$. We describe oracle $\mathcal{O}\mathsf{vote}$ for Game 2 in Figure 3.20.

We show that Game 1 and Game 2 are indistinguishable if signature of knowledge σ satisfies simulatability. We give a distinguishing algorithm in Figure 3.21. $\mathcal{D}_1$ aims to guess a bit β^* in the simulatability experiment for SOK and is given access to a signing oracle $\mathcal{O}\mathsf{SoKSign}$, which returns a real (i.e., the output of algorithm $\mathsf{SoK.Sign}$) or simulated signature (i.e., the output of algorithm $\mathsf{SoK.SimSign}$), depending on bit β^*. We show that, when $\beta^* = 0$,

$\mathsf{Exp}^{\mathsf{dtBS},\beta}_{\Gamma^*_{\mathsf{mini}},\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda)$

$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow (); \ V \leftarrow \emptyset; \ \mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset; \ V_0 \leftarrow \{\{\}\}; \ V_1 \leftarrow \{\{\}\}$
$pp_{\mathsf{THR}} \leftarrow\!\!\$\ \mathsf{THR.Setup}(1^\lambda)$
$(pk_{\mathsf{THR}}, sk_{\mathsf{THR}} = (sk_1, \ldots, sk_n)) \leftarrow\!\!\$\ \mathsf{THR.KGen}(pp_{\mathsf{THR}})$
$(pp_{\mathsf{SOK}}, \tau) \leftarrow\!\!\$\ \mathsf{SoK.SimSetup}(1^\lambda)$
$pk \leftarrow (pp_{\mathsf{THR}}, pp_{\mathsf{SOK}}, pk_{\mathsf{THR}})$
$sk \leftarrow (sk_{\mathsf{THR}}, pk)$
$(st_1, \{i_1, \ldots, i_{t-1}\}) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
$(\mathsf{bb}, st_2) \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote}}(st_1, \{sk_{i_1}, \ldots, sk_{i_{t-1}}\})$
$(r, \rho) \leftarrow\!\!\$\ \mathsf{Tally}(\mathsf{bb}, \mathcal{L}, (sk_{i_1}, \ldots, sk_{i_{t-1}}, sk_{j,(j\in\{1,\ldots,n\}\setminus\{i_1,\ldots,i_{t-1}\})}))$
for $i = 1, \ldots, |\mathsf{bb}|$
  if $\mathsf{bb}[i]$ appears in a tuple in set V and is the final tuple in set V that contains a voter public credential pk_{id}
    $V_0 \leftarrow V_0 \cup \{\{v_0\}\}; \ V_1 \leftarrow V_1 \cup \{\{v_1\}\}$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_3(r, \rho, st_2)$
if $V_0 = V_1$ **return** β'

Figure 3.19: The modified dtBS experiment for Game 1 in the proof of Theorem 3.

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb},\tau,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},V)}(id, v_0, v_1)$

if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt} \ \vee \ v_0, v_1 \notin \mathcal{C}$ **return** $\perp$
parse pk as $(pp_{\mathsf{THR}}, pp_{\mathsf{SOK}}, pk_{\mathsf{THR}})$
$c \leftarrow\!\!\$\ \mathsf{THR.Enc}(pp_{\mathsf{THR}}, pk_{\mathsf{THR}}, v_\beta; \mathsf{coins})$
$\sigma \leftarrow\!\!\$\ \mathsf{SoK.SimSign}(pp_{\mathsf{SOK}}, \tau, (c, pk_{\mathsf{THR}}, \mathbf{pk}[id]), c)$
$b \leftarrow (\mathbf{pk}[id], c, \sigma)$
$V \leftarrow V \cup \{(\mathbf{pk}[id], v_0, v_1, b)\}$
return b

Figure 3.20: The oracle $\mathcal{O}\mathsf{vote}$ for Game 2 in the proof of Theorem 3.

inputs to $\mathcal{A}$ are identical to Game 1 and, when $\beta^* = 1$, inputs to $\mathcal{A}$ are identical to Game 2. Note that Game 1 and Game 2 are identical with the exception of oracle $\mathcal{O}\mathsf{vote}$. If $\beta^* = 0$, $\mathcal{D}_1$ is input a real signature of knowledge as in Game 1. If $\beta^* = 1$, $\mathcal{D}_1$ is input a simulated signature as in Game 2. Therefore,

$$\big|\Pr[S_1] - \Pr[S_2]\big| \leq \mathsf{negl}_{\mathsf{SOK\text{-}sim}}$$

where $\mathsf{negl}_{\mathsf{SOK\text{-}sim}}$ is the advantage in breaking the simulatability property of SOK.

Game 3 is identical to Game 2 except for a change to the tallying process. Rather than decrypting the list of ballots bb', we run algorithm SOK.Extract to recover the plaintext message in each ballot, where the ballot is not queried to oracle $\mathcal{O}\mathsf{vote}$ (i.e., the ballot does not appear in a tuple in set V). Then, the election result is computed by applying result

$\mathcal{D}_1^{\mathcal{O}\mathsf{SoKSign}}(pp_{\mathsf{SOK}})$

$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow (); \ V \leftarrow \emptyset; \ \mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset; \ V_0 \leftarrow \{\{\}\}; \ V_1 \leftarrow \{\{\}\}$
$pp_{\mathsf{THR}} \leftarrow\!\!\$\ \mathsf{THR.Setup}(1^\lambda)$
$(pk_{\mathsf{THR}}, sk_{\mathsf{THR}} = (sk_1, \ldots, sk_n)) \leftarrow\!\!\$\ \mathsf{THR.KGen}(pp_{\mathsf{THR}})$
$pk \leftarrow (pp_{\mathsf{THR}}, pp_{\mathsf{SOK}}, pk_{\mathsf{THR}})$
$sk \leftarrow (sk_{\mathsf{THR}}, pk)$
$(st_1, \{i_1, \ldots, i_{t-1}\}) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
$(\mathsf{bb}, st_2) \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote}}(st_1, \{sk_{i_1}, \ldots, sk_{i_{t-1}}\})$
$(r, \rho) \leftarrow\!\!\$\ \mathsf{Tally}(\mathsf{bb}, \mathcal{L}, (sk_{i_1}, \ldots, sk_{i_{t-1}}, sk_{j,(j\in\{1,\ldots,n\}\setminus\{i_1,\ldots,i_{t-1}\})}))$
for $i = 1, \ldots, |\mathsf{bb}|$
  if $\mathsf{bb}[i]$ appears in a tuple in set V and is the final tuple in set V that contains a voter public credential pk_{id}
    $V_0 \leftarrow V_0 \cup \{\{v_0\}\}; \ V_1 \leftarrow V_1 \cup \{\{v_1\}\}$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_3(r, \rho, st_2)$
if $V_0 = V_1$ **return** β'

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb},\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathcal{Q}\mathsf{vote},\mathbf{pk},\mathbf{sk},V)}(id, v_0, v_1)$

if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt} \ \vee \ v_0, v_1 \notin \mathcal{C}$ **return** $\perp$
parse pk as $(pp_{\mathsf{THR}}, pp_{\mathsf{SOK}}, pk_{\mathsf{THR}})$
$c \leftarrow\!\!\$\ \mathsf{THR.Enc}(pp_{\mathsf{THR}}, pk_{\mathsf{THR}}, v_\beta; \mathsf{coins})$
$\sigma \leftarrow\!\!\$\ \mathcal{O}\mathsf{SoKSign}((c, pk_{\mathsf{THR}}, \mathbf{pk}[id]), (\mathsf{coins}, \mathbf{sk}[id], v_\beta), c)$
$b \leftarrow (\mathbf{pk}[id], c, \sigma)$
$V \leftarrow V \cup \{(\mathbf{pk}[id], v_0, v_1, b)\}$
return b

$\mathcal{O}\mathsf{reg}$

As in Game 0.

Figure 3.21: Distinguisher $\mathcal{D}_1$ that distinguishes Game 1 and Game 2 in the proof of Theorem 3.

function f to the outputs of algorithm SOK.Extract, and the voter/vote tuples (id, v_β) that are input to oracle $\mathcal{O}$vote. We describe the modified experiment in Figure 3.22. If the SOK scheme satisfies extractability then, with overwhelming probability, the witness output by algorithm SoK.Extract is identical to the input to algorithm THR.Enc to generate ciphertext c in the ballot. As such, the view of the adversary is identical following this game hop, unless the SOK scheme does not satisfy extractability. Therefore,

$$|\Pr[S_2] - \Pr[S_3]| \leq k' \cdot \mathsf{negl}_{\mathsf{SOK\text{-}Ext}}$$

where $\mathsf{negl}_{\mathsf{SOK\text{-}Ext}}$ is the probability that the SOK scheme does not satisfy extractability and k' is the maximum number of ballots in ballot box bb' that are not the output of oracle $\mathcal{O}$vote.

We define Game 4 to be identical to Game 3 but with the following change to oracle $\mathcal{O}$vote. When $\beta = 0$, the oracle encrypts vote v_1 rather than vote v_0. Else, oracle $\mathcal{O}$vote proceeds as usual. We define the modified oracle $\mathcal{O}$vote in Figure 3.23. Moreover, the tally is computed as in Game 3 except that, for queries to oracle $\mathcal{O}$vote, the result function takes as input v_1 rather than v_β.

We show that Game 3 and Game 4 are indistinguishable if threshold public-key encryption scheme THR satisfies tIND-CPA security. We give a distinguishing algorithm $\mathcal{D}_2 = (\mathcal{D}_{2,1}, \mathcal{D}_{2,2})$ in Figure 3.24 that guesses a bit β^* in the tIND-CPA experiment for multiple encryptions against scheme THR, and is given access to oracle $\mathcal{O}$enc. In this experiment, instead of receiving a single ciphertext from oracle $\mathcal{O}$enc, the adversary can

$\mathsf{Exp}^{\mathsf{dtBS},\beta}_{\Gamma^*_{\mathsf{mini}},\mathcal{A},\mathcal{V},\mathcal{C}}(\lambda)$

$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow (); V \leftarrow \emptyset; \ \mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset; \ V_0 \leftarrow \{\{\}\}; \ V_1 \leftarrow \{\{\}\}$
$pp_{\mathsf{THR}} \leftarrow\!\!\$ \ \mathsf{THR.Setup}(1^\lambda)$
$(pk_{\mathsf{THR}}, sk_{\mathsf{THR}} = (sk_1, \ldots, sk_n)) \leftarrow\!\!\$ \ \mathsf{THR.KGen}(pp_{\mathsf{THR}})$
$(pp_{\mathsf{SOK}}, \tau) \leftarrow\!\!\$ \ \mathsf{SoK.SimSetup}(1^\lambda)$
$pk \leftarrow (pp_{\mathsf{THR}}, pp_{\mathsf{SOK}}, pk_{\mathsf{THR}})$
$sk \leftarrow (sk_{\mathsf{THR}}, pk)$
$(st_1, \{i_1, \ldots, i_{t-1}\}) \leftarrow\!\!\$ \ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
$(\mathsf{bb}, st_2) \leftarrow\!\!\$ \ \mathcal{A}_2^{\mathcal{O}\mathsf{vote}}(st_1, \{sk_{i_1}, \ldots, sk_{i_{t-1}}\})$
Run algorithm Tally to obtain $\mathsf{bb}' = ((pk_{id_1}, b_1), \ldots, (pk_{id_{|bb'|}}, b_{|\mathsf{bb}'|}))$.
$r = f((id_1, v^1), \ldots, (id_{|\mathsf{bb}'|}, v^{|\mathsf{bb}'|}))$
where $v^i = v_\beta$ if $(\mathbf{pk}[id_i], v_0, v_1, b_i) \in V$
and $(\mathsf{coins}, \mathbf{sk}[id_i], v^i) \leftarrow\!\!\$ \ \mathsf{SOK.Extract}(pp_{\mathsf{SOK}}, \tau, (b_i, pk_{\mathsf{THR}}, \mathbf{pk}[id_i]), c_i)$ for $b_i = (\mathbf{pk}[id_i], c_i, \sigma_i)$ otherwise
$\rho \leftarrow \perp$
for $i = 1, \ldots, |\mathsf{bb}|$
  if $\mathsf{bb}[i]$ appears in a tuple in set V and is the final tuple in
  set V that contains a voter public credential pk_{id}
    $V_0 \leftarrow V_0 \cup \{\{v_0\}\}; \ V_1 \leftarrow V_1 \cup \{\{v_1\}\}$
$\beta' \leftarrow\!\!\$ \ \mathcal{A}_3(r, \rho, st_2)$
if $V_0 = V_1$ **return** β'

Figure 3.22: The modified dtBS experiment for Game 3 in the proof of Theorem 3.

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb},\tau,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},V)}(id, v_0, v_1)$

if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt} \ \vee \ v_0, v_1 \notin \mathcal{C}$ **return** $\perp$
parse pk as $(pp_{\mathsf{THR}}, pp_{\mathsf{SOK}}, pk_{\mathsf{THR}})$
if $\beta = 0$
  $c \leftarrow\!\!\$ \ \mathsf{THR.Enc}(pp_{\mathsf{THR}}, pk_{\mathsf{THR}}, v_1; \mathsf{coins})$
  $\sigma \leftarrow\!\!\$ \ \mathsf{SoK.SimSign}(pp_{\mathsf{SOK}}, \tau, (c, pk_{\mathsf{THR}}, \mathbf{pk}[id]), c)$
  $b \leftarrow (\mathbf{pk}[id], c, \sigma)$
  $V \leftarrow V \cup \{(\mathbf{pk}[id], v_0, v_1, b)\}$
  return b
else Proceed as usual

Figure 3.23: The oracle $\mathcal{O}\mathsf{vote}$ for Game 4 in the proof of Theorem 3.

make several queries of the form $\mathcal{O}\mathsf{enc}(m_0, m_1)$ and receives a ciphertext for each query. We show that, when $\beta^* = 0$, inputs to $\mathcal{A}$ are identical to Game 3 and, when $\beta^* = 1$, inputs to $\mathcal{A}$ are identical to Game 4. Game 3 and Game 4 are identical with the exception of oracle $\mathcal{O}\mathsf{vote}$ when $\beta = 0$. When $\beta = 0$ and $\beta^* = 0$, $\mathcal{D}_2$ is input an encryption of v_0 and the inputs to $\mathcal{A}$ are identical to Game 3. If $\beta = 0$ and $\beta^* = 1$, $\mathcal{D}_2$ is input an encryption of v_1 and the inputs to $\mathcal{A}$ are identical to Game 4. Therefore,

$$\big|\Pr[S_3] - \Pr[S_4]\big| \leq \mathsf{negl}_{\mathsf{tIND\text{-}CPA\text{-}mult}}$$

where $\mathsf{negl}_{\mathsf{tIND\text{-}CPA\text{-}mult}}$ is the advantage in breaking the $\mathsf{tIND\text{-}CPA}$ for multiple encryptions

property of THR. By a standard argument,

$$\left|\Pr[S_3] - \Pr[S_4]\right| \leq k \cdot \mathsf{negl}_{\mathsf{tIND\text{-}CPA}}$$

where $\mathsf{negl}_{\mathsf{tIND\text{-}CPA}}$ is the advantage in breaking the tIND-CPA property of THR and k is the maximum number of queries to oracle $\mathcal{O}\mathsf{vote}$.

$\mathcal{D}_{2,1}(pp_{\mathsf{THR}}, pk_{\mathsf{THR}})$

$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow (); V \leftarrow \emptyset; \ \mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset; \ V_0 \leftarrow \{\{\}\}; \ V_1 \leftarrow \{\{\}\}$
$(pp_{\mathsf{SOK}}, \tau) \leftarrow\!\!\$\ \mathsf{SoK.SimSetup}(1^\lambda)$
$pk \leftarrow (pp_{\mathsf{THR}}, pp_{\mathsf{SOK}}, pk_{\mathsf{THR}})$
$(st_1, \{i_1, \ldots, i_{t-1}\}) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
return $(st_1, \{i_1, \ldots, i_{t-1}\})$

$\mathcal{D}_{2,2}^{\mathcal{O}\mathsf{enc}}(st_1, \{sk_{\mathsf{THR},i_1}, \ldots, sk_{\mathsf{THR},i_{t-1}}\})$

$(\mathsf{bb}, st_2) \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote}}(st_1, \{sk_{\mathsf{THR},i_1}, \ldots, sk_{\mathsf{THR},i_{t-1}}\})$
Run algorithm Tally to obtain $\mathsf{bb}' = ((pk_{id_1}, b_1), \ldots, (pk_{id_{|\mathsf{bb}'|}}, b_{|\mathsf{bb}'|}))$.
$r = f((id_1, v^1), \ldots, (id_{|\mathsf{bb}'|}, v^{|\mathsf{bb}'|}))$ where $v^i = v_1$ if $(\mathbf{pk}[id_i], v_0, v_1, b_i) \in V$
and $(\mathsf{coins}, \mathbf{sk}[id_i], v^i) \leftarrow\!\!\$\ \mathsf{SOK.Extract}(pp_{\mathsf{SOK}}, \tau, (b_i, pk_{\mathsf{THR}}, \mathbf{pk}[id_i]), c_i)$ for $b_i = (\mathbf{pk}[id_i], c_i, \sigma_i)$ otherwise
$\rho \leftarrow \perp$
for $i = 1, \ldots, |\mathsf{bb}|$
  if $\mathsf{bb}[i]$ appears in a tuple in set V and is the final tuple in
  set V that contains a voter public credential pk_{id}
    $V_0 \leftarrow V_0 \cup \{\{v_0\}\}; \ V_1 \leftarrow V_1 \cup \{\{v_1\}\}$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_3(r, \rho, st_2)$
if $V_0' = V_1'$ **return** β'

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb},\tau,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathcal{Q}\mathsf{vote},\mathbf{pk},\mathbf{sk},V)}(id, v_0, v_1)$

if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt} \ \vee \ v_0, v_1 \notin \mathcal{C}$ **return** $\perp$
parse pk as $(pp_{\mathsf{THR}}, pp_{\mathsf{SOK}}, pk_{\mathsf{THR}})$
if $\beta = 0$
  $c \leftarrow\!\!\$\ \mathcal{O}\mathsf{enc}(v_0, v_1)$
  $\sigma \leftarrow\!\!\$\ \mathsf{SoK.SimSign}(pp_{\mathsf{SOK}}, \tau, (c, pk_{\mathsf{THR}}, \mathbf{pk}[id]), c)$
  $b \leftarrow (\mathbf{pk}[id], c, \sigma)$
  $V \leftarrow V \cup \{(\mathbf{pk}[id], v_0, v_1, b)\}$
  return b
else Proceed as usual

$\mathcal{O}\mathsf{reg}$

As in Game 0.

Figure 3.24: Distinguisher $\mathcal{D}_2$ that distinguishes Game 3 and Game 4 in the proof of Theorem 3.

In Game 4, the inputs to $\mathcal{A}$ are identical for $\beta = 0$ and $\beta = 1$. In fact, the output of oracle $\mathcal{O}\mathsf{vote}$ and the election result returned to $\mathcal{A}$ contains ballots submitted by $\mathcal{A}$ to the bulletin board and ballots computed for v_1 that are submitted to oracle $\mathcal{O}\mathsf{vote}$, regardless of the bit β chosen in the experiment. Therefore, $\Pr[S_4] = 1/2$. We now have that $|\Pr[S_0] - 1/2| \leq \mathsf{negl}_{\mathsf{SOK\text{-}sim}} + k' \cdot \mathsf{negl}_{\mathsf{SOK\text{-}Ext}} + k \cdot \mathsf{negl}_{\mathsf{tIND\text{-}CPA}}$ and conclude that the advantage of the adversary in Game 0 is negligible as required. □

3.6 Concluding Remarks

Definitions of ballot secrecy in the honest model are well-studied. BPRIV, in particular, has received a lot of attention and is often regarded as the *de facto* definition of ballot secrecy. In contrast, the ballot secrecy definitions introduced in this paper follow the approach of [18]. Consequently, our definitions are not affected by the limitations of BPRIV, namely, the need for additional properties in order to prove security of an e-voting scheme, and the difficulties associated with extendibility. In fact, our definitions inherit the benefits of [18]:

our definitions are intuitive, based on long-established techniques for indistinguishability experiments, and are well-suited to extensions into the malicious setting. Moreover, our definitions differ from existing definitions that also follow the approach of [18]. In particular, our definitions model e-voting schemes with a registration phase and, in comparison to existing definitions, BSec captures *adaptive* voter corruption in an e-voting scheme with a registration phase, whilst also modelling revoting policies. Moreover, though we do restrict the set of result functions that can be considered, we do not require a partial tally assumption. We believe that, to model realistic attack scenarios, the way forward is ballot secrecy definitions that model corrupted election officials. Our definitions mbbBS and dtBS model such attack scenarios and provide a spring-board for future research in this direction.

Chapter 4

Receipt-Freeness for E-Voting

Contents

In this chapter, we analyse three definitions of receipt-freeness, uncovering weaknesses in two of the definitions and limitations in all three. We also show that each definition captures a different attacker model and examine those differences. We then present new definitions of receipt-freeness. In particular, we capture two variants of receipt-freeness, ensuring that we define receipt-freeness for a wide class of e-voting schemes. We analyse our new definitions, proving that one variant of receipt-freeness is stronger than the other. We conclude by comparing our new definitions with existing definitions from the literature. Part of the work in this chapter appears in [66], which is joint work with Elizabeth A. Quaglia and Ben Smyth.

4.1 Introduction

A potential problem in elections is that of vote buying, whereby, during an election, an attacker may bribe a voter to vote for a particular candidate. Vote buying is an issue in real elections. In fact, Transparency International[1], an organisation that works to end global corruption, reported that, in some countries, 28% of voters were offered a bribe during an election [128]. In the event that vote buying is a concern, a voting system should be secure against vote buying attacks. The e-voting literature has addressed the threat of vote buying attacks through the introduction of the *receipt-freeness* property. Benaloh and Tuinstra first defined receipt-freeness as the property that "no voter should be able to convince any other participant of the value of its vote" [17]. Initially known as uncoercibility, this property has since been rebranded as receipt-freeness, a term that was first used by Killian and Sako in [116]. Receipt-freeness is now widely understood to capture the notion that a voter cannot prove their vote (see, for example, [7, 32, 35, 38, 41, 52, 53, 80, 81, 86, 87, 88, 91, 96, 101, 104, 116, 123]). Thus, intuitively, a receipt-free e-voting scheme protects vote buying. Moreover, receipt-freeness is widely understood to be stronger than ballot secrecy [51]. Indeed, if a voter cannot prove their vote, it is certainly the case that the voter's vote is secret.

Receipt-freeness has been captured in several formal receipt-freeness definitions in the literature. The early formal definitions, with the exception of Moran and Naor's simulation-based definition [101], are formulated in the symbolic model, for example, [7, 32, 52, 58, 86, 87]. These definitions use a variety of logical languages to capture the intuition of receipt-freeness. More recently, there has been a movement towards definitions that describe indistinguishability experiments that capture receipt-freeness. This approach has possibly been driven by the simplicity of proof techniques in the model, and possibly because rigorous definitions of ballot secrecy that can be extended to receipt-freeness exist in this model (cf. Chapter 3). To the best of our knowledge, there is no examination that tests the rigour of receipt-freeness definitions, and we believe it is useful to revisit existing definitions in the literature and perform a critical analysis. To this end, in this chapter, we analyse three definitions of receipt-freeness that are formulated as indistinguishability experiments. We discover weaknesses and limitations in these definitions and then present new definitions of receipt-freeness that address these issues.

[1] `https://www.transparency.org/en/`

4.1.1 Our Contributions

In this chapter, we analyse definitions of receipt-freeness from the literature and discuss the attacker model of each definition. Then, we present new definitions of receipt-freeness, and compare our new definitions with existing work. We summarise the contributions of this chapter here.

Analysing existing definitions. We cast three definitions of receipt-freeness into the syntax of an e-voting scheme with internal voter registration (Definition 43) that we defined in Chapter 3. We consider the following three definitions: a receipt-freeness definition by Kiayias *et al.*, which we call KZZ [91]; one by Chaidos *et al.*, which we call CCFG [38]; and one by Bernhard *et al.* for schemes that use deniable vote updating, a process that allows a voter to change their vote without detection, which we call BKV [24, 25]. We analyse each definition, uncovering weaknesses and limitations. We categorise these weaknesses and limitations as soundness issues and completeness issues. Soundness issues occur when a definition is satisfied by a scheme, but the scheme is not intuitively receipt-free. Conversely, completeness issues arise when a scheme is declared not to satisfy a receipt-freeness definition, despite being intuitively receipt-free.

We find that KZZ and CCFG are not sound. The soundness issue in KZZ arises because the definition is satisfied by schemes that reveal how voters vote when not all voters vote. An issue arises in CCFG because Chaidos *et al.* do not consider *strong consistency* or *strong correctness*, properties that are defined to accompany ballot secrecy definition BPRIV [22], upon which CCFG is based, and are used to detect some attacks against ballot secrecy. All three definitions are incomplete because some schemes are out of scope. Schemes that count votes in some particular ways and others that allow voters to submit multiple ballots are out of scope of KZZ. We prove that neither KZZ nor CCFG is satisfiable by JCJ [88], an e-voting scheme that is intuitively coercion-resistant. Moreover, CCFG adopts an adaptive voter corruption strategy that can be leveraged by an adversary to trivially succeed in the CCFG experiment, even if the scheme in question is intuitively receipt-free. Finally, BKV limits the class of schemes considered to those that use deniable vote updating.

Discussing attacker models. We discuss the attacker model adopted by each definition, showing that each definition considers a different attacker model. We find that KZZ models a voter that attempts to prove their vote to an attacker without allowing the voter to interact with the attacker before voting. In particular, the attacker cannot provide instructions to the voter. We demonstrate that the attacker model in CCFG is stronger, capturing an attacker with some control over the voter. We also demonstrate that BKV does not model a voter who attempts to prove their vote, but only asks whether an attacker can determine whether a voter has updated their vote from the attacker's choice or not. We use these discussions to explore the boundary between receipt-freeness and coercion-resistance, a stronger privacy notion that captures receipt-freeness and provides additional protection against randomisation, simulation, and abstention attacks.

New definitions of receipt-freeness. We then present two new definition of receipt-freeness, which we call N-RF and E-RF. We design our two definitions to be applicable to two different classes of e-voting schemes. Specifically, e-voting schemes may achieve receipt-freeness by providing the voter with an *evasion strategy* (see, for example, [25, 88, 100]), a procedure that a voter can follow to evade coercion. Thus, we define E-RF, which indicates Evasion strategy Receipt-Freeness, to capture schemes that require an evasion strategy. Other e-voting schemes do not require an evasion strategy to achieve receipt-freeness. To this end, we define N-RF, a definition of receipt-freeness for schemes that are Non evasion strategy Receipt-Freeness. Our definitions build upon the ballot secrecy definition BPRIV [22], and address the soundness and completeness issues in previous definitions.

We conclude with an analysis of our new definitions. In particular, we compare N-RF and E-RF, showing that N-RF is stronger than E-RF. That is, if an e-voting scheme satisfies N-RF, then the e-voting scheme can be shown to satisfy E-RF. However, the reverse implication does not hold. In fact, it transpires that there exists e-voting schemes that are intuitively coercion-resistant but can only satisfy our weaker notion of receipt-freeness: E-RF. This hints at an interesting relationship between receipt-freeness and coercion-resistance that we highlight in this chapter. We also contextualise our new definitions by discussing how they relate to the existing receipt-freeness literature. This allows us to emphasise the benefits and limitations of our new definitions, and we demonstrate potential avenues for future research on this topic. We summarise the results of our analysis as follows.

$$\mathsf{CCFG} \overset{\rightarrow}{\not\leftarrow} \mathsf{N-RF} \overset{\rightarrow}{\not\leftarrow} \mathsf{E-RF} \rightarrow \mathsf{BPRIV}$$

4.2 Chapter Preliminaries

The receipt-freeness definitions that we analyse and introduce in this chapter are all defined for e-voting schemes with internal voter registration. Therefore, throughout this chapter, we rely on the e-voting scheme syntax introduced in Definition 43 of Chapter 3. Furthermore, to prove several results in this chapter, we rely on the JCJ [88] e-voting scheme. We present the JCJ scheme here.

4.2.1 The JCJ E-Voting Scheme

JCJ [88] was the first coercion-resistant e-voting scheme to be presented in the literature, and has been implemented as Civitas [42, 43]. JCJ has been shown to satisfy definitions of coercion-resistance [88] and, as a result, JCJ should satisfy receipt-freeness. However, in this chapter, we show that JCJ does not satisfy a number of receipt-freeness definitions, demonstrating that several receipt-freeness definitions are incomplete. We define JCJ in Figure 4.1 by adapting the definition of generalised JCJ presented in [122] to our syntax. JCJ requires a public-key encryption scheme PKE, several non-interactive zero-knowledge proof systems, and a mixnet. We now provide a brief description of JCJ.

The election administrator runs algorithm Setup to generate the public parameters for PKE and the NIZK system, and generates a key pair for PKE. For simplicity in our construction presentation, we run a single instance of NIZK.Setup to generate public parameters for all NIZK proof systems required in the JCJ construction. However, in practice, it may be the case that each proof system requires a different set of public parameters. JCJ requires the following NIZK proofs: a proof of plaintext knowledge (NIZK_{enc}), conjunctive plaintext knowledge ($\mathsf{NIZK}_{\mathsf{conj}}$), plaintext equivalence test ($\mathsf{NIZK}_{\mathsf{PET}}$), correct mixing ($\mathsf{NIZK}_{\mathsf{mix}}$) and correct decryption ($\mathsf{NIZK}_{\mathsf{Dec}}$).

In the registration phase, algorithm Register is run to generate credentials for voters. The voter's secret credential is an element chosen from the message space of PKE and the public credential is an encryption of the secret credential under the public key for PKE.

During the voting phase of the election, voters produce ballots by running algorithm Vote. Ballots consist of the following four elements: 1) an encryption of the voter's choice under the public key for PKE; 2) an encryption of the voter's secret credential under the public key for PKE; 3) a NIZK proof that the first ciphertext encrypts the voter's choice; 4) a NIZK proof of conjunctive plaintext knowledge of the voter's choice and secret credential. Ballots are appended to the ballot box (without modification or any checks) by the ballot box manager running algorithm Append. The public bulletin board output by algorithm Publish is identical to the ballot box.

During the tally phase, the tallier runs algorithm Tally. First, the ballots are filtered to retain only ballots for which the NIZK proofs in the ballot verify. Then, a single ballot for each voter is retained, implementing a last-vote-counts re-vote policy. Next, the remaining ballots are mixed, which involves permuting the ballots and re-encrypting each ballot. A NIZK proof of correct mixing is generated. Following mixing, ballots are filtered to retain only ballots submitted on behalf of registered voters. The remaining ballots are decrypted, and a proof of correct decryption is generated. The election result is assumed to be a vector that indicates the number of votes cast for each candidate. Anyone can verify that the election result is correct. We omit the details of verification for simplicity in our presentation, noting that, to verify the result, it is necessary to check each proof generated by algorithm Tally. For full details of the verification of the tallying procedure, the interested reader can consult [122].

JCJ is additionally equipped with an algorithm FakeKey that a voter can run to generate a fake secret credential that is indistinguishable from a real secret credential. This allows a voter to evade coercion. For example, if a voter is coerced to vote for a particular candidate, they can submit a ballot for that candidate with their fake credential. Then, the voter can prove that they followed the coercer's instructions by providing the coercer with their fake credential and the randomness used to construct the ballot. Furthermore, the voter can give the coercer their fake credential, and the coercer can cast (fake) ballots on behalf of the voter. Meanwhile, a coerced voter can submit ballots using their real secret credential. Ballots submitted with the fake credential are discarded during the tallying phase of the

election and the voter's real vote (i.e., the vote submitted with the voter's real secret credential) is included in the election result.

$\mathsf{Setup}(1^\lambda)$

$\mathcal{L} \leftarrow \emptyset;\ \mathsf{bb} \leftarrow ()$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}})$
$pp_{\mathsf{NIZK}} \leftarrow\!\!\$\ \mathsf{NIZK.Setup}(1^\lambda)$
$pk \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$sk \leftarrow (sk_{\mathsf{PKE}}, pk)$
return (pk, sk)

$\mathsf{Register}(id, pk)$

parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$sk_{id} \leftarrow\!\!\$\ \mathsf{M}_{\mathsf{PKE}}$
$pk_{id} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, sk_{id})$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{pk_{id}\}$
return (pk_{id}, sk_{id})

$\mathsf{Vote}(v, pk_{id}, sk_{id}, pk)$

parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$c_1 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v; r_{c_1})$
$c_2 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, sk_{id}; r_{c_2})$
$\rho_{c_1} \leftarrow\!\!\$\ \mathsf{NIZK.Prove}_{\mathsf{Enc}}(pp_{\mathsf{NIZK}}, (pk_{\mathsf{PKE}}, c_1, \mathcal{C}), (r_{c_1}, v))$
$\rho_{c_2} \leftarrow\!\!\$\ \mathsf{NIZK.Prove}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, (pk_{\mathsf{PKE}}, c_1, c_2), (r_{c_1}, v, r_{c_2}, sk_{id}))$
return $b \leftarrow (c_1, c_2, \rho_{c_1}, \rho_{c_2})$

$\mathsf{FakeKey}(pk_{id}, sk_{id}, pk)$

return sk'_{id}

$\mathsf{Valid}(b, \mathsf{bb}, \mathcal{L}, pk)$

return 1

$\mathsf{Append}(b, \mathsf{bb}, \mathcal{L}, pk)$

return $\mathsf{bb} \leftarrow \mathsf{bb} \parallel b$

$\mathsf{Publish}(\mathsf{bb})$

return bb

$\mathsf{Tally}(\mathsf{bb}, \mathcal{L}, sk)$

parse sk as (sk_{PKE}, pk) $\wedge$ **parse** pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$r \leftarrow ();\ A_1 \leftarrow ();\ A'_1 \leftarrow ();\ B_1 \leftarrow ();\ B'_1 \leftarrow ();\ A_3 \leftarrow ()$
for $i = 1, \ldots, |\mathsf{bb}|$
  parse $\mathsf{bb}[i]$ as $(c_1, c_2, \rho_{c_1}, \rho_{c_2})$
  if $\mathsf{NIZK.Verify}_{\mathsf{Enc}}(pp_{\mathsf{NIZK}}, (pk_{\mathsf{PKE}}, c_1, \mathcal{C}), \rho_{c_1}) = 1\ \wedge\ \mathsf{NIZK.Verify}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, (pk_{\mathsf{PKE}}, c_1, c_2), \rho_{c_2}) = 1$
    $A_1 \leftarrow A_1 \parallel c_1;\ B_1 \leftarrow B_1 \parallel c_2$
for $i = 1, \ldots, |B_1|$
  if $B_1[i]$ encrypts a voter secret credential that does not appear in any other entry in B_1 (i.e., applying $\mathsf{NIZK}_{\mathsf{PET}}$)
    $A'_1 \leftarrow A'_1 \parallel A_1[i];\quad B'_1 \leftarrow B'_1 \parallel B_1[i]$
$A_2 \leftarrow\!\!\$\ \mathsf{MixNet}(pk, A'_1; \phi_1);\quad \rho_{A_2} \leftarrow\!\!\$\ \mathsf{NIZK.Prove}_{\mathsf{mix}}(pp_{\mathsf{NIZK}}, (pk, A'_1, A_2), \phi_1)$
$B_2 \leftarrow\!\!\$\ \mathsf{MixNet}(pk, B'_1; \phi_1);\quad \rho_{B_2} \leftarrow\!\!\$\ \mathsf{NIZK.Prove}_{\mathsf{mix}}(pp_{\mathsf{NIZK}}, (pk, B'_1, B_2), \phi_1)$
$L' \leftarrow\!\!\$\ \mathsf{MixNet}(pk, \mathcal{L}; \phi_2);\quad \rho_{L'} \leftarrow\!\!\$\ \mathsf{NIZK.Prove}_{\mathsf{mix}}(pp_{\mathsf{NIZK}}, (pk, \mathcal{L}, L'), \phi_2)$
for $i = 1, \ldots, |B_2|$
  if $B_2[i]$ encrypts a plaintext such that L' contains an encryption of the same plaintext (i.e., applying $\mathsf{NIZK}_{\mathsf{PET}}$)
    $A_3 \leftarrow A_3 \parallel A_2[i]$
for $i = 1, \ldots, |A_3|$: $v_i \leftarrow \mathsf{PKE.Dec}(pp_{\mathsf{PKE}}, sk_{\mathsf{PKE}}, A_3[i]);\ r[v] \leftarrow r[v] + 1$
$\rho_{\mathsf{Dec}} \leftarrow\!\!\$\ \mathsf{NIZK.Prove}_{\mathsf{Dec}}(pp_{\mathsf{NIZK}}, (pk_{\mathsf{PKE}}, A_3, v_1, \ldots, v_{|A_3|}), sk_{\mathsf{PKE}})$
Let ρ contain all proof generated by algorithm Tally
return (r, ρ)

Figure 4.1: The JCJ e-voting scheme [88].

4.2.2 A Useful Property

In this chapter, we prove that e-voting schemes satisfy (or do not satisfy) a given definition of receipt-freeness. Occasionally, our proofs rely on a property of an e-voting scheme called *injectivity*, as defined in [122]. Intuitively, injectivity requires that two ballots output by algorithm Vote are identical only if the underlying votes are the same. Thus, injectivity ensures that ballots are unambiguous; any honestly produced ballot can only encode

one vote from the set of candidates. We note that both the Helios and JCJ e-voting systems satisfy injectivity [122, Theorems 19 and 28]. We capture this property formally in Definition 51.

Definition 51 (Injectivity [122]). *An e-voting scheme* Γ *for a result function* f*, set of voters* $\mathcal{V}$ *and set of candidates* $\mathcal{C}$*, satisfies* injectivity *if for any public key* pk *output by algorithm* Setup*, any voters* $id_0, id_1 \in \mathcal{V}$*, and any vote choices* $v_0, v_1 \leftarrow\!\!\$\, \mathcal{C}$ *with* $v_0 \neq v_1$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l}(pk_{id_0},sk_{id_0}) \leftarrow\!\!\$\, \mathsf{Register}(pk,id_0);\\ (pk_{id_1},sk_{id_1}) \leftarrow\!\!\$\, \mathsf{Register}(pk,id_1);\\ b_0 \leftarrow\!\!\$\, \mathsf{Vote}(v_0,pk_{id_0},sk_{id_0},pk);\\ b_1 \leftarrow\!\!\$\, \mathsf{Vote}(v_1,pk_{id_1},sk_{id_1},pk)\end{array} : \; b_0 \neq b_1 \wedge b_0 \neq \bot \wedge b_1 \neq \bot\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

4.3 Receipt-Freeness by Kiayias, Zacharias & Zhang (KZZ)

We now analyse the receipt-freeness definition by Kiayias *et al.* [91], which we call KZZ. First, we cast KZZ into our syntax, and then we explore completeness and soundness issues of the definition. In particular, we demonstrate a soundness issue with KZZ, namely, that KZZ guarantees receipt-freeness only if all voters vote. Moreover, KZZ is incomplete because there are schemes that are receipt-free but do not satisfy KZZ. We conclude with a discussion on the attacker model of KZZ.

4.3.1 The KZZ Experiment

KZZ describes an experiment in which an adversary can submit ballots on behalf of corrupted voters and submits two voters (left and right votes) on behalf of non-corrupted voters. The KZZ experiment always provides the adversary with a view of a bulletin board, and the corresponding election result, that contains ballots for either the left or right votes of non-corrupted voters (in addition to the ballots of corrupted voters). Additionally, KZZ requires that a balancing condition holds. Accordingly, KZZ follows the 'tally the viewed election' approach to defining privacy for e-voting (cf. Chapter 3.3.2). To capture receipt-freeness, the KZZ experiment also provides the adversary with the "internal state"[91][2] of non-corrupted voters during voting. This models the 'receipt' that the voter

[2]The internal state refers to any information that the voter inputs to the voting client to produce a ballot, including, but not necessarily limited to, secret credentials and the randomness input to algorithm Vote.

may present to an attacker to prove their vote. The adversary is provided with the real internal state of non-corrupted voters, or with a simulated internal state, which models a voter that attempts to prove that they submitted a different vote. If an adversary cannot determine whether the bulletin board and result contain the left or right votes submitted by non-corrupted voters, a scheme is said to satisfy KZZ.

We formally cast KZZ into our syntax in Definition 52 and describe the KZZ experiment for adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ in Figure 4.2. In the KZZ experiment, sets V_r, V_c, Cor and Receipts are initialised as empty. Respectively, these sets track non-corrupted voters, corrupted voters, secret credentials of corrupted voters, and the receipts of non-corrupted voters. An election key pair is generated and all $|\mathcal{V}|$ eligible voters are registered. Adversary $\mathcal{A}_1 = (\mathcal{A}_{1,1}, \ldots, \mathcal{A}_{1,|\mathcal{V}|})$ is given the public key pk, the list of public credentials $\mathcal{L}$ and (initially) empty sets Receipts and Cor. For all $|\mathcal{V}|$ eligible voters, $\mathcal{A}_{1,i}$ ($i \in [|\mathcal{V}|]$) decides whether voter id_i is corrupt by outputting a bit. If the voter is corrupt (i.e., bit $d_i = 0$), $\mathcal{A}_{2,i}$ outputs a ballot b_i, on input of the voter's secret credential sk_{id_i}. The ballot is appended to the ballot box, and sets V_c and Cor are updated to include the identity and secret credential of the corrupt voter respectively. If the voter is not corrupt (i.e., $d_i = 1$), $\mathcal{A}_{2,i}$ outputs votes v_0^i, v_1^i, and the challenger computes a ballot b_i for vote v_β^i The challenger updates set Receipts with the ballot and the receipt Rcpt_i. The receipt is equal to the real internal state of the voter if $\beta = 0$. If $\beta = 1$, a simulator, denoted $\mathcal{S}$, generates Rcpt_i on input the election public key pk, ballot b_i and vote v_0^i. The ballot is appended to the ballot box, and set V_r is updated to include the identity of the non-corrupted voter. $\mathcal{A}_3$ is provided with the election result, tallying proof and bulletin board, and outputs a bit β'. The experiment returns β' if the number of corrupted voters is bounded by t, and $f(\langle (id_i, v_0^i) \rangle_{id_i \in V_r}) = f(\langle (id_i, v_1^i) \rangle_{id_i \in V_r})$,[3] i.e., with respect to uncorrupted voters the outcome of the election computed via the result function f is the same, regardless of whether $\beta = 0$ or $\beta = 1$.

Definition 52 (KZZ [91])**.** *An e-voting scheme Γ for a result function f, set of voters $\mathcal{V}$ and set of candidates $\mathcal{C}$ satisfies* KZZ *for at most t corrupted voters if there exists a simulator $\mathcal{S}$ such that, for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there exists a negligible function* negl *such that*

[3] We borrow the notation $f(\langle (id, v) \rangle_{id \in V_r})$ from [91] to mean that function f is applied to all tuples of the form (id, v) such that the condition $id \in V_r$ is satisfied.

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{KZZ},t,0}_{\Gamma,\mathcal{A},\mathcal{S},\mathcal{V},\mathcal{C}}(\lambda)=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{KZZ},t,1}_{\Gamma,\mathcal{A},\mathcal{S},\mathcal{V},\mathcal{C}}(\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{KZZ},t,\beta}_{\Gamma,\mathcal{A},\mathcal{S},\mathcal{V},\mathcal{C}}(\lambda)$ *is the experiment defined in Figure 4.2 for* $\beta\in\{0,1\}$.

$\mathsf{Exp}^{\mathsf{KZZ},t,\beta}_{\Gamma,\mathcal{A},\mathcal{S},\mathcal{V},\mathcal{C}}(\lambda)$

$V_r\leftarrow\emptyset;\ V_c\leftarrow\emptyset;\ \mathsf{Cor}\leftarrow\emptyset;\ \mathsf{Receipts}\leftarrow\emptyset$
$(pk,sk)\leftarrow_{\$}\mathsf{Setup}(1^\lambda)$
for $i=1,\ldots,|\mathcal{V}|$: $(pk_{id_i},sk_{id_i})\leftarrow_{\$}\mathsf{Register}(id_i,pk)$
$\mathcal{L}\leftarrow\{pk_{id_1},\ldots,pk_{id_{|\mathcal{V}|}}\}$
for $i=1,\ldots,|\mathcal{V}|$: $d_i\leftarrow_{\$}\mathcal{A}_{1,i}(pk,\mathcal{L},\mathsf{Receipts},\mathsf{Cor})$
  if $d_i=0$
    $V_c\leftarrow V_c\cup\{id_i\}$
    $\mathsf{Cor}\leftarrow\mathsf{Cor}\cup\{sk_{id_i}\}$
    $b_i\leftarrow_{\$}\mathcal{A}_{2,i}(pk,\mathcal{L},\mathsf{Receipts},\mathsf{Cor})$
  else
    $V_r\leftarrow V_r\cup\{id_i\}$
    $(v_0^i,v_1^i)\leftarrow_{\$}\mathcal{A}_{2,i}(pk,\mathcal{L},\mathsf{Receipts})$
    $b_i\leftarrow_{\$}\mathsf{Vote}(v_\beta^i,pk_{id_i},sk_{id_i},pk)$
    if $\beta=0$: $\mathsf{Receipts}\leftarrow\mathsf{Receipts}\cup\{(\mathsf{Rcpt}_i,b_i)\}$
    if $\beta=1$: $\mathsf{Receipts}\leftarrow\mathsf{Receipts}\cup\{(\mathcal{S}(pk,b_i,v_0^i),b_i)\}$
  $\mathsf{bb}\leftarrow_{\$}\mathsf{Append}(b_i,\mathsf{bb},\mathcal{L},pk)$
$\mathcal{BB}\leftarrow_{\$}\mathsf{Publish}(\mathsf{bb})$
$(r,\rho)\leftarrow_{\$}\mathsf{Tally}(\mathsf{bb},\mathcal{L},sk)$
$\beta'\leftarrow_{\$}\mathcal{A}_3(r,\rho,\mathcal{BB},pk,\mathcal{L},\mathsf{Receipts},\mathsf{Cor})$
if $V_c\leq t\ \wedge\ f(\langle(id_i,v_0^i)\rangle_{id_i\in V_r})=f(\langle(id_i,v_1^i)\rangle_{id_i\in V_r})$ **return** β'

Figure 4.2: The receipt-freeness experiment $\mathsf{Exp}^{\mathsf{KZZ},t,\beta}_{\mathcal{A},\Gamma,\mathcal{S},\mathcal{V},\mathcal{C}}(\lambda)$ for $\beta\in\{0,1\}$ from [91] and cast into our syntax.

4.3.2 Soundness of KZZ

KZZ requires that a single ballot is submitted to the ballot box on behalf of each voter. Though this may appear to be a minor technical issue, we show that, in fact, KZZ declares schemes as receipt-free that reveal how voters vote, when not all voters vote. To illustrate this, we define an e-voting scheme Tally.Γ (Definition 53) for which the election result lists each voter that voted and the vote submitted by that voter, if not all voters vote. Intuitively, this scheme is not receipt-free. Indeed, the scheme does not satisfy ballot secrecy because the result announces the link between voter and vote. However, in Proposition 1, we show that Tally.Γ satisfies KZZ. This occurs because, in the KZZ experiment, a ballot must be submitted for every voter, so this privacy leakage will not be identified. Consequently, if a

real-world deployment cannot ensure that all voters vote, a proven secure scheme may leak every voter's vote. Hence, there may exist schemes that are proven secure but, in practice, do not offer any degree of privacy for voters.

Definition 53 (E-voting scheme $\mathsf{Tally}.\Gamma$)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Valid}, \mathsf{Append}, \mathsf{Publish}, \mathsf{Tally}, \mathsf{Verify})$ *be an e-voting scheme that satisfies* KZZ*. We define a modified scheme* $\mathsf{Tally}.\Gamma = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Valid}, \mathsf{Append}, \mathsf{Publish}, \mathsf{Tally}', \mathsf{Verify})$ *such that*

$\mathsf{Tally}'(\mathsf{bb}, \mathcal{L}, sk)$ *On input ballot box* bb*, set* $\mathcal{L}$ *and election secret key* sk*, algorithm* Tally' *checks if* $|\mathsf{bb}| \leq n_v - 1$*. If* $|\mathsf{bb}| \leq n_v - 1$*,* Tally' *returns* $(r, \rho) = (\{(pk_{id_1}, v_1), \ldots, (pk_{id_{|\mathsf{bb}|}}, v_{|\mathsf{bb}|})\}, \perp)$ *where tuples* $(pk_{id_1}, v_1), \ldots, (pk_{id_i}, v_i)$ *are input to algorithm* Vote *and a corresponding ballot appears on* bb*. Else, algorithm* Tally' *computes* $(r, \rho) \leftarrow_{\$} \mathsf{Tally}(\mathsf{bb}, \mathcal{L}, sk)$ *and returns* (r, ρ)*.*

Proposition 1. $\mathsf{Tally}.\Gamma$ *satisfies* KZZ *if* Γ *satisfies* KZZ*.*

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ be an adversary in the KZZ experiment against scheme $\mathsf{Tally}.\Gamma$. We show that we can construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3)$ that succeeds in the KZZ experiment for construction Γ. Adversary $\mathcal{B}$ internally runs $\mathcal{A}$ and attempts to guess a bit β^* in the KZZ experiment. We present adversary $\mathcal{B}$ in Figure 4.3.

We first note that the inputs to adversary $\mathcal{A}$ provided by adversary $\mathcal{B}$ are identically distributed to the inputs in the KZZ experiment. Indeed, the public key pk, list $\mathcal{L}$, bulletin board $\mathcal{BB}$ and sets Receipts and Cor are generated identically. Moreover, $|\mathsf{bb}| = n_v$ because $\mathcal{A}$ outputs a single ballot or vote pair for each eligible voter. As such, algorithm Tally' always runs algorithm Tally. Therefore, the result r and proof ρ input to $\mathcal{A}$ by $\mathcal{B}$ is identical to the input to $\mathcal{A}$ in the KZZ experiment. Thus, $\mathcal{B}$ perfectly simulates the challenger in the KZZ experiment to $\mathcal{A}$.

We now show that $\mathcal{B}$ succeeds in the KZZ experiment against scheme $\mathsf{Tally}.\Gamma$. That is, we show that $\beta' = \beta^*$, $V_c \leq t$ and $f(\langle v_{0,i} \rangle_{id_i \in V_h}) = f(\langle v_{1,i} \rangle_{id_i \in V_h})$. By assumption, the second two conditions hold. Otherwise, $\mathcal{A}$ cannot succeed in the KZZ experiment against scheme $\mathsf{Tally}.\Gamma$. Moreover, if $\mathcal{A}$ succeeds then $\mathcal{A}$ outputs $\beta' = \beta$. As $\beta = \beta^*$ and is chosen in the KZZ experiment against scheme $\mathsf{Tally}.\Gamma$, $\mathcal{B}$ returns $\beta' = \beta^*$ and succeeds in the KZZ experiment for scheme Γ. Therefore, by contradiction, the result holds. □

$\underline{\mathcal{B}_{1,i}(pk, \mathcal{L}, \mathsf{Receipts}, \mathsf{Cor})}$

$V_c \leftarrow \emptyset;\ V_r \leftarrow \emptyset$
$d_i \leftarrow\!\!\$\ \mathcal{A}_{1,i}(pk, \mathcal{L}, \mathsf{Receipts})$
return d_i

$\underline{\mathcal{B}_3(r, \rho, \mathcal{BB}, pk, \mathcal{L}, \mathsf{Receipts}, \mathsf{Cor})}$

$\beta' \leftarrow\!\!\$\ \mathcal{A}_3(r, \rho, \mathcal{BB}, pk, \mathcal{L}, \mathsf{Receipts}, \mathsf{Cor})$
return β'

$\underline{\mathcal{B}_{2,i}(pk, \mathcal{L}, \mathsf{Receipts}, \mathsf{Cor})\ //\ \text{if } d_i = 0}$

$V_c \leftarrow V_c \cup \{id_i\}$
$b_i \leftarrow\!\!\$\ \mathcal{A}_{2,i}(pk, \mathcal{L}, \mathsf{Receipts}, \mathsf{Cor})$
return b_i

$\underline{\mathcal{B}_{2,i}(pk, \mathcal{L}, \mathsf{Receipts}, \mathsf{Cor})\ //\ \text{if } d_i = 1}$

$V_r \leftarrow V_r \cup \{id_i\}$
$(v_0^i, v_1^i) \leftarrow\!\!\$\ \mathcal{A}_{2,i}(pk, \mathcal{L}, \mathsf{Receipts})$
return (v_0^i, v_1^i)

Figure 4.3: Adversary $\mathcal{B}$ that breaks the KZZ security of Γ in the proof of Proposition 1.

Fortunately, the soundness issue with KZZ can be remedied in a straightforward way. The experiment can provide adversary $\mathcal{A}$ with access to oracles that allow the submission of ballots or two votes on behalf of corrupt and non-corrupt voters respectively (i.e., as defined for our ballot secrecy definition BSec (Definition 48) and other formal definitions of ballot secrecy (cf. Chapter 3.3). In this way, an adversary can submit ballots and votes but does not have to submit a single query on behalf of each voter.

4.3.3 Completeness of KZZ

We now discuss that KZZ captures a limited class of schemes, thereby demonstrating completeness issues with KZZ. First, we recall that KZZ requires a balancing condition to hold. In fact, as in [16], KZZ requires the partial tally assumption (cf. Chapter 3.3.2). Therefore, KZZ is limited to considering result functions for which the partial tally assumption holds. We now discuss a further completeness issue that we uncover with KZZ that arises from the requirement that a single ballot must be submitted on behalf of each voter.

Schemes with multiple ballots are out of scope. KZZ defines an adversary that can only submit a single ballot or vote pair on behalf of each voter. In addition to the soundness issue this causes, we now show that this requirement means that schemes with multiple ballots are out of scope of KZZ. That is to say, if an intuitively receipt-free e-voting scheme requires multiple ballots to achieve receipt-freeness, it may not satisfy KZZ. We demonstrate this by showing that JCJ [88] (Figure 4.1), a coercion-resistant scheme, does not satisfy KZZ. Recall that JCJ (Figure 4.1) provides a mechanism by which voters can generate fake secret credentials (that are indistinguishable from real secret credentials).

Voters can cast dummy ballots using fake credentials and prove the contents of dummy ballots (rather than real ballots) to an attacker. A voter can then cast a ballot for a different vote using their real credential. Accordingly, it is necessary that a voter submits two ballots to submit a vote but prove that they submitted a different vote. However, KZZ captures the requirement that a voter proves they submitted a different vote, but only allows the submission of a single ballot. As a result, JCJ does not satisfy KZZ. We obtain the result in Proposition 2.

Proposition 2. JCJ *(Figure 4.1) does not satisfy* KZZ *if* JCJ *satisfies injectivity.*

Proof. We first define the form of Rcpt_i for a voter $id_i \in \mathcal{V}$. Let $\mathsf{Rcpt}_i = (v_0^i, b_i, pk_{id_i}, sk_{id_i}, r_{c_1}, r_{c_2})$ where $b_i \leftarrow\!\!\$\, \mathsf{Vote}(v_0^i, pk_{id_i}, sk_{id_i}, pk; r_{c_1}, r_{c_2})$ for some vote choice $v_0^i \in \mathcal{C}$, voter credential pair (pk_{id_i}, sk_{id_i}) output by algorithm Register, and some public election key pk output by algorithm Setup. Moreover, we define the simulated receipt to be $(v_0^i, b_i', pk_{id_i}, sk_{id_i}', r_{c_1}'', r_{c_2}'') \leftarrow\!\!\$\, \mathcal{S}(pk, b_i', v_0^i) =$ for some $b_i' \leftarrow\!\!\$\, \mathsf{Vote}(v_i^1, pk_{id_i}, sk_{id_i}, pk; r_{c_1}', r_{c_2}')$ and $sk_{id_i}' \leftarrow\!\!\$\, \mathsf{FakeKey}(pk_{id_i}, sk_{id_i}, pk)$. That is, the output of algorithm $\mathcal{S}$ is a transcript such that b_i^* appears to encode a vote for v_0^i when, in fact, it encodes a vote for v_i^1.

We now show that we can construct an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ that succeeds in the KZZ experiment for the JCJ scheme. We define adversary $\mathcal{A}$ in Figure 4.4 to select all $|\mathcal{V}|$ voters to be non-corrupted. On behalf of $|\mathcal{V}|/2$ voters, $\mathcal{A}_{2,i}$ submits vote choices (v_0, v_1) and, on behalf of the remaining voters, submits vote choices (v_1, v_0). In this way, the balancing condition of the KZZ experiment is satisfied, and no voters are corrupted, ensuring that $V_c \leq t$ as required. Note that this implicitly assumes that $\mathcal{V}$ contains an even number of eligible voters. If $\mathcal{V}$ contains an odd number, $\mathcal{B}_{2,|\mathcal{V}|}$ can submit (v_0, v_0) to ensure that the balancing condition holds.

Consider voter id_1. Bulletin board $\mathcal{BB}$ contains the ballot $b_1 = \mathsf{Vote}(v_0, pk_{id_1}, sk_{id_1}, pk; r_{c_1}, r_{c_2})$ (if $\beta = 0$) or $b_1 = \mathsf{Vote}(v_1, pk_{id_1}, sk_{id_1}, pk; r_{c_3}, r_{c_4})$ (if $\beta = 1$). With access to $\mathsf{Rcpt}_1 = (v_0^i, b_i, pk_{id_i}, sk_{id_i}^*, r_{c_1}^*, r_{c_2}^*)$, the adversary can locally compute a ballot b^*. That is, adversary $\mathcal{A}$ can compute $b^* \leftarrow\!\!\$\, \mathsf{Vote}(v_0, pk_{id_1}, sk_{id_1}^*, pk; r_{c_1}^*, r_{c_2}^*)$. As JCJ is injective, if $b^* = b_1$ it must be the case that $\beta = 0$. Else, $\beta = 1$. Therefore, $\mathcal{A}$ returns $\beta' = \beta$ in the KZZ experiment and e-voting scheme JCJ does not satisfy KZZ. □

$\mathcal{A}_{1,i}(pk, \mathcal{L}, \mathsf{Receipts}, \mathsf{Cor})$	$\mathcal{A}_{2,i}(pk, \mathcal{L}, \mathsf{Receipts}, \mathsf{Cor})$ // if $d_i = 0$
return 1	**return** b_i
$\mathcal{A}_3(r, \rho, \mathcal{BB}, pk, \mathcal{L}, \mathsf{Receipts}, \mathsf{Cor})$	$\mathcal{A}_{2,i}(pk, \mathcal{L}, \mathsf{Receipts}, \mathsf{Cor})$ // if $d_i = 1$
parse Receipts as $\{\mathsf{Rcpt}_1 = (v_0, b_1, pk_{id_1}, sk^*_{id_1}, r^*_{c_1}, r^*_{c_2}), \dots\}$ $b^* \leftarrow\!\!\$\ \mathsf{Vote}(v_0, pk_{id_1}, sk^*_{id_1}, pk; r^*_{c_1}, r^*_{c_2})$ **if** $b^* = b_1$ **return** 0 **if** $b^* \neq b_1$ **return** 1	**if** $i \leq \lvert\mathcal{V}\rvert/2$: **return** (v_0, v_1) **else return** (v_1, v_0)

Figure 4.4: Adversary $\mathcal{A}$ that succeeds in the KZZ experiment for the JCJ scheme in the proof of Proposition 2.

4.3.4 Further Discussion

Receipt-freeness intuitively captures the property that a voter cannot prove their vote. That is, an attacker cannot be convinced of how a voter voted, even if the voter provides 'proof' of their vote to the attacker. KZZ captures this intuition. Specifically, KZZ models attack scenarios in which a voter provides evidence of their vote (including their secret credential) to the attacker after voting. The intuition of receipt-freeness does not indicate whether the voter can interact with the attacker (i.e., by revealing their secret credentials or any other material) before they have voted, which raises the following question: should an attacker that interacts with a voter before voting fall under the umbrella of receipt-freeness? To answer this question, we first illustrate that KZZ does not capture such an attacker. Then, we discuss this question more widely and, hence, discuss whether KZZ captures a reasonable attacker model.

We show that DEMOS, an e-voting scheme that satisfies KZZ [91, Theorem 5], is no longer receipt-free if an attacker can compel a voter to reveal their credentials before voting. DEMOS provides each eligible voter with a voting card (which is a secret credential in our terminology). This voting card consists of two parts: the first part contains a list of candidates and a unique vote code associated with each candidate. This is repeated on the second part of the voting card, although the vote codes associated with each candidate are different. To cast a ballot, each voter selects a part of their voting card (part '0' or part '1', which we call the randomness, using our terminology). The selected part and the vote code listed next to their chosen candidate constitute the voter's ballot. The ballot box is updated with the ballot. Intuitively, DEMOS satisfies KZZ because voters can swap vote codes on the voting card and can make the vote code on their ballot correspond to any candidate they wish, before revealing the voting card to an attacker as a 'receipt'.

Therefore, the voter can convince the attacker that the submitted vote code corresponds to the attacker's choice of candidate after voting.

However, consider the following scenario: an attacker wants a voter to vote for candidate A, but the voter wants to vote for candidate B. The attacker requests to see the voter's voting card *before* voting. Only after seeing the voting card, the attacker requests that the voter cast a ballot for A. In this scenario, the voter may not have switched vote codes for A and B. Thus, the voter cannot vote for B *and* convince the attacker that they voted for A. By contrast, if an attacker does not see the voting card until after voting, the voter can switch the vote codes for A and B. Therefore, DEMOS is receipt-freeness only if it is assumed that the voting card is revealed after voting.

The scenario above describes an attacker who interacts with a voter before voting, which is outside the scope of KZZ. Many definitions of receipt-freeness concur with this, suggesting that providing information to an attacker before voting is captured by coercion-resistance, not receipt-freeness. In fact, Delaune *et al.*, define receipt-freeness as the property that "a voter does not gain any information (a receipt) that can be used to prove to a coercer that she voted in a certain way", and coercion-resistance as the property that "a voter cannot cooperate with a coercer to prove to him that she voted in a certain way" [52]. This suggests that providing information to an attacker before voting is captured by coercion-resistance, not receipt-freeness. Moreover, Delaune *et al.*'s definition of receipt-freeness implies that a voter uses information to prove their vote only *after* voting, whereas providing information to an attacker *before* voting is considered cooperation with an attacker. Therefore, we conclude that KZZ does capture a reasonable attack model. However, stronger variants of receipt-freeness exist, as we shall see in the next section. Therefore, we return to this discussion at the end of Section 4.4.

4.4 Receipt-Freeness by Chaidos *et al.* (CCFG)

We now consider a definition of receipt-freeness by Chaidos *et al.* [38], which we call CCFG. As with KZZ, we cast CCFG into our syntax. We then show that CCFG is unsound as it overlooks the needs for additional properties and is incomplete, limiting the class of schemes that can be declared receipt-free. We conclude this section by returning to our

discussion on attacker models from the previous section.

4.4.1 The CCFG Experiment

Chaidos *et al.* define CCFG as an extension of BPRIV [22] such that an adversary is provided with a view of bulletin board $\mathcal{BB}_0$ or $\mathcal{BB}_1$, but the election result is always computed for the contents of ballot box bb_0. Like BPRIV, the CCFG experiment allows the adversary to submit ballots on behalf of corrupt voters and two votes on behalf of honest voters. Moreover, the adversary can submit *two* ballots (the left and right ballot) on behalf of corrupt voters. The left (resp., right) ballot is appended to ballot box bb_0 (resp., bb_1). This additional capability captures the intuition that a receipt-freeness attacker may be provided with (or may provide a voter with) all the material used to construct the voter's ballot such that the attacker can locally reconstruct the ballot submitted by the voter. If the adversary cannot determine whether they are viewing $\mathcal{BB}_0$ or $\mathcal{BB}_1$, a scheme is said to satisfy CCFG.

We cast CCFG into our syntax in Definition 54 and describe the CCFG experiment in Figure 4.5. Like BPRIV, CCFG relies on algorithms Sim and Sim_ρ, defined in Chapter 3.3.1, that facilitate the ability to simulate a tallying proof ρ. With these algorithms, CCFG can be described as follows. Ballot boxes bb_0 and bb_1 are initialised as empty lists, lists **pk** and **sk** are initialised as empty to track the public and secret credentials of voters, and sets $\mathcal{Q}\mathsf{reg}$ and $\mathcal{Q}\mathsf{corrupt}$ track, respectively, registered and corrupted voters. Adversary $\mathcal{A}$ can query oracles defined in Figure 4.5, under the constraint that $\mathcal{O}\mathsf{setup}$ must be queried before any other oracles and $\mathcal{O}\mathsf{tally}$ appears only as the final oracle call. Oracle $\mathcal{O}\mathsf{setup}$ generates an election key pair (and auxiliary information τ if $\beta = 1$) and returns election public key pk to adversary $\mathcal{A}$. Oracle $\mathcal{O}\mathsf{tally}$ returns the election result for bb_0 and the tallying proof, if $\beta = 0$, or a simulated proof, if $\beta = 1$. As expected, oracles $\mathcal{O}\mathsf{reg}$ and $\mathcal{O}\mathsf{corrupt}$ allow the adversary to register eligible voters and obtain the secret credentials of corrupt voters. As in BPRIV, adversary $\mathcal{A}$ can submit two votes on behalf of an honest voter via oracle $\mathcal{O}\mathsf{vote}$ and can submit ballots on behalf of corrupt voters via oracle $\mathcal{O}\mathsf{cast}$. CCFG is additionally equipped with oracle $\mathcal{O}\mathsf{RF}$ that allows adversary $\mathcal{A}$ to submit two ballots for corrupt voters. The left-hand ballot is appended to bb_0, and the right-hand ballot is appended to bb_1. At any time, adversary $\mathcal{A}$ can view bulletin board $\mathcal{BB}_\beta$ by calling oracle $\mathcal{O}\mathsf{board}$. The experiment terminates by returning a bit β' that is output by

adversary $\mathcal{A}$.

Definition 54 (CCFG [38]). *An e-voting scheme Γ for a result function f, set of voters $\mathcal{V}$, and set of candidates $\mathcal{C}$ satisfies* CCFG *if there exists algorithms* Sim *and* Sim_ρ *such that, for any PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{CCFG},0}_{\Gamma,\mathcal{A},\mathsf{Sim},\mathsf{Sim}_\rho,\mathcal{V},\mathcal{C}}(\lambda)=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{CCFG},1}_{\Gamma,\mathcal{A},\mathsf{Sim},\mathsf{Sim}_\rho,\mathcal{V},\mathcal{C}}(\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{CCFG},\beta}_{\Gamma,\mathcal{A},\mathsf{Sim},\mathsf{Sim}_\rho,\mathcal{V},\mathcal{C}}(\lambda)$ *is the experiment defined in Figure 4.5 for $\beta\in\{0,1\}$.*

$\mathsf{Exp}^{\mathsf{CCFG},\beta}_{\Gamma,\mathcal{A},\mathsf{Sim},\mathsf{Sim}_\rho,\mathcal{V},\mathcal{C}}(\lambda)$

$\mathsf{bb}_0\leftarrow();\ \mathsf{bb}_1\leftarrow();\ \mathbf{pk}\leftarrow();\ \mathbf{sk}\leftarrow()$
$\mathcal{Q}\mathsf{reg}\leftarrow\emptyset;\ \mathcal{Q}\mathsf{corrupt}\leftarrow\emptyset$
$\beta'\leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{setup},\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}()$
return β'

$\mathcal{O}\mathsf{setup}()$

if $\beta=0:\ (pk,sk)\leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
if $\beta=1:\ (pk,sk,\tau)\leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda)$
return pk

$\mathcal{O}\mathsf{reg}_{(pk,\mathcal{L},\mathcal{Q}\mathsf{reg},\mathbf{pk},\mathbf{sk})}(id)$

if $id\in\mathcal{Q}\mathsf{reg}$ **return** $\bot$
$(\mathbf{pk}[id],\mathbf{sk}[id])\leftarrow\!\!\$\ \mathsf{Register}(id,pk)$
$\mathcal{L}\leftarrow\mathcal{L}\cup\{\mathbf{pk}[id]\}$
$\mathcal{Q}\mathsf{reg}\leftarrow\mathcal{Q}\mathsf{reg}\cup\{id\}$
return $\mathbf{pk}[id]$

$\mathcal{O}\mathsf{corrupt}_{(\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{sk})}(id)$

if $id\notin\mathcal{Q}\mathsf{reg}$ **return** $\bot$
$\mathcal{Q}\mathsf{corrupt}\leftarrow\mathcal{Q}\mathsf{corrupt}\cup\{id\}$
return $(\mathbf{pk}[id],\mathbf{sk}[id])$

$\mathcal{O}\mathsf{board}_{(\mathsf{bb}_\beta)}()$

return $\mathsf{Publish}(\mathsf{bb}_\beta)$

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathbf{pk},\mathbf{sk})}(id,v_0,v_1)$

$b_0\leftarrow\!\!\$\ \mathsf{Vote}(v_0,\mathbf{pk}[id],\mathbf{sk}[id],pk)$
$b_1\leftarrow\!\!\$\ \mathsf{Vote}(v_1,\mathbf{pk}[id],\mathbf{sk}[id],pk)$
if $\mathsf{Valid}(b,\mathsf{bb}_\beta,\mathcal{L},pk)=0$ **return** $\bot$
$\mathsf{bb}_0\leftarrow\!\!\$\ \mathsf{Append}(b_0,\mathsf{bb}_0,\mathcal{L},pk)$
$\mathsf{bb}_1\leftarrow\!\!\$\ \mathsf{Append}(b_1,\mathsf{bb}_1,\mathcal{L},pk)$
return $\top$

$\mathcal{O}\mathsf{RF}_{(pk,\mathcal{L},\mathcal{Q}\mathsf{corrupt},\mathsf{bb}_0,\mathsf{bb}_1)}(id,b_0,b_1)$

if $id\notin\mathcal{Q}\mathsf{corrupt}$ **return** $\bot$
if $\mathsf{Valid}(b_0,\mathsf{bb}_0,\mathcal{L},pk)=0\ \vee\ \mathsf{Valid}(b_1,\mathsf{bb}_1,\mathcal{L},pk)=0$ **return** $\bot$
$\mathsf{bb}_0\leftarrow\!\!\$\ \mathsf{Append}(b_0,\mathsf{bb}_0,\mathcal{L},pk)$
$\mathsf{bb}_1\leftarrow\!\!\$\ \mathsf{Append}(b_1,\mathsf{bb}_1,\mathcal{L},pk)$
return $\top$

$\mathcal{O}\mathsf{cast}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1)}(id,b)$

if $\mathsf{Valid}(b,\mathsf{bb}_\beta,\mathcal{L},pk)=0$ **return** $\bot$
$\mathsf{bb}_0\leftarrow\!\!\$\ \mathsf{Append}(b,\mathsf{bb}_0,\mathcal{L},pk)$
$\mathsf{bb}_1\leftarrow\!\!\$\ \mathsf{Append}(b,\mathsf{bb}_1,\mathcal{L},pk)$
return $\top$

$\mathcal{O}\mathsf{tally}_{(sk,\tau,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1)}()$

if $\beta=0:\ (r,\rho)\leftarrow\!\!\$\ \mathsf{Tally}(\mathsf{bb}_0,\mathcal{L},sk)$
else
  $(r,\rho')\leftarrow\!\!\$\ \mathsf{Tally}(\mathsf{bb}_0,\mathcal{L},sk)$
  $\rho\leftarrow\!\!\$\ \mathsf{Sim}_\rho(\tau,\mathsf{bb}_1,r)$
return (r,ρ)

Figure 4.5: The receipt-freeness experiment $\mathsf{Exp}^{\mathsf{CCFG},\beta}_{\Gamma,\mathcal{A},\mathsf{Sim},\mathsf{Sim}_\rho,\mathcal{V},\mathcal{C}}(\lambda)$ for $\beta\in\{0,1\}$ from [38] and cast into our syntax.

4.4.2 Soundness of CCFG

Recall that, in Chapter 3.3.1, we discussed that BPRIV must be accompanied by two additional properties, namely, strong consistency and strong correctness, both of which are defined in [22]. These properties are necessary to capture attacks that can be used to break the ballot secrecy property of an e-voting scheme, and, without them, BPRIV

would be unsound. Indeed, in [22], it was shown that there exists e-voting schemes that are not intuitively ballot secret yet satisfy BPRIV. We refer the reader to Chapter 3.3.1 for an outline of the strong consistency and strong correctness properties and the attacks captured by these properties, and to the original definitions in [22] for a full explanation. As CCFG builds on BPRIV it must also be accompanied with these additional properties. Otherwise, there may exist schemes satisfying CCFG that do not even satisfy ballot secrecy. However, Chaidos *et al.* do not consider strong correctness or strong consistency in [38], which results in an unsound definition of receipt-freeness.

We believe that this is an oversight and can be corrected by presenting definitions of strong consistency and strong correctness for CCFG. Unfortunately, CCFG cannot simply adopt the original definitions by Bernhard *et al.*, because CCFG is defined over different syntax. In particular, CCFG is defined for e-voting schemes with internal credentials, whereas BPRIV is defined for schemes with external credentials. That being said, it was shown in [44] that syntax for e-voting schemes with internal credentials can be used to define these properties through straightforward modifications to the original definitions of strong consistency and strong correctness. Moreover, in Chapter 4.6, we show that we can define these properties in our syntax. Therefore, we believe that the soundness issue of CCFG can be remedied by adopting definitions of strong consistency and strong correctness similar to those presented in [44] and Chapter 4.6 of this thesis.

4.4.3 Completeness of CCFG

We now describe two completeness issues with CCFG. We uncover the first issue and the second issue builds upon a result from [44], which we elaborate on below.

CCFG limits the set of schemes that can be declared receipt-free We observe that CCFG is unsatisfiable by schemes for which $\mathsf{Append}(b, \mathsf{bb}, \mathcal{L}, pk)$ outputs $\mathsf{bb} \parallel b$ and $\mathsf{Publish}(\mathsf{bb})$ outputs bb. That is, $\mathsf{Append}(b, \mathsf{bb}, \mathcal{L}, pk)$ appends ballot b to ballot box bb without processing the ballot in any way and $\mathsf{Publish}(\mathsf{bb})$ outputs bb such that the ballot that appears on the public view of bb is identical to the ballot submitted by the voter. Formally, we have the following result.

Proposition 3. *Let* Γ *be an e-voting scheme for which* $\mathsf{Append}(b, \mathsf{bb}, \mathcal{L}, pk)$ *outputs* $\mathsf{bb} \parallel b$

$\mathcal{A}^{\mathcal{O}\mathsf{setup},\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}()$

$pk \leftarrow\!\$\ \mathcal{O}\mathsf{setup}();\ pk_{id} \leftarrow\!\$\ \mathcal{O}\mathsf{reg}(id);\ sk_{id} \leftarrow\!\$\ \mathcal{O}\mathsf{corrupt}(id)$
$v_0 \leftarrow\!\$\ \mathcal{C};\ v_1 \leftarrow\!\$\ \mathcal{C};\ b_0 \leftarrow\!\$\ \mathsf{Vote}(v_0, pk_{id}, sk_{id}, pk);\ b_1 \leftarrow\!\$\ \mathsf{Vote}(v_1, pk_{id}, sk_{id}, pk)$
$\top \leftarrow\!\$\ \mathcal{O}\mathsf{RF}(id, b_0, b_1);\ \mathcal{BB}_\beta \leftarrow\!\$\ \mathcal{O}\mathsf{board}()$
if $\mathcal{BB}_\beta = (b_0)$ **return** 0
if $\mathcal{BB}_\beta = (b_1)$ **return** 1

Figure 4.6: Adversary $\mathcal{A}$ that succeeds in the CCFG experiment for Γ in the proof of Proposition 3.

and Publish(bb) *outputs* bb. *Then* Γ *does not satisfy* CCFG.

Proof. We define an adversary $\mathcal{A}$ that succeeds in the CCFG experiment for scheme Γ in Figure 4.6. Adversary $\mathcal{A}$, upon obtaining the election public key from oracle $\mathcal{O}$setup, registers and corrupts a single voter by calling oracles $\mathcal{O}$reg and $\mathcal{O}$corrupt. Adversary $\mathcal{A}$ then constructs two ballots for two vote choices chosen randomly from the set of candidates $\mathcal{C}$ and submits the ballots via a query to $\mathcal{O}$RF. Then, adversary $\mathcal{A}$ queries $\mathcal{O}$board. Bulletin board $\mathcal{BB}_\beta$ that is returned to adversary $\mathcal{A}$ contains the single entry b_0 (if $\beta = 0$) or b_1 (if $\beta = 1$). Therefore, $\mathcal{A}$ can correctly guess β and e-voting scheme Γ does not satisfy CCFG. □

CCFG is unsatisfiable by these schemes because, in the CCFG experiment, the adversary submits two ballots to $\mathcal{O}$receipt. To satisfy CCFG, the adversary must be unable to distinguish a bulletin board that contains ballot b_0 and a bulletin board that contains ballot b_1, where the adversary queries $\mathcal{O}\mathsf{receipt}(id, b_0, b_1)$ in the CCFG game. This requires that ballots are modified in some way before they are appended to bb_0 and bb_1, or before $\mathcal{BB}_\beta$ is published. Otherwise, the adversary can trivially distinguish as shown in the proof of Proposition 3. Partly, CCFG excludes these schemes by design. Chaidos *et al.* acknowledge that a scheme satisfies CCFG only if it achieves receipt-freeness without the voter relying on an *evasion strategy* [38]. An evasion strategy is a procedure that an e-voting scheme provides that allows voters to evade coercion. Generally, schemes that provide voters with an evasion strategy do not rely on ballot modification but instead on the use of an evasion strategy to achieve receipt-freeness. This means that schemes that rely on evasion strategies to achieve receipt-freeness cannot satisfy CCFG despite the fact that they are receipt-free. For example, JCJ relies on fake credentials, a type of evasion strategy, to achieve receipt-freeness (cf. Chapter 4.2). Moreover, JCJ ballots are not modified before

they are appended to the ballot box and Publish(bb) outputs bb. Thus, we have the following corollary, which follows from Proposition 3.

Corollary 1. JCJ *does not satisfy* CCFG.

Intuitively receipt-free schemes are out of scope Recall that CCFG extends BPRIV to capture receipt-freeness. Additionally, CCFG is defined for schemes with internal voter registration, whereas BPRIV is defined for external voter registration. Within the internal credentials setting, CCFG captures adaptive voter corruption. In [44], Cortier *et al.* demonstrate that extending BPRIV to internal voter registration with adaptive voter corruption results in a definition that is not satisfiable by schemes for which algorithm Append accepts duplicate ballots. That is $\mathsf{Append}(b, \mathsf{bb}, \mathcal{L}, pk)$ appends ballot b to ballot box bb even if b appears as a previous entry on bb. We formalise the result from [44], showing that it also applies to CCFG.

Proposition 4. *Let* Γ *be an e-voting scheme that allows revoting and implements a last-vote-counts policy. Moreover, let algorithm* $\mathsf{Append}(b, \mathsf{bb}, \mathcal{L}, pk)$ *return* $\mathsf{bb} \parallel b$. *Then* Γ *does not satisfy* CCFG.

Proof. We define an adversary $\mathcal{A}$ that succeeds in the CCFG experiment for scheme Γ in Figure 4.7. Adversary $\mathcal{A}$ obtains the election public key from oracle $\mathcal{O}\mathsf{setup}$ and registers a single voter by calling oracle $\mathcal{O}\mathsf{reg}$. Then, adversary $\mathcal{A}$ selects two vote choices v_0 and v_1 randomly from the candidate set $\mathcal{C}$ and submits the choices to oracle $\mathcal{O}\mathsf{vote}$. Adversary $\mathcal{A}$ then queries $\mathcal{O}\mathsf{board}$ and submits the single ballot that appears on $\mathcal{BB}_\beta$ to oracle $\mathcal{O}\mathsf{cast}$. Finally $\mathcal{A}$ queries $\mathcal{O}\mathsf{tally}$ and outputs a bit β'. If the result includes a single ballot for v_0 (resp., v_1), then adversary $\mathcal{A}$ viewed a ballot for v_0 (resp., v_1) on $\mathcal{BB}_\beta$ and queried a ballot for v_0 (resp., v_1) to oracle $\mathcal{O}\mathsf{cast}$. Then, adversary $\mathcal{A}$ can correctly guess β and e-voting scheme Γ does not satisfy CCFG. □

As a result of Proposition 4, which is derived from the argument set out in [44], e-voting schemes that are intuitively receipt-free may be declared not to satisfy CCFG. This holds because an adversary can query $\mathcal{O}\mathsf{vote}(id, v_0, v_1)$ and then subsequently corrupt the voter, submitting the ballot that appears on bb_β on behalf of voter id to oracle $\mathcal{O}\mathsf{cast}$. However, as noted in [44], this does not translate to a real-world attack. For example, suppose a voter submits the same ballot multiple times. In that case, the election result does not

$\mathcal{A}^{\mathcal{O}\mathsf{setup},\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}()$

$pk \leftarrow_{\$} \mathcal{O}\mathsf{setup}();\ pk_{id} \leftarrow_{\$} \mathcal{O}\mathsf{reg}(id);\ v_0 \leftarrow_{\$} \mathcal{C};\ v_1 \leftarrow_{\$} \mathcal{C}$
$\top \leftarrow_{\$} \mathcal{O}\mathsf{vote}(id, v_0, v_1);\ sk_{id} \leftarrow_{\$} \mathcal{O}\mathsf{corrupt}(id);\ \mathcal{BB}_\beta = (b_\beta) \leftarrow_{\$} \mathcal{O}\mathsf{board}();\ \top \leftarrow_{\$} \mathcal{O}\mathsf{cast}(id, b_\beta)$
$(r, \rho) \leftarrow_{\$} \mathcal{O}\mathsf{tally}()$
if $r = f((id, v_0))$ **return** 0
if $r = f((id, v_1))$ **return** 1

Figure 4.7: Adversary $\mathcal{A}$ that succeeds in the CCFG experiment for scheme Γ in the proof of Proposition 4.

change, and the attacker does not learn any more information from multiple identical ballots than can be learned from a single ballot. Therefore, an e-voting scheme may be intuitively receipt-free yet cannot satisfy CCFG.

A solution to this completeness issue is to adopt the approach of [44] and define CCFG for static voter corruption. This can be achieved by defining adversary $\mathcal{A}$ in the CCFG experiment as $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. $\mathcal{A}_1$ has access to oracles $\mathcal{O}\mathsf{setup}$, $\mathcal{O}\mathsf{reg}$ and $\mathcal{O}\mathsf{corrupt}$ and outputs state information. Then, $\mathcal{A}_2$ obtains state information from $\mathcal{A}_1$ and, with access to the remaining oracles defined for the CCFG experiment, outputs a bit β'.

4.4.4 Further Discussion

CCFG captures the attack scenario in which an honest voter constructs their ballot and gives the attacker the randomness used (or possibly uses randomness provided by the attacker) to construct their ballot. The attacker can use this information to reconstruct the ballot locally and check whether the ballot appears on the bulletin board. CCFG captures this scenario through the oracle $\mathcal{O}\mathsf{receipt}$, which allows the adversary to construct ballots on behalf of voters and then submit these ballots to $\mathcal{O}\mathsf{receipt}$. The adversary can then view $\mathcal{BB}_\beta$, and expects to see a ballot corresponding to one of those submitted to $\mathcal{O}\mathsf{receipt}$.

Reflecting on the discussion at the end of Chapter 4.3, it is clear that CCFG captures a stronger attack model than KZZ. In particular, Chaidos *et al.* consider a voter that cooperates with an attacker (e.g., by using randomness provided by the attacker and providing the attacker with a secret voting credential *before* voting) to fall within the scope of receipt-freeness. The literature suggests that such an attacker is beyond the scope of receipt-freeness [52]. However, CCFG is *not* a definition of coercion-resistance. Specifically,

CCFG does not capture abstention attacks, whereby an attacker can prevent a voter from casting a ballot, which are captured by coercion-resistance. This suggests that there is no strict boundary between receipt-freeness and coercion-resistance. In Chapter 4.6, we prove results that demonstrate relations between these two properties.

4.5 Receipt-Freeness by Bernhard, Kulyk & Volkamer (BKV)

In this section, we analyse a definition of receipt-freeness by Bernhard *et al.* [24, 25] for schemes that use *deniable vote updating*, which we call BKV. Bernhard *et al.* construct a definition of receipt-freeness for KTV-Helios [93], a variant of the Helios e-voting scheme that uses deniable vote updating whereby a voter casts a ballot, and then changes their vote, without an attacker detecting the change. In [25] it was recognized that CCFG does not apply to KTV-Helios because deniable vote updating is a type of evasion strategy and the strategy is required to achieve receipt-freeness. Therefore, Bernhard *et al.* introduce a new receipt-freeness definition that modifies CCFG to schemes that use deniable vote updating.

4.5.1 The BKV Experiment

BKV captures the following idea: the attacker should be unable to distinguish a voter who submits a vote and a voter who submits the same vote but then deniably updates their vote. BKV considers e-voting schemes with timestamps such that algorithm Vote is redefined to take additional input of a timestamp t, indicating the time at which a ballot is to be cast.

We cast BKV into our syntax in Definition 55 and describe the BKV experiment in Figure 4.8. BKV relies on algorithms Sim and Sim_ρ (cf. Chapter 3.3.1) and, additionally, algorithms Upd and Obf such that:

$\mathsf{Upd}(v_0, v_1, sk_{id}, t_u, pk)$ On input of votes v_0, v_1, private credential sk_{id}, timestamp t_u from some probability space $\mathbb{P}$ and public key pk, algorithm Upd outputs a ballot that updates a vote for v_0 to a vote for v_1 at timestamp t_u.

$\mathsf{Obf}(\mathsf{bb}, id)$ On input of ballot box bb and voter id, algorithm Obf casts dummy ballots for voter id to hide ballots cast by id in the event that id updates their vote, and outputs the updated ballot box.

The purpose of these algorithms is that, if a voter is coerced to vote for v_0, the voter can cast this ballot and then deniably update their ballot to a vote for a different candidate. Then, to ensure that the attacker cannot detect such updating (i.e., in the event that more than one ballot appears on behalf of the coerced voter), an authority can cast dummy ballots on behalf of the coerced voter, ensuring that an attacker cannot determine whether subsequent ballots are genuine updates output by the voter, or dummy ballots posted by an authority.

With these algorithms, BKV is described as follows. Ballot boxes bb_0 and bb_1 are initialised as empty and lists **pk** and **sk** are initialised to track the public and secret credentials of registered voters. An election key pair is generated and all $|\mathcal{V}|$ eligible voters are registered. Adversary $\mathcal{A}$ is input pk, list $\mathcal{L}$ and bulletin board $\mathcal{BB}_\beta$. Adversary $\mathcal{A}$ can query a number of oracles. Oracles $\mathcal{O}\mathsf{vote}$, $\mathcal{O}\mathsf{cast}$ and $\mathcal{O}\mathsf{tally}$ operate identically to the oracles in the CCFG experiment. Oracle $\mathcal{O}\mathsf{RF}$ allows adversary $\mathcal{A}$ to submit two votes on behalf of a voter. A ballot is computed for the left-hand vote and appended to both ballot boxes, and a ballot that updates the left-hand vote to the right-hand vote is appended to bb_1. Oracle $\mathcal{O}\mathsf{RF}$ then runs algorithm Obf on both ballot boxes. The experiment terminates by returning a bit β' that is output by adversary $\mathcal{A}$.

Definition 55 (BKV [24, 25])**.** *An e-voting scheme with timestamps* Γ *for a result function* f*, set of voters* $\mathcal{V}$*, and set of candidates* $\mathcal{C}$ *satisfies* BKV *if there exists algorithms* Sim*,* Sim_ρ*,* Upd *and* Obf *such that, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{BKV},0}_{\Gamma,\mathcal{A},\mathsf{Sim},\mathsf{Sim}_\rho,\mathsf{Upd},\mathsf{Obf},\mathcal{V},\mathcal{C}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{BKV},1}_{\Gamma,\mathcal{A},\mathsf{Sim},\mathsf{Sim}_\rho,\mathsf{Upd},\mathsf{Obf},\mathcal{V},\mathcal{C}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{BKV},\beta}_{\Gamma,\mathcal{A},\mathsf{Sim},\mathsf{Sim}_\rho,\mathsf{Upd},\mathsf{Obf},\mathcal{V},\mathcal{C}}(\lambda)$ *is the experiment defined in Figure 4.8 for* $\beta \in \{0, 1\}$*.*

$\mathsf{Exp}^{\mathsf{BKV},\beta}_{\Gamma,\mathcal{A},\mathsf{Sim},\mathsf{Sim}_\rho,\mathsf{Upd},\mathsf{Obf},\mathcal{V},\mathcal{C}}(\lambda)$

$\mathsf{bb}_0 \leftarrow (); \ \mathsf{bb}_1 \leftarrow (); \ \mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow ()$
if $\beta = 0$: $(pk, sk) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
if $\beta = 1$: $(pk, sk, \tau) \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda)$
for $i = 1, \ldots, |\mathcal{V}|$: $(\mathbf{pk}[id_i], \mathbf{sk}[id_i]) \leftarrow\!\!\$\ \mathsf{Register}(id_i, pk)$
$\mathcal{L} \leftarrow \{\mathbf{pk}[id_1], \ldots, \mathbf{pk}[id_{|\mathcal{V}|}]\}$
$\beta' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{tally}}(pk, \mathcal{L}, \mathcal{BB}_\beta)$
return β'

$\mathcal{O}\mathsf{RF}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathbf{pk},\mathbf{sk})}(id, v_0, v_1, t)$

if $v_0 \notin \mathcal{C} \ \vee \ v_1 \notin \mathcal{C}$ **return** $\bot$
$b_0 \leftarrow\!\!\$\ \mathsf{Vote}(v_0, \mathbf{pk}[id], \mathbf{sk}[id], t, pk)$
$\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_0, \mathcal{L}, pk)$
$\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_1, \mathcal{L}, pk)$
$t_u \leftarrow\!\!\$\ \mathbb{P}$
$b_1 \leftarrow\!\!\$\ \mathsf{Upd}(v_0, v_1, \mathbf{sk}[id], t_u, pk)$
$\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_1, \mathsf{bb}_1, \mathcal{L}, pk)$
$\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Obf}(\mathsf{bb}_0, id)$
$\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Obf}(\mathsf{bb}_1, id)$
return $\top$

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathbf{pk},\mathbf{sk})}(id, v_0, v_1, t)$

$b_0 \leftarrow \mathsf{Vote}(v_0, \mathbf{pk}[id], \mathbf{sk}[id], t, pk)$
$b_1 \leftarrow \mathsf{Vote}(v_1, \mathbf{pk}[id], \mathbf{sk}[id], t, pk)$
if $\mathsf{Valid}(b_\beta, \mathsf{bb}_\beta, \mathcal{L}, pk) = 0$ **return** $\bot$
$\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_0, \mathcal{L}, pk)$
$\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_1, \mathsf{bb}_1, \mathcal{L}, pk)$
return $\top$

$\mathcal{O}\mathsf{cast}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1)}(id, b)$

if $\mathsf{Valid}(b, \mathsf{bb}_\beta, \mathcal{L}, pk) = 0$ **return** $\bot$
$\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b, \mathsf{bb}_0, \mathcal{L}, pk)$
$\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b, \mathsf{bb}_1, \mathcal{L}, pk)$
return $\top$

$\mathcal{O}\mathsf{tally}_{(sk,\tau,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1)}()$

if $\beta = 0$: $(r, \rho) \leftarrow\!\!\$\ \mathsf{Tally}(\mathsf{bb}_0, \mathcal{L}, sk)$
else
  $(r, \rho') \leftarrow\!\!\$\ \mathsf{Tally}(\mathsf{bb}_0, \mathcal{L}, sk)$
  $\rho \leftarrow\!\!\$\ \mathsf{Sim}_\rho(\tau, \mathsf{bb}_1, r)$
return (r, ρ)

Figure 4.8: The receipt-freeness experiment $\mathsf{Exp}^{\mathsf{BKV},\beta}_{\Gamma,\mathcal{A},\mathsf{Sim},\mathsf{Sim}_\rho,\mathsf{Upd},\mathsf{Obf},\mathcal{V},\mathcal{C}}(\lambda)$ for $\beta \in \{0, 1\}$ from [24, 25] and cast into our syntax.

4.5.2 Soundness and Completeness of BKV

We did not find any soundness issues with BKV. In particular, although BKV uses the same framework as CCFG, BKV does not overlook the need for strong consistency and strong correctness and defines these properties in their syntax in [24]. Clearly, BKV is incomplete because it limits the class of e-voting schemes that can be declared receipt-free to schemes with timestamps that achieve receipt-freeness through the use of deniable vote updating, although this is by design.

4.5.3 Further Discussion

BKV models an attack scenario in which voters submit a ballot and then either deniably update their vote, or do not. BKV does not model a voter who interacts with an attacker to prove their vote. That is, there is no mechanism to capture the fact that a voter may try to pass their credentials or randomness to an attacker either before or after voting. Certainly, BKV does not pose any issues with respect to whether it captures attack

scenarios that should be considered under the heading of coercion-resistance. However, it does raise questions about whether this definition captures receipt-freeness. Recall that receipt-freeness is the property that a voter cannot prove their vote. BKV captures e-voting schemes with an evasion strategy (i.e., deniable vote updating) but does not provide a mechanism for a voter to attempt to prove their vote. As such, we conclude that receipt-freeness is guaranteed under the assumption that the voter does not pass any proof of their vote to the attacker.

4.6 New Definitions of Receipt-Freeness

We are now ready to present new definitions of receipt-freeness. Like CCFG and BKV, our definitions build upon BPRIV. As such, our definitions inherits the benefits of the BPRIV approach (cf. Chapter 3.3.1). Additionally, our definitions adopt the intuition of KZZ, capturing a voter that submits a vote and proves that they submitted that vote or proves that they submitted a different vote.

Our definitions take on board the lessons learned from our analysis of existing definitions, avoiding the soundness and completeness issues discussed in Chapters 4.3 – 4.5. Notably, our definitions do not require that all voters submit a single ballot, and we define the additional properties of strong correctness and strong consistency in our syntax to accompany our receipt-freeness definitions. Hence, we avoid the soundness issues of KZZ and CCFG respectively. Moreover, we also address the completeness issues of existing definitions. First, our definitions are restricted to *static* corruption of voters only, avoiding the second completeness issue of CCFG. Second, we address the common limitation that existing definitions are restricted to schemes that do not require an evasion strategy (i.e., KZZ and CCFG) or schemes that require a specific evasion strategy (i.e., deniable vote updating in BKV) by presenting two variants of receipt-freeness. We call our first variant N-RF, indicating *non*-evasion receipt-freeness, that, as the name suggests, captures schemes that do not require an evasion strategy. Our second variant is intended to capture a range of evasion strategies. We call this variant E-RF to indicate evasion receipt-freeness.

In this section, we proceed as follows. First, we present our definitions within a common framework and show that they capture a broad class of e-voting schemes, particularly

focusing on how E-RF captures well-known evasion strategies from the literature. We then analyse our definitions. First, we show that N-RF is stronger than E-RF and discuss the implications of this relationship. Then, we compare our definitions with BPRIV, showing that our definitions capture the ballot secrecy property, which is required of any receipt-freeness definition. Finally, we draw comparisons between our definitions and existing receipt-freeness definitions, in particular, discussing the relationship between N-RF, KZZ and CCFG.

4.6.1 Our Receipt-Freeness Definitions (N-RF and E-RF)

Our receipt-freeness definitions, N-RF and E-RF, describe an experiment in which an adversary is presented with a view of a real or a fake election (depending on a bit β) and a result computed for the real election. During the experiment, the adversary can submit two votes (the real and fake vote) on behalf of honest voters, and ballots on behalf of corrupted voters. Additionally, the adversary can submit two votes on behalf of honest voters and receives a receipt, which captures the evidence that a voter presents to an attacker to prove their vote. To satisfy N-RF and E-RF, we require that the adversary cannot determine whether the bulletin board contains real or fake votes of honest voters.

We formally define the N-RF and E-RF experiments in Figure 4.9. Our experiments rely on algorithms Sim and Sim_ρ, as defined for BPRIV(cf. Chapter 3.3.1) and, additionally, simulator $\mathcal{S}$ such that

$\mathcal{S}(pk, b, v)$ On input of election public key pk, ballot b and vote v, simulator $\mathcal{S}$ outputs a receipt Rcpt indicating that b is a ballot encoding vote v.

With these algorithms, we define N-RF and E-RF for an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. As we shall see, the N-RF and E-RF experiments are identical, with the exception of a single oracle $\mathcal{O}\mathsf{RF}$. The experiments can be described as follows. Ballot boxes bb_0 and bb_1 are initialised as empty. Lists **pk** and **sk**, which track public and secret credentials of voters respectively are set to empty. Additionally, sets $\mathcal{L}$, $\mathcal{Q}\mathsf{reg}$ and $\mathcal{Q}\mathsf{corrupt}$, which track public credentials of registered voters, identities of registered voters, and identities of corrupt voters respectively, are initialised as empty sets. A flag flag is initialised as 0. We use flag to ensure that $\mathcal{O}\mathsf{tally}$ is the final oracle query that an adversary can make. That is,

flag is updated to 1 following a query to $\mathcal{O}\mathsf{tally}$ and all other oracles return $\perp$ if $\mathsf{flag} = 1$. The election key pair (pk, sk) (and auxiliary information τ if $\beta = 1$) is generated and the election public key is input to adversary $\mathcal{A}_1$. Adversary $\mathcal{A}_1$ can query $\mathcal{O}\mathsf{reg}$, which is described as follows.

$\mathcal{O}\mathsf{reg}_{(pk,\mathcal{L},\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk})}(id,d)$ registers an eligible voter and, if $d = 1$, corrupts the voter. If id is an eligible but unregistered voter, oracle $\mathcal{O}\mathsf{reg}$ runs algorithm $\mathsf{Register}$ to generate a credential pair (pk_{id}, sk_{id}). A set $\mathcal{Q}\mathsf{reg}$ is updated to include the voter's identity id. If $d = 0$, the oracle returns the voter's public credential pk_{id} to $\mathcal{A}_1$. If $d = 1$, a set $\mathcal{Q}\mathsf{corrupt}$ is updated to include id, and the oracle returns the credential pair (pk_{id}, sk_{id}) to $\mathcal{A}_1$.

Adversary $\mathcal{A}_1$ outputs state information that is input to adversary $\mathcal{A}_2$, which can query several oracles that we now describe.

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)$ generates ballots b_0 and b_1 for v_0 and v_1 respectively, and appends the ballots to bb_0 and bb_1. If voter id is registered but not corrupt, oracle $\mathcal{O}\mathsf{vote}$ runs algorithm Vote on behalf of voter id to generate ballots b_0 and b_1. If ballot b_β is valid with respect to ballot box bb_β, oracle $\mathcal{O}\mathsf{vote}$ runs algorithm Append for both ballots.

$\mathcal{O}\mathsf{cast}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{corrupt},\mathsf{flag})}(id, b)$ submits a ballot on behalf of a voter. If voter id is corrupt and ballot b is valid with respect to ballot box bb_β, oracle $\mathcal{O}\mathsf{cast}$ runs algorithm Append to append ballot b to ballot boxes bb_0 and bb_1.

$\mathcal{O}\mathsf{RF}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)//\mathsf{N\text{-}RF}$ generates and submits ballots b_0 and b_1 for v_0 and v_1 respectively, and outputs a receipt Rcpt. If voter id is registered but not corrupt, oracle $\mathcal{O}\mathsf{vote}$ runs algorithm Vote on behalf of voter id to generate ballots b_0 and b_1. If ballot b_β is valid with respect to ballot box bb_β, oracle $\mathcal{O}\mathsf{vote}$ runs algorithm Append for both ballots. $\mathcal{O}\mathsf{RF}$ generates a receipt Rcpt_0 that includes vote v_0, the credential of voter id, the randomness used to compute the ballot coins and the ballot b_0. That is, Rcpt_0 includes all material used to construct ballot b_0. Then, $\mathcal{O}\mathsf{RF}$ runs simulator $\mathcal{S}$ to generate receipt Rcpt_1, a simulated receipt such that b_1 appears to encode a vote for v_0. Finally, $\mathcal{O}\mathsf{RF}$ returns Rcpt_β to $\mathcal{A}_2$.

$\mathcal{O}\mathsf{RF}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)//\mathsf{E\text{-}RF}$ generates and submits a ballot b_0

for v_0, implements the voter's evasion strategy, and outputs a receipt Rcpt. If voter id is registered but not corrupt, oracle $\mathcal{O}\mathsf{vote}$ runs algorithm Vote on behalf of voter id to generate ballots b_0 and runs algorithm Append to append b_0 to ballot box bb_0. $\mathcal{O}\mathsf{RF}$ then runs algorithm $\mathsf{CoercedVote}$ that is defined as follows.

$\mathsf{CoercedVote}(v_0, v_1, pk_{id}, sk_{id}, b_0, pk, \mathsf{coins})$ On input of votes v_0, v_1, credentials pk_{id} and sk_{id}, ballot b_0, election public key pk and randomness used to construct ballot b_0 coins, algorithm $\mathsf{CoercedVote}$ returns ballots b_1, b_2 and b_3. Below, we provide concrete examples of algorithm $\mathsf{CoercedVote}$.

$\mathcal{O}\mathsf{RF}$ then runs algorithm Append to append b_2 to bb_0, and b_1, b_3 to bb_1. Then, receipts Rcpt_0 and Rcpt_1 are generated as in $\mathcal{O}\mathsf{RF}$ for $\mathsf{N\text{-}RF}$. Finally, $\mathcal{O}\mathsf{RF}$ returns Rcpt_β to $\mathcal{A}_2$.

$\mathcal{O}\mathsf{board}_{(\mathsf{bb}_\beta)}()$ returns bulletin board $\mathcal{BB}_\beta$ to $\mathcal{A}_2$.

$\mathcal{O}\mathsf{tally}_{(sk,\tau,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathsf{flag})}()$ computes and returns the election result and tallying proof. Oracle $\mathcal{O}\mathsf{tally}$ runs algorithm Tally to compute the result r and tallying proof ρ for ballot box bb_0. If $\beta = 0$, $\mathcal{O}\mathsf{tally}$ returns the result and proof to $\mathcal{A}_2$. If $\beta = 1$, algorithm Sim_ρ is run to compute a simulated proof such that result r appears to be computed for ballot box bb_1. The result and simulated proof are returned to $\mathcal{A}_2$.

Adversary $\mathcal{A}_2$ outputs a bit that is returned by the experiment.

Definition 56 ($\mathsf{X\text{-}RF}$). *An e-voting scheme Γ for a result function f, set of voters $\mathcal{V}$, and set of candidates $\mathcal{C}$ satisfies* $\mathsf{X\text{-}RF}$ *for* $\mathsf{X} \in \{\mathsf{N}, \mathsf{E}\}$ *if there exists a simulator $\mathcal{S}$ and algorithms* Sim *and* Sim_ρ *(and, if* $\mathsf{X} = \mathsf{E}$*, algorithm* $\mathsf{CoercedVote}$*) such that, for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* negl *such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{X\text{-}RF},0}_{\Gamma,\mathcal{A},\mathcal{S},\mathsf{Sim},\mathsf{Sim}_\rho,\mathcal{V},\mathcal{C}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{X\text{-}RF},1}_{\Gamma,\mathcal{A},\mathcal{S},\mathsf{Sim},\mathsf{Sim}_\rho,\mathcal{V},\mathcal{C}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{X\text{-}RF},\beta}_{\Gamma,\mathcal{A},\mathcal{S},\mathsf{Sim},\mathsf{Sim}_\rho,\mathcal{V},\mathcal{C}}(\lambda)$ *is the experiment defined in Figure 4.9 for* $\beta \in \{0,1\}$ *and* $\mathsf{X} \in \{\mathsf{N}, \mathsf{E}\}$.

Applying our definitions. Our definitions can be applied to various strategies used to achieve receipt-freeness. Firstly, if a scheme does not require an evasion strategy to achieve receipt-freeness (i.e., the voter can simply cast their ballot and provide fake evidence to an

$\underline{\mathsf{Exp}^{\mathsf{X\text{-}RF},\beta}_{\Gamma,\mathcal{A},\mathcal{S},\mathsf{Sim},\mathsf{Sim}_\rho,\mathcal{V},\mathcal{C}}(\lambda)}$

$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow (); \ \mathsf{bb}_0 \leftarrow (); \ \mathsf{bb}_1 \leftarrow ()$
$\mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset$
$\mathsf{flag} \leftarrow 0$
if $\beta = 0$: $(pk, sk) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
if $\beta = 1$: $(pk, sk, \tau) \leftarrow\!\!\$\ \mathsf{Sim}(1^\lambda)$
$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}(st)$
return β'

$\underline{\mathcal{O}\mathsf{reg}_{(pk,\mathcal{L},\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk})}(id, d)}$

if $id \notin \mathcal{V} \ \vee \ id \in \mathcal{Q}\mathsf{reg}$ **return** $\perp$
$(\mathbf{pk}[id], \mathbf{sk}[id]) \leftarrow\!\!\$\ \mathsf{Register}(id, pk)$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathbf{pk}[id]\}$
$\mathcal{Q}\mathsf{reg} \leftarrow \mathcal{Q}\mathsf{reg} \cup \{id\}$
if $d = 0$: **return** $\mathbf{pk}[id]$
if $d = 1$
  $\mathcal{Q}\mathsf{corrupt} \leftarrow \mathcal{Q}\mathsf{corrupt} \cup \{id\}$
  return $(\mathbf{pk}[id], \mathbf{sk}[id])$
else return $\perp$

$\underline{\mathcal{O}\mathsf{cast}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{corrupt},\mathsf{flag})}(id, b)}$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
if $\mathsf{Valid}(b, \mathsf{bb}_\beta, \mathcal{L}, pk) = 0$ **return** $\perp$
$\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b, \mathsf{bb}_0, \mathcal{L}, pk)$
$\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b, \mathsf{bb}_1, \mathcal{L}, pk)$
return $\top$

$\underline{\mathcal{O}\mathsf{board}_{(\mathsf{bb}_\beta)}()}$

return $\mathsf{Publish}(\mathsf{bb}_\beta)$

$\underline{\mathcal{O}\mathsf{tally}_{(sk,\tau,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathsf{flag})}()}$

$\mathsf{flag} \leftarrow 1$
if $\beta = 0$: $(r, \rho) \leftarrow\!\!\$\ \mathsf{Tally}(\mathsf{bb}_0, \mathcal{L}, sk)$
if $\beta = 1$
  $(r, \rho') \leftarrow\!\!\$\ \mathsf{Tally}(\mathsf{bb}_0, \mathcal{L}, sk)$
  $\rho \leftarrow\!\!\$\ \mathsf{Sim}_\rho(\tau, \mathsf{bb}_1, r)$
return (r, ρ)

$\underline{\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)}$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
$b_0 \leftarrow\!\!\$\ \mathsf{Vote}(v_0, \mathbf{pk}[id], \mathbf{sk}[id], pk)$
$b_1 \leftarrow\!\!\$\ \mathsf{Vote}(v_1, \mathbf{pk}[id], \mathbf{sk}[id], pk)$
if $\mathsf{Valid}(b_\beta, \mathsf{bb}_\beta, \mathcal{L}, pk) = 0$ **return** $\perp$
$\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_0, \mathcal{L}, pk)$
$\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_1, \mathsf{bb}_1, \mathcal{L}, pk)$
return $\top$

$\underline{\mathcal{O}\mathsf{RF}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1) \backslash\backslash \mathsf{N\text{-}RF}}$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
$b_0 \leftarrow\!\!\$\ \mathsf{Vote}(v_0, \mathbf{pk}[id], \mathbf{sk}[id], pk; \mathsf{coins})$
$b_1 \leftarrow\!\!\$\ \mathsf{Vote}(v_1, \mathbf{pk}[id], \mathbf{sk}[id], pk)$
$\mathsf{Rcpt}_0 \leftarrow (v_0, \mathbf{pk}[id], \mathbf{sk}[id], \mathsf{coins}, b_0)$
$\mathsf{Rcpt}_1 \leftarrow\!\!\$\ \mathcal{S}(pk, b_1, v_0)$
if $\mathsf{Valid}(b_\beta, \mathsf{bb}_\beta, \mathcal{L}, pk) = 0$ **return** $\perp$
$\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_0, \mathcal{L}, pk)$
$\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_1, \mathsf{bb}_1, \mathcal{L}, pk)$
return Rcpt_β

$\underline{\mathcal{O}\mathsf{RF}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1) \backslash\backslash \mathsf{E\text{-}RF}}$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
$b_0 \leftarrow\!\!\$\ \mathsf{Vote}(v_0, \mathbf{pk}[id], \mathbf{sk}[id], pk; \mathsf{coins})$
$\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_0, \mathcal{L}, pk)$
$(b_1, b_2, b_3) \leftarrow\!\!\$\ \mathsf{CoercedVote}(v_0, v_1, \mathbf{pk}[id], \mathbf{sk}[id], b_0, pk, \mathsf{coins})$
$\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_1, \mathsf{bb}_1, \mathcal{L}, pk)$
$\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_2, \mathsf{bb}_0, \mathcal{L}, pk)$
$\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_3, \mathsf{bb}_1, \mathcal{L}, pk)$
$\mathsf{Rcpt}_0 \leftarrow (v_0, \mathbf{pk}[id], \mathbf{sk}[id], \mathsf{coins}, b_0)$
$\mathsf{Rcpt}_1 \leftarrow\!\!\$\ \mathcal{S}(\mathsf{Rcpt}_0, B)$
return Rcpt_β

Figure 4.9: The receipt-freeness experiment $\mathsf{Exp}^{\mathsf{X\text{-}RF},\beta}_{\Gamma,\mathcal{A},\mathcal{S},\mathsf{Sim},\mathsf{Sim}_\rho,\mathcal{V},\mathcal{C}}(\lambda)$ for $\mathsf{X} \in \{\mathsf{N}, \mathsf{E}\}$ and $\beta \in \{0, 1\}$.

attacker), then our N-RF variant can be used to determine whether the scheme satisfies receipt-freeness. If the e-voting scheme furnishes voters with an evasion strategy, E-RF should be used in place of N-RF. We show that E-RF captures schemes that use two

popular evasion strategies from the literature: deniable vote updating and the use of fake credentials.

Recall from Chapter 4.5 that the **deniable vote updating** strategy allows voters to submit ballots for a vote, say v_0, and then deniably update their ballot to a different vote, say v_1. For this evasion strategy to be successful, a receipt-freeness attacker should be unable to distinguish a voter that deniably updates their vote from a voter that does not. We show that algorithm $\mathsf{CoercedVote}$ can be defined to capture deniable vote updating. We adopt the deniable updating strategy of [25, 100], requiring that, to deniably update a vote, voters produce a second ballot for $v_1 - v_0$ that cancels their original ballot for v_0. Algorithm $\mathsf{CoercedVote}$ is defined as follows.

$\underline{\mathsf{CoercedVote}(v_0, v_1, pk_{id}, sk_{id}, b_0, pk, \mathsf{coins})}$

$b_1 \leftarrow\!\!\$\ \mathsf{Vote}(v_0, pk_{id}, sk_{id}, pk; \mathsf{coins})$

$b_2 \leftarrow\!\!\$\ \mathsf{Vote}(\epsilon, pk_{id}, sk_{id}, pk)$

$b_3 \leftarrow\!\!\$\ \mathsf{Vote}(v_1 - v_0, pk_{id}, sk_{id}, pk)$

return (b_1, b_2, b_3)

With algorithm $\mathsf{CoercedVote}$ defined in this way, an adversary in the $\mathsf{E\text{-}RF}$ experiment is presented with one of two views of the bulletin board as follows. If $\beta = 0$, the adversary views a bulletin board that contains of a ballot b_0 for v_0 and a dummy ballot for an empty string b_2. This captures the scenario that a voter does not use their evasion strategy, but the e-voting scheme casts a dummy ballot to hide whether or not deniable vote updating was applied. If $\beta = 1$, the adversary views a bulletin board that consists of a ballot b_1 for v_0 and a ballot, denoted b_3, that deniably updates b_1 to a vote for v_1. Then, if $\beta = 1$, the adversary views a fake election in which the voter applies their evasion strategy. Moreover, the adversary is presented with a receipt. If $\beta = 1$, we define the receipt as

$$\mathsf{Rcpt}_1 = (v_0, pk_{id}, sk_{id}, \mathsf{coins}, b_1).$$

That is, regardless of the value of β, the adversary is presented with a receipt that allows the adversary to reconstruct the first ballot posted to bb_β by oracle $\mathcal{O}\mathsf{RF}$.

Our second example captures **fake credentials** as defined for JCJ. This strategy permits a voter to produce a fake secret credential (by running algorithm $\mathsf{FakeKey}$) and submit a

ballot for vote v_0 using their fake credential. Then, the voter can submit a second ballot for vote v_1 using their real credential. For this strategy to be successful, it must be the case that an attacker cannot distinguish ballots generated with real secret credentials from ballots generated with fake secret credentials. Specifically, we define algorithm CoercedVote below.

$\underline{\mathsf{CoercedVote}(v_0, v_1, pk_{id}, sk_{id}, b_0, pk, \mathsf{coins})}$

$sk'_{id} \leftarrow\!\!\$\ \mathsf{FakeKey}(pk, pk_{id}, sk_{id})$

$b_1 \leftarrow\!\!\$\ \mathsf{Vote}(v_0, pk_{id}, sk'_{id}, pk; \mathsf{coins}')$

$b_2 \leftarrow\!\!\$\ \mathsf{Vote}(v \leftarrow\!\!\$\ \mathcal{C}, pk_{id}, sk'_{id}, pk)$

$b_3 \leftarrow\!\!\$\ \mathsf{Vote}(v_1, pk_{id}, sk_{id}, pk)$

return (b_1, b_2, b_3)

For JCJ the view of the adversary in the E-RF experiment is as follows. If $\beta = 0$, the bulletin board returned to the adversary contains a ballot b_0 for v_0 generated with the voter's real credential and a ballot b_2 for some v chosen randomly from the set of all possible candidates and using the voter's fake secret credential. We require that b_2 is posted to bb_1 as, otherwise, bulletin board $\mathcal{BB}_0$ will contain less ballots than $\mathcal{BB}_1$. In this event, the adversary can trivially output $\beta' = \beta$ in the E-RF experiment. However, by using the fake credential to generate ballot b_2, we ensure that b_2 is not included in the election result. As such, if $\beta = 0$, the adversary views an election in which the voter does not apply their evasion strategy. If $\beta = 1$, the adversary views a bulletin board that contains a ballot b_1 for v_0, generated with the voter's fake secret credential, and a ballot b_3 for v_1 generated with the voter's real secret credential. That is to say, if $\beta = 1$, the adversary views an election in which the voter applies their evasion strategy. We define the receipt returned to the adversary if $\beta = 1$ as

$$\mathsf{Rcpt}_1 = (v_0, pk_{id}, sk'_{id}, \mathsf{coins}, b_1).$$

Therefore, the adversary is presented with a receipt that contains all the material used to construct ballot b_β.

The above examples of deniable vote updating and fake credentials illustrate that E-RF can be defined to capture established evasion strategies from the literature. Moreover, as we shall see in Chapter 4.6.2, JCJ satisfies E-RF with algorithm CoercedVote and receipt

Rcpt_1 defined as above. However, before presenting this result, we introduce the additional properties strong correctness and strong consistency for our receipt-freeness definitions.

Additional properties for our definitions. To ensure soundness of our definitions, we define the strong correctness and strong consistency properties from [22] in our syntax. Recall that strong correctness is the property that requires that, with respect to a malicious ballot box, ballots generated via algorithm Vote are valid.

Definition 57 (Strong correctness)**.** *An e-voting scheme* Γ *for a result function* f*, set of voters* $\mathcal{V}$*, and set of candidates* $\mathcal{C}$ *satisfies* strong correctness *if, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\begin{array}{l}\mathbf{pk}\leftarrow();\ \mathbf{sk}\leftarrow();\\ (pk,sk)\leftarrow\!\$\ \mathsf{Setup}(1^\lambda);\\ \textbf{for } i=1,\ldots,|\mathcal{V}|:\ (\mathbf{pk}[id_i],\mathbf{sk}[id_i])\leftarrow\!\$\ \mathsf{Register}(id_i,pk);\\ \mathcal{L}\leftarrow\{\mathbf{pk}[id_1],\ldots,\mathbf{sk}[id_{|\mathcal{V}|}]\};\\ (id,v,\mathsf{bb})\leftarrow\!\$\ \mathcal{A}(pk,\mathbf{pk},\mathbf{sk});\\ b\leftarrow\!\$\ \mathsf{Vote}(v,\mathbf{pk}[id],\mathbf{sk}[id],pk)\end{array}:\ id\in\mathcal{V}\ \wedge\ \mathsf{Valid}(b,\mathsf{bb},\mathcal{L},sk)\coloneqq 0\right]\leq\mathsf{negl}(\lambda)\,.$$

Strong consistency requires that all valid ballots are included in the result output by algorithm Tally. It relies on the existence of two algorithms Extract and ValidInd that are defined as follows.

$\mathsf{Extract}(b, sk)$ On input of a ballot b and election secret key sk, algorithm Extract outputs a tuple (pk_{id}, v) where v is the vote encoded in ballot b that is produced by a voter with public credential pk_{id}.

$\mathsf{ValidInd}(b, pk)$ On input a ballot b and an election public key pk, algorithm ValidInd outputs 1 if the ballot is well-formed, and 0 otherwise.

Definition 58 (Strong consistency)**.** *An e-voting scheme* Γ *for a result function* f*, set of voters* $\mathcal{V}$*, and set of candidates* $\mathcal{C}$ *satisfies* strong consistency *if there exists algorithms* Extract *and* ValidInd *such that the following conditions hold*

- *For any* $v\in\mathcal{C}$*, any* (pk, sk) *output by algorithm* Setup *and any* (pk_{id}, sk_{id}) *output by algorithm* Register *on behalf of any voter* $id\in\mathcal{V}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\begin{array}{l}b\leftarrow\!\$\ \mathsf{Vote}(v,pk_{id},sk_{id},pk);\\ \mathsf{bb}\leftarrow\!\$\ \mathsf{Append}(b,\mathsf{bb},\mathcal{L},pk)\end{array}:\ \mathsf{Extract}(\mathsf{bb}[1],sk)\coloneqq(pk_{id},v)\right]\geq 1-\mathsf{negl}(\lambda)\,.$$

- *For any* (pk, sk) *output by algorithm* Setup *and any adversary* $\mathcal{A}(pk, sk)$ *that outputs tuple* $(\mathsf{bb}, b, \mathcal{L})$*, if* $\mathsf{Valid}(b, \mathsf{bb}, \mathcal{L}, pk) = 1$*, then* $\mathsf{ValidInd}(b, pk) = 1$*.*

- *For all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\begin{array}{l}\mathbf{pk}\leftarrow();\ \mathbf{sk}\leftarrow();\ \mathsf{bb}'\leftarrow() \\ (pk,sk) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda); \\ \mathbf{for}\ i{=}1,...|\mathcal{V}| \\ \quad (\mathbf{pk}[id_i],\mathbf{sk}[id_i]) \leftarrow\!\!\$\ \mathsf{Register}(id_i,pk); \\ \mathcal{L}\leftarrow\{\mathbf{pk}[id_1],...,\mathbf{sk}[id_{|\mathcal{V}|}]\}; \\ \mathsf{bb} \leftarrow\!\!\$\ \mathcal{A}(pk,\mathbf{pk},\mathbf{sk}); \\ (r,\rho) \leftarrow\!\!\$\ \mathsf{Tally}(\mathsf{bb},\mathcal{L},sk); \\ \mathbf{for}\ i\ =\ 1,...,|\mathsf{bb}| \\ \quad \mathsf{bb}'[i] \leftarrow\!\!\$\ \mathsf{Extract}(\mathsf{bb}[i],sk); \\ r'\leftarrow f(\mathsf{bb}'[1],...,\mathsf{bb}'[|\mathsf{bb}'|])\end{array} : \begin{array}{l} r \neq r' \ \wedge \\ \mathbf{for}\ i{=}1,...,|\mathsf{bb}|:\ \mathsf{ValidInd}(\mathsf{bb}[i],pk)\ :=\ 1\end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

4.6.2 Analysis of Our Definitions

To aid our understanding of our new definitions, we formally compare N-RF and E-RF, showing that N-RF is stronger than E-RF. We first show that N-RF implies E-RF. Then, we prove that E-RF does not imply N-RF by demonstrating that there exists an e-voting scheme, namely JCJ, that satisfies E-RF but not N-RF. We present these results and then reflect on their meaning. We formally prove that N-RF implies E-RF in Theorem 4.

Theorem 4 (N-RF ⇒ E-RF)**.** *Let* Γ *be an e-voting scheme that satisfies* N-RF*. Then* Γ *also satisfies* E-RF*.*

Proof. We first note that, by assumption, Γ does not require an evasion strategy as it satisfies N-RF. Therefore, it is possible to define algorithm CoercedVote as follows.

$\underline{\mathsf{CoercedVote}(v_0, v_1, pk_{id}, sk_{id}, b_0, pk, \mathsf{coins})}$

$b_1 \leftarrow\!\!\$\ \mathsf{Vote}(v_1, pk_{id}, sk_{id}, pk)$

$b_2 \leftarrow ()$

$b_3 \leftarrow ()$

return (b_1, b_2, b_3)

In this way, algorithm CoercedVote captures that, if a voter does not need to implement an evasion strategy, the voter can simply casts a ballot for a vote of their choice. Additionally, we define receipt Rcpt_1 as the tuple $(v_0, pk_{id}, sk_{id}, \mathsf{coins}', b_1)$, which provides simulated evidence that b_1 is a ballot for v_0.

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary that succeeds in the E-RF experiment for e-voting scheme Γ. We show that we can construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that succeeds in the N-RF experiment. Adversary $\mathcal{B}$ internally runs $\mathcal{A}$ and attempts to guess a bit β^* in the N-RF experiment. We present adversary $\mathcal{B}$ in Figure 4.10.

It is clear that the inputs to $\mathcal{A}_1$ and $\mathcal{A}_2$ produced respectively by $\mathcal{B}_1$ and $\mathcal{B}_2$ are distributed identically to the E-RF experiment when e-voting scheme Γ does not require an evasion strategy. Indeed, election public key pk is generated identically. Moreover, with the exception of oracle $\mathcal{O}\mathsf{RF}$, $\mathcal{B}$ can respond to $\mathcal{A}$'s oracle queries by querying the identical oracle in the N-RF experiment. $\mathcal{B}$ can perfectly simulate oracle $\mathcal{O}\mathsf{RF}$, as we now describe. $\mathcal{B}_2$ queries $\mathcal{O}\mathsf{RF}(id, v_0, v_1)$ which generates ballots b_0 and b_1 for votes v_0 and v_1 respectively. These ballots are appended to ballot boxes bb_0 and bb_1 and a receipt Rcpt_β is returned to $\mathcal{B}_2$. Additionally, $\mathcal{B}_2$ sets ballots b_2 and b_3 to be empty tuples as no further ballots need to be submitted if the e-voting scheme does not require an evasion strategy. $\mathcal{B}_2$ returns receipt Rcpt_β to $\mathcal{A}_2$, perfectly simulating the oracle $\mathcal{O}\mathsf{RF}$ in the E-RF experiment.

We now show that $\mathcal{B}$ succeeds in the N-RF experiment. That is, we show that $\mathcal{B}$ returns $\beta' = \beta$. By assumption, if $\mathcal{A}$ succeeds in the E-RF experiment then $\mathcal{A}$ outputs $\beta' = \beta$. As $\beta = \beta^*$ and is chosen in the E-RF experiment, $\mathcal{B}$ returns $\beta' = \beta^*$ and succeeds in the N-RF experiment. Hence, the result holds by contradiction. □

$\underline{\mathcal{B}_1^{\mathcal{O}\mathsf{reg}}(pk)}$

$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$

return st

$\underline{\mathcal{B}_2^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}(st)}$

$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}(st)$

return β'

$\underline{\mathcal{O}\mathsf{RF}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1) // \mathsf{E\text{-}RF}$

if $\mathsf{flag} = 1$ **return** $\perp$

if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$

$\mathsf{Rcpt}_\beta \leftarrow\!\!\$\ \mathcal{O}\mathsf{RF}(id, v_0, v_1)$ //N-RF

$b_2 \leftarrow (); \ b_3 \leftarrow ()$

return Rcpt_β

$\underline{\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{vote}/\mathcal{O}\mathsf{cast}/\mathcal{O}\mathsf{board}/\mathcal{O}\mathsf{tally}}$

Run the identical oracle in the N-RF experiment.

Figure 4.10: Adversary $\mathcal{B}$ that breaks the N-RF security of Γ in the proof of Theorem 4.

We now show that the reverse implication does not hold, that is, E-RF does not imply N-RF. We obtain the result in Theorem 5.

Theorem 5 (E-RF $\not\Rightarrow$ N-RF)**.** *There exists an e-voting scheme, namely* JCJ*, that satisfies* E-RF *but does not satisfy* N-RF*.*

We prove Theorem 5 through two Lemmata. In Lemma 1, we show that JCJ does not satisfy N-RF, and, in Lemma 2, we show that JCJ satisfies E-RF.

Lemma 1. JCJ *(Figure 4.1) does not satisfy* N-RF *if* JCJ *satisfies injectivity.*

Proof. The proof of this result is largely similar to the proof of Proposition 2. Indeed, we define the form of Rcpt identically. That is, $\mathsf{Rcpt}_0 = (v_0, b_0, pk_{id}, sk_{id}, r_{c_1}, r_{c_2})$ where, on behalf of voter id with credential pair (pk_{id}, sk_{id}), $b_0 \leftarrow\!\!\$\ \mathsf{Vote}(v_0, pk_{id}, sk_{id}, pk; r_{c_1}, r_{c_2})$. We also define $\mathsf{Rcpt}_1 = (v_0, b_1, pk_{id}, sk'_{id}, r'_{c_1}, r'_{c_2})$ for some sk'_{id} output by algorithm FakeKey and $b_1 \leftarrow\!\!\$\ \mathsf{Vote}(v_0, pk_{id}, sk'_{id}, pk; r''_{c_1}, r''_{c_2})$.

We construct an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, defined in Figure 4.11, that succeeds in the N-RF experiment for the JCJ scheme. $\mathcal{A}_1$ makes a single query to oracle $\mathcal{O}$reg to register an elegible voter id. Then, $\mathcal{A}_2$ makes a single query to oracle $\mathcal{O}$RF on behalf of the registered voter. If $\beta = 0$, oracle $\mathcal{O}$RF returns $\mathsf{Rcpt}_0 = (v_0, b_0, pk_{id}, sk_{id}, r_{c_1}, r_{c_2})$ where $b_0 \leftarrow\!\!\$\ \mathsf{Vote}(v_0, pk_{id}, sk_{id}, pk; r_{c_1}, r_{c_2})$. If $\beta = 1$, oracle $\mathcal{O}$RF returns $\mathsf{Rcpt}_1 = (v_0, b_1, pk_{id}, sk'_{id}, r'_{c_1}, r'_{c_2})$ where $b_1 \leftarrow\!\!\$\ \mathsf{Vote}(v_1, pk_{id}, sk_{id}, pk; r''_{c_1}, r''_{c_2})$. That is, a ballot b_1 is constructed for vote v_1, and the receipt simulates evidence that b_1 encodes vote v_0. $\mathcal{A}_2$ can locally construct a ballot $b \leftarrow\!\!\$\ \mathsf{Vote}(v_0, pk_{id}, sk^*_{id}, pk; r^*_{c_1}, r^*_{c_2})$ where $\mathsf{Rcpt}_\beta = (v_0, b^*, pk_{id}, sk^*_{id}, r^*_{c_1}, r^*_{c_2})$. As JCJ is injective, if $\beta = 0$, $b = b^*$. Else, if $\beta = 1$, $b \neq b^*$. Therefore, $\mathcal{A}$ returns $\beta' = \beta$ and succeeds in the N-RF experiment. Thus, e-voting scheme JCJ does not satisfy N-RF. □

$\mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$	$\mathcal{A}_2^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}(st)$
$pk_{id} \leftarrow\!\!\$\ \mathcal{O}\mathsf{reg}(id, 0)$	**parse** st as (id, pk_{id}, pk); $v_0 \leftarrow\!\!\$\ \mathcal{C}$; $v_1 \leftarrow\!\!\$\ \mathcal{C}$
return $st = (id, pk_{id}, pk)$	$\mathsf{Rcpt}_\beta \leftarrow\!\!\$\ \mathcal{O}\mathsf{RF}(id, v_0, v_1)$
	parse Rcpt as $(v_0, b^*, pk_{id}, sk^*_{id}, r^*_{c_1}, r^*_{c_2})$
	$b \leftarrow\!\!\$\ \mathsf{Vote}(v_0, pk_{id}, sk^*_{id}, pk; r^*_{c_1}, r^*_{c_2})$
	if $b = b^*$ **return** 0
	else return 1

Figure 4.11: Adversary $\mathcal{A}$ that succeeds in the N-RF experiment for the JCJ scheme in the proof of Lemma 1.

Lemma 2. JCJ *satisfies* E-RF.

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary in the E-RF experiment against the JCJ scheme that makes at most k_1 queries to oracle $\mathcal{O}$vote and at most k_2 queries to oracle $\mathcal{O}$RF. We proceed through a series of game hops that we show are indistinguishable to the adversary. In the final game, the view of the adversary will be identical for $\beta = 0$ and $\beta = 1$. We

define Game 0 as the E-RF experiment with β chosen randomly. Let S_i be the event that adversary $\mathcal{A}$ correctly guesses β in Game i.

Game 1 is identical to Game 0 except that, when $\beta = 0$ and the election key pair is generated, we run algorithm Sim to generate a trapdoor τ in addition to election key pair (pk, sk). We define the modified experiment for Game 1 in Figure 4.12. As this change is superficial and does not change the view of adversary $\mathcal{A}$, we have,

$$\Big|\Pr[S_0] - \Pr[S_1]\Big| = 0.$$

$\mathsf{Exp}^{\mathsf{E\text{-}RF},\beta}_{\mathsf{JCJ},\mathcal{A},\mathcal{S},\mathsf{Sim},\mathsf{Sim}_\rho,\mathcal{V},\mathcal{C}}(\lambda)$

$\mathbf{pk} \leftarrow ();\ \mathbf{sk} \leftarrow ();\ \mathsf{bb}_0 \leftarrow ();\ \mathsf{bb}_1 \leftarrow ()$
$\mathcal{Q}\mathsf{reg} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset;\ \mathcal{L} \leftarrow \emptyset$
$\mathsf{flag} \leftarrow 0$
if $\beta = 0$: $(pk, sk, \tau) \leftarrow_\$ \mathsf{Sim}(1^\lambda)$
  $st \leftarrow_\$ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
  $\beta' \leftarrow_\$ \mathcal{A}_2^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}(st)$
  return β'
else Proceed as usual.

Figure 4.12: The modified E-RF experiment for Game 1 in the proof of Lemma 2.

Game 2 is identical to Game 1 except for a change to oracle $\mathcal{O}$tally. When $\beta = 0$, rather than generating a real tallying proof, $\mathcal{O}$tally runs algorithm Sim_ρ to simulate the tallying proof. We describe oracle $\mathcal{O}$tally for Game 2 in Figure 4.13.

$\mathcal{O}\mathsf{tally}_{(sk,\tau,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathsf{flag})}()$

$\mathsf{flag} \leftarrow 1$
if $\beta = 0$: $(r, \rho') \leftarrow_\$ \mathsf{Tally}(\mathsf{bb}_0, \mathcal{L}, sk);\ \rho \leftarrow_\$ \mathsf{Sim}_\rho(\tau, \mathsf{bb}_0, r)$
  return (r, ρ)
else Proceed as usual.

Figure 4.13: The oracle $\mathcal{O}$tally for Game 2 in the proof of Lemma 2.

We show that Game 1 and Game 2 are indistinguishable if the NIZK proof system used to generate the tallying proof satisfies the zero-knowledge property.[4] We give a distinguishing algorithm $\mathcal{D}_1$ in Figure 4.14 that aims to guess a bit β^* in the zero-knowledge experiment for NIZK and is given access to a proving oracle $\mathcal{O}$Prove. Oracle $\mathcal{O}$Prove returns a real (i.e., generated by algorithm NIZK.Prove) or a simulated (i.e., generated by algorithm

[4]Note that the tallying proof consists of a number of NIZK proofs. For simplicity of presentation, we choose to capture this in a single game hop, rather than defining multiple game hops that modify each part of the tallying proof.

NIZK.SimProve) proof, depending on bit β^*. We show that, when $\beta^* = 0$, inputs to $\mathcal{A}$ are identical to Game 1 and, when $\beta^* = 1$, inputs to $\mathcal{A}$ are identical to Game 2. Note that Game 1 and Game 2 are identical with the exception of oracle $\mathcal{O}\mathsf{tally}$ when $\beta = 0$. In particular, the election public key pk is honestly generated and identical in Games 1 and 2. Moreover, all other oracles queried by $\mathcal{A}$ are identical. If $\beta^* = 0$, $\mathcal{D}_1$ is input a real tallying proof as in Game 0. If $\beta^* = 1$, $\mathcal{D}_1$ is input a simulated tallying proof, as in Game 1. Therefore,

$$\big|\Pr[S_1] - \Pr[S_2]\big| \leq \mathsf{negl}_{\mathsf{NIZK\text{-}ZK}}$$

where $\mathsf{negl}_{\mathsf{NIZK\text{-}ZK}}$ is the advantage in breaking the zero-knowledge property of the NIZK proof system.

$\mathcal{D}_1^{\mathcal{O}\mathsf{Prove}}(pp_{\mathsf{NIZK}})$

$\beta \leftarrow\!\!\$\ \{0,1\}$
$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow (); \ \mathsf{bb}_0 \leftarrow (); \ \mathsf{bb}_1 \leftarrow ()$
$\mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset$
$\mathsf{flag} \leftarrow 0$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}})$
$pk \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$sk \leftarrow (sk_{\mathsf{PKE}}, pk)$
$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}(st)$
if $\beta' = \beta$ **return** 1

$\mathcal{O}\mathsf{tally}_{(sk,\tau,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathsf{flag})}()$

$\mathsf{flag} \leftarrow 1$
if $\beta = 0$
  $(r, \rho') \leftarrow\!\!\$\ \mathsf{Tally}(\mathsf{bb}_0, \mathcal{L}, sk);$
  $\rho \leftarrow\!\!\$\ \mathcal{O}\mathsf{NIZK.Prove}((\mathsf{bb}_0, r), (sk_{\mathsf{PKE}}, \phi_1, \phi_2))$
  return (r, ρ)
else Proceed as usual.

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{vote}/\mathcal{O}\mathsf{RF}/\mathcal{O}\mathsf{cast}/\mathcal{O}\mathsf{board}$

As in Game 0.

Figure 4.14: Distinguisher $\mathcal{D}_1$ that distinguishes Game 1 and Game 2 in the proof of Lemma 2

Game 3 is identical to Game 2 except for a change to oracles $\mathcal{O}\mathsf{vote}$ and $\mathcal{O}\mathsf{RF}$. For every ballot $b = (c_1, c_2, \rho_{c_1}, \rho_{c_2})$ output by algorithm Vote, we simulate the zero-knowledge proofs ρ_{c_1} and ρ_{c_2}, rather than generating real proofs. We describe oracles $\mathcal{O}\mathsf{vote}$ and $\mathcal{O}\mathsf{RF}$ for Game 3 in Figure 4.15. In this game hop, and subsequent game hops, we underline the changes made for clarity.

We show that Game 2 and Game 3 are indistinguishable if the NIZK proof system used to generate the proof of plaintext knowledge and proof of conjunctive plaintext knowledge satisfy the zero-knowledge property. We give a distinguishing algorithm $\mathcal{D}_2$ in Figure 4.16 that aims to guess a bit β^* in the zero-knowledge experiment for NIZK and has access to a proving oracle $\mathcal{O}\mathsf{Prove}$, as in the previous game hop. We show that, when $\beta^* = 0$, inputs to $\mathcal{A}$ are identical to Game 2 and, when $\beta^* = 1$, inputs to $\mathcal{A}$ are identical to Game 3. As before, election public key pk input to $\mathcal{A}$ is identical in Games 1 and 2 and oracles are

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$c_{1,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,0})$; $c_{2,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,0})$
$\underline{\rho_{c_1,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}))}$; $\underline{\rho_{c_2,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}))}$
$c_{1,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_1; r_{c_1,1})$; $c_{2,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,1})$
$\underline{\rho_{c_1,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, \mathcal{C}))}$; $\underline{\rho_{c_2,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, c_{2,1}))}$
$b_0 \leftarrow (c_{1,0}, c_{2,0}, \rho_{c_1,0}, \rho_{c_2,0})$; $b_1 \leftarrow (c_{1,1}, c_{2,1}, \rho_{c_1,1}, \rho_{c_2,1})$
if $\mathsf{Valid}(b_\beta, \mathsf{bb}_\beta, \mathcal{L}, pk) = 1$: $\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_0, \mathcal{L}, sk)$; $\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_1, \mathsf{bb}_1, \mathcal{L}, sk)$
return $\top$

$\mathcal{O}\mathsf{RF}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$c_{1,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,0})$; $c_{2,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,0})$
$\underline{\rho_{c_1,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}))}$; $\underline{\rho_{c_2,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}))}$
$b_0 \leftarrow (c_{1,0}, c_{2,0}, \rho_{c_1,0}, \rho_{c_2,0})$;
$sk'_{id} \leftarrow\!\!\$\ \mathsf{FakeKey}(pk, \mathbf{pk}[id], \mathbf{sk}[id])$
$c_{1,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,1})$; $c_{2,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, sk'_{id}; r_{c_2,1})$
$\underline{\rho_{c_1,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, \mathcal{C}))}$; $\underline{\rho_{c_2,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, c_{2,1}))}$
$b_1 \leftarrow (c_{1,1}, c_{2,1}, \rho_{c_1,1}, \rho_{c_2,1})$
$c_{1,2} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v \leftarrow\!\!\$\ \mathcal{C}; r_{c_1,2})$; $c_{2,2} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, sk'_{id}; r_{c_2,2})$
$\underline{\rho_{c_1,2} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,2}, \mathcal{C}))}$; $\underline{\rho_{c_2,2} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,2}, c_{2,2}))}$
$b_2 \leftarrow (c_{1,2}, c_{2,2}, \rho_{c_1,2}, \rho_{c_2,2})$
$c_{1,3} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_1; r_{c_1,3})$; $c_{2,3} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,3})$
$\underline{\rho_{c_1,3} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}))}$; $\underline{\rho_{c_2,3} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}))}$
$b_3 \leftarrow (c_{1,3}, c_{2,3}, \rho_{c_1,3}, \rho_{c_2,3})$;
$\mathsf{Rcpt}_0 \leftarrow (v_0, \mathbf{pk}[id], \mathbf{sk}[id], r_{c_1,0}, r_{c_2,0}, b_0)$; $\mathsf{Rcpt}_1 \leftarrow (v_0, \mathbf{pk}[id], sk'_{id}, r_{c_1,1}, r_{c_2,1}, b_1)$
if $\mathsf{Valid}(b_\beta, \mathsf{bb}_\beta, \mathcal{L}, pk) = 0$ **return** $\perp$
$\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_0, \mathcal{L}, sk)$; $\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_1, \mathsf{bb}_1, \mathcal{L}, sk)$; $\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_2, \mathsf{bb}_0, \mathcal{L}, sk)$; $\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_3, \mathsf{bb}_1, \mathcal{L}, sk)$
return Rcpt_β

Figure 4.15: Oracles $\mathcal{O}\mathsf{vote}$ and $\mathcal{O}\mathsf{RF}$ for Game 3 in the proof of Lemma 2.

identical with the exception of oracles $\mathcal{O}\mathsf{vote}$ and $\mathcal{O}\mathsf{RF}$. If $\beta^* = 0$, $\mathcal{D}_2$ is input real proofs as in Game 2. If $\beta^* = 1$, $\mathcal{D}_2$ is input simulated proofs as in Game 3. Therefore,

$$\big|\Pr[S_2] - \Pr[S_3]\big| \leq \mathsf{negl}_{\mathsf{ENC\text{-}ZK}} + \mathsf{negl}_{\mathsf{CONJ\text{-}ZK}}$$

where $\mathsf{negl}_{\mathsf{ENC\text{-}ZK}}$ is the advantage in breaking the zero-knowledge property of the NIZK proof of plaintext knowledge and $\mathsf{negl}_{\mathsf{CONJ\text{-}ZK}}$ is the advantage in breaking the zero-knowledge property of the NIZK proof of conjunctive plaintext knowledge.

We now define Game 4 to be identical to Game 3 but with the following change to oracle $\mathcal{O}\mathsf{vote}$. To construct ballot b_0, the oracle encrypts vote v_1 rather than vote v_0. We define the modified oracle $\mathcal{O}\mathsf{vote}$ in Figure 4.17.

$\mathcal{D}_2^{\mathcal{O}\mathsf{NIZK.Prove}}(pp_{\mathsf{NIZK}})$

$\beta \leftarrow\!\!\$\ \{0,1\}$
$\mathbf{pk} \leftarrow ();\ \mathbf{sk} \leftarrow ();\ \mathsf{bb}_0 \leftarrow ();\ \mathsf{bb}_1 \leftarrow ()$
$\mathcal{Q}\mathsf{reg} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset;\ \mathcal{L} \leftarrow \emptyset$
$\mathsf{flag} \leftarrow 0$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}})$
$pk \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$sk \leftarrow (sk_{\mathsf{PKE}}, pk)$
$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}(st)$
if $\beta' = \beta$ **return** 1

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$c_{1,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,0});\quad c_{2,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,0})$
$\underline{\rho_{c_1,0} \leftarrow\!\!\$\ \mathcal{O}\mathsf{NIZK.Prove}_{\mathsf{enc}}((pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}), (r_{c_1,0}, v_0))}$
$\underline{\rho_{c_2,0} \leftarrow\!\!\$\ \mathcal{O}\mathsf{NIZK.Prove}_{\mathsf{conj}}((pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}), (r_{c_1,0}, v_0, r_{c_2,0}, \mathbf{sk}[id]))}$
$c_{1,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_1; r_{c_1,1});\quad c_{2,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,1})$
$\underline{\rho_{c_1,1} \leftarrow\!\!\$\ \mathcal{O}\mathsf{NIZK.Prove}_{\mathsf{enc}}((pk_{\mathsf{PKE}}, c_{1,1}, \mathcal{C}), (r_{c_1,1}, v_1))}$
$\underline{\rho_{c_2,1} \leftarrow\!\!\$\ \mathcal{O}\mathsf{NIZK.Prove}_{\mathsf{conj}}((pk_{\mathsf{PKE}}, c_{1,1}, c_{2,1}), (r_{c_1,1}, v_1, r_{c_2,1}, \mathbf{sk}[id]))}$
$b_0 \leftarrow (c_{1,0}, c_{2,0}, \rho_{c_1,0}, \rho_{c_2,0});\ b_1 \leftarrow (c_{1,1}, c_{2,1}, \rho_{c_1,1}, \rho_{c_2,1})$
if $\mathsf{Valid}(b_\beta, \mathsf{bb}_\beta, \mathcal{L}, pk) = 1$: $\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_0, \mathcal{L}, sk);\ \mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_1, \mathsf{bb}_1, \mathcal{L}, sk)$
return $\top$

$\underline{\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{cast}/\mathcal{O}\mathsf{board}/\mathcal{O}\mathsf{tally}}$

As in Game 0.

$\mathcal{O}\mathsf{RF}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$c_{1,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,0});\quad c_{2,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,0})$
$\underline{\rho_{c_1,0} \leftarrow\!\!\$\ \mathcal{O}\mathsf{NIZK.Prove}_{\mathsf{enc}}((pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}), (r_{c_1,0}, v_0))};\quad \underline{\rho_{c_2,0} \leftarrow\!\!\$\ \mathcal{O}\mathsf{NIZK.Prove}_{\mathsf{conj}}((pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}), (r_{c_1,0}, v_0, r_{c_2,0}, \mathbf{sk}[id]))}$
$b_0 \leftarrow (c_{1,0}, c_{2,0}, \rho_{c_1,0}, \rho_{c_2,0});$
$sk'_{id} \leftarrow\!\!\$\ \mathsf{FakeKey}(pk, \mathbf{pk}[id], \mathbf{sk}[id])$
$c_{1,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,1});\quad c_{2,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, sk'_{id}; r_{c_2,1})$
$\underline{\rho_{c_1,1} \leftarrow\!\!\$\ \mathcal{O}\mathsf{NIZK.Prove}_{\mathsf{enc}}((pk_{\mathsf{PKE}}, c_{1,1}, \mathcal{C}), (r_{c_1}, v_0))};\quad \underline{\rho_{c_2,1} \leftarrow\!\!\$\ \mathcal{O}\mathsf{NIZK.Prove}_{\mathsf{conj}}((pk_{\mathsf{PKE}}, c_{1,1}, c_{2,1}), (r_{c_1,1}, v_0, r_{c_2,1}, sk'_{id}))}$
$b_1 \leftarrow (c_{1,1}, c_{2,1}, \rho_{c_1,1}, \rho_{c_2,1})$
$c_{1,2} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v \leftarrow\!\!\$\ \mathcal{C}; r_{c_1,3});\quad c_{2,2} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, sk'_{id}; r_{c_2,3})$
$\underline{\rho_{c_1,2} \leftarrow\!\!\$\ \mathcal{O}\mathsf{NIZK.Prove}_{\mathsf{enc}}((pk_{\mathsf{PKE}}, c_{1,2}, \mathcal{C}), (r_{c_1,3}, v))};\quad \underline{\rho_{c_2,2} \leftarrow\!\!\$\ \mathcal{O}\mathsf{NIZK.Prove}_{\mathsf{conj}}((pk_{\mathsf{PKE}}, c_{1,2}, c_{2,2}), (r_{c_1,3}, v, r_{c_2,3}, sk'_{id}))}$
$b_2 \leftarrow (c_{1,2}, c_{2,2}, \rho_{c_1,2}, \rho_{c_2,2})$
$c_{1,3} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_1, r_{c_1,4});\quad c_{2,3} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id], r_{c_2,4})$
$\underline{\rho_{c_1,3} \leftarrow\!\!\$\ \mathcal{O}\mathsf{NIZK.Prove}_{\mathsf{enc}}((pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}), (r_{c_1,4}, v_1))};\quad \underline{\rho_{c_2,3} \leftarrow\!\!\$\ \mathcal{O}\mathsf{NIZK.Prove}_{\mathsf{conj}}((pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}), (r_{c_1,4}, v_1, r_{c_2,4}, \mathbf{sk}[id]))}$
$b_3 \leftarrow (c_{1,3}, c_{2,3}, \rho_{c_1,3}, \rho_{c_2,3});$
$\mathsf{Rcpt}_0 \leftarrow (v_0, \mathbf{pk}[id], \mathbf{sk}[id], r_{c_1,0}, r_{c_2,0}, b_0);\ \mathsf{Rcpt}_1 \leftarrow (v_0, \mathbf{pk}[id], sk'_{id}, r_{c_1,1}, r_{c_2,1}, b_1)$
if $\mathsf{Valid}(b_\beta, \mathsf{bb}_\beta, \mathcal{L}, pk) = 0$ **return** $\perp$
$\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_0, \mathcal{L}, sk);\ \mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_1, \mathsf{bb}_1, \mathcal{L}, sk);\ \mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_2, \mathsf{bb}_0, \mathcal{L}, sk);\ \mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_3, \mathsf{bb}_1, \mathcal{L}, sk)$
return Rcpt_β

Figure 4.16: Distinguisher $\mathcal{D}_2$ that distinguishes Game 2 and Game 3 in the proof of Lemma 2.

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$\underline{c_{1,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_1; r_{c_1,0})};\quad c_{2,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,0})$
$\rho_{c_1,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}));\quad \rho_{c_2,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}))$
$c_{1,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_1; r_{c_1,1});\quad c_{2,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,1})$
$\rho_{c_1,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, \mathcal{C}));\quad \rho_{c_2,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, c_{2,1}))$
$b_0 \leftarrow (c_{1,0}, c_{2,0}, \rho_{c_1,0}, \rho_{c_2,0});\ b_1 \leftarrow (c_{1,1}, c_{2,1}, \rho_{c_1,1}, \rho_{c_2,1})$
if $\mathsf{Valid}(b_\beta, \mathsf{bb}_\beta, \mathcal{L}, pk) = 1$: $\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_0, \mathcal{L}, sk);\ \mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_1, \mathsf{bb}_1, \mathcal{L}, sk)$
return $\top$

Figure 4.17: The oracle $\mathcal{O}\mathsf{vote}$ for Game 4 in the proof of Lemma 2.

We show that Game 3 and Game 4 are indistinguishable if the PKE scheme satisfies NM-CPA security. We give a distinguishing algorithm $\mathcal{D}_3$ in Figure 4.18 that guesses a

bit β^* in the NM-CPA for multiple encryptions experiment for PKE and is given access to oracles $\mathcal{O}$enc and $\mathcal{O}$dec. We show that, when $\beta^* = 0$, inputs to $\mathcal{A}$ are identical to Game 3 and, when $\beta^* = 1$, inputs to $\mathcal{A}$ are identical to Game 4. Note that pk input to $\mathcal{A}$ is honestly generated and identical in each game. Furthermore, sk_{PKE} is unknown but is not required. Games 3 and 4 are identical with the exception of oracles $\mathcal{O}$vote and $\mathcal{O}$tally. To compute the tally, algorithm $\mathcal{D}_3$ queries a vector of ciphertexts to $\mathcal{O}$Dec in the NM-CPA experiment. The vector consists of the ciphertexts included in ballots from ballot box bb_0 that are to be included in the tally (with the exception of ballots generated via queries to $\mathcal{O}$vote). Oracle $\mathcal{O}$dec returns a vector of plaintext messages, which algorithm $\mathcal{D}_3$ inputs to the result function to compute the election result. In a call to oracle $\mathcal{O}$vote, if $\beta^* = 0$, $\mathcal{D}_3$ is input an encryption of v_0 as in Game 3. If $\beta^* = 1$, $\mathcal{D}_3$ is input an encryption of v_1 as in Game 4. Therefore,

$$\big|\Pr[S_3] - \Pr[S_4]\big| \leq \mathsf{negl}_{\mathsf{NM\text{-}CPA\text{-}mult}}$$

where $\mathsf{negl}_{\mathsf{NM\text{-}CPA\text{-}mult}}$ is the advantage in breaking the NM-CPA for multiple encryptions property of PKE. By a standard argument,

$$\big|\Pr[S_3] - \Pr[S_4]\big| \leq k_1 \cdot \mathsf{negl}_{\mathsf{NM\text{-}CPA}}$$

where $\mathsf{negl}_{\mathsf{NM\text{-}CPA}}$ is the advantage in breaking the NM-CPA property of PKE and k_1 is the maximum number of queries to oracle $\mathcal{O}$vote.

We now define Game 5 to be identical to Game 4 but with the following change to oracle $\mathcal{O}$RF. To construct ballot b_3, the oracle encrypts vote v_0 rather than vote v_1. We define the modified oracle $\mathcal{O}$RF in Figure 4.19.

We show that Game 4 and Game 5 are indistinguishable if the PKE scheme satisfies NM-CPA security. We give a distinguishing algorithm $\mathcal{D}_4$ in Figure 4.20 that guesses a bit β^* in the NM-CPA for multiple encryptions experiment for PKE. We show that, when $\beta^* = 0$, inputs to $\mathcal{A}$ are identical to Game 4 and, when $\beta^* = 1$, inputs to $\mathcal{A}$ are identical to Game 5. Note that pk input to $\mathcal{A}$ is honestly generated and identical in each game. Furthermore, sk_{PKE} is unknown but is not required. Also, as Game 5 changes the contents of ballot box bb_1, the tally does not change and can be computed as in Game 4. Therefore, Games 4 and 5 are identical with the exception of oracle $\mathcal{O}$RF. If $\beta^* = 0$, $\mathcal{D}_4$ is input an encryption of v_1 as in Game 4. If $\beta^* = 1$, $\mathcal{D}_4$ is input an encryption of v_0 as in Game 5.

$\mathcal{D}_3^{\mathcal{O}\mathsf{enc},\mathcal{O}\mathsf{dec}}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}})$

$\beta \leftarrow\!\!\$\ \{0,1\}$
$\mathbf{pk} \leftarrow ();\ \mathbf{sk} \leftarrow ();\ \mathsf{bb}_0 \leftarrow ();\ \mathsf{bb}_1 \leftarrow ()$
$\mathcal{Q}\mathsf{reg} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset;\ \mathcal{L} \leftarrow \emptyset$
$\mathsf{flag} \leftarrow 0$
$(pp_{\mathsf{NIZK}}, \tau) \leftarrow\!\!\$\ \mathsf{NIZK.SimSetup}(1^\lambda)$
$pk \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}(st)$
if $\beta' = \beta$ **return** 1

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{cast}/\mathcal{O}\mathsf{board}$

As in Game 0.

$\mathcal{O}\mathsf{RF}$

As in Game 3.

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$c_{1,0} \leftarrow\!\!\$\ \mathcal{O}\mathsf{enc}(v_0, v_1)$; $c_{2,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,0})$
$\rho_{c_1,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}))$; $\rho_{c_2,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}))$
$c_{1,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_1; r_{c_1,1})$; $c_{2,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,1})$
$\rho_{c_1,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, \mathcal{C}))$; $\rho_{c_2,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, c_{2,1}))$
$b_0 \leftarrow (c_{1,0}, c_{2,0}, \rho_{c_1,0}, \rho_{c_2,0});\ b_1 \leftarrow (c_{1,1}, c_{2,1}, \rho_{c_1,1}, \rho_{c_2,1})$
if $\mathsf{Valid}(b_\beta, \mathsf{bb}_\beta, \mathcal{L}, pk) = 1$: $\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_0, \mathcal{L}, sk);\ \mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_1, \mathsf{bb}_1, \mathcal{L}, sk)$
return $\top$

$\mathcal{O}\mathsf{tally}_{(sk,\tau,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,t)}()$

$\mathsf{flag} \leftarrow 1;\ \mathbf{c} \leftarrow ()$
Perform steps of tally to obtain A_1' and B_1' for the contents of bb_0
for $i = 1, \ldots |B_1'|$
  if $B_1'[i]$ encrypts a plaintext such that $\mathcal{L}$ contains an encryption of the same plaintext : $A_3 \leftarrow A_3 \parallel A_1'[i]$
parse A_3 as $(c_{1,1}, \ldots, c_{1,|A_3|})$
if $(id_i, b_i = (c_{1,i}, c_{2,i}, \rho_{c_1,i}, \rho_{c_2,i}))$ is queried to $\mathcal{O}\mathsf{cast}$: $\mathbf{c} \leftarrow \mathbf{c} \parallel A_3[i]$
$\mathbf{m} \leftarrow\!\!\$\ \mathcal{O}\mathsf{dec}(\mathbf{c})$; **parse** $\mathbf{m}$ as $(m_1, \ldots m_{|\mathbf{m}|})$
$r = f((id_1, v^1), \ldots, (id_{|A_3|}, v^{|A_3|}))$ where $v^i = m_j$ if $(id_i, b_i = (c_{1,i}, c_{2,i}, \rho_{c_1,i}, \rho_{c_2,i}))$ is queried to $\mathcal{O}\mathsf{cast}$ and $m_j = \mathsf{PKE.Dec}(pp_{\mathsf{PKE}}, sk_{\mathsf{PKE}}, \mathbf{c}[j] = c_{1,i})$,
$v^i = v_1$ if (id_i, v_0, v_1) is queried to $\mathcal{O}\mathsf{vote}$ and $v^i = v_0$ if (id_i, v_0, v_1) is queried to $\mathcal{O}\mathsf{RF}$
$\rho \leftarrow\!\!\$\ \mathsf{Sim}_\rho(\tau, \mathsf{bb}_\beta, r)$
return (r, ρ)

Figure 4.18: Distinguisher $\mathcal{D}_3$ that distinguishes Game 3 and Game 4 in the proof of Lemma 2.

Therefore,

$$\left|\Pr[S_4] - \Pr[S_5]\right| \leq \mathsf{negl}_{\mathsf{NM\text{-}CPA\text{-}mult}}$$

where $\mathsf{negl}_{\mathsf{NM\text{-}CPA\text{-}mult}}$ is the advantage in breaking the NM-CPA for multiple encryptions property of PKE'. By a standard argument,

$$\left|\Pr[S_4] - \Pr[S_5]\right| \leq k_2 \cdot \mathsf{negl}_{\mathsf{NM\text{-}CPA}}$$

$\mathcal{O}\mathsf{RF}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$c_{1,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,0})$; $c_{2,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,0})$
$\rho_{c_1,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}))$; $\rho_{c_2,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}))$
$b_0 \leftarrow (c_{1,0}, c_{2,0}, \rho_{c_1,0}, \rho_{c_2,0})$;
$sk'_{id} \leftarrow\!\!\$\ \mathsf{FakeKey}(pk, \mathbf{pk}[id], \mathbf{sk}[id])$
$c_{1,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,1})$; $c_{2,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, sk'_{id}; r_{c_2,1})$
$\rho_{c_1,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, \mathcal{C}))$; $\rho_{c_2,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, c_{2,1}))$
$b_1 \leftarrow (c_{1,1}, c_{2,1}, \rho_{c_1,1}, \rho_{c_2,1})$
$c_{1,2} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v \leftarrow\!\!\$\ \mathcal{C}; r_{c_1,2})$; $c_{2,2} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, sk'_{id}; r_{c_2,2})$
$\rho_{c_1,2} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,2}, \mathcal{C}))$; $\rho_{c_2,2} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,2}, c_{2,2}))$
$b_2 \leftarrow (c_{1,2}, c_{2,2}, \rho_{c_1,2}, \rho_{c_2,2})$
$c_{1,3} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,3})$; $c_{2,3} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,3})$
$\rho_{c_1,3} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}))$; $\rho_{c_2,3} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}))$
$b_3 \leftarrow (c_{1,3}, c_{2,3}, \rho_{c_1,3}, \rho_{c_2,3})$;
$\mathsf{Rcpt}_0 \leftarrow (v_0, \mathbf{pk}[id], \mathbf{sk}[id], r_{c_1,0}, r_{c_2,0}, b_0)$; $\mathsf{Rcpt}_1 \leftarrow (v_0, \mathbf{pk}[id], sk'_{id}, r_{c_1,1}, r_{c_2,1}, b_1)$
if $\mathsf{Valid}(b_\beta, \mathsf{bb}_\beta, \mathcal{L}, pk) = 0$ **return** $\perp$
$\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_0, \mathcal{L}, sk)$; $\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_1, \mathsf{bb}_1, \mathcal{L}, sk)$; $\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_2, \mathsf{bb}_0, \mathcal{L}, sk)$; $\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_3, \mathsf{bb}_1, \mathcal{L}, sk)$
return Rcpt_β

Figure 4.19: Oracle $\mathcal{O}\mathsf{RF}$ for Game 5 in the proof of Lemma 2.

where $\mathsf{negl}_{\mathsf{NM\text{-}CPA}}$ is the advantage in breaking the NM-CPA property of PKE and k_2 is the maximum number of queries to oracle $\mathcal{O}\mathsf{RF}$.

We now define Game 6 to be identical to Game 5 but with the following change to oracle $\mathcal{O}\mathsf{RF}$. To construct ballot b_2, the oracle encrypts vote v_0 rather than vote $v \leftarrow\!\!\$\ \mathcal{C}$. We define the modified oracle $\mathcal{O}\mathsf{RF}$ in Figure 4.21.

We show that Game 5 and Game 6 are indistinguishable if the PKE scheme satisfies NM-CPA security. We give a distinguishing algorithm $\mathcal{D}_5$ in Figure 4.22 that guesses a bit β^* in the NM-CPA for multiple encryptions experiment for PKE. We show that, when $\beta^* = 0$, inputs to $\mathcal{A}$ are identical to Game 5 and, when $\beta^* = 1$, inputs to $\mathcal{A}$ are identical to Game 6. Note that pk input to $\mathcal{A}$ is honestly generated and identical in each game. Furthermore, sk_{PKE} is unknown but is not required. Moreover, as ballot b_2 is not included in the tally (i.e., it is submitted with a fake credential and is therefore filtered during the tallying process) the tally can be computed as in Game 4. Therefore, Games 5 and 6 are identical with the exception of oracle $\mathcal{O}\mathsf{RF}$. If $\beta^* = 0$, $\mathcal{D}_5$ is input an encryption of $v \leftarrow\!\!\$\ \mathcal{C}$ as in Game 5. If $\beta^* = 1$, $\mathcal{D}_5$ is input an encryption of v_0 as in Game 6. Therefore,

$$\left|\Pr[S_5] - \Pr[S_6]\right| \leq \mathsf{negl}_{\mathsf{NM\text{-}CPA\text{-}mult}}$$

where $\mathsf{negl}_{\mathsf{NM\text{-}CPA\text{-}mult}}$ is the advantage in breaking the NM-CPA for multiple encryptions

$\mathcal{D}_4^{\mathcal{O}\mathsf{enc},\mathcal{O}\mathsf{dec}}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}})$

$\beta \leftarrow\!\!\$\ \{0,1\}$
$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow (); \ \mathsf{bb}_0 \leftarrow (); \ \mathsf{bb}_1 \leftarrow ()$
$\mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset$
$\mathsf{flag} \leftarrow 0$
$(pp_{\mathsf{NIZK}}, \tau) \leftarrow\!\!\$\ \mathsf{NIZK.SimSetup}(1^\lambda)$
$pk \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}(st)$
if $\beta' = \beta$ **return** 1

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{cast}/\mathcal{O}\mathsf{board}$

As in Game 0.

$\mathcal{O}\mathsf{vote}/\mathcal{O}\mathsf{tally}$

As in Game 4.

$\mathcal{O}\mathsf{RF}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$c_{1,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,0})$; $c_{2,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,0})$
$\rho_{c_1,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}))$; $\rho_{c_2,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}))$
$b_0 \leftarrow (c_{1,0}, c_{2,0}, \rho_{c_1,0}, \rho_{c_2,0})$;
$sk'_{id} \leftarrow\!\!\$\ \mathsf{FakeKey}(pk, \mathbf{pk}[id], \mathbf{sk}[id])$
$c_{1,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,1})$; $c_{2,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, sk'_{id}; r_{c_2,1})$
$\rho_{c_1,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, \mathcal{C}))$; $\rho_{c_2,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, c_{2,1}))$
$b_1 \leftarrow (c_{1,1}, c_{2,1}, \rho_{c_1,1}, \rho_{c_2,1})$
$c_{1,2} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v \leftarrow\!\!\$\ \mathcal{C}; r_{c_1,2})$; $c_{2,2} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, sk'_{id}; r_{c_2,2})$
$\rho_{c_1,2} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,2}, \mathcal{C}))$; $\rho_{c_2,2} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,2}, c_{2,2}))$
$b_2 \leftarrow (c_{1,2}, c_{2,2}, \rho_{c_1,2}, \rho_{c_2,2})$
$c_{1,3} \leftarrow\!\!\$\ \mathcal{O}\mathsf{enc}(v_1, v_0)$; $c_{2,3} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,3})$
$\rho_{c_1,3} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}))$; $\rho_{c_2,3} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}))$
$b_3 \leftarrow (c_{1,3}, c_{2,3}, \rho_{c_1,3}, \rho_{c_2,3})$;
$\mathsf{Rcpt}_0 \leftarrow (v_0, \mathbf{pk}[id], \mathbf{sk}[id], r_{c_1,0}, r_{c_2,0}, b_0)$; $\mathsf{Rcpt}_1 \leftarrow (v_0, \mathbf{pk}[id], sk'_{id}, r_{c_1,1}, r_{c_2,1}, b_1)$
if $\mathsf{Valid}(b_\beta, \mathsf{bb}_\beta, \mathcal{L}, pk) = 0$ **return** $\perp$
$\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_0, \mathcal{L}, sk)$; $\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_1, \mathsf{bb}_1, \mathcal{L}, sk)$; $\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_2, \mathsf{bb}_0, \mathcal{L}, sk)$; $\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_3, \mathsf{bb}_1, \mathcal{L}, sk)$
return Rcpt_β

Figure 4.20: Distinguisher $\mathcal{D}_4$ that distinguishes Game 4 and Game 5 in the proof of Lemma 2.

property of PKE. By a standard argument,

$$\left|\Pr[S_5] - \Pr[S_6]\right| \leq k_2 \cdot \mathsf{negl}_{\mathsf{NM\text{-}CPA}}$$

where $\mathsf{negl}_{\mathsf{NM\text{-}CPA}}$ is the advantage in breaking the NM-CPA property of PKE and k_2 is the maximum number of queries to oracle $\mathcal{O}\mathsf{RF}$.

We now define a final game hop. Game 7 is identical to Game 6 except that we change the order in which $\mathcal{O}\mathsf{RF}$ appends ballots to bb_1 and change the receipt Rcpt_1. That is, we append b_3 before b_1 and produce a receipt for b_3 instead of b_1. We describe oracle $\mathcal{O}\mathsf{RF}$ in Figure 4.23. If $\beta = 1$, following a query to $\mathcal{O}\mathsf{RF}$, $\mathcal{A}$ expects to see two ballots, where the first encrypts a vote for v_0, and a receipt that can be used to reconstruct the first ballot. By our previous game hops, both ballots that appear on bb_1 encrypt a vote for v_0.

$\mathcal{O}\mathsf{RF}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$c_{1,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,0})$; $c_{2,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,0})$
$\rho_{c_1,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}))$; $\rho_{c_2,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}))$
$b_0 \leftarrow (c_{1,0}, c_{2,0}, \rho_{c_1,0}, \rho_{c_2,0})$;
$sk'_{id} \leftarrow\!\!\$\ \mathsf{FakeKey}(pk, \mathbf{pk}[id], \mathbf{sk}[id])$
$c_{1,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,1})$; $c_{2,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, sk'_{id}; r_{c_2,1})$
$\rho_{c_1,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, \mathcal{C}))$; $\rho_{c_2,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, c_{2,1}))$
$b_1 \leftarrow (c_{1,1}, c_{2,1}, \rho_{c_1,1}, \rho_{c_2,1})$
$c_{1,2} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,2})$; $c_{2,2} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, sk'_{id}; r_{c_2,2})$
$\rho_{c_1,2} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,2}, \mathcal{C}))$; $\rho_{c_2,2} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,2}, c_{2,2}))$
$b_2 \leftarrow (c_{1,2}, c_{2,2}, \rho_{c_1,2}, \rho_{c_2,2})$
$c_{1,3} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,3})$; $c_{2,3} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,3})$
$\rho_{c_1,3} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}))$; $\rho_{c_2,3} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}))$
$b_3 \leftarrow (c_{1,3}, c_{2,3}, \rho_{c_1,3}, \rho_{c_2,3})$;
$\mathsf{Rcpt}_0 \leftarrow (v_0, \mathbf{pk}[id], \mathbf{sk}[id], r_{c_1,0}, r_{c_2,0}, b_0)$; $\mathsf{Rcpt}_1 \leftarrow (v_0, \mathbf{pk}[id], sk'_{id}, r_{c_1,1}, r_{c_2,1}, b_1)$
if $\mathsf{Valid}(b_\beta, \mathsf{bb}_\beta, \mathcal{L}, pk) = 0$ **return** $\perp$
$\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_0, \mathcal{L}, sk)$; $\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_1, \mathsf{bb}_1, \mathcal{L}, sk)$; $\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_2, \mathsf{bb}_0, \mathcal{L}, sk)$; $\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_3, \mathsf{bb}_1, \mathcal{L}, sk)$
return Rcpt_β

Figure 4.21: Oracle $\mathcal{O}\mathsf{RF}$ for Game 6 in the proof of Lemma 2.

Therefore, we conclude that this step does not change the view of the adversary. Therefore,

$$\left|\Pr[S_6] - \Pr[S_7]\right| = 0$$

.

In Game 7, the inputs to $\mathcal{A}$ are identical for $\beta = 0$ and $\beta = 1$. In fact, for a voter $id \in \mathcal{Q}\mathsf{corrupt}$ (i.e. the voter is corrupt), the inputs to $\mathcal{A}$ are independent of β. If $id \notin \mathcal{Q}\mathsf{corrupt}$, the voter can be input to oracles $\mathcal{O}\mathsf{reg}$, $\mathcal{O}\mathsf{vote}$ and $\mathcal{O}\mathsf{RF}$. Our game hops ensure that these oracles are identical and do not depend on β. Moreover, oracle $\mathcal{O}\mathsf{board}$ outputs a public bulletin board that is independent of β, and the tally does not reveal the bit β (it is always computed for the left-hand oracle inputs of $\mathcal{A}$). Therefore $\Pr[S_7] = 1/2$. We now have that

$$\left|\Pr[S_0] - \frac{1}{2}\right| \leq \mathsf{negl}_{\mathsf{NIZK\text{-}ZK}} + \mathsf{negl}_{\mathsf{ENC\text{-}ZK}} + \mathsf{negl}_{\mathsf{CONJ\text{-}ZK}} + (k_1 + 2 \cdot k_2) \cdot \mathsf{negl}_{\mathsf{NM\text{-}CPA}}$$

which is negligible as required. We conclude that JCJ satisfies E-RF. □

Reflections on Theorems 4 and 5. In the introduction to Chapter 3, we recalled intuitive definitions of ballot secrecy, receipt-freeness and coercion-resistance that form a

$\mathcal{D}_5^{\mathcal{O}\mathsf{enc},\mathcal{O}\mathsf{dec}}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}})$

$\beta \leftarrow\!\!\$\ \{0,1\}$
$\mathbf{pk} \leftarrow ();\ \mathbf{sk} \leftarrow ();\ \mathsf{bb}_0 \leftarrow ();\ \mathsf{bb}_1 \leftarrow ()$
$\mathcal{Q}\mathsf{reg} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset;\ \mathcal{L} \leftarrow \emptyset$
$\mathsf{flag} \leftarrow 0$
$(pp_{\mathsf{NIZK}}, \tau) \leftarrow\!\!\$\ \mathsf{NIZK.SimSetup}(1^\lambda)$
$pk \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}(st)$
if $\beta' = \beta$ **return** 1

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{cast}/\mathcal{O}\mathsf{board}$

As in Game 0.

$\mathcal{O}\mathsf{vote}/\mathcal{O}\mathsf{tally}$

As in Game 5.

$\mathcal{O}\mathsf{RF}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$c_{1,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,0});\quad c_{2,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,0})$
$\rho_{c_1,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}));\quad \rho_{c_2,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}))$
$b_0 \leftarrow (c_{1,0}, c_{2,0}, \rho_{c_1,0}, \rho_{c_2,0});$
$sk'_{id} \leftarrow\!\!\$\ \mathsf{FakeKey}(pk, \mathbf{pk}[id], \mathbf{sk}[id])$
$c_{1,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,1});\quad c_{2,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, sk'_{id}; r_{c_2,1})$
$\rho_{c_1,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, \mathcal{C}));\quad \rho_{c_2,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, c_{2,1}))$
$b_1 \leftarrow (c_{1,1}, c_{2,1}, \rho_{c_1,1}, \rho_{c_2,1})$
$c_{1,2} \leftarrow\!\!\$\ \mathcal{O}\mathsf{enc}(v \leftarrow\!\!\$\ \mathcal{C}, v_0);\quad c_{2,2} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, sk'_{id}; r_{c_2,2})$
$\rho_{c_1,2} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,2}, \mathcal{C}));\quad \rho_{c_2,2} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,2}, c_{2,2}))$
$b_2 \leftarrow (c_{1,2}, c_{2,2}, \rho_{c_1,2}, \rho_{c_2,2})$
$c_{1,3} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,3});\quad c_{2,3} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,3})$
$\rho_{c_1,3} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}));\quad \rho_{c_2,3} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}))$
$b_3 \leftarrow (c_{1,3}, c_{2,3}, \rho_{c_1,3}, \rho_{c_2,3});$
$\mathsf{Rcpt}_0 \leftarrow (v_0, \mathbf{pk}[id], \mathbf{sk}[id], r_{c_1,0}, r_{c_2,0}, b_0);\ \mathsf{Rcpt}_1 \leftarrow (v_0, \mathbf{pk}[id], sk'_{id}, r_{c_1,1}, r_{c_2,1}, b_1)$
if $\mathsf{Valid}(b_\beta, \mathsf{bb}_\beta, \mathcal{L}, pk) = 0$ **return** $\perp$
$\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_0, \mathcal{L}, sk);\ \mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_1, \mathsf{bb}_1, \mathcal{L}, sk);\ \mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_2, \mathsf{bb}_0, \mathcal{L}, sk);\ \mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_3, \mathsf{bb}_1, \mathcal{L}, sk)$
return Rcpt_β

Figure 4.22: Distinguisher $\mathcal{D}_5$ that distinguishes Game 5 and Game 6 in the proof of Lemma 2.

hierarchy of privacy properties for e-voting schemes [51]. That is, the relationship between these properties is believed to be linear, with receipt-freeness strengthening ballot secrecy and coercion-resistance strengthening receipt-freeness. In particular, coercion-resistance can be defined as receipt-freeness plus protection against abstention, randomisation and simulation attacks. The linear relationship suggests that coercion-resistant e-voting schemes should satisfy receipt-freeness. However, Theorems 4 and 5 show that there exists an e-voting scheme, namely JCJ, that can only satisfy our weaker notion of receipt-freeness E-RF, despite the fact that JCJ is intuitively coercion-resistant. Moreover, there exists schemes that are intuitively receipt-free but not coercion-resistant (and do not require an evasion strategy) that satisfy our stronger notion N-RF. We do not prove this formally but note that, in [38], Cortier *et al.* define an intuitively receipt-free e-voting scheme that satisfies CCFG. We will show in Chapter 4.6.4 that CCFG is strictly stronger than N-RF,

$\mathcal{O}\mathsf{RF}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
parse pk as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pk_{\mathsf{PKE}})$
$c_{1,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,0})$; $c_{2,0} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,0})$
$\rho_{c_1,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}))$; $\rho_{c_2,0} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}))$
$b_0 \leftarrow (c_{1,0}, c_{2,0}, \rho_{c_1,0}, \rho_{c_2,0})$;
$sk'_{id} \leftarrow\!\!\$\ \mathsf{FakeKey}(pk, \mathbf{pk}[id], \mathbf{sk}[id])$
$c_{1,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,1})$; $c_{2,1} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, sk'_{id}; r_{c_2,1})$
$\rho_{c_1,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, \mathcal{C}))$; $\rho_{c_2,1} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,1}, c_{2,1}))$
$b_1 \leftarrow (c_{1,1}, c_{2,1}, \rho_{c_1,1}, \rho_{c_2,1})$
$c_{1,2} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,2})$; $c_{2,2} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, sk'_{id}; r_{c_2,2})$
$\rho_{c_1,2} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,2}, \mathcal{C}))$; $\rho_{c_2,2} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,2}, c_{2,2}))$
$b_2 \leftarrow (c_{1,2}, c_{2,2}, \rho_{c_1,2}, \rho_{c_2,2})$
$c_{1,3} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, v_0; r_{c_1,3})$; $c_{2,3} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}}, \mathbf{sk}[id]; r_{c_2,3})$
$\rho_{c_1,3} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{enc}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, \mathcal{C}))$; $\rho_{c_2,3} \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}_{\mathsf{conj}}(pp_{\mathsf{NIZK}}, \tau, (pk_{\mathsf{PKE}}, c_{1,0}, c_{2,0}))$
$b_3 \leftarrow (c_{1,3}, c_{2,3}, \rho_{c_1,3}, \rho_{c_2,3})$;
$\mathsf{Rcpt}_0 \leftarrow (v_0, \mathbf{pk}[id], \mathbf{sk}[id], r_{c_1,0}, r_{c_2,0}, b_0)$; $\underline{\mathsf{Rcpt}_1 \leftarrow (v_0, \mathbf{pk}[id], \mathbf{sk}[id], r_{c_1,3}, r_{c_2,3}, b_3)}$
if $\mathsf{Valid}(b_\beta, \mathsf{bb}_\beta, \mathcal{L}, pk) = 0$ **return** $\perp$
$\underline{\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_0, \mathsf{bb}_0, \mathcal{L}, sk)}$; $\underline{\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_3, \mathsf{bb}_1, \mathcal{L}, sk)}$; $\underline{\mathsf{bb}_0 \leftarrow\!\!\$\ \mathsf{Append}(b_2, \mathsf{bb}_0, \mathcal{L}, sk)}$; $\underline{\mathsf{bb}_1 \leftarrow\!\!\$\ \mathsf{Append}(b_1, \mathsf{bb}_1, \mathcal{L}, sk)}$
return Rcpt_β

Figure 4.23: Oracle $\mathcal{O}\mathsf{RF}$ for Game 7 in the proof of Lemma 2.

which indicates that the scheme defined in [38] will also satisfy N-RF. We conclude that e-voting schemes that do not require an evasion strategy can satisfy our stronger notion of receipt-freeness. On the other hand, intuitively coercion-resistant schemes that require an evasion strategy may provide additional protection from abstention, randomisation and simulation attacks, but may only satisfy weak notions of receipt-freeness. Therefore, our results suggests that the relationship between receipt-freeness and coercion-resistance is not strictly linear, and rigorously defining this relationship is an open problem.

4.6.3 A Comparison with BPRIV

As receipt-freeness strengthens ballot secrecy, we expect that our definitions of receipt-freeness will imply BPRIV, and we formally prove that this is the case. Specifically, we obtain Theorem 6, which proves that E-RF implies BPRIV.

Theorem 6 (E-RF $\Rightarrow$ BPRIV)**.** *Let* Γ *be an e-voting scheme that satisfies* E-RF*. Then* Γ *also satisfies* BPRIV *when adapted to schemes with internal voter registration.*

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary that succeeds in the BPRIV experiment for e-voting scheme Γ when BPRIV is adapted to schemes with internal voter registration. That is, in the BPRIV experiment (Figure 3.2), adversary $\mathcal{A}$ is defined as $(\mathcal{A}_1, \mathcal{A}_2)$ such

that $\mathcal{A}_1$, on input the election public key pk can query oracle $\mathcal{O}\mathsf{reg}$ as defined for N-RF and E-RF. Then, $\mathcal{A}_2$ proceeds as $\mathcal{A}$ in the original BPRIV experiment. We show that we can construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that succeeds in the E-RF experiment. We define $\mathcal{B}$ in Figure 4.24.

It is clear that the inputs to $\mathcal{A}_1$ and $\mathcal{A}_2$ produced respectively by $\mathcal{B}_1$ and $\mathcal{B}_2$ are distributed identically to the BPRIV experiment. Indeed, election public key pk is generated identically and $\mathcal{B}$ can respond to $\mathcal{A}$'s oracle queries by calling the identical oracle in the E-RF experiment. In particular, $\mathcal{B}_2$ never queries oracle $\mathcal{O}\mathsf{RF}$ as $\mathcal{A}_2$ does not have access to this oracle. Therefore, $\mathcal{B}$ perfectly simulates the BPRIV experiment to $\mathcal{A}$.

We now show that $\mathcal{B}$ succeeds in the E-RF experiment. That is, we show that $\mathcal{B}$ returns $\beta' = \beta$. By assumption, if $\mathcal{A}$ succeeds in the BPRIV experiment then $\mathcal{A}$ outputs $\beta' = \beta$. Hence, $\mathcal{B}$ returns $\beta' = \beta$ and the result holds by contradiction. □

$\underline{\mathcal{B}_1^{\mathcal{O}\mathsf{reg}}(pk)}$

$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$

return st

$\underline{\mathcal{B}_2^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}(st)}$

$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}(st)$

return β'

$\underline{\mathcal{O}\mathsf{vote}/\mathcal{O}\mathsf{RF}/\mathcal{O}\mathsf{cast}/\mathcal{O}\mathsf{board}/\mathcal{O}\mathsf{tally}}$

Run the identical oracle in the E-RF experiment.

Figure 4.24: Adversary $\mathcal{B}$ that breaks the BPRIV security of Γ in the proof of Theorem 6.

Recall that, as a result of Theorems 4 and 5, N-RF is stronger than E-RF. Combining this with the result in Theorem 6, we conclude that N-RF is stronger than BPRIV. Theorem 6 implies that E-RF is at least as strong (i.e., implies) BPRIV. We now discuss whether E-RF is stronger than BPRIV, showing that we cannot conclusively confirm this to be the case.

Consider algorithm $\mathsf{CoercedVote}(v_0, v_1, pk_{id}, sk_{id}, b_0, pk, \mathsf{coins})$ that is defined to perform the following actions:

$$b_1 \leftarrow\!\!\$\ \mathsf{Vote}(v_0, pk_{id}, sk_{id}, pk; \mathsf{coins});\ b_2 \leftarrow ();\ b_3 \leftarrow ()$$

Then, oracle $\mathcal{O}\mathsf{RF}$ appends ballot b_1 to bb_1 and sets $\mathsf{Rcpt}_1 = \mathsf{Rcpt}_0$. That is to say, oracle $\mathcal{O}\mathsf{RF}$ appends identical ballots to bb_0 and bb_1 and returns a receipt to the adversary that is independent of bit β. If algorithm CoercedVote is defined in this way, $\mathcal{O}\mathsf{RF}$ does not provide the adversary with any input that the adversary can use to guess β. In essence,

this means that an adversary in the E-RF experiment does not obtain any more information that can be used to guess β than an adversary in the BPRIV experiment.

Defining algorithm CoercedVote in this way goes against the spirit of the algorithm, which is intended to model evasion strategies. However, CoercedVote is defined generically to capture a range of evasion strategies, and is therefore open to this kind of abuse. Indeed, a scheme that is intuitively only ballot secret may satisfy E-RF if algorithm CoercedVote is defined in this way. Therefore, we can conclude that E-RF is only at least as strong as BPRIV, but is not necessarily stronger. Despite this, we believe that our definitions are valuable. In particular, the analysis presented in this section adds valuable insights to the current understanding of the relationship between receipt-freeness and coercion-resistance. Furthermore, we believe that formal definitions of receipt-freeness for e-voting is an open research problem and that our definitions can inform future research in this field.

4.6.4 Comparing N-RF and E-RF with Existing Definitions

We conclude this chapter by comparing our definitions with the definitions we analyse in Chapters 4.3 – 4.5, summarising how we address issues with existing definitions.

Comparing E-RF and existing work We introduce E-RF to capture e-voting schemes that provide voters with an evasion strategy and show that E-RF captures two popular strategies: deniable vote updating and fake credentials. In comparison to existing work, only BKV considers evasion strategies, but is limited to deniable vote updating. E-RF, on the other hand, captures schemes such as JCJ, which is out of scope of KZZ, CCFG and BKV. Therefore, we believe that E-RF is a first step in defining receipt-freeness as an indistinguishability experiment for schemes with an evasion strategy.

Comparing N-RF and existing work N-RF is similar in spirit to KZZ. That is, RF assumes that a voter attempts to prove their vote through an interaction with the attacker, during which the voter provides the attacker with a 'receipt'. Accordingly, recalling our discussions at the end of Chapter 4.3, which discussed the attacker model of KZZ, we surmise that N-RF captures a well-established intuition of receipt-freeness. We also note that N-RF avoids the completeness and soundness issues of KZZ. Indeed, N-RF does not

limit the adversary to one, and only one, query on behalf of each eligible voter. Moreover, by working in the BPRIV model, RF does not limit the class of result functions that can be considered. That being said, RF introduces a new limitation: RF allows *static* corruption of voters only. Thus, we see that there is a trade-off with respect to the features and benefits that a definition can provide.

N-RF also addresses the completeness and soundness issues of CCFG. We present definitions of strong consistency and strong correctness for our syntax, and limit to static voter corruption only. In contrast to CCFG, we define oracle $\mathcal{O}$RF to take two *votes* as input, rather than two *ballots*. In this way, to satisfy N-RF, we do not require that a submitted ballot is modified before it is appended to the ballot box and/or the public bulletin board is output. However, like CCFG, N-RF does not capture JCJ. That being said, we also define E-RF that does capture JCJ. Therefore, N-RF improves upon the existing literature for e-voting schemes without an evasion strategy.

As promised in the discussion on Theorems 4 and 5 in Chapter 4.6.2, we now prove that CCFG is stronger than N-RF. We expect this result as CCFG captures an intuitively stronger attacker whereas N-RF captures the intuition of KZZ. We first show that CCFG implies N-RF, and then prove that the reverse implication does not hold. We obtain Theorems 7 and 8.

Theorem 7 (CCFG $\Rightarrow$ N-RF)**.** *Let* Γ *be an e-voting scheme that satisfies* CCFG*. Then* Γ *also satisfies* N-RF*.*

Proof. To prove this result, we first note that algorithms Append and Publish modify ballots submitted to the ballot box and bulletin board respectively. As such, whenever a ballot is posted by a voter, the ballot submitted by the voter will be altered by the voting scheme. For this reason, we can define $\mathsf{Rcpt}_0 = \mathsf{Rcpt}_1 = (v_0, pk_{id}, sk_{id}, \mathsf{coins}, b_\beta)$ where v_0 is the vote choice of voter id with credentials (pk_{id}, sk_{id}) and b_0 is the ballot computed by algorithm Vote for vote v_0 with randomness coins. In particular, Rcpt_1 indicates the receipt of a voter that attempts to prove a vote for v_0 where the voter actually submitted a ballot for a vote v_1. If the ballot submitted by the voter only appears on the public bulletin board in a modified form, then the voter can simply claim that they submitted b_0.

With the receipt defined as described, we define an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that succeeds

in the N-RF experiment for e-voting scheme Γ. We show that we can construct an adversary $\mathcal{B}$ that succeeds in the CCFG experiment. We define $\mathcal{B}$ in Figure 4.25.

The inputs to $\mathcal{A}$ produced by $\mathcal{B}$ are distributed identically to the RF experiment. Indeed, the election public key pk is generated identically and $\mathcal{B}$ responds to queries to oracles $\mathcal{O}$vote, $\mathcal{O}$cast, $\mathcal{O}$board and $\mathcal{O}$tally by querying the identical oracle in the CCFG experiment. $\mathcal{B}$ answers queries to $\mathcal{O}$reg by calling its own $\mathcal{O}$reg and, if $d = 1$, queries oracle $\mathcal{O}$corrupt to obtain the secret credential of the registered voter. $\mathcal{B}$ then returns the voter's public credential or the voter's credential pair depending on whether $\mathcal{A}$ indicates the voter should be corrupt or not. Finally, $\mathcal{B}$ answers queries to $\mathcal{O}$RF by constructing ballots on behalf of the voter and submitting the two ballots to oracle $\mathcal{O}$RF in the CCFG experiment. $\mathcal{B}$ then returns Rcpt to $\mathcal{A}$, which is defined identically regardless of β. Therefore, $\mathcal{B}$ perfectly simulates the RF experiment to $\mathcal{A}$.

We now show that $\mathcal{B}$ succeeds in the CCFG experiment. By assumption. if $\mathcal{A}$ succeeds in the RF experiment then $\mathcal{A}$ outputs $\beta' = \beta$. Hence, $\mathcal{B}$ returns $\beta' = \beta$ and the result holds by contradiction. □

To show that the reverse implication does not hold, we define an e-voting scheme that satisfies N-RF but does not satisfy CCFG.

Definition 59 (E-voting scheme Γ')**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Valid}, \mathsf{Append}, \mathsf{Tally}, \mathsf{Publish}, \mathsf{Verify})$ *be an e-voting scheme that satisfies* RF*. We define a modified scheme* $\Gamma' = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}', \mathsf{Valid}', \mathsf{Append}', \mathsf{Tally}, \mathsf{Publish}, \mathsf{Verify})$ *such that*

$\mathsf{Vote}'(v, pk_{id}, sk_{id}, pk)$ On input of vote v, voter credentials pk_{id} and sk_{id}, and election public key pk, algorithm Vote' runs $b' \leftarrow\!\!\$\ \mathsf{Vote}(v, pk_{id}, sk_{id}, pk)$ and returns $b \coloneqq 0 \parallel b'$.

$\mathsf{Valid}'(b, \mathsf{bb}, \mathcal{L}, pk)$ On input of ballot b, ballot box bb, list $\mathcal{L}$ and election public key pk, algorithm Valid' parses ballot b as $\alpha \parallel b'$. If $\alpha \notin \{0, 1\}$, return 0. Else, return $\mathsf{Valid}(b', \mathsf{bb}, \mathcal{L}, pk)$.

$\mathsf{Append}'(b, \mathsf{bb}, \mathcal{L}, pk)$ On input of ballot b, ballot box bb, list $\mathcal{L}$ and election public key pk, algorithm Append' returns 0 if $\mathsf{Valid}'(b, \mathsf{bb}, \mathcal{L}, pk) = 0$. Else, parse ballot b as $\alpha \parallel b'$. If $\alpha = 0$, return $\mathsf{bb} \leftarrow\!\!\$\ \mathsf{Append}(b', \mathsf{bb}, \mathcal{L}, pk)$. If $\alpha = 1$, return $\mathsf{bb} \leftarrow \mathsf{bb} \parallel b'$.

$\mathcal{B}^{\mathcal{O}\mathsf{setup},\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}()$

$\mathbf{pk} \leftarrow (); \ \mathbf{sk} \leftarrow ()$
$\mathcal{Q}\mathsf{reg} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset; \ \mathcal{L} \leftarrow \emptyset$
$\mathsf{flag} \leftarrow 0$
$pk \leftarrow_\$ \mathcal{O}\mathsf{setup}()$
$st \leftarrow_\$ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
$\beta' \leftarrow_\$ \mathcal{A}_2^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}(st)$
return β'

$\mathcal{O}\mathsf{reg}_{(pk,\mathcal{L},\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk})}(id, d)$

if $id \notin \mathcal{V} \ \vee \ id \in \mathcal{Q}\mathsf{reg}$ **return** $\perp$
$pk_{id} \leftarrow_\$ \mathcal{O}\mathsf{reg}(id)$
$\mathcal{Q}\mathsf{reg} \leftarrow \mathcal{Q}\mathsf{reg} \cup \{id\}$
$\mathbf{pk}[id] \leftarrow pk_{id}$
if $d = 0$: **return** pk_{id}
if $d = 1$
  $\mathcal{Q}\mathsf{corrupt} \leftarrow \mathcal{Q}\mathsf{corrupt} \cup \{id\}$
  $(pk_{id}, sk_{id}) \leftarrow_\$ \mathcal{O}\mathsf{corrupt}(id)$
  $\mathbf{sk}[id] \leftarrow sk_{id}$
  return (pk_{id}, sk_{id})
else return $\perp$

$\mathcal{O}\mathsf{tally}_{(sk,\tau,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathsf{flag})}()$

$\mathsf{flag} \leftarrow 1$
$(r, \rho) \leftarrow_\$ \mathcal{O}\mathsf{tally}()$
return (r, ρ)

$\mathcal{O}\mathsf{RF}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
$(pk_{id}, sk_{id}) \leftarrow_\$ \mathcal{O}\mathsf{corrupt}(id)$
$b_0 \leftarrow_\$ \mathsf{Vote}(v_0, pk_{id}, sk_{id}, pk; \mathsf{coins})$
$b_1 \leftarrow_\$ \mathsf{Vote}(v_1, pk_{id}, sk_{id}, pk; \mathsf{coins}')$
$\top \leftarrow_\$ \mathcal{O}\mathsf{RF}(id, b_0, b_1)$
return $\mathsf{Rcpt} = (v_0, pk_{id}, sk_{id}, \mathsf{coins}, b_0)$

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},\mathbf{pk},\mathbf{sk},\mathsf{flag})}(id, v_0, v_1)$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
$\top \leftarrow_\$ \mathcal{O}\mathsf{vote}(id, v_0, v_1)$
return $\top$

$\mathcal{O}\mathsf{cast}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{corrupt},\mathsf{flag})}(id, b)$

if $\mathsf{flag} = 1$ **return** $\perp$
if $id \notin \mathcal{Q}\mathsf{corrupt}$ **return** $\perp$
$\top \leftarrow_\$ \mathcal{O}\mathsf{cast}(id, b)$
return $\top$

$\mathcal{O}\mathsf{board}$

As usual.

Figure 4.25: Adversary $\mathcal{B}$ that succeeds in the CCFG experiment in the proof of Theorem 7.

We obtain Theorem 8, which we prove in Lemmata 3 and 4.

Theorem 8 (N-RF $\not\Rightarrow$ CCFG)**.** *If there exists an e-voting scheme* Γ *that satisfies* N-RF *then we can define e-voting scheme* Γ' *(Definition 59) that satisfies* RF *but does not satisfy* CCFG*.*

Lemma 3. *Let* Γ *be an e-voting scheme that satisfies* N-RF*. Then* Γ' *(Definition 59) also satisfies* N-RF*.*

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary that succeeds in the N-RF experiment for scheme Γ'. We show that we can construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that succeeds in the N-RF experiment for Γ. We define $\mathcal{B}$ in Figure 4.26.

The inputs to $\mathcal{A}_1$ and $\mathcal{A}_2$ produced respectively by $\mathcal{B}_1$ and $\mathcal{B}_2$ are distributed identically to the N-RF experiment for Γ'. Indeed, the election public key pk is generated identically, and oracle queries can be answered identically. Specifically, $\mathcal{B}$ responds to queries to

oracles $\mathcal{O}\mathsf{reg}$ and $\mathcal{O}\mathsf{tally}$ by querying the identical oracle. Moreover, $\mathcal{B}$ can respond to queries to oracles $\mathcal{O}\mathsf{vote}$ and $\mathcal{O}\mathsf{RF}$ by calling the identical oracles because, unless a ballot is pre-appended with a bit 1 (which cannot happen in the event of queries to $\mathcal{O}\mathsf{vote}$ or $\mathcal{O}\mathsf{RF}$ as the oracles call algorithm Vote which always pre-appends the ballot with a bit 0), the oracles are identical. To answer queries to $\mathcal{O}\mathsf{cast}$, $\mathcal{B}$ parses the ballot, removes the pre-appended bit, and submits a query to $\mathcal{O}\mathsf{cast}$. If the pre-appended bit is 1, $\mathcal{B}$ maintains a list of the submitted ballots and the ballot that is appended to the public bulletin board. Then, when $\mathcal{A}_2$ queries $\mathcal{O}\mathsf{board}$, $\mathcal{B}$ can query the identical oracle in the $\mathsf{N\text{-}RF}$ experiment and replace any ballots on the returned bulletin board that are submitted by $\mathcal{A}_2$ and are pre-appended with a bit 1, with the ballot submitted by $\mathcal{A}_2$ to oracle $\mathcal{O}\mathsf{cast}$. In this way, if the ballot is pre-appended with a bit 1 (and, hence, appears on the ballot box as submitted) $\mathcal{B}$ can perfectly simulate the view of $\mathcal{A}$ in the RF experiment for Γ'. Therefore, $\mathcal{B}$ perfectly simulates the $\mathsf{N\text{-}RF}$ experiment for Γ' to $\mathcal{A}$.

We now show that $\mathcal{B}$ succeeds in the $\mathsf{N\text{-}RF}$ experiment. That is, we show that $\mathcal{B}$ returns $\beta' = \beta$. By assumption, if $\mathcal{A}$ succeeds in the $\mathsf{N\text{-}RF}$ experiment for Γ' then $\mathcal{A}$ outputs $\beta' = \beta$. Hence, $\mathcal{B}$ returns $\beta' = \beta$ and the result holds by contradiction. □

$\mathcal{B}_1^{\mathcal{O}\mathsf{reg}}(pk)$

$B \leftarrow \emptyset$
$st_1 \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg}}(pk)$
return st_1

$\mathcal{B}_2^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}(st_1)$

$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}$

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{vote}/\mathcal{O}\mathsf{RF}/\mathcal{O}\mathsf{tally}$

As usual.

$\mathcal{O}\mathsf{cast}_{(pk,\mathcal{L},\mathsf{bb}_0,\mathsf{bb}_1,\mathcal{Q}\mathsf{corrupt},\mathsf{flag})}(id, b)$

parse b as $\alpha \parallel b'$
$\top \leftarrow\!\!\$\ \mathcal{O}\mathsf{cast}(id, b')$
$\mathcal{BB}_\beta = (\ldots, b^*) \leftarrow\!\!\$\ \mathcal{O}\mathsf{board}()$
if $\alpha = 1 \quad B \leftarrow B \cup \{(b', b^*)\}$

$\mathcal{O}\mathsf{board}_{(\mathsf{bb}_\beta)}()$

$\mathcal{BB}_\beta = (b_1, \ldots, b_{|\mathcal{BB}_\beta|}) \leftarrow\!\!\$\ \mathcal{O}\mathsf{board}()$
for every entry b in $\mathcal{BB}_\beta$, if there exists $(b', b) \in B$, replace b with b'
return $\mathcal{BB}'_\beta$

Figure 4.26: Adversary $\mathcal{B}$ that breaks the $\mathsf{N\text{-}RF}$ security of scheme Γ in the proof of Lemma 3.

Lemma 4. Γ' *(Definition 59) does not satisfy* CCFG.

Proof. We define an adversary $\mathcal{A}$ that succeeds in the CCFG experiment against scheme Γ' in Figure 4.27. In particular, $\mathcal{A}$ constructs two ballots pre-appended with 1 and submits them via a query to $\mathcal{O}\mathsf{RF}$. Then, $\mathcal{A}$ queries $\mathcal{O}\mathsf{board}$. $\mathcal{BB}_\beta$ contains the single entry b_0^* (if $\beta = 0$) or b_1^* (if $\beta = 1$). Therefore, $\mathcal{A}$ can correctly guess β and e-voting scheme Γ' does not satisfy CCFG. □

$\mathcal{A}^{\mathcal{O}\mathsf{setup},\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{RF},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{tally}}()$

$pk \leftarrow\!\$\ \mathcal{O}\mathsf{setup}();\ pk_{id} \leftarrow\!\$\ \mathcal{O}\mathsf{reg}(id);\ (pk_{id}, sk_{id}) \leftarrow\!\$\ \mathcal{O}\mathsf{corrupt}(id)$
$b_0 \leftarrow\!\$\ \mathsf{Vote}'(v_0, pk_{id}, sk_{id}, pk);\ b_1 \leftarrow\!\$\ \mathsf{Vote}'(v_1, pk_{id}, sk_{id}, pk)$
parse b_0 as $0 \parallel b'_0 \ \wedge\ b_1$ as $0 \parallel b'_1$
$b^*_0 \leftarrow 1 \parallel b'_0;\quad b^*_1 \leftarrow 1 \parallel b'_1$
$\top \leftarrow\!\$\ \mathcal{O}\mathsf{RF}(id, b^*_0, b^*_1);\ \mathcal{BB}_\beta \leftarrow\!\$\ \mathcal{O}\mathsf{board}()$
if $\mathcal{BB}_\beta = (b'_0)$ **return** 0
if $\mathcal{BB}_\beta = (b'_1)$ **return** 1

Figure 4.27: Adversary $\mathcal{A}$ that breaks the CCFG security of scheme Γ' in the proof of Lemma 4.

4.7 Concluding Remarks

In this chapter, we analysed three definitions of receipt-freeness and uncovered completeness and soundness issues. We also found that each definition considers a different attacker model. We then presented new definitions of receipt-freeness N-RF and E-RF that address the completeness and soundness issues of existing definitions. Finally, we concluded this chapter with an analysis of our definitions. From our analysis, we abstract two key takeaways. Firstly, the relationship between receipt-freeness and coercion-resistance is unclear and does not appear to be linear. We expect that an intuitively coercion-resistant e-voting scheme will satisfy receipt-freeness. However, in this chapter, we showed that JCJ does not satisfy *three* definitions of receipt-freeness, namely, KZZ, CCFG and N-RF. Moreover, we discovered that there exists intuitively receipt-free (but not coercion-resistant) e-voting schemes that can satisfy the stronger notions of N-RF and CCFG, but intuitively coercion-resistant schemes can only satisfy the weaker notion of E-RF. We believe the relationship between these properties warrants further exploration and hope that the results in this chapter inspire this. Secondly, our analysis of existing definitions demonstrated that it is challenging to provide rigorous and complete definitions of receipt-freeness for e-voting schemes. As a result, we took the approach of presenting two definitions of receipt-freeness. Unifying these two variants is an open problem; the answer could present more rigorous definitions and contribute to understanding the relationship between receipt-freeness and coercion-resistance.

More generally, we have seen in this chapter that defining advanced notions of privacy for e-voting is complex, and early formal definitions are prone to weaknesses and limitations.

In the next chapter, we question whether defining these complex notions simplifies in a different setting and whether anything can be learned about these notions when considered in a different setting. To this end, in the next chapter, we consider receipt-freeness and coercion-resistance in the more fundamental setting of digital signatures.

Chapter 5

On the Incoercibility of Digital Signatures

Contents

In this chapter, we introduce incoercible signatures, a variant of a standard digital signature scheme that enables signers, when coerced to produce a signature for a message chosen by an attacker, to generate fake signatures that are indistinguishable from real signatures. We require that this holds even if the signer is compelled to reveal their full history (including their secret signing keys and any randomness used to produce keys/signatures) to the attacker. We present a formal security model for incoercible signatures and demonstrate that an incoercible signature scheme can be viewed as a transformation of any generic signature scheme. Finally, we demonstrate a relationship between receipt-freeness for incoercible signatures and the closely related notion of deniability, namely, that a variant of deniable encryption is necessary to build incoercible signature schemes.

5.1 Introduction

In Chapters 3 and 4 we saw that receipt-freeness and coercion-resistance were originally defined for the setting of e-voting, and that these properties can be formalised as a hierarchy [52]. However, the boundary and relationship between these properties is not always clear, as we saw in Chapter 4. In this chapter, we direct our attention away from e-voting and towards a more fundamental cryptographic primitive: digital signature schemes. Specifically, we consider receipt-freeness and coercion-resistance within the context of digital signatures. We collectively refer to receipt-freeness and coercion-resistance as incoercibility in this chapter. In doing so, we adopt the naming convention from definitions of incoercibility that have been put forth in the generic multi-party protocol setting [35, 127], which includes e-voting. In particular, in [127], incoercibility is defined as a hierarchy of the receipt-freeness and coercion-resistance properties.

Traditionally, security models for signatures schemes focused on the property of existential unforgeability against an adaptive chosen-message attack (EUF-CMA) [71] (cf., Definition 18). This captures the property that an attacker, with access only to public information, cannot output a valid message/signature pair (i.e., a forgery). In recognition of the threat posed by attackers with stronger, potentially coercive, capabilities, several security notions have been proposed that strengthen the traditional security model for signature schemes (cf. Chapter 5.2). These notions have predominantly focused on *key exposure attacks* [11, 56, 57, 76, 82, 83, 84, 102], whereby a signer is coerced to reveal (part of) their secret key to an attacker. The proposed schemes in the literature generally allow the signer to recover from the key exposure attack, most commonly by updating their secret key. The security of such schemes requires that an attacker cannot produce a valid signature on behalf of a signer whose key has been exposed.

Previous works on coercive attacks do not consider techniques that allow signers to evade coercion without detection, which can arise in the context of digitally signed messages. For example, consider a protocol in which members of an organisation can produce endorsements on behalf of other members. Such endorsements can be used, for instance, to recommend a particular member for a role within the organisation. It may be desirable that members *publicly* endorse other members; endorsements accompanied with publicly verifiable signatures can facilitate this. However, it is also possible that a member may be

coerced by another member (the coercer) to produce an endorsement. In this scenario, if a standard digital signature is used to sign the endorsement, the non-repudiation property of digital signatures ensures that the signer cannot deny having signed the recommendation, and other users are convinced that the signer intentionally provided it. Therefore, the signer cannot evade the coercion attempt without detection. That is, the signer produces the signature or does not. In either case, the signer's choice is clear to the coercer. As such, there is a need for signatures schemes that provide signers with some form of protection against such coercive attacks.

Some works have addressed coercive attacks and evasion of such attacks [59, 94]. These works consider an attacker that may request a signer to produce a signature for a particular message. The solution to this attack introduces a trusted authority that can detect coercion, and the security model requires that the attacker cannot distinguish a signer evading coercion from a signer who cooperates with the attacker. However, the solutions in [59, 94] model signers that reveal only their secret key to the attacker, and require interactive key generation. We address these limitations by introducing incoercible signature schemes. Broadly, incoercible signatures follow the approach of [59, 94], but distinguish themselves in the following way. Firstly, we allow the attacker to demand the *full* transcript of the signer, thereby including, along with the secret key, any randomness chosen in key generation and signing. Secondly, in addressing incoercibility, we preserve the essence of a standard signature scheme, i.e., our primitive is fully non-interactive during key generation, signing and verification.

5.1.1 Our Contributions

In this chapter we present the syntax and security model for incoercible signature schemes. We then present constructions of receipt-free and coercion-resistant incoercible signature schemes, and prove that they satisfy our security model. We close this chapter with a discussion that relates receipt-freeness and deniability, a technique from the cryptographic literature that is closely related to incoercibility. We now provide an overview of the results presented in this chapter.

Defining incoercible signatures. We present the syntax and security model for incoercible signature schemes. Incoercibility means that a signer can sign messages as they desire and, conversely, need not produce signatures for messages that they do not want to sign. Our syntax builds upon a standard digital signature scheme and captures two new requirements. Firstly, we introduce the ability to produce 'fake' signatures that are indistinguishable from real signatures. In this way, a coerced signer can produce a fake signature for a message chosen by the attacker. Fake signatures must pass public verification to ensure that an attacker cannot trivially determine whether a signature is real or not. Secondly, we require a trusted authority that can distinguish real signatures and fake signatures, which we call the authenticator. Otherwise, the signer can produce fake signatures, but no entity can detect coercion. Effectively, without a trusted authority, fake signatures are identical to real signatures. We note that the authenticator can only determine whether a signature is real or fake and learns nothing about the signer's secrets. Crucially, this means that the authenticator cannot produce forgeries on behalf of any signer. We adopt the convention of calling a signature valid or verifiable if it passes public verification, and, additionally, call a signature authentic if the designated authenticator deems the signature to be real, as opposed to fake.

Our syntax implicitly assumes that signers keep a transcript of their actions. Informally, the transcript is a record of all computations performed, and inputs generated, by the signer. We include such a transcript in our syntax to model the fact that a signer may present an attacker with 'evidence' (i.e., the transcript) to prove that they have followed the attacker's instructions. As such, the transcript is similar to the receipt generated on behalf of a voter in our receipt-freeness definitions (cf. Chapter 4.6). We describe the form of a transcript, and provide a concrete example, in Chapter 5.3. Our syntax also explicitly models the generation of a fake transcript. The fake transcript is used by a signer who attempts to evade coercion by providing a transcript that will convince an attacker that the signer followed the attacker's instructions.

Our security model for incoercible signatures captures standard notions of correctness and unforgeability, adapted to our syntax. We also introduce a further property that we call completeness, which captures the notion that an honestly generated signature is authentic. We then define receipt-freeness and the stronger notion of coercion-resistance.

In the context of digital signatures, receipt-freeness intuitively captures an attacker that

provides a coerced signer with a message to be signed and requests a 'receipt' proving that the signer produced an authentic signature. Signers either produce real signatures for the attacker's messages, or produce fake signatures. Then, the signer provides the attacker with their real or fake transcript, respectively. Following the intuition of receipt-freeness for e-voting, we capture an attacker that coerces signers, but does not attempt to control them. That is, we define the receipt-freeness attacker to simply require proof that a signer has followed their instructions; the attacker does not attempt to act on behalf of coerced signers (e.g., by producing signatures on behalf of the signer). Formally, we capture this intuition in two requirements: indistinguishability and soundness. *Indistinguishability* models an attacker that requests signatures and transcripts from signers for adversarially-chosen messages, and requires that a cooperating signer is indistinguishable from a signer that does not cooperate (i.e., produces fake signatures and transcripts). *Soundness* guarantees that a coerced signer can communicate coercion to the authenticator. In other words, the authenticator detects fake signatures.

We then introduce the stronger notion of coercion-resistance that, in contrast to receipt-freeness, captures an attacker that accesses the signer's transcript throughout the protocol and, crucially, before providing instructions to the signer. Additionally, we consider that a coercion-resistance attacker may attempt to act on behalf of signers by producing signatures on their behalf. Coercion-resistance requires indistinguishability and strong soundness. Analogously to receipt-freeness, *indistinguishability* models an attacker that views signatures and transcripts. By contrast, the coercion-resistance adversary can request the transcripts of coerced signers throughout the experiment. *Strong soundness* is the property that an attacker cannot output an authentic signature on behalf of a coerced, but uncooperative, signer, that is, a signer that presents an attacker with a fake transcript.

Achieving incoercible signatures. We present two incoercible signature scheme constructions that rely on a generic standard signature scheme and a sender-deniable encryption scheme. We prove that our first construction satisfies receipt-freeness and our second construction satisfies coercion-resistance.

Briefly, in our first construction, signers generate a key pair for a signature scheme, and the authenticator generates a key pair for a deniable encryption scheme. To sign a message, the signer produces a signature for a standard signature scheme, and encrypts a single

bit (which indicates whether the signature is coerced or not) under the public key of the authenticator. In this way, the signer communicates coercion to the authenticator, and can produce a fake transcript that proves encryption of a different bit. We show that this construction satisfies correctness, completeness, unforgeability and receipt-freeness.

In our second construction, the authenticator generates a key pair for a deniable encryption scheme, as in our receipt-freeness construction. The signer additionally produces a real and a fake secret during key generation. The real secret is encrypted under the deniable encryption scheme and included in the signer's public key. When a signer wishes to produce a real (resp., fake) signature, the signer deniably encrypts the real (resp., fake) secret and produces a signature for the message and the ciphertext. The final signature includes the ciphertext and the signature for the message/ciphertext tuple. To detect coercion, the authenticator decrypts the ciphertext included in the signature and the signer's public key. If the decrypted messages match, the signature is authentic. Otherwise, the signature is coerced. Therefore, the signer can communicate coercion to the authenticator and the deniability property of the deniable encryption scheme allows the signer to produce a fake transcript that can be revealed to an attacker. Furthermore, an attacker cannot construct an authentic signature on behalf of a coerced, but uncooperative, signer as the attacker does not have the signer's real key. Our second construction satisfies correctness, completeness, unforgeability and coercion-resistance.

Further results. In the final section of the chapter, we discuss the link between deniability and incoercibility. Deniability is closely linked to coercion, as we discuss in Chapter 5.2, and we use deniable encryption schemes in both of our incoercible signature scheme constructions. We therefore ask whether deniable encryption is necessary to achieve incoercible signature schemes. We show that, given a receipt-free incoercible signature scheme, we can construct a variant of a deniable encryption scheme. This suggests that a variant of deniable encryption is necessary to construct receipt-free incoercible signature schemes.

5.1.2 Application of Incoercible Signatures

Incoercible signatures address scenarios in which a signer is coerced to sign a message and reveal some secret information. They can be used in the scenario in which organisation members can produce endorsements on behalf of other members, which we outlined at the beginning of this chapter. In particular, incoercible signatures ensure that the coerced member can indicate coercion to an authenticator (i.e., the entity who ultimately determines which members are appointed to advertised roles). The authenticator can subsequently refuse to appoint the coercer to the role, without revealing the fact that the endorsing member evaded the coercion attempt. Accordingly, the use of incoercible signatures achieves a balance of public verifiability and protection from coercive attacks.

In this scenario, the authenticator need not publicly reveal the coercion attempt and can fabricate a reason for not appointing the coercer to the role. This is crucial to the utility of incoercible signatures. Indeed, in other scenarios, the actions of the authenticator may indicate an unsuccessful coercion attempt to the coercer. For example, if a coercer requests the transfer of funds, the sender can indicate coercion to the entity that will perform the transfer by producing an incoercible signature for the transfer request. The entity will be alerted to the coercion attack and will not transfer the funds. However, if the coercer does not receive the funds, they will be alerted to the fact that the sender evaded the coercion attempt. In this scenario it is more difficult for the authenticator to provide a convincing reason not to send the funds, and so incoercible signatures may not be useful. We also note that the authenticator should be carefully chosen such that they are the entity that can act on the coercion attempt without alerting the coercer (or any other entity) to the coercion attempt. In our syntax and constructions, we model a single authenticator, noting that, in practice, the authenticator can be chosen by the signer to ensure that the correct authority is alerted to coercion.

5.2 Related Work

In the digital signatures literature, coercion has been addressed in several ways. Security models and schemes have been proposed to protect against (partial) key exposure attacks, and mechanisms have been introduced to enable the signer to warn an authority of such

attacks, either via direct communication or by embedding a coercion warning into the signature. We discuss these approaches in more detail next, but first we explore how the work in this chapter relates to the e-voting setting.

5.2.1 Incoercibility and E-Voting

As noted, one of the earliest and most notable considerations of incoercibility was in relation to e-voting schemes. In fact, the earliest definition of incoercibility, which later and more commonly became known as the receipt-freeness property, was presented in [17]. Furthermore, it was recognised that receipt-freeness, though sufficient in some circumstances, does not fully capture the notion of incoercibility. In response, the property of coercion-resistance was defined in [88]. More generally, in [127], the Composable Incoercibility framework is introduced. In this framework, the authors provide a generic definition of incoercibility that captures notions of receipt-freeness and coercion-resistance. Similarly, in this work, we present definitions of receipt-freeness and coercion-resistance for digital signatures.

Our intuitive definitions of receipt-freeness and coercion-resistance for incoercible signature schemes are inspired by the e-voting literature. In particular, recalling that receipt-freeness in the e-voting setting protects against vote buying attacks, receipt-freeness for incoercible signatures can be though of as protecting against 'signature buying' attacks. In other words, rather than 'a voter cannot prove their vote', we consider that 'a signer cannot prove their signature (is authentic)'. Moreover, our receipt-freeness attacker is assumed not to act on behalf of signers, or demand transcripts before the signer acts on the attacker's instructions. Therefore, receipt-freeness for incoercible signatures captures a similar attacker model to receipt-freeness definition KZZ [91] (cf, Chapter 4.3) and our receipt-freeness definitions N-RF and E-RF (cf. Chapter 4.6).

Our intuitive definition of coercion-resistance captures an attacker that controls the signer. Recall that coercion-resistance for e-voting captures an attacker that controls a voter, providing protection against vote buying and randomisation, simulation and abstention attacks. Control is a key aspect of coercion-resistance for e-voting and we attempt to capture a similar intuition in the setting of incoercible signatures. Hence, we model an attacker that attempts to 'buy' signatures, and can interact with the signer (i.e., obtain

transcripts) throughout the run of the protocol. In our setting of incoercible signatures, we capture simulation attacks as defined for coercion-resistance in the e-voting setting via our soundness requirement, which allows the adversary to submit signatures on behalf of coerced signers. We do not, however, capture randomisation or abstention attacks. For incoercible signatures, generation of a randomised signature is unlikely to achieve anything and the signature will likely not verify. Finally, we omit abstention attacks, preferring instead to model a signer that may continue to produce valid and authentic signatures even whilst under the attacker's control. We believe this is realistic in the incoercible signature setting. In particular, as incoercible signatures are publicly verifiable, an abstention attack becomes trivial. That is, if a signer abstains, no signatures are produced. Conversely, if a signer does not abstain, it is trivial to see that the signer has produced a publicly verifiable signature. By contrast, in the e-voting setting, the attacker may gain something from an abstention or randomisation attack (i.e., preventing ballot casting and, as a result, potentially changing the election result). Therefore, our coercion-resistance and receipt-freeness definitions are inspired by the e-voting literature but we introduce some key differences to reflect the differences in settings.

5.2.2 Key Exposure Attacks

An attacker with the capability to obtain the secret key of a signer can easily construct valid forgeries on behalf of the signer. This is known as a key exposure attack and several solutions that allow recovery from such an attack have been presented in the literature. The earliest solutions required key distribution [54, 79, 106], but the most prevalent solution allows the signer to update their secret key [11, 56, 57, 82, 83, 84]. This approach became prominent following the introduction of forward secure signatures [11], which guarantee that an attacker cannot produce valid forgeries for any time period prior to key update. Subsequent works, for example [56, 57, 82, 83, 84], extend this guarantee, providing an assurance that an attacker cannot produce valid forgeries for time periods prior to, and following, the key update. However, many of these solutions provide these guarantees for partial key exposure only [56, 57, 83, 84]. By contrast, [11, 82, 84] consider full key exposure attacks. In a departure from this approach, monotonic signature schemes [102] allow the signer, in response to a partial key exposure attack, to update the verification algorithm, rather than the secret signing key. Monotonic signatures ensure that the attacker cannot produce valid forgeries after the verification algorithm is updated, but forgeries created

before the algorithm update are valid. In this work, we consider an attacker that demands a full transcript from the signer, and require that the attacker cannot distinguish cooperating and non-cooperating signers. Additionally, our strongest security model guarantees that an attacker cannot produce valid forgeries on behalf of a coerced signer.

5.2.3 Communicating Key Exposure

Generally, signature schemes that are secure against key exposure attacks present a limited window of time within which the attacker can produce a valid forgery, e.g., the time period when key exposure occurs in the case of [56, 57, 82, 83, 84]. Moreover, the requirement to update keys/algorithms may alert an attacker to an unsuccessful key exposure attack. For example, in the case of monotonic signatures, the attacker can check if their forgery is valid. If the forgery is invalid, it is clear that the signer updated their verification algorithm and thus evaded coercion. Similar arguments hold for other schemes secure against key exposure attacks. To address this, key evolving signature schemes [84] introduced a trusted authority with whom the signer can communicate. The signer is required to update their keys at regular intervals; if the signer does not contact the authority before key update occurs, any signatures generated since the previous key update period become invalid. In this way, forgeries created by the attacker can be invalidated. Moreover, the signer can communicate to the authority that the signatures were coerced. In doing so, the signer can evade coercion and, if the authority does not publicly invalidate the signatures, the attacker is not alerted to the unsuccessful coercion. Similarly, funkspiel schemes [76] introduce a trusted authority with whom the signer can communicate the compromise of keys. In our work, we also introduce a trusted authority that can detect coercion, though we do not require that the signer communicate with the authority. Rather, we include a 'coercion alert' in the signature that can be recovered only by the trusted authority.

5.2.4 Embedded Secret Signature Schemes

Embedded secret signatures [59] allow signers to produce signatures that contain an embedded warning that can be extracted from the signature only by a trusted authority. As such, the signer can evade coercion to sign a message chosen by the attacker, without detection by the attacker. For this reason, embedded secret signature schemes are most

closely related to our work. A key difference, however, is that the trusted authority must interact with every signer during key generation in an embedded secret signature scheme. In particular, these schemes rely on a shared secret between the trusted authority and the signer that is established during key generation. By contrast, incoercible signature schemes are non-interactive.

Our security model for incoercible signatures has a similar approach to that of embedded secret signature schemes [59]. In fact, embedded secret signatures must satisfy an indistinguishability requirement similar to our receipt-freeness indistinguishability property. That is, the security model for an embedded secret signatures captures the requirement that an attacker cannot determine whether the signer signed the attacker's message or not, when provided with the signer's secret key. In comparison, our indistinguishability requirements for receipt-freeness and coercion-resistance consider an attacker with access to full signer transcripts that include randomness used during signing and key generation, in addition to the signer's secret key, resulting in a stronger security property.

Additionally, embedded secret signatures must also satisfy a soundness requirement, called embedded secret unforgeability, which ensures that an attacker with a fake secret key cannot output a signature that does not contain an embedded warning. We capture a similar property in our coercion-resistance definition. However, our receipt-freeness definition considers a weaker attack model where the attacker never takes full control of a coerced signer. Hence, we present a hierarchy of security properties for incoercible signature schemes that incorporates notions of receipt-freeness and coercion-resistance.

5.2.5 Further Related Work

We now discuss existing work involving concepts and techniques closely related to the notion of incoercibility for signature schemes.

Subliminal channels. Subliminal channels [118] use cryptographic structures to communicate information secretly without modifying the transmitted message. Early subliminal channels were discovered in the ElGamal signature scheme [61], in which an authority can recover the subliminal message if the authority has the signer's secret signing key.

For the DSA signature scheme, which is derived from the ElGamal scheme, a subliminal channel can be achieved without knowledge of the signing key, but this comes at the cost of efficiency [119]. Subliminal channels can be used to communicate a coercion attempt to a designated authority. In fact, subliminal channels are used by embedded secret signature schemes [59, 94]. Our incoercible signature scheme constructions, on the other hand, do not require a subliminal channel, though it may be possible to construct an incoercible signature scheme with a subliminal channel.

Subversion attacks. A related area of work considers the formalisation of subversion attacks. Introduced in [131], this area initially considered the ability of an attacker to modify or replace a cryptographic algorithm with the aim of recovering secret information. In [5], subversion attacks against signatures schemes were explored and subversion resilient signature schemes were introduced. Such signature schemes ensure that an attacker cannot use the output of a modified signing algorithm to generate forgeries, if the signer cannot detect that the signing algorithm has been modified. In our work, we ensure that signers can use a modified signing algorithm to generate fake signatures without detection. In this way, the signer can indicate coercion to a trusted authority. Therefore, the signer in our work and the attacker in a subversion attack take on similar roles, but with different intended outcomes. That is, both the subversion attacker and the signer in our work make use of a modified algorithm. However, the signer in our work does this to evade coercion (and, hence, not reveal their secrets) whereas the subversion attacker uses the modified algorithm to learn secrets.

Deniability in cryptography. Deniability is a property that can be useful in the context of coercion within cryptographic protocols. Indeed, if a protocol participant is coerced to perform a particular action, deniability can be used to allow the participant to perform a different action (or no action) and prove that they followed the attackers instructions. Conversely, the participant can, when coerced *not* to do something, perform the action and deny it afterwards. In other words, deniability captures the ability of a participant to repudiate their actions.

Deniability has been considered in the context of several cryptographic primitives. Most notably, deniable encryption [20, 34, 37, 105] addresses the scenario in which the sender, if coerced to encrypt a particular message by an attacker, can produce a ciphertext for

a different message and retrospectively claim that the ciphertext encrypts the attacker's message. Deniable encryption can be used to build other primitives that offer some protection against coercive attacks [59] and, as we demonstrate in this chapter, deniable encryption can be used to construct incoercible signature schemes. Deniable authentication protocols [60] allow a sender to authenticate a message such that the receiver is convinced that authentication occurred, but cannot convince a third party that the authentication occurred. Forward deniable authentication [55] additionally ensures that the authentication protocol remains deniable even if the sender attempts to prove that authentication occurred. In designated verifier signatures schemes [85], which, as the name suggests, allow for signatures to be signed with respect to a designated verifier, signers are able to deny signatures because the designated verifier can simulate signatures on behalf of signers. Furthermore, strong designated verifier signatures schemes [85, 113] allow signers to deny signatures even when the designated verifier is trusted not to simulate signatures because signatures authored by different signers are indistinguishable to all but the designated verifier.

In connection with digital signatures, deniability, if defined analogously to other cryptographic primitives, would provide an assurance that the signer can deny having produced a signature. Under this definition, deniability captures coercive attacks where the attacker instructs the signer *not* to sign messages. However, deniability in this context seems difficult to achieve. Certainly, a message/signature pair are public, which suggests that a signer cannot deny producing the signature. In this way, incoercible signatures retain the non-repudiability property of standard signatures, That is, in an incoercible signature scheme, if a signer produces a signatures, they cannot later deny this fact, regardless of whether the signature is real or fake. Rather, our incoercible signatures allow a signer to produce *inauthentic* signatures (which are still valid and non-repudiable) and deny the fact that they are inauthentic, i.e., prove that they are authentic. As such, we balance the requirements of incoercibility and non-repudiability in the incoercible signature setting.

5.3 Syntax and Security Model for Incoercible Signatures

We now introduce the syntax of an incoercible signature scheme. We define incoercible signatures as a variant of a standard signature schemes, providing standard algorithms

$\mathsf{KGen}(pp_{\mathsf{SIG}})$	$\mathsf{Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, m)$	$\mathsf{Verify}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, m, \sigma)$
$sk_{\mathsf{SIG}} \leftarrow\!\!\$\, \mathbb{Z}_q$;	$k \leftarrow\!\!\$\, \mathbb{Z}_q$; $r \leftarrow G(g^k)$;	**if** $r \neq G(g^{H(m)\cdot s^{-1}} \cdot pk_{\mathsf{SIG}}^{r\cdot s^{-1}})$ **return** 0
$pk_{\mathsf{SIG}} \leftarrow g^{sk_{\mathsf{SIG}}}$;	$s \leftarrow (k^{-1} \cdot (H(m) + sk_{\mathsf{SIG}} \cdot r)) \mod q$;	**return** 1
return $(pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}})$	**return** $\sigma = (r, s)$	

Figure 5.1: The DSA/ECDSA signature scheme for which we define the transcript in Example 1.

for key generation, signing and public verification of a signature. Additionally, incoercible signatures protect against an attacker that attempts to coerce a signer to produce a signature for a particular message. As such, we incorporate the following additions into our syntax. Incoercible signatures are equipped with an algorithm $\mathsf{FakeSign}$ that allow signers to generate fake signatures. Moreover, we introduce a new authority, which we call the authenticator, whose role is to distinguish real signatures produced by running standard signature algorithm Sign and signatures that are the output of algorithm $\mathsf{FakeSign}$. In fact, we introduce an algorithm $\mathsf{Authenticate}$ that is run by the authenticator and outputs a bit that indicates whether the signature is authentic (i.e. a real signature that is the output of algorithm Sign) or not.

Our syntax implicitly assumes that signers maintain a transcript of their actions, which we denote trans. A transcript contains all secrets and randomness generated by the signer during key generation and signing, and its form is specific to the signature scheme being used. We provide a concrete example of this next, where we illustrate the form of trans in the case of the DSA/ECDSA signature scheme, whose description is taken from [89].

Example 1 (DSA/ECDSA)**.** *DSA (respectively, ECDSA) is defined relative to a set of public parameters* $pp_{\mathsf{SIG}} = (\mathbb{G}, g, q)$ *where* $\mathbb{G}$ *is a subgroup of* $\mathbb{Z}_p$ *(resp.,* $E(\mathbb{Z}_p)$*) of order* q *for* p *a prime. We write* g *is the generator of* $\mathbb{G}$*. We also assume that functions* $G : \mathbb{G} \to \mathbb{Z}_q$ *and* $H : \{0,1\}^* \to \mathbb{Z}_q$ *are defined during setup. Then, DSA/ECDSA is defined in Figure 5.1. The transcript* trans *(after key generation and the signing of one message* m*) is defined as the tuple* $((pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}), (m, k, \sigma))$*. We note that, with the transcript, it is possible to verify that* $pk_{\mathsf{SIG}} = g^{sk_{\mathsf{SIG}}}$ *and recompute the signature* σ *generated by the signer.*

We also introduce a fake transcript $\mathsf{trans}_{\mathsf{fake}}$ into our syntax. Maintaining a real and fake transcript is necessary to ensure a signer can evade coercion, the main security goal of incoercible signatures.

Formally, we define the syntax for an incoercible signature scheme as follows.

Definition 60 (Incoercible signature scheme)**.** *An* incoercible signature scheme INC-SIG *is a tuple of polynomial time algorithms* (Setup, AKGen, SKGen, FakeTrans, Sign, FakeSign, Verify, Authenticate) *such that:*

Setup(1^λ). On input of security parameter 1^λ, algorithm Setup outputs public parameters pp. We assume that pp includes the message space M.

AKGen(pp) On input of public parameters pp, algorithm AKGen outputs an authenticator key pair (pk_A, sk_A) where pk_A is the authenticator's public key and sk_A is the authenticator's secret key.

SKGen(pp, pk_A) On input of public parameters pp and authenticator public key pk_A, algorithm SKGen outputs a signer key pair (pk_S, sk_S) where pk_S is the signer's public key and sk_S is the signer's secret key.

FakeTrans($pp, pk_A, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}}$) On input of public parameters pp, authenticator public key pk_A, a transcript trans and a fake transcript $\mathsf{trans}_{\mathsf{fake}}$, algorithm FakeTrans outputs an updated fake transcript $\mathsf{trans}_{\mathsf{fake}}$. We write that $\mathsf{trans}_{\mathsf{fake}} \leftarrow \mathsf{FakeTrans}(pp, pk_A, \mathsf{trans}, \perp)$ generates an initial fake transcript.

Sign(pp, sk_S, pk_A, m) On input of public parameters pp, signer secret key sk_S, authenticator public key pk_A and message m, algorithm Sign outputs a signature σ.

FakeSign($pp, sk_S, pk_A, m, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}}$) On input of public parameters pp, signer secret key sk_S, authenticator public key pk_A, message m, a transcript trans and a fake transcript $\mathsf{trans}_{\mathsf{fake}}$, algorithm FakeSign outputs a signature σ and updated fake transcript $\mathsf{trans}_{\mathsf{fake}}$.

Verify(pp, pk_S, m, σ) On input of public parameters pp, signer public key pk_S, message m and signature σ, algorithm Verify outputs 1 if σ is a valid signature, and 0 otherwise.

Authenticate($pp, pk_S, sk_A, m, \sigma$) On input of public parameters pp, signer public key pk_S, authenticator secret key sk_A, message m and signature σ, algorithm Authenticate outputs 1 if σ is an authentic signature, and 0 otherwise.

We require that an incoercible signature scheme satisfies *correctness*, which, as for standard signature schemes, requires that signatures output by algorithm Sign are verifiable. Additionally, an incoercible signature scheme must satisfy *completeness*, the property that signatures output by algorithm Sign are authentic.

Definition 61 (Correctness)**.** *An incoercible signature scheme* INC-SIG *satisfies* correctness *if, for any message* $m \in \mathsf{M}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} pp \leftarrow\!\$\ \mathsf{Setup}(1^\lambda); \\ (pk_A, sk_A) \leftarrow\!\$\ \mathsf{AKGen}(pp); \\ (pk_S, sk_S) \leftarrow\!\$\ \mathsf{SKGen}(pp, pk_A); \\ \sigma \leftarrow\!\$\ \mathsf{Sign}(pp, sk_S, pk_A, m) \end{array} : \mathsf{Verify}(pp, pk_S, m, \sigma) \coloneqq 1\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Definition 62 (Completeness)**.** *An incoercible signature scheme* INC-SIG *satisfies* completeness *if, for any message* $m \in \mathsf{M}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} pp \leftarrow\!\$\ \mathsf{Setup}(1^\lambda); \\ (pk_A, sk_A) \leftarrow\!\$\ \mathsf{AKGen}(pp); \\ (pk_S, sk_S) \leftarrow\!\$\ \mathsf{SKGen}(pp, pk_A); \\ \sigma \leftarrow\!\$\ \mathsf{Sign}(pp, sk_S, pk_A, m) \end{array} : \mathsf{Authenticate}(pp, pk_S, sk_A, m, \sigma) \coloneqq 1\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

5.3.1 Security Model

We present a security model for our syntax capturing a standard definition of existential unforgeability against a chosen message attack [71], adapted to the setting of incoercible signature schemes. Then, we define receipt-freeness and coercion-resistance, which are new security properties for the incoercible signature setting. Before we introduce our security properties for incoercible signatures, we discuss the corruption and coercion strategies an attacker can employ, and introduce the oracles that an adversary can query in our security experiments.

Corruption and coercion strategies. Both receipt-freeness and coercion-resistance consider signers that are either honest, corrupt, or coerced. Honest signers follow the protocol and do not interact with the attacker. Corrupt signers are controlled by the attacker; the attacker can act on their behalf and is assumed to have the secret signing key of corrupt signers. Coerced signers are signers that are provided with instructions by the attacker. Additionally, the attacker can request proof from the signer that they followed the attacker's instructions. Signers in the coerced category can follow one of two approaches. Coerced signers can cooperate with the attacker, generating authentic signatures for the attacker's messages and providing the attacker with their real transcript. Alternatively, coerced signers may appear to cooperate with the attacker when they are, in fact, deceiving. In this case, the coerced signer generates inauthentic signatures, thereby alerting the authenticator to the coercion, and presents a fake transcript to the attacker. Our security model captures an attacker that adaptively corrupt or coerce signers.

We assume that signers can fall into only one of these categories. That is, corrupt signers cannot be coerced and coerced signers cannot be corrupt. We believe that this is a reasonable assumption. Indeed, in [127], it is established that a notion of incoercibility that allows simultaneous corruption and coercion is, arguably, too strong. For example, a corrupt signer may reveal real secrets. If the corrupt signer is later coerced, they cannot deceive the attacker as they have already revealed their real secrets (i.e., their real transcript).

Oracles. Our security experiments rely on a number of oracles, which we define formally in Figure 5.2. In our oracles and the corresponding experiments, we write $\mathbf{pk_S}$ and $\mathbf{sk_S}$ to be the vector of signer public keys and secret keys respectively. Moreover, we write trans and $\mathsf{trans}_{\mathsf{fake}}$ to denote the vector of signer transcripts and fake transcripts. We provide details of our oracles here.

$\mathcal{O}\mathsf{reg}_{(pp,pk_A,\mathbf{pk_S},\mathbf{sk_S},\mathsf{L},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id)$ registers a signer. Oracle $\mathcal{O}\mathsf{reg}$ adds id to a list of registered signers L and runs algorithm SKGen to generate a signer key pair. Algorithm $\mathsf{FakeTrans}$ generates an initial fake transcript and the signer's public key is output.

$\mathcal{O}\mathsf{corrupt}_{(\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans})}(id)$ corrupts a signer. If the signer is registered but not coerced, oracle $\mathcal{O}\mathsf{corrupt}$ adds the signer's identity id to a list of corrupt signers corL and returns the signer's secret key and transcript.

$\mathcal{O}\mathsf{coerce}_{(\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id)$ outputs a real or fake transcript, depending on the value of a bit β chosen in the security experiment. Oracle $\mathcal{O}\mathsf{coerce}$ also adds the signer to a list of coerced signers crcL if the signer is registered but not corrupt.

$\mathcal{O}\mathsf{fakecoerce}_{(\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans}_{\mathsf{fake}})}(id)$ outputs a fake transcript. It also adds the signer to a list of coerced signers crcL if the signer is registered but not corrupt.

$\mathcal{O}\mathsf{sign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathcal{Q},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id,m)$ produces a signature for message m on behalf of a registered signer id. Oracle $\mathcal{O}\mathsf{sign}$ runs algorithm Sign to generate a signature for a signer. The oracle outputs the signatures and updates the fake transcript of the signer. Additionally, the tuple (id, m) is added to a set $\mathcal{Q}$. We define oracle $\mathcal{O}\mathsf{sign}$ to take as input *any* registered signer. Accordingly, our security experiments capture the fact that coerced signers, in addition to following instructions provided by the attacker, may also produce honestly generated signatures for messages of their choosing.

$\mathcal{O}\mathsf{sign}^*_{(pp,pk_A,sk_S,\mathcal{Q}^*)}(m)$ generates a signature on behalf of a signer with secret key sk_S. Oracle $\mathcal{O}\mathsf{sign}^*$ runs algorithm Sign to generate a signature for a message m, outputs the signature and adds message m to a set $\mathcal{Q}^*$.

$\mathcal{O}\mathsf{coercesign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id,m)$ coerces a signer and generates a real or a fake signature depending on the value of a bit β chosen in the security experiment. That is, oracle $\mathcal{O}\mathsf{coercesign}$ adds the signer to a list of coerced signers crcL if the signer is registered and not corrupt. Then, if $\beta = 0$ (resp., $\beta = 1$), algorithm Sign (resp., algorithm $\mathsf{FakeSign}$) generates a signature for message m and returns the signature.

$\mathcal{O}\mathsf{fakesign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id,m)$ generates a fake signature for a signer. If the signer is registered and not corrupt, the signer is added to a list of coerced signers crcL. The oracle runs algorithm $\mathsf{FakeSign}$ to generate a fake signature for message m and returns the signature.

With these oracles, we can now define unforgeability, receipt-freeness and coercion-resistance for incoercible signature schemes.

$\underline{\mathcal{O}\mathsf{reg}_{(pp,pk_A,\mathbf{pk_S},\mathbf{sk_S},\mathsf{L},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id)}$
if $id \in \mathsf{L}$ **return** $\perp$
$\mathsf{L} \leftarrow \mathsf{L} \cup \{id\}$
$(\mathbf{pk_S}[id], \mathbf{sk_S}[id]) \leftarrow\!\!\$\ \mathsf{SKGen}(pp, pk_A)$
$\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow\!\!\$\ \mathsf{FakeTrans}(pp, pk_A, \mathsf{trans}[id], \perp)$
return $\mathbf{pk_S}[id]$

$\underline{\mathcal{O}\mathsf{corrupt}_{(\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans})}(id)}$
if $id \notin \mathsf{L} \setminus \mathsf{crcL}$ **return** $\perp$
$\mathsf{corL} \leftarrow \mathsf{corL} \cup \{id\}$
return $\mathbf{sk_S}[id], \mathsf{trans}[id]$

$\underline{\mathcal{O}\mathsf{coerce}_{(\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id)}$
if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
$\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
if $\beta = 0$ **return** $\mathsf{trans}[id]$
if $\beta = 1$ **return** $\mathsf{trans}_{\mathsf{fake}}[id]$

$\underline{\mathcal{O}\mathsf{fakecoerce}_{(\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans}_{\mathsf{fake}})}(id)}$
if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
$\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
return $\mathsf{trans}_{\mathsf{fake}}[id]$

$\underline{\mathcal{O}\mathsf{sign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathcal{Q},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id,m)}$
if $id \notin \mathsf{L}$ **return** $\perp$
$\sigma \leftarrow\!\!\$\ \mathsf{Sign}(pp, \mathbf{sk_S}[id], pk_A, m)$
$\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow\!\!\$\ \mathsf{FakeTrans}(pp, pk_A, \mathsf{trans}[id], \mathsf{trans}_{\mathsf{fake}}[id])$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(id, m)\}$
return σ

$\underline{\mathcal{O}\mathsf{sign}^*_{(pp,pk_A,sk_S,\mathcal{Q}^*)}(m)}$
$\sigma \leftarrow\!\!\$\ \mathsf{Sign}(pp, sk_S, pk_A, m)$
$\mathcal{Q}^* \leftarrow \mathcal{Q}^* \cup \{m\}$
return σ

$\underline{\mathcal{O}\mathsf{coercesign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id,m)}$
if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
$\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
if $\beta = 0$ $\sigma \leftarrow\!\!\$\ \mathsf{Sign}(pp, \mathbf{sk_S}[id], pk_A, m)$
if $\beta = 1$ $(\sigma, \mathsf{trans}_{\mathsf{fake}}[id]) \leftarrow\!\!\$\ \mathsf{FakeSign}(pp, \mathbf{sk_S}[id], pk_A, m, \mathsf{trans}[id], \mathsf{trans}_{\mathsf{fake}}[id])$
return σ

$\underline{\mathcal{O}\mathsf{fakesign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id,m)}$
if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
$\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
$(\sigma, \mathsf{trans}_{\mathsf{fake}}[id]) \leftarrow\!\!\$\ \mathsf{FakeSign}(pp, \mathbf{sk_S}[id], pk_A, m, \mathsf{trans}[id], \mathsf{trans}_{\mathsf{fake}}[id])$
return σ

Figure 5.2: Oracles used in the experiments for unforgeability, receipt-freeness and coercion-resistance of an incoercible signature scheme $\mathsf{INC\text{-}SIG}$.

Unforgeability. We define existential unforgeability against a chosen message attack (EUF-CMA) for incoercible signatures. Like EUF-CMA for a standard signature scheme, we define an adversary $\mathcal{A}$ that has access to signing oracle $\mathcal{O}\mathsf{sign}^*$ and the public key of a signer. If the adversary cannot output a valid signature on behalf of the signer, where the signature is not the output of $\mathcal{O}\mathsf{sign}^*$, the incoercible signature scheme satisfies EUF-CMA security. Additionally, in our incoercible signature setting, we provide the adversary with the key pair of the authenticator, meaning that EUF-CMA security holds even if the authenticator is corrupt.

Definition 63 (EUF-CMA)**.** *An incoercible signature scheme* INC-SIG *satisfies* EUF-CMA *security if, for any PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l}\mathcal{Q}^*\leftarrow\emptyset;\\ pp\leftarrow\$\,\mathsf{Setup}(1^\lambda);\\ (pk_A,sk_A)\leftarrow\$\,\mathsf{AKGen}(pp);\\ (pk_S,sk_S)\leftarrow\$\,\mathsf{SKGen}(pp,pk_A);\\ (m^*,\sigma^*)\leftarrow\$\,\mathcal{A}^{\mathcal{O}\mathsf{sign}^*}(pp,pk_S,pk_A,sk_A)\end{array} : \mathsf{Verify}(pp,pk_S,m^*,\sigma^*) \coloneqq 1 \wedge m^*\notin\mathcal{Q}^*\right] \leq \mathsf{negl}(\lambda)$$

where oracle $\mathcal{O}\mathsf{sign}^*$ is the oracle defined in Figure 5.2.

Receipt-freeness. Receipt-freeness captures an attacker that coerces a signer to sign messages and, afterwards, demands evidence of the signer's cooperation. In our syntax, evidence refers to the transcript of the signer. We present a definition of receipt-freeness that captures two properties: indistinguishability and soundness.

First, we describe our indistinguishability experiment RF-IND that is formally defined in Figure 5.3. Intuitively, indistinguishability requires that an attacker, who can request transcripts from coerced signers, cannot distinguish a signer that cooperates with an attacker and presents a real transcript from a signer that evades coercion and presents a fake transcript. In the RF-IND experiment, an adversary can call oracles $\mathcal{O}\mathsf{reg}$, $\mathcal{O}\mathsf{corrupt}$ and $\mathcal{O}\mathsf{coerce}$ to register, corrupt and coerce signers respectively. Additionally, the adversary can request honestly generated signatures on behalf of any signer via oracle $\mathcal{O}\mathsf{sign}$, and requests signatures on behalf of coerced signers via queries to oracle $\mathcal{O}\mathsf{coercesign}$. Depending on a bit β, $\mathcal{O}\mathsf{coercesign}$ runs algorithm Sign or FakeSign to produce a signature. At the end of the experiment, the adversary can request to see the transcript of coerced signers by calling oracle $\mathcal{O}\mathsf{coerce}$, which returns real or fake transcripts depending on the bit β. An incoercible signature scheme satisfies the indistinguishability requirement if the adversary can guess β with probability only negligibly more than 1/2.

We assume that a receipt-freeness attacker will not generate signatures on behalf of a coerced signer. Therefore, we require only a basic soundness requirement such that, if a signature is fake, the authenticator will determine the signature to be inauthentic. Formally, we describe our soudness experiment sound in Figure 5.3, which captures the property that, if a signature is the output of algorithm FakeSign, then algorithm Authenticate will output 1 will negligible probability.

Definition 64 (Receipt-Freeness)**.** *An incoercible signature scheme* INC-SIG *satisfies* receipt-freeness *if the following conditions hold.*

- ***Indistinguishability****: for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* negl *such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{RF\text{-}IND},0}_{\mathsf{INC\text{-}SIG},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{RF\text{-}IND},1}_{\mathsf{INC\text{-}SIG},\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)\,.$$

- ***Soundness****: for any message* $m \in \mathsf{M}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{Exp}^{\mathsf{sound}}_{\mathsf{INC\text{-}SIG}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,.$$

where $\mathsf{Exp}^{\mathsf{RF\text{-}IND},\beta}_{\mathsf{INC\text{-}SIG},\mathcal{A}}(\lambda)$ *for* $\beta \in \{0,1\}$ *and* $\mathsf{Exp}^{\mathsf{sound}}_{\mathsf{INC\text{-}SIG}}(\lambda)$ *are the experiments defined in Figure 5.3.*

$\mathsf{Exp}^{\mathsf{RF\text{-}IND},\beta}_{\mathsf{INC\text{-}SIG},\mathcal{A}}(\lambda)$

$\mathbf{pk_S} \leftarrow (); \ \mathbf{sk_S} \leftarrow (); \ \mathsf{trans} \leftarrow (); \ \mathsf{trans}_{\mathsf{fake}} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset; \ \mathsf{crcL} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q} \leftarrow \emptyset$
$pp \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(pk_A, sk_A) \leftarrow\!\!\$\ \mathsf{AKGen}(pp)$
$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{coercesign}}(pp, pk_A)$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{coerce}}(st)$
return β'

$\mathsf{Exp}^{\mathsf{sound}}_{\mathsf{INC\text{-}SIG}}(\lambda)$

$pp \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(pk_A, sk_A) \leftarrow\!\!\$\ \mathsf{AKGen}(pp)$
$(pk_S, sk_S) \leftarrow\!\!\$\ \mathsf{SKGen}(pp, pk_A)$
$\mathsf{trans}_{\mathsf{fake}} \leftarrow\!\!\$\ \mathsf{FakeTrans}(pp, pk_A, \mathsf{trans}, \perp)$
$(\sigma, \mathsf{trans}_{\mathsf{fake}}) \leftarrow\!\!\$\ \mathsf{FakeSign}(pp, sk_S, pk_A, m, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}})$
$b \leftarrow\!\!\$\ \mathsf{Authenticate}(pp, pk_S, sk_A, m, \sigma)$
return b

$\mathsf{Exp}^{\mathsf{CR\text{-}IND},\beta}_{\mathsf{INC\text{-}SIG},\mathcal{A}}(\lambda)$

$\mathbf{pk_S} \leftarrow (); \ \mathbf{sk_S} \leftarrow (); \ \mathsf{trans} \leftarrow (); \ \mathsf{trans}_{\mathsf{fake}} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset; \ \mathsf{crcL} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q} \leftarrow \emptyset$
$pp \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(pk_A, sk_A) \leftarrow\!\!\$\ \mathsf{AKGen}(pp)$
$\beta' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{coerce},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{coercesign}}(pp, pk_A)$
return β'

$\mathsf{Exp}^{\mathsf{st\text{-}sound}}_{\mathsf{INC\text{-}SIG},\mathcal{A}}(\lambda)$

$\mathbf{pk_S} \leftarrow (); \ \mathbf{sk_S} \leftarrow (); \ \mathsf{trans} \leftarrow (); \ \mathsf{trans}_{\mathsf{fake}} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset; \ \mathsf{crcL} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q} \leftarrow \emptyset$
$pp \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(pk_A, sk_A) \leftarrow\!\!\$\ \mathsf{AKGen}(pp)$
$(id^*, m^*, \sigma^*) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{fakecoerce},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{fakesign}}(pp, pk_A)$
if $id^* \in \mathsf{crcL} \ \wedge \ (id^*, m^*) \notin \mathcal{Q} \ \wedge \ \mathsf{Authenticate}(pp, \mathbf{pk_S}[id^*], sk_A, m^*, \sigma^*) = 1$
  return 1
else return 0

Figure 5.3: The receipt-freeness and coercion-resistance experiments where the adversary has access to oracles defined in Figure 5.2.

Coercion-resistance. Coercion-resistance captures an attacker that can demand the transcripts of coerced signers at any point, and can attempt to produce signatures on their behalf. As for receipt-freeness, our coercion-resistance definition captures indistinguishability and strong soundness.

Our indistinguishability requirement is similar to indistinguishability as defined for receipt-freeness, with one key difference. In our coercion-resistance experiment, the adversary can query oracle $\mathcal{O}\mathsf{coerce}$ throughout the experiment. This models the fact that the attacker, rather than requesting transcripts of a coerced signer at the end of a protocol run, may demand the transcripts at any point. With respect to all other oracles queries, the coercion-resistance adversary is identical to the receipt-freeness adversary. We require that the adversary cannot determine whether the coerced signers are cooperating or evading coercion. Formally, as in our receipt-freeness indistinguishability experiment, this means that the adversary can guess the bit β with probability only negligibly more than $1/2$.

We require that an attacker, who can request fake transcripts and signatures of coerced, but uncooperative, signers, cannot output an authentic signature on behalf of a coerced signer. We formally define a strong soundness experiment in which an adversary can register and corrupt signers via oracles $\mathcal{O}\mathsf{reg}$ and $\mathcal{O}\mathsf{corrupt}$ respectively. Moreover, the adversary can request honestly generated signatures for any signer by calling oracle $\mathcal{O}\mathsf{sign}$. An incoercible signature scheme satisfies strong soundness if the adversary cannot output an authentic signature on behalf of a coerced signer, where the signature is not the output of the signing oracle.

Definition 65 (Coercion-Resistance)**.** *An incoercible signature scheme* INC-SIG *satisfies* coercion-resistance *if the following conditions hold.*

- ***Indistinguishability****: for any PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{CR\text{-}IND},0}_{\mathsf{INC\text{-}SIG},\mathcal{A}}(\lambda)=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{CR\text{-}IND},1}_{\mathsf{INC\text{-}SIG},\mathcal{A}}(\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)\,.$$

- ***Strong soundness****: for any PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{Exp}^{\mathsf{st\text{-}sound}}_{\mathsf{INC\text{-}SIG},\mathcal{A}}(\lambda)=1\right]\leq \mathsf{negl}(\lambda)\,.$$

where $\mathsf{Exp}^{\mathsf{CR\text{-}IND},\beta}_{\mathsf{INC\text{-}SIG},\mathcal{A}}(\lambda)$ *for* $\beta \in \{0,1\}$ *and* $\mathsf{Exp}^{\mathsf{st\text{-}sound}}_{\mathsf{INC\text{-}SIG},\mathcal{A}}(\lambda)$ *are the experiments defined in Figure 5.3.*

Relating receipt-freeness and coercion-resistance. A coercion-resistant attacker can generate signatures for coerced signers and can demand transcripts from signers throughout the protocol. A receipt-freeness attacker, on the other hand, only demands transcripts at the end of the protocol and does not attempt to generate signatures on behalf of coerced signers. Therefore, it appears that coercion-resistance is stronger than receipt-freeness. We now formally prove this. That is, we have Theorem 9, which proves that coercion-resistance implies receipt-freeness. Moreover, in Chapter 5.4, we introduce an incoercible signature scheme construction that satisfies receipt-freeness but not coercion-resistance. Therefore, we have that coercion-resistance is stronger than receipt-freeness.

Theorem 9. *Let* INC-SIG *be an incoercible signature scheme that satisfies coercion-resistance. Then* INC-SIG *also satisfies receipt-freeness.*

Proof. We first consider the indistinguishability requirement for receipt-freeness. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary that succeeds in the RF-IND experiment for incoercible signature scheme INC-SIG. We show that we can construct an adversary $\mathcal{B}$ that succeeds in the CR-IND experiment. We define adversary $\mathcal{B}$ in Figure 5.4. It is clear that the inputs to $\mathcal{A}$ produced by $\mathcal{B}$ are distributed identically to the RF-IND experiment. In particular, the public parameters pp and authenticator public key pk_A are identical. In this reduction, the authenticator's secret key is not known but not required. Moreover, $\mathcal{B}$ can respond to $\mathcal{A}$'s oracle queries by calling the identical oracles in the CR-IND experiment. Therefore, $\mathcal{B}$ perfectly simulates the RF-IND experiment to $\mathcal{A}$. We now show that $\mathcal{B}$ succeeds in the CR-IND experiment, that is, $\mathcal{B}$ returns $\beta' = \beta$. By assumption, adversary $\mathcal{A}$ succeeds in the RF-IND experiment and outputs $\beta' = \beta$. Hence, $\mathcal{B}$ returns $\beta' = \beta$ and the result holds by contradiction.

We now consider the soundness requirement. Assume that incoercible signature scheme INC-SIG does not satisfy the soundness property for receipt-freeness. That is, a signature output by algorithm FakeSign is authentic. If this is true then an adversary in the strong soundness experiment can output a tuple (id^*, m^*, σ^*) where $\sigma^* \leftarrow\!\!\$\, \mathcal{O}\mathsf{fakesign}(id^*, m^*)$ for a coerced signer id^*. As such, the adversary succeeds in the strong soundness experiment, i.e., algorithm Authenticate returns 1, id^* is coerced, and (id^*, m^*) is not queried to oracle

$\mathcal{B}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{coerce},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{coercesign}}(pp, pk_A)$

$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{coercesign}}(pp, pk_A)$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{coerce}}(st)$
return β'

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{coerce}/\mathcal{O}\mathsf{sign}/\mathcal{O}\mathsf{coercesign}$

Run the identical oracle in the CR-IND experiment.

Figure 5.4: Adversary $\mathcal{B}$ that breaks the CR-IND security of scheme INC-SIG in the proof of Theorem 9.

$\mathcal{O}$sign. Therefore, if INC-SIG does not satisfy the receipt-free soundness requirement, then an adversary in the strong soundness experiment can succeed and the result holds by contradiction. □

5.4 A Receipt-Free Incoercible Signature Scheme Construction

Having introduced new definitions, it is natural to consider how to achieve them. To this end, we present two incoercible signature scheme constructions. First, we present a receipt-freeness construction and prove that it is secure. Then, in Chapter 5.5, we present a coercion-resistant construction. We choose to present a construction that satisfies our weaker notion of security because, as we shall see, our receipt-free construction is more efficient that our coercion-resistant construction. Therefore, our receipt-free construction may be appropriate in application scenarios where a balance must be achieved between efficiency and security.

We present a receipt-free incoercible signature scheme construction (RF.SIG) in Figure 5.5. Our construction relies on a standard signature scheme SIG and a deniable encryption scheme DEN. We also make use of a collision resistant hash function $H : \{0,1\}^* \rightarrow \mathsf{M}$, where M is the message space of the signature scheme SIG.

We now describe the details of our construction. A trusted third party runs algorithm RF.Setup to generate public parameters for the scheme. This includes performing the setup for the SIG and DEN schemes. Then, the authenticator generates a key pair for the DEN scheme via algorithm RF.AKGen, and signers generate a key pair for the SIG scheme via

algorithm RF.SKGen. To sign a message, the signer runs algorithm RF.Sign or RF.FakeSign. Algorithm RF.Sign (resp., RF.FakeSign) deniably encrypts bit 1 (resp., 0) under the public key of the authenticator. Then, the signer generates a signature over the resulting ciphertext and the message. Additionally, algorithm RF.FakeSign updates the signer's fake transcript. By using deniable encryption, our construction ensures that, if a signer is coerced, they can produce fake randomness such that they appear to have encrypted a different bit (i.e., produce randomness such that the ciphertext appears to encrypt 1 rather than 0). Our construction includes a public verification algorithm RF.Verify. Additionally, the authenticator can run algorithm RF.Authenticate, which decrypts the ciphertext contained in a signature, thus revealing whether the signature is real or fake.

RF.Setup(1^λ)

$pp_{\mathsf{DEN}} \leftarrow\!\!\$\ \mathsf{DEN.Setup}(1^\lambda)$
$pp_{\mathsf{SIG}} \leftarrow\!\!\$\ \mathsf{SIG.Setup}(1^\lambda)$
$pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
return pp

RF.AKGen(pp)

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
$(pk_A, sk_A) \leftarrow\!\!\$\ \mathsf{DEN.KGen}(pp_{\mathsf{DEN}})$
return (pk_A, sk_A)

RF.SKGen(pp, pk_A)

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
$(pk_S, sk_S) \leftarrow\!\!\$\ \mathsf{SIG.KGen}(pp_{\mathsf{SIG}}; r_{\mathsf{SIG}})$
return (pk_S, sk_S)

RF.Verify(pp, pk_S, m, σ)

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$ $\wedge$ **parse** σ as (σ_1, σ_2)
if $\mathsf{SIG.Verify}(pp_{\mathsf{SIG}}, pk_S, H(m||\sigma_1), \sigma_2) = 0$ **return** 0
return 1

RF.Authenticate($pp, pk_S, sk_A, m, \sigma$)

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$ $\wedge$ **parse** σ as (σ_1, σ_2)
if $\mathsf{SIG.Verify}(pp_{\mathsf{SIG}}, pk_S, H(m||\sigma_1), \sigma_2) \coloneqq 0$ **return** 0
$t \leftarrow\!\!\$\ \mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, \sigma_1)$
if $t = 1$ **return** 1
else **return** 0

RF.Sign(pp, sk_S, pk_A, m)

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
$\sigma_1 \leftarrow\!\!\$\ \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, 1; r_{\sigma_1})$
$\sigma_2 \leftarrow\!\!\$\ \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_S, H(m||\sigma_1); r_{\sigma_2})$
return $\sigma = (\sigma_1, \sigma_2)$

RF.FakeSign($pp, sk_S, pk_A, m, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}}$)

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
$\sigma_1 \leftarrow\!\!\$\ \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, 0; r_{\sigma_1})$
$\sigma_2 \leftarrow\!\!\$\ \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_S, H(m||\sigma_1); r_{\sigma_2})$
$r'_{\sigma_1} \leftarrow\!\!\$\ \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, \sigma_1, 1)$
$\mathsf{trans}_{\mathsf{fake}} \leftarrow\!\!\$\ \mathsf{trans}_{\mathsf{fake}} \cup \{(m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
return $(\sigma = (\sigma_1, \sigma_2), \mathsf{trans}_{\mathsf{fake}})$

RF.FakeTrans($pp, pk_A, \mathsf{trans}, \bot$)

return $\mathsf{trans}_{\mathsf{fake}} \coloneqq \mathsf{trans} = \{(r_{\mathsf{SIG}}, pk_S, sk_S)\}$

RF.FakeTrans($pp, pk_A, \mathsf{trans}, \mathsf{trans}_{\mathsf{fake}}$)

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
parse trans as $\{(r_{\mathsf{SIG}}, pk_S, sk_S), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
For each new entry $(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)$ from RF.Sign
  $\mathsf{trans}_{\mathsf{fake}} \leftarrow\!\!\$\ \mathsf{trans}_{\mathsf{fake}} \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
return $\mathsf{trans}_{\mathsf{fake}}$

Figure 5.5: Our incoercible signature scheme construction RF.SIG.

5.4.1 Security of Our Construction

Our receipt-free construction RF.SIG satisfies correctness, completeness, unforgeability and receipt-freeness.

Theorem 10 (Correctness)**.** RF.SIG *is* correct *if signature scheme* SIG *is correct.*

Proof. Consider a signature $\sigma = (\sigma_1, \sigma_2)$ output by algorithm $\mathsf{RF.Sign}(pp, sk_S, pk_A, m)$ with respect to message $m \in \mathsf{M}$ for keys sk_S and pk_A generated by algorithms RF.SKGen and RF.AKGen respectively and parameters pp output by algorithm RF.Setup. Then, by definition of correctness, RF.SIG is correct if $\mathsf{RF.Verify}(pp, pk_S, m, \sigma)$ outputs 1 with overwhelming probability. Assume that RF.Verify does not return 1. Then it must be the case that $\mathsf{SIG.Verify}(pp_{\mathsf{SIG}}, pk_S, H(m||\sigma_1), \sigma_2) = 0$. By assumption, signature scheme SIG is correct. Therefore, algorithm SIG.Verify returns 1 with overwhelming probability where pp_{SIG} and pk_S are generated according to the construction description and σ_2 is the output of SIG.Sign for message $H(m||\sigma_1)$. Then, by contradiction, RF.SIG is correct. □

Theorem 11 (Completeness)**.** RF.SIG *is* complete *if deniable encryption scheme* DEN *is correct and* RF.SIG *is correct.*

Proof. Consider a signature $\sigma = (\sigma_1, \sigma_2)$ output by algorithm $\mathsf{RF.Sign}(pp, sk_S, pk_A, m)$ with respect to message $m \in \mathsf{M}$ for keys sk_S and pk_A generated by algorithms RF.SKGen and RF.AKGen respectively and parameters pp output by algorithm RF.Setup. By definition of completeness, RF.SIG is complete if $\mathsf{RF.Authenticate}(pp, pk_S, sk_A, m, (\sigma_1, \sigma_2))$ outputs 1 with overwhelming probability. Assume that RF.Authenticate does not return 1. Then, either $\mathsf{SIG.Verify}(pp_{\mathsf{SIG}}, pk_S, H(m||\sigma_1), \sigma_2) = 0$ or $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, \sigma_1) \neq 1$. By correctness of RF.SIG, SIG.Verify returns 1 with overwhelming probability. Moreover, by correctness of the deniable encryption scheme, DEN.Dec returns 1 where $\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, 1)$. Then, by contradiction, RF.SIG is complete. □

Theorem 12 (Unforgeability)**.** RF.SIG *satisfies* EUF-CMA *security if digital signature scheme* SIG *satisfies* EUF-CMA *security and the hash function* H *is collision resistant.*

Proof. Let $\mathcal{A}$ be an adversary that succeeds in the EUF-CMA experiment (Definition 63) for the RF.SIG construction. We show that we can construct an adversary $\mathcal{B}$ that succeeds in the EUF-CMA experiment (Definition 18) for signature scheme SIG. We present $\mathcal{B}$ in Figure 5.6. It is clear that inputs to $\mathcal{A}$ are distributed identically to the EUF-CMA experiment of Definition 63 because keys pk_A and pk_S are generated identically. Moreover, oracle $\mathcal{O}\mathsf{sign}^*$ is distributed identically because $\mathcal{O}\mathsf{sign}(H(m||\sigma_1))$ returns $\mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_S, H(m||\sigma_1))$. We now show that $\mathcal{B}$ is successful in the EUF-CMA experiment of Definition 18. That is, we

show that $(H(m^*||\sigma_1^*), \sigma_2^*)$ is a valid message/signature pair that has not been input to $\mathcal{O}\mathsf{sign}$. If $\mathcal{A}$ succeeds in the EUF-CMA experiment then $(m^*, (\sigma_1^*, \sigma_2^*))$ is a valid message/signature pair, i.e., $\mathsf{RF.Verify}(pp, pk_S, m^*, (\sigma_1^*, \sigma_2^*)) = 1$. Then, by definition of algorithm RF.Verify, $\mathsf{SIG.Verify}(pp_{\mathsf{SIG}}, pk_S, H(m^*||\sigma_1^*), \sigma_2^*) = 1$. Moreover, by assumption, $m^* \notin \mathcal{Q}^*$ and, therefore, $H(m^*||\sigma_1^*)$ is not input to oracle $\mathcal{O}\mathsf{sign}$ unless a hash collision occurs, which occurs with negligible probability if hash function H is collision resistant. Therefore, $\mathcal{B}$ succeeds in the EUF-CMA experiment and, by contradiction, the result holds. □

$\mathcal{B}^{\mathcal{O}\mathsf{sign}}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}})$	$\mathcal{O}\mathsf{sign}^*_{(pp,pk_A,sk_S,\mathcal{Q}^*)}(m)$
$\mathcal{Q}^* \leftarrow \emptyset$	$\sigma_1 \leftarrow\!\!\$\ \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, 1)$
$pp_{\mathsf{DEN}} \leftarrow\!\!\$\ \mathsf{DEN.Setup}(1^\lambda)$	$\sigma_2 \leftarrow\!\!\$\ \mathcal{O}\mathsf{sign}(H(m\|\|\sigma_1))$
$pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$	$\mathcal{Q}^* \leftarrow \mathcal{Q}^* \cup \{m\}$
$(pk_A, sk_A) \leftarrow\!\!\$\ \mathsf{RF.AKGen}(pp)$	**return** $\sigma = (\sigma_1, \sigma_2)$
$pk_S \leftarrow pk_{\mathsf{SIG}}$	
$(m^*, (\sigma_1^*, \sigma_2^*)) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{sign}^*}(pp, pk_S, pk_A, sk_A)$	
return $(H(m^*\|\|\sigma_1^*), \sigma_2^*)$	

Figure 5.6: Adversary $\mathcal{B}$ that breaks the EUF-CMA security of signature scheme SIG in the proof of Theorem 12.

Theorem 13 (Receipt-freeness)**.** RF.SIG *satisfies* receipt-freeness *if deniable encryption scheme* DEN *is correct,* IND-CPA *secure and* IND-EXP *secure, and signature scheme* SIG *is correct.*

We prove this result in Lemmata 5 and 6, which demonstrate that the indistinguishability and soundness conditions of receipt-freeness are satisfied.

Lemma 5 (Indistinguishability)**.** RF.SIG *satisfies* RF-IND *security if deniable encryption scheme* DEN *satisfies* IND-CPA *security and* IND-EXP *security.*

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary in the RF-IND experiment against scheme RF.SIG that makes at most k queries to oracle $\mathcal{O}\mathsf{coercesign}$. We proceed through a series of game hops that we show are indistinguishable to the adversary. In the final game, the view of the adversary will be identical for $\beta = 0$ and $\beta = 1$. We define Game 0 as the RF-IND experiment with β chosen randomly. Let S_i be the event that adversary $\mathcal{A}$ correctly guesses β in Game i.

Game 1 is identical to Game 0 except for a change to the oracle $\mathcal{O}\mathsf{coercesign}$. When $\beta = 1$ we encrypt '1' rather than '0'. Both the real and fake transcript are updated as normal. We

$\underline{\mathcal{O}\mathsf{coercesign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans_{fake}})}(id,m)}$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
$\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
if $\beta = 0$: $\sigma \leftarrow\!\!\$\ \mathsf{Sign}(pp, \mathbf{sk_S}[id], pk_A, m)$
if $\beta = 1$
  parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
  $\sigma_1 \leftarrow\!\!\$\ \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, 1; r_{\sigma_1})$
  $\sigma_2 \leftarrow\!\!\$\ \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_S, H(m||\sigma_1); r_{\sigma_2})$
  $r'_{\sigma_1} \leftarrow\!\!\$\ \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, \sigma_1, 1)$
  $\mathsf{trans_{fake}} \leftarrow\!\!\$\ \mathsf{trans_{fake}} \cup \{(m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  $\sigma \leftarrow (\sigma_1, \sigma_2)$
return σ

Figure 5.7: Oracle $\mathcal{O}\mathsf{coercesign}$ used in Game 1 in the proof of Lemma 5.

define oracle $\mathcal{O}\mathsf{coercesign}$ used in Game 1 in Figure 5.7. We show that Game 0 and Game 1 are indistinguishable if the deniable encryption scheme DEN is IND-CPA secure. We give a distinguishing algorithm $\mathcal{D}_1$ in Figure 5.8. $\mathcal{D}_1$ aims to guess a bit β^* in the IND-CPA game for multiple encryptions. Recall that, in this experiment, instead of receiving one ciphertext, the adversary can make several queries of the form $\mathcal{O}\mathsf{enc}(m_0, m_1)$ and will receive $\mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, m_{\beta^*})$ for each query.

$\underline{\mathcal{D}_1^{\mathcal{O}\mathsf{enc}}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}})}$

$\beta \leftarrow\!\!\$\ \{0,1\}$; $\mathbf{pk_S} \leftarrow ()$; $\mathbf{sk_S} \leftarrow ()$; $\mathsf{trans} \leftarrow ()$; $\mathsf{trans_{fake}} \leftarrow ()$; $\mathsf{L} \leftarrow \emptyset$; $\mathsf{crcL} \leftarrow \emptyset$; $\mathsf{corL} \leftarrow \emptyset$; $\mathcal{Q} \leftarrow \emptyset$
$pp_{\mathsf{SIG}} \leftarrow\!\!\$\ \mathsf{SIG.Setup}(1^\lambda)$; $pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$; $pk_A \leftarrow pk_{\mathsf{DEN}}$
$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{coercesign}}(pp, pk_A)$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{coerce}}(st)$
if $\beta' = \beta$ **return** 1

$\underline{\mathcal{O}\mathsf{coercesign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans_{fake}})}(id,m)}$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
$\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
if $\beta = 1$: **parse** pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
  $\sigma_1 \leftarrow\!\!\$\ \mathcal{O}\mathsf{enc}(0,1)$; $\sigma_2 \leftarrow\!\!\$\ \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, \mathbf{sk_S}[id]; H(m||\sigma_1); r_{\sigma_2})$
  $\sigma \leftarrow (\sigma_1, \sigma_2)$
  $r'_{\sigma_1} \leftarrow\!\!\$\ \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, \sigma_1, 1)$; $\mathsf{trans_{fake}} \leftarrow \mathsf{trans_{fake}} \cup \{(m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  return σ
else as normal

$\underline{\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{coerce}/\mathcal{O}\mathsf{sign}}$

As in Game 0.

Figure 5.8: Distinguisher $\mathcal{D}_1$ that distinguishes Game 0 and Game 1 in the proof of Lemma 5.

We show that, when $\beta^* = 0$, inputs to $\mathcal{A}$ are identical to Game 0 and, when $\beta^* = 1$, inputs

to $\mathcal{A}$ are identical to Game 1. Note that pp and pk_A input to $\mathcal{A}$ are honestly generated and are identical in Games 0 and 1. Furthermore, sk_A is not known but is never used. Games 0 and 1 are identical with the exception of oracle $\mathcal{O}\mathsf{coercesign}$ when $\beta = 1$. If $\beta^* = 0$, $\mathcal{D}_1$ is input an encryption of '0' as in Game 0. If $\beta^* = 1$, $\mathcal{D}_1$ is input an encryption of '1' as in Game 1. Therefore,

$$|\Pr[S_0] - \Pr[S_1]| \leq \mathsf{negl}_{\mathsf{IND\text{-}CPA\text{-}mult}}$$

where $\mathsf{negl}_{\mathsf{IND\text{-}CPA\text{-}mult}}$ is the advantage in breaking the IND-CPA for multiple encryptions property of deniable encryption scheme DEN. By a standard argument

$$|\Pr[S_0] - \Pr[S_1]| \leq k \cdot \mathsf{negl}_{\mathsf{IND\text{-}CPA}}$$

where $\mathsf{negl}_{\mathsf{IND\text{-}CPA}}$ is the advantage in breaking the IND-CPA property of DEN and at most k queries are made to oracle $\mathcal{O}\mathsf{coercesign}$.

Game 2 is identical to Game 1 except for a further change to oracle $\mathcal{O}\mathsf{coercesign}$. When $\beta = 1$, we include the real randomness used to encrypt in the fake transcript, rather than obtaining fake randomness from algorithm DEN.Exp. We describe oracle $\mathcal{O}\mathsf{coercesign}$ for Game 2 in Figure 5.9. We show that Game 1 and Game 2 are indistinguishable assuming the deniable encryption scheme DEN is IND-EXP secure. We give a distinguishing algorithm $\mathcal{D}_2$ in Figure 5.10. $\mathcal{D}_2$ aims to guess a bit β^* in the IND-EXP game which is adjusted similarly to the IND-CPA experiment for multiple encryptions used in Game 1. That is, the adversary can submit multiple queries to an oracle $\mathcal{O}\mathsf{exp}$ that, on input a message, returns a ciphertext and randomness. If $\beta^* = 0$, oracle $\mathcal{O}\mathsf{exp}$ returns the randomness used to encrypt and, if $\beta^* = 1$, it returns randomness obtained via the algorithm DEN.Exp.

We show that, when $\beta^* = 0$, inputs to $\mathcal{A}$ are identical to Game 2 and, when $\beta^* = 1$ inputs to $\mathcal{A}$ are identical to Game 1. As before, pp and pk_A input to $\mathcal{A}$ are identical in Games 1 and 2 and sk_A is not known but is never used. Games 1 and 2 are identical with the exception of oracle $\mathcal{O}\mathsf{coercesign}$ when $\beta = 1$. Regardless of β^*, oracle $\mathcal{O}\mathsf{exp}$ always returns an encryption of '1' under pk_A. If $\beta^* = 0$, $\mathcal{D}_2$ is input the real randomness used to encrypt as used in Game 2. If $\beta^* = 1$, $\mathcal{D}_2$ is input the randomness output by $\mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, \sigma_1, 1)$

$\underline{\mathcal{O}\mathsf{coercesign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans_{fake}})}(id,m)}$
if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
$\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
if $\beta = 0$: $\sigma \leftarrow\!\!\$\ \mathsf{Sign}(pp, \mathbf{sk_S}[id], pk_A, m)$
if $\beta = 1$
  parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
  $\sigma_1 \leftarrow\!\!\$\ \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, 1; r_{\sigma_1})$
  $\sigma_2 \leftarrow\!\!\$\ \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_S, H(m||\sigma_1); r_{\sigma_2})$
  $\mathsf{trans_{fake}} \leftarrow\!\!\$\ \mathsf{trans_{fake}} \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  $\sigma = (\sigma_1, \sigma_2)$
return σ

Figure 5.9: Oracle $\mathcal{O}\mathsf{coercesign}$ used in Game 2 in the proof of Lemma 5.

which is identical to the input to $\mathcal{A}$ in Game 1. Therefore,

$$|\Pr[S_1] - \Pr[S_2]| \leq \mathsf{negl}_{\mathsf{IND\text{-}EXP\text{-}mult}}$$

where $\mathsf{negl}_{\mathsf{IND\text{-}EXP\text{-}mult}}$ is the advantage in breaking the IND-EXP for multiple messages property of deniable encryption scheme DEN. By a standard argument,

$$|\Pr[S_1] - \Pr[S_2]| \leq k \cdot \mathsf{negl}_{\mathsf{IND\text{-}EXP}}$$

where $\mathsf{negl}_{\mathsf{IND\text{-}EXP}}$ is the advantage in breaking the IND-EXP property of DEN and at most k queries are made to oracle $\mathcal{O}\mathsf{coercesign}$.

In Game 2, the inputs to $\mathcal{A}$ are identical for $\beta = 0$ and $\beta = 1$. In fact, for a signer $id \in \mathsf{corL}$ or $id \in \mathsf{L} \setminus (\mathsf{crcL} \cup \mathsf{corL})$ (i.e., the signer is corrupt or honest respectively), the inputs to $\mathcal{A}$ are independent of β. If $id \in \mathsf{crcL}$ (i.e., the signer is coerced), when $\beta = 1$, all signatures generated by algorithms RF.Sign and RF.FakeSign contain an encryption of '1'. Moreover, the fake transcript is updated with the real encryption randomness. As such, the outputs of oracles $\mathcal{O}\mathsf{coerce}$ and $\mathcal{O}\mathsf{coercesign}$ are identical for $\beta = 0$ and $\beta = 1$. Therefore, $\Pr[S_2] = 1/2$. We now have that

$$|\Pr[S_0] - 1/2| \leq k \cdot (\mathsf{negl}_{\mathsf{IND\text{-}CPA}} + \mathsf{negl}_{\mathsf{IND\text{-}EXP}})$$

and conclude that the advantage of the adversary in Game 0 is negligible as required. □

Lemma 6 (Soundness)**.** RF.SIG *satisfies* soundness *if deniable encryption scheme* DEN *is*

$\mathcal{D}_2^{\mathcal{O}\mathsf{exp}}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}})$

$\beta \leftarrow_{\$} \{0,1\}$; $\mathbf{pk_S} \leftarrow ()$; $\mathbf{sk_S} \leftarrow ()$; $\mathsf{trans} \leftarrow ()$; $\mathsf{trans_{fake}} \leftarrow ()$; $\mathsf{L} \leftarrow \emptyset$; $\mathsf{crcL} \leftarrow \emptyset$; $\mathsf{corL} \leftarrow \emptyset$; $\mathcal{Q} \leftarrow \emptyset$
$pp_{\mathsf{SIG}} \leftarrow_{\$} \mathsf{SIG.Setup}(1^\lambda)$; $pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$; $pk_A \leftarrow pk_{\mathsf{DEN}}$
$st \leftarrow_{\$} \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{coercesign}}(pp, pk_A)$
$\beta' \leftarrow_{\$} \mathcal{A}_2^{\mathcal{O}\mathsf{coerce}}(st)$
if $\beta' = \beta$ **return** 1

$\mathcal{O}\mathsf{coercesign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans_{fake}})}(id, m)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
$\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
if $\beta = 1$ **parse** pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
  $(\sigma_1, r_{\sigma_1}) \leftarrow_{\$} \mathcal{O}\mathsf{exp}(1)$; $\sigma_2 \leftarrow_{\$} \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, \mathbf{sk_S}[id], H(m||\sigma_1); r_{\sigma_2})$
  $\sigma \leftarrow (\sigma_1, \sigma_2)$
  $\mathsf{trans_{fake}} \leftarrow \mathsf{trans_{fake}} \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  return σ
else as normal

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{coerce}/\mathcal{O}\mathsf{sign}$

As in Game 0.

Figure 5.10: Distinguisher $\mathcal{D}_2$ that distinguishes Game 1 and Game 2 in the proof of Lemma 5.

correct and signature scheme SIG *is correct.*

Proof. Consider a signature $\sigma = (\sigma_1, \sigma_2)$ and fake transcript $\mathsf{trans_{fake}} = \{(r_{\mathsf{SIG}}, pk_S, sk_S), (m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$ output by algorithm RF.FakeSign with respect to message m for keys sk_S and pk_A generated by algorithms RF.SKGen and RF.AKGen respectively and parameters pp output by algorithm RF.Setup. Then, by definition of soundness, algorithm $\mathsf{RF.Authenticate}(pp, pk_S, sk_A, m, \sigma)$ returns 0 with overwhelming probability. Assume that RF.Authenticate does not return 0. Then, it must be the case that $\mathsf{SIG.Verify}(pp_{\mathsf{SIG}}, pk_S, H(m \parallel \sigma_1), \sigma_2) = 1$ and $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, \sigma_1) = 1$. By correctness of the signature scheme, SIG.Verify returns 1 with overwhelming probability. However, by correctness of the deniable encryption scheme, algorithm DEN.Dec returns 0 where $\sigma_1 \leftarrow \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, 0)$. Then, by contradiction, RF.SIG is sound. □

5.5 A Coercion-Resistant Incoercible Signature Scheme Construction

Our receipt-free construction RF.SIG does not satisfy coercion-resistance. In fact, the fake transcript $\{(r_{\mathsf{SIG}}, pk_S, sk_S), (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \cdots\}$ of a coerced signer contains the real secret key of the signer, which a coercive attacker can use to create valid and authentic forgeries on behalf of the signer by running algorithm RF.Sign. As such, RF.SIG cannot satisfy the strong soundness property that is necessary for coercion-resistance. Indeed, an adversary in the strong soundness experiment against scheme RF.SIG can corrupt a signer and obtain their transcript via oracle $\mathcal{O}$coerce. The initial real and fake transcripts are identical and contain the secret key of the signer. Therefore, the adversary can run algorithm RF.Sign to generate a signature and can output this signature in the strong soundness experiment. The conditions of the experiment are satisfied and, if RF.SIG is complete, algorithm Authenticate outputs 1, and the adversary succeeds in the experiment.

Therefore, we now introduce a construction that we call CR.SIG that satisfies coercion-resistance. We present CR.SIG in Figure 5.11 that relies on a standard signature scheme SIG and a deniable encryption scheme DEN. Additionally, our construction uses two hash functions $H_1 : \{0,1\}^* \to \{0,1\}^*$, $H_2 : \{0,1\}^* \to \mathsf{M}$, where M is the message space of the signature scheme SIG. The first is assumed to be a random oracle, whereas the second is required to be collision resistant.

We briefly note the key differences between our constructions. During key generation, in addition to generating a key pair for signature scheme SIG, a signer generates a random string s and deniably encrypts this under the authenticator's public key. The signer's secret key now consists of a secret key for SIG and a string s. The corresponding public key consists of the public key for SIG and the ciphertext that encrypts s. To sign a message, the signer generates an encryption of s, rather than a single bit. Signing then proceeds as in our receipt-free construction. Signers can signal coercion by encrypting a different random string s'. The authenticator can detect coercion by decrypting the ciphertexts contained in the public key and the signature, and comparing the two. The signer creates a fake transcript that contains s', rather than s. In this way, by security of the deniable encryption scheme, the attacker cannot distinguish a real and a fake transcript. Moreover, the coercive attacker cannot forge an authentic signature without knowledge of s, and our

construction achieves strong soundness.

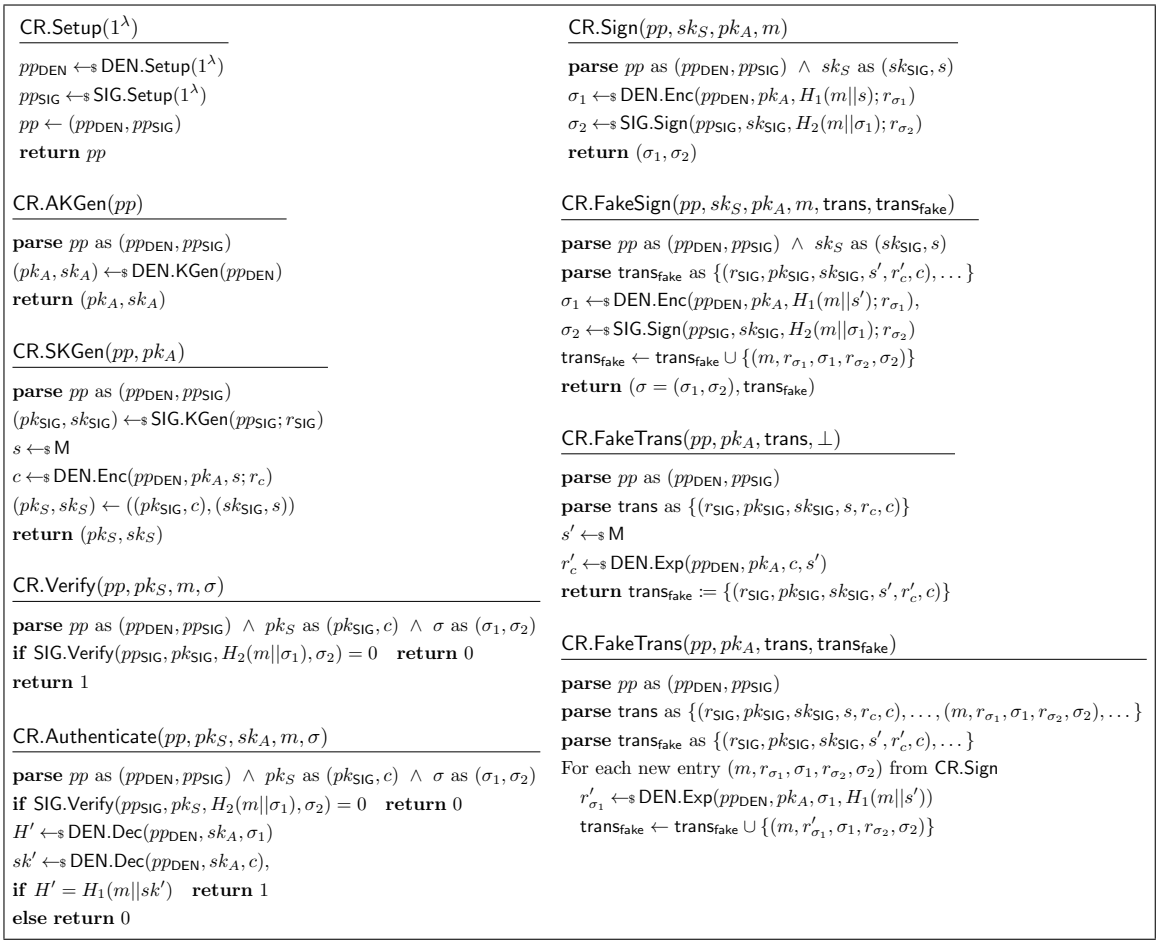

Figure 5.11: Our coercion-resistant construction CR.SIG.

5.5.1 Security of Our Construction

Our coercion-resistance construction satisfies correctness, completeness, unforgeability and coercion-resistance.

Theorem 14 (Correctness)**.** CR.SIG *is* correct *if signature scheme* SIG *is correct.*

Proof. Consider a signature $\sigma = (\sigma_1, \sigma_2)$ output by algorithm $\mathsf{CR.Sign}(pp, sk_S, pk_A, m)$ with respect to message m for keys sk_S and pk_A generated by algorithms CR.SKGen and CR.AKGen respectively and parameters pp output by algorithm CR.Setup. Then, by definition of correctness, CR.SIG is correct if $\mathsf{CR.Verify}(pp, pk_S, m, \sigma)$ outputs 1 with overwhelming probability. Assume that CR.Verify does not return 1. Then it must be the case that $\mathsf{SIG.Verify}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, H_2(m||\sigma_1), \sigma_2) = 0$. By assumption, signature scheme SIG is correct. Therefore, algorithm SIG.Verify returns 1 with overwhelming probability

where pp_{SIG} and pk_{SIG} are generated according to the construction description and σ_2 is the output of SIG.Sign for message $H_2(m||\sigma_1)$. Then, by contradiction, CR.SIG is correct. □

Theorem 15 (Completeness)**.** CR.SIG *is* complete *if deniable encryption scheme* DEN *is correct and* CR.SIG *is correct.*

Proof. Consider a signature $\sigma = (\sigma_1, \sigma_2)$ output by algorithm $\mathsf{CR.Sign}(pp, sk_S, pk_A, m)$ with respect to message m for keys sk_S and pk_A generated by algorithms CR.SKGen and CR.AKGen respectively and parameters pp output by algorithm CR.Setup. By definition of completeness, CR.SIG is complete if $\mathsf{CR.Authenticate}(pp, pk_S, sk_A, m, (\sigma_1, \sigma_2))$ outputs 1 with overwhelming probability. Assume that CR.Authenticate does not return 1. Then, either $\mathsf{SIG.Verify}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, H_2(m||\sigma_1), \sigma_2) = 0$ or $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, \sigma_1) \neq H_1(m||$ $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, c))$. By correctness of CR.SIG, SIG.Verify returns 1 with overwhelming probability. Moreover, by correctness of the deniable encryption scheme, $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}$, $sk_A, c)$ returns s, and $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, \sigma_1)$ returns $H_1(m||s)$. Then, by contradiction, CR.SIG is complete. □

Theorem 16 (Unforgeability)**.** CR.SIG *satisfies* EUF-CMA *security if digital signature scheme* SIG *satisfies* EUF-CMA *security and the hash function* H_2 *is collision resistant.*

Proof. Let $\mathcal{A}$ be an adversary that succeeds in the EUF-CMA experiment (Definition 63) for the CR.SIG construction. We show that we can construct an adversary $\mathcal{B}$ that succeeds in the EUF-CMA experiment (Definition 18) for signature scheme SIG. We present $\mathcal{B}$ in Figure 5.12. It is clear that inputs to $\mathcal{A}$ are distributed identically to the EUF-CMA experiment of Definition 63 because keys pk_A and pk_S are generated identically. Moreover, oracle $\mathcal{O}\mathsf{sign}^*$ is distributed identically because $\mathcal{O}\mathsf{sign}(H_2(m||\sigma_1))$ returns $\mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, H_2(m||\sigma_1))$. We now show that $\mathcal{B}$ is successful in the EUF-CMA experiment of Definition 18. That is, we show that $(H_2(m^*||\sigma_1^*), \sigma_2^*)$ is a valid message/signature pair that has not been input to $\mathcal{O}\mathsf{sign}$. If $\mathcal{A}$ succeeds in the EUF-CMA experiment then $(m^*, (\sigma_1^*, \sigma_2^*))$ is a valid message/signature pair, i.e., $\mathsf{CR.Verify}(pp, pk_S, m^*, (\sigma_1^*, \sigma_2^*)) = 1$. Then, by definition of algorithm CR.Verify, $\mathsf{SIG.Verify}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, H_2(m^*||\sigma_1^*), \sigma_2^*) = 1$. Moreover, by assumption, $m^* \notin \mathcal{Q}^*$ and, therefore, $H_2(m^*||\sigma_1^*)$ is not input to $\mathcal{O}\mathsf{sign}$ unless a hash collision occurs, which occurs with negligible probability if hash function H_2 is collision resistant. Therefore, $\mathcal{B}$ succeeds in the EUF-CMA experiment and, by contradiction, the result holds. □

$\mathcal{B}^{\mathcal{O}\mathsf{sign}}(pp_{\mathsf{SIG}}, pk_{\mathsf{SIG}})$

$\mathcal{Q}^* \leftarrow \emptyset$
$pp_{\mathsf{DEN}} \leftarrow\!\!\$\ \mathsf{DEN.Setup}(1^\lambda)$
$pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
$(pk_A, sk_A) \leftarrow\!\!\$\ \mathsf{CR.AKGen}(pp)$
$s \leftarrow\!\!\$\ \mathsf{M};\ c \leftarrow\!\!\$\ \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, s);\ pk_S \leftarrow (pk_{\mathsf{SIG}}, c)$
$(m^*, (\sigma_1^*, \sigma_2^*)) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{sign}^*}(pp, pk_S, pk_A, sk_A)$
return $(H_2(m^*||\sigma_1^*), \sigma_2^*)$

$\mathcal{O}\mathsf{sign}^*_{(pp,pk_A,sk_S,\mathcal{Q}^*)}(m)$

$\sigma_1 \leftarrow\!\!\$\ \mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_A, H_1(m||s))$
$\sigma_2 \leftarrow\!\!\$\ \mathcal{O}\mathsf{sign}(H_2(m||\sigma_1))$
$\mathcal{Q}^* \leftarrow \mathcal{Q}^* \cup \{m\}$
return (σ_1, σ_2)

Figure 5.12: Adversary $\mathcal{B}$ that breaks the EUF-CMA security of signature scheme SIG in the proof of Theorem 16.

Theorem 17 (Coercion-resistance)**.** CR.SIG *satisfies* coercion-resistance *if deniable encryption scheme is* IND-CPA *secure and* IND-EXP *secure, and hash function* H_1 *is a random oracle.*

We prove this result in Lemmata 7 and 8 , which demonstrate that the indistinguishability and strong soundness requirements of coercion-resistance are satisfied.

Lemma 7 (Indistinguishability)**.** CR.SIG *satisfies* CR-IND *security if the deniable encryption scheme satisfies* IND-CPA *security and* IND-EXP *security.*

Proof. Let $\mathcal{A}$ be an adversary in the CR-IND experiment against scheme CR.SIG that makes at most k_1 queries to oracle $\mathcal{O}\mathsf{reg}$, at most k_2 queries to oracle $\mathcal{O}\mathsf{sign}$ per signer, and at most k_3 queries to oracle $\mathcal{O}\mathsf{coercesign}$ per signer. We proceed through a series of game hops that we show are indistinguishable to the adversary. In the final game, the view of the adversary will be identical for $\beta = 0$ and $\beta = 1$. We define Game 0 as the CR-IND experiment with β chosen randomly. Let S_i be the event that adversary $\mathcal{A}$ correctly guesses b in Game i.

Game 1 is identical to Game 0, except that the fake transcript is only generated when a signer is first added to the coerced list crcL. Moreover, oracle $\mathcal{O}\mathsf{sign}$ only updates the fake transcript if the signer is coerced. We give the adjusted oracles for this game in Figure 5.13. As the fake transcript is only used after a signer has been coerced this is only a superficial change and does not affect the distribution of inputs to the adversary. Therefore,

$$\Big|\Pr[S_0] - \Pr[S_1]\Big| = 0.$$

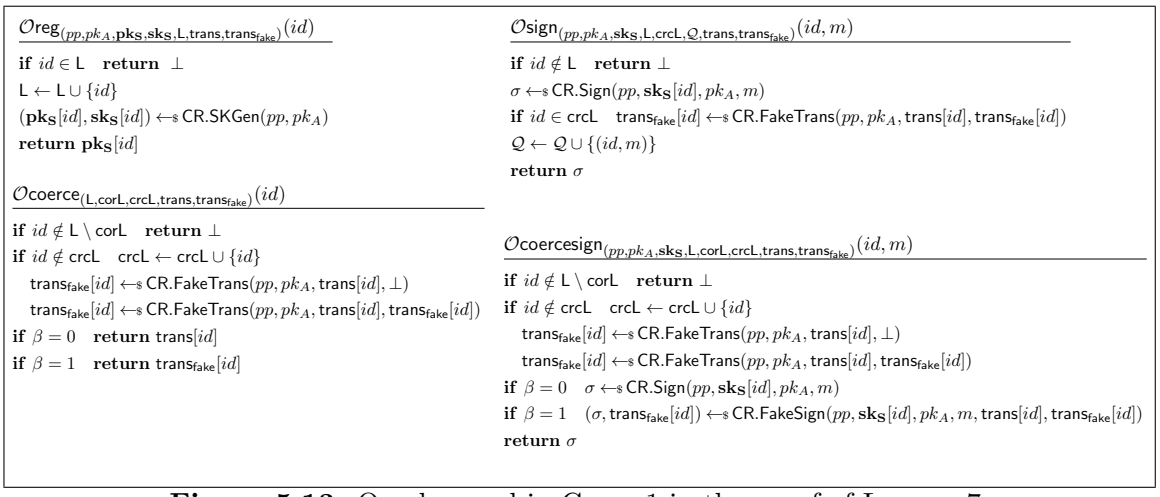

Figure 5.13: Oracles used in Game 1 in the proof of Lemma 7.

Game 2 is identical to Game 1, except for a change to oracles $\mathcal{O}\mathsf{coerce}$ and $\mathcal{O}\mathsf{coercesign}$. In the fake transcript, we include s rather than s'. We define oracles $\mathcal{O}\mathsf{coerce}$ and $\mathcal{O}\mathsf{coercesign}$ used in Game 2 in Figure 5.14. This game hop will make use of a hybrid argument. We split the reduction into k_1 steps and define Game 1.i to be identical to Game 2 for the first i signers submitted to the $\mathcal{O}\mathsf{reg}$ oracle, and, otherwise, to be identical to Game 1. Clearly, Game 1.k_1 is identical to Game 2 and Game 1.0 is identical to Game 1.

We show that Game 1.i and Game 1.$(i+1)$ are indistinguishable if deniable encryption scheme DEN is $\mathsf{IND\text{-}CPA}$ secure. We give a distinguishing algorithm $\mathcal{D}_1$ in Figure 5.15. $\mathcal{D}_1$ aims to guess a bit β^* in the $\mathsf{IND\text{-}CPA}$ game for multiple encryptions. In this game, instead of receiving one ciphertext, the adversary can make several queries of the form (m_0, m_1) and will receive $\mathsf{DEN.Enc}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}}, m_b)$ for each query to an oracle $\mathcal{O}\mathsf{enc}$.

We show that, when $\beta^* = 0$, inputs to $\mathcal{A}$ are identical to Game 1.i and, when $\beta^* = 1$, inputs to $\mathcal{A}$ are identical to Game 1.$(i+1)$. Note that pp and pk_A input to $\mathcal{A}$ are honestly generated and are identical in Games 1.i and 1.$(i+1)$. Furthermore, sk_A is not known but is never used. Oracles $\mathcal{O}\mathsf{reg}$ and $\mathcal{O}\mathsf{sign}$ are distributed identically in Games 1.i and 1.$(i+1)$, because when $\beta = 1$, $d = 1$ and $id = \hat{id}$, $s^*_{\beta^*}$ is used as the secret key and is encrypted in both the public key and in honest signatures. As the real transcript is generated identically, oracle $\mathcal{O}\mathsf{corrupt}$ is distributed identically in Games 1.i and 1.$(i+1)$, provided it does not abort, i.e., $id \neq \hat{id}$ or $d = 0$. Therefore, games 1.i and 1.$(i+1)$ are identical with the exception of oracles $\mathcal{O}\mathsf{coerce}$ and $\mathcal{O}\mathsf{coercesign}$ when $id = \hat{id}$ where $\hat{id}$ is the $i+1$th signer created. When $\beta = 0$, the fake transcript is never used and the real transcript is generated normally. Therefore the outputs of the $\mathcal{O}\mathsf{coerce}$ and $\mathcal{O}\mathsf{coercesign}$ oracles are distributed

$\mathcal{O}\mathsf{coerce}_{(pp,pk_A,\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
parse $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id \notin \mathsf{crcL}$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
  $r'_c \leftarrow\!\!\$\ \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, c, s)$
  $\mathsf{trans}_{\mathsf{fake}}[id] \coloneqq \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r'_c, c)\}$
  For each entry $(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)$ from $\mathsf{CR.Sign}$
    $r'_{\sigma_1} \leftarrow\!\!\$\ \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, \sigma_1, H_1(m||s))$
    $\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{trans}_{\mathsf{fake}}[id] \cup \{(m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
if $\beta = 0$ **return** $\mathsf{trans}[id]$
if $\beta = 1$ **return** $\mathsf{trans}_{\mathsf{fake}}[id]$

$\mathcal{O}\mathsf{coercesign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id, m)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
parse $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id \notin \mathsf{crcL}$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
  $r'_c \leftarrow\!\!\$\ \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, c, s)$
  $\mathsf{trans}_{\mathsf{fake}}[id] \coloneqq \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r'_c, c)\}$
  For each entry $(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)$ from $\mathsf{CR.Sign}$
    $r'_{\sigma_1} \leftarrow\!\!\$\ \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, \sigma_1, H_1(m||s))$
    $\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{trans}_{\mathsf{fake}}[id] \cup \{(m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
if $\beta = 0$ $\sigma \leftarrow\!\!\$\ \mathsf{CR.Sign}(pp, \mathbf{sk_S}[id], pk_A, m)$
if $\beta = 1$ $(\sigma, \mathsf{trans}_{\mathsf{fake}}[id]) \leftarrow\!\!\$\ \mathsf{CR.FakeSign}(pp, \mathbf{sk_S}[id], pk_A, m, \mathsf{trans}[id], \mathsf{trans}_{\mathsf{fake}}[id])$
return σ

Figure 5.14: Oracles $\mathcal{O}\mathsf{coerce}$ and $\mathcal{O}\mathsf{coercesign}$ used in Game 2 in the proof of Lemma 7.

identically in Games 1.i and 1.$(i+1)$. When $d = 0$, both oracles abort. When $d = 1$ and $\beta = 1$, if $\beta^* = 0$ the oracle is distributed identically to Game 1 because s_0^* is used as the secret key and s_1^* is used in the fake transcript. When $d = 1$ and $\beta = 1$, if $\beta^* = 1$ the oracle is distributed identically to Game 2 because s_1^* is used as both the secret key and in the fake transcript. The probability that distinguisher $\mathcal{D}_1$ aborts is at most 1/2. Therefore

$$\left|\Pr[S_{1.i}] - \Pr\left[S_{1.(i+1)}\right]\right| \leq 2 \cdot \mathsf{negl}_{\mathsf{IND\text{-}CPA\text{-}mult}}$$

where $\mathsf{negl}_{\mathsf{IND\text{-}CPA\text{-}mult}}$ is the advantage in breaking the IND-CPA for multiple encryptions property of deniable encryption scheme DEN. By a standard argument,

$$\left|\Pr[S_{1.i}] - \Pr\left[S_{1.(i+1)}\right]\right| \leq 2 \cdot (1 + k_2) \cdot \mathsf{negl}_{\mathsf{IND\text{-}CPA}}$$

$\mathcal{D}_1^{\mathcal{O}\mathsf{enc}}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}})$

$\beta \leftarrow\!\!\$\ \{0,1\};\ q \leftarrow 0;\ d \leftarrow\!\!\$\ \{0,1\};\ \mathbf{pk_S} \leftarrow ();\ \mathbf{sk_S} \leftarrow ();\ \mathsf{trans} \leftarrow ();\ \mathsf{trans_{fake}} \leftarrow ();\ \mathsf{L} \leftarrow \emptyset;\ \mathsf{crcL} \leftarrow \emptyset;\ \mathsf{corL} \leftarrow \emptyset;\ \mathcal{Q} \leftarrow \emptyset$
$pp_{\mathsf{SIG}} \leftarrow\!\!\$\ \mathsf{SIG.Setup}(1^\lambda);\ pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}});\ pk_A \leftarrow pk_{\mathsf{DEN}}$
$\beta' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{coerce},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{coercesign}}(pp, pk_A)$
if $\beta' = \beta$ **return** 1

$\mathcal{O}\mathsf{reg}_{(pp,pk_A,\mathbf{pk_S},\mathbf{sk_S},\mathsf{L},\mathsf{trans},\mathsf{trans_{fake}})}(id)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
if $id \in \mathsf{L}$ **return** $\perp$; $\mathsf{L} \leftarrow \mathsf{L} \cup \{id\}$; $q \leftarrow q+1$
if $q = i+1$ $\hat{id} \leftarrow id$
if $\beta = 1 \ \wedge\ d = 1 \ \wedge\ id = \hat{id}$
  $(pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}) \leftarrow\!\!\$\ \mathsf{SIG.KGen}(pp_{\mathsf{SIG}}; r_{\mathsf{SIG}})$
  $s_0^* \leftarrow\!\!\$\ \mathsf{M};\ s_1^* \leftarrow\!\!\$\ \mathsf{M};\ c \leftarrow \mathcal{O}\mathsf{enc}(s_0^*, s_1^*)$
  $(\mathbf{pk_S}[id], \mathbf{sk_S}[id]) \leftarrow ((pk_{\mathsf{SIG}}, c), (sk_{\mathsf{SIG}}, \perp))$
  return $\mathbf{pk_S}[id]$
else as in Game 1

$\mathcal{O}\mathsf{sign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathcal{Q},\mathsf{trans},\mathsf{trans_{fake}})}(id, m)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$ $\wedge$ **parse** $\mathbf{sk_S}[id]$ as (sk_{SIG}, s)
if $\beta = 1 \ \wedge\ d = 1 \ \wedge\ id = \hat{id}$
  $\sigma_1 \leftarrow\!\!\$\ \mathcal{O}\mathsf{enc}(H_1(m||s_0^*), H_1(m||s_1^*))$
  $\sigma_2 \leftarrow\!\!\$\ \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, H_2(m||\sigma_1))$
  $\sigma \leftarrow (\sigma_1, \sigma_2)$
  if $id \in \mathsf{crcL}$
    $\mathsf{trans_{fake}}[id] \leftarrow\!\!\$\ \mathsf{CR.FakeTrans}(pp, pk_A, \mathsf{trans}[id], \mathsf{trans_{fake}}[id])$
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(id, m)\}$
  return σ
else as in Game 1

$\mathcal{O}\mathsf{corrupt}_{(\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans})}(id)$

if $id \notin \mathsf{L} \setminus \mathsf{crcL}$ **return** $\perp$
$\mathsf{corL} \leftarrow \mathsf{corL} \cup \{id\}$
if $d = 1 \ \wedge\ id = \hat{id}$: $\mathcal{D}_1$ aborts
else return $\mathbf{sk_S}[id], \mathsf{trans}[id]$

$\mathcal{O}\mathsf{coerce}_{(pp,pk_A,\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans_{fake}})}(id)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id = \hat{id}$
  if $\beta = 0$: $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$; **return** $\mathsf{trans}[id]$
  if $d = 0$: $\mathcal{D}_1$ aborts
  parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
  parse $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, \perp, \perp, c), \ldots, (m, \perp, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
  if $id \notin \mathsf{crcL}$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
    $r'_c \leftarrow\!\!\$\ \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, c, s_1^*)$
    $\mathsf{trans_{fake}}[id] \coloneqq \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s_1^*, r'_c, c)$
    For each entry $(m, \perp, \sigma_1, r_{\sigma_2}, \sigma_2)$ from $\mathsf{CR.Sign}$
      $r'_{\sigma_1} \leftarrow\!\!\$\ \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, \sigma_1, H_1(m||s_1^*))$
      $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{trans_{fake}}[id] \cup \{(m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  return $\mathsf{trans_{fake}}[id]$
else As in Game 1.i

$\mathcal{O}\mathsf{coercesign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{crcL},\mathsf{corL},\mathsf{trans},\mathsf{trans_{fake}})}(id, m)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id = \hat{id}$
  if $\beta = 0$: $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$; **return** $\sigma \leftarrow\!\!\$\ \mathsf{CR.Sign}(pp, \mathbf{sk_S}[id], pk_A, m)$
  if $d = 0$: $\mathcal{D}_1$ aborts
  parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$ $\wedge$ **parse** $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, \perp, \perp, c), \ldots, (m, \perp, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
  if $id \notin \mathsf{crcL}$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
    $r'_c \leftarrow\!\!\$\ \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, c, s_1^*)$
    $\mathsf{trans_{fake}}[id] \coloneqq \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s_1^*, r'_c, c)$
    For each entry $(m, \perp, \sigma_1, r_{\sigma_2}, \sigma_2)$ from $\mathsf{CR.Sign}$
      $r'_{\sigma_1} \leftarrow\!\!\$\ \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, \sigma_1, H_1(m||s_1^*))$
      $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{trans_{fake}}[id] \cup \{(m, r'_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  $(\sigma, \mathsf{trans_{fake}}[id]) \leftarrow\!\!\$\ \mathsf{CR.FakeSign}(pp, \mathbf{sk_S}[id], pk_A, m, \mathsf{trans}[id], \mathsf{trans_{fake}}[id])$
  return σ
else As in Game 1.i

Figure 5.15: Distinguisher $\mathcal{D}_1$ that distinguishes between Game 1.i and Game 1.(i+1) in the proof of Lemma 7.

where $\mathsf{negl}_{\mathsf{IND\text{-}CPA}}$ is the advantage in breaking the IND-CPA property of DEN and k_2 is the maximum number of queries to oracle $\mathcal{O}\mathsf{sign}$ per signer. Therefore

$$\left|\Pr[S_1] - \Pr[S_2]\right| \leq 2 \cdot k_1 \cdot (1 + k_2) \cdot \mathsf{negl}_{\mathsf{IND\text{-}CPA}}$$

where k_1 is the maximum number of queries to oracle $\mathcal{O}\mathsf{reg}$.

Game 3 is identical to Game 2, except for another change to the $\mathcal{O}\mathsf{sign}$, $\mathcal{O}\mathsf{coerce}$ and $\mathcal{O}\mathsf{coercesign}$ oracles. We include the real randomness used to encrypt in the fake transcript, instead of obtaining fake randomness from algorithm DEN.Exp. Additionally, in $\mathcal{O}\mathsf{coercesign}$ we use the honest signing algorithm CR.Sign regardless of β and update the fake transcript as in the real transcript. We give the $\mathcal{O}\mathsf{sign}$, $\mathcal{O}\mathsf{coerce}$ and $\mathcal{O}\mathsf{coercesign}$ oracles used in Game 3 in Figure 5.16. This game hop will again make use of a hybrid argument. We split the reduction into k_1 steps. We define Game 2.i to be identical to Game 3 for the first i signers submitted to oracle $\mathcal{O}\mathsf{reg}$ and otherwise be identical to Game 2. Clearly, Game 2.k_1 is identical to Game 3 and Game 2.0 is identical to Game 2.

$\underline{\mathcal{O}\mathsf{sign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{crcL},\mathcal{Q},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id, m)}$

if $id \notin \mathsf{L}$ **return** $\perp$
$\sigma \leftarrow\!\!\$\ \mathsf{CR.Sign}(pp, \mathbf{sk_S}[id], pk_A, m)$
if $id \in \mathsf{crcL}$: $\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{trans}[id]$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(id, m)\}$
return σ

$\underline{\mathcal{O}\mathsf{coerce}_{(\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id)}$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id \notin \mathsf{crcL}$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
$\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{trans}[id]$
if $\beta = 0$ **return** $\mathsf{trans}[id]$
if $\beta = 1$ **return** $\mathsf{trans}_{\mathsf{fake}}[id]$

$\underline{\mathcal{O}\mathsf{coercesign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans}_{\mathsf{fake}})}(id, m)}$

parse $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
$\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
$\sigma \leftarrow\!\!\$\ \mathsf{CR.Sign}(pp, \mathbf{sk_S}[id], pk_A, m)$
$\mathsf{trans}_{\mathsf{fake}}[id] \leftarrow \mathsf{trans}[id]$
return σ

Figure 5.16: Oracles $\mathcal{O}\mathsf{sign}$, $\mathcal{O}\mathsf{coerce}$ and $\mathcal{O}\mathsf{coercesign}$ used in Game 3 in the proof of Lemma 7.

We show that Game 2.i and Game 2.$(i + 1)$ are indistinguishable if deniable encryption scheme DEN is IND-EXP secure. We give a distinguishing algorithm $\mathcal{D}_2$ in Figure 5.17. $\mathcal{D}_2$ aims to guess a bit β^* in the IND-EXP game, which is adjusted similarly to the IND-CPA

experiment for multiple encryptions. That is, the adversary can submit multiple queries to an oracle $\mathcal{O}\mathsf{exp}$ that, on input a message, returns a ciphertext and randomness. If $\beta^* = 0$, oracle $\mathcal{O}\mathsf{exp}$ returns the randomness used to encrypt and, if $\beta^* = 1$, it returns randomness obtained via the algorithm $\mathsf{DEN.Exp}$.

$\mathcal{D}_2^{\mathcal{O}\mathsf{exp}}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}})$

$\beta \leftarrow\!\!\$\ \{0,1\};\ q \leftarrow 0;\ d \leftarrow\!\!\$\ \{0,1\};\ \mathbf{pk_S} \leftarrow ();\ \mathbf{sk_S} \leftarrow ();\ \mathsf{trans} \leftarrow ();\ \mathsf{trans_{fake}} \leftarrow ();\ \mathsf{L} \leftarrow \emptyset;\ \mathsf{crcL} \leftarrow \emptyset;\ \mathsf{corL} \leftarrow \emptyset;\ \mathcal{Q} \leftarrow \emptyset$
$pp_{\mathsf{SIG}} \leftarrow\!\!\$\ \mathsf{SIG.Setup}(1^\lambda);\ pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}});\ pk_A \leftarrow pk_{\mathsf{DEN}}$
$\beta' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{coerce},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{coercesign}}(pp, pk_A)$
if $\beta' = \beta$ **return** 1

$\mathcal{O}\mathsf{reg}_{(pp,pk_A,\mathbf{pk_S},\mathbf{sk_S},\mathsf{L},\mathsf{trans},\mathsf{trans_{fake}})}(id)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
if $id \in \mathsf{L}$ **return** $\perp$; $\mathsf{L} \leftarrow \mathsf{L} \cup \{id\}$; $q \leftarrow q+1$
if $q = i+1$: $\hat{id} \leftarrow id$
if $\beta = 1 \ \wedge\ d = 1 \ \wedge\ id = \hat{id}$
  $(pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}) \leftarrow\!\!\$\ \mathsf{SIG.KGen}(pp_{\mathsf{SIG}}; r_{\mathsf{SIG}})$
  $s \leftarrow\!\!\$\ \mathsf{M};\ (c, r_c) \leftarrow\!\!\$\ \mathcal{O}\mathsf{exp}(s)$
  $\mathsf{trans}[id] \coloneqq \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c)$
  $(\mathbf{pk_S}[id], \mathbf{sk_S}[id]) \leftarrow ((pk_{\mathsf{SIG}}, c), (sk_{\mathsf{SIG}}, s))$
  return $\mathbf{pk_S}[id]$
else as in Game 2

$\mathcal{O}\mathsf{sign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathcal{Q},\mathsf{trans},\mathsf{trans_{fake}})}(id, m)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$ $\wedge$ **parse** $\mathbf{sk_S}[id]$ as (sk_{SIG}, s)
if $\beta = 1 \ \wedge\ d = 1 \ \wedge\ id = \hat{id}$
  $(\sigma_1, r_{\sigma_1}) \leftarrow\!\!\$\ \mathcal{O}\mathsf{exp}(H_1(m||s))$
  $\sigma_2 \leftarrow\!\!\$\ \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, H_2(m||\sigma_1); r_{\sigma_2})$
  $\sigma\ (\sigma_1, \sigma_2)$
  $\mathsf{trans}[id] \leftarrow \mathsf{trans}[id] \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  if $id \in \mathsf{crcL}$
    $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{trans_{fake}}[id] \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(id, m)\}$
  return σ
else as in Game 2

$\mathcal{O}\mathsf{corrupt}_{(\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathbf{sk_S},\mathsf{trans})}(id)$

if $id \notin \mathsf{L} \setminus \mathsf{crcL}$ **return** $\perp$
$\mathsf{corL} \leftarrow \mathsf{corL} \cup \{id\}$
if $d = 1 \ \wedge\ id = \hat{id}$: $\mathcal{D}_2$ aborts
else return $\mathbf{sk_S}[id], \mathsf{trans}[id]$

$\mathcal{O}\mathsf{coerce}_{(\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans_{fake}})}(id)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id = \hat{id}$
  if $\beta = 0$: $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$; **return** $\mathsf{trans}[id]$
  if $d = 0$: $\mathcal{D}_2$ aborts
  parse $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
  if $id \notin \mathsf{crcL}$: $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
    $\mathsf{trans_{fake}}[id] \coloneqq \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
  return $\mathsf{trans_{fake}}[id]$
else As in Game 1.i

$\mathcal{O}\mathsf{coercesign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans},\mathsf{trans_{fake}})}(id, m)$

if $id \notin \mathsf{L} \setminus \mathsf{corL}$ **return** $\perp$
if $id = \hat{id}$
  if $\beta = 0$: $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$; **return** $\sigma \leftarrow\!\!\$\ \mathsf{CR.Sign}(pp, \mathbf{sk_S}[id], pk_A, m)$
  if $d = 0$ $\mathcal{D}_1$ aborts
  parse $\mathsf{trans}[id]$ as $\{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
  if $id \notin \mathsf{crcL}$ $\mathsf{crcL} \leftarrow \mathsf{crcL} \cup \{id\}$
    $\mathsf{trans_{fake}}[id] \coloneqq \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s, r_c, c), \ldots, (m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2), \ldots\}$
  $(\sigma_1, r_{\sigma_1}) \leftarrow\!\!\$\ \mathcal{O}\mathsf{exp}(H_1(m||s))$
  $\sigma_2 \leftarrow\!\!\$\ \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, H_2(m||\sigma_1); r_{\sigma_2});\ \sigma \leftarrow (\sigma_1, \sigma_2)$
  $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{trans_{fake}}[id] \cup \{(m, r_{\sigma_1}, \sigma_1, r_{\sigma_2}, \sigma_2)\}$
  return σ
else As in Game 1.i

Figure 5.17: Distinguisher $\mathcal{D}_2$ that distinguishes between Game 2.i and Game 2.$(i+1)$ in the proof of Lemma 7.

We show that, when $\beta^* = 0$, inputs to $\mathcal{A}$ are identical to Game 2.$(i+1)$ and, when $\beta^* = 1$, inputs to $\mathcal{A}$ are identical to Game 2.i. Note that pp and pk_A input to $\mathcal{A}$ are honestly generated and are identical in Games 2.i and 2.$(i+1)$. Furthermore, sk_A is not known but is never used. The output of oracles $\mathcal{O}\mathsf{reg}$ and $\mathcal{O}\mathsf{sign}$ are distributed identically to both Games 2.i and 2.$(i+1)$, because when $\beta = 1, d = 1$ and $id = \hat{id}$, the secret key s is always encrypted in both the public key and in signatures. The $\mathcal{O}\mathsf{corrupt}$ oracle is distributed identically to both Games 2.i and 2.$(i+1)$, provided it doesn't abort. This is because the real transcript is generated identically provided $id \neq \hat{id}$ or $d = 0$.

Games 2.i and 2.$(i+1)$ are identical with the exception of oracles $\mathcal{O}\mathsf{sign}$, $\mathcal{O}\mathsf{coerce}$ and $\mathcal{O}\mathsf{coercesign}$ when $id = \hat{id}$. When $\beta = 0$, the fake transcript is never used and the real transcript is generated normally. Therefore the outputs of the $\mathcal{O}\mathsf{coerce}$ and $\mathcal{O}\mathsf{coercesign}$ oracles are distributed identically in Games 2.i and 2.$(i+1)$. When $d = 0$, the $\mathcal{O}\mathsf{coerce}$ and $\mathcal{O}\mathsf{coercesign}$ oracles both abort. When $d = 1$ and $\beta = 1$, if $\beta^* = 0$ the fake transcript is distributed identically to Game 3 because $\mathcal{D}_2$ is input the real randomness used in the encryption. When $d = 1$ and $\beta = 1$, if $\beta^* = 1$ the fake transcript is distributed identically to Game 2 because $\mathcal{D}_2$ is input the randomness output by $\mathsf{DEN.Exp}$. In $\mathcal{O}\mathsf{coercesign}$, as the real key is included in the fake transcript in both Games 2 and Games 3, the signature output is distributed identically in Game 2 and Game 3. The probability that distinguisher $\mathcal{D}_2$ aborts is at most 1/2. Therefore

$$\left| \Pr[S_{2.i}] - \Pr\left[S_{2.(i+1)}\right] \right| \leq 2 \cdot \mathsf{negl}_{\mathsf{IND\text{-}EXP\text{-}mult}}$$

where $\mathsf{negl}_{\mathsf{IND\text{-}EXP\text{-}mult}}$ is the advantage in breaking the $\mathsf{IND\text{-}EXP}$ for multiple messages property of deniable encryption scheme DEN. By a standard argument,

$$\left| \Pr[S_{2.i}] - \Pr\left[S_{2.(i+1)}\right] \right| \leq 2 \cdot (1 + k_2 + k_3) \cdot \mathsf{negl}_{\mathsf{IND\text{-}EXP}}$$

where $\mathsf{negl}_{\mathsf{IND\text{-}EXP}}$ is the advantage in breaking the $\mathsf{IND\text{-}EXP}$ property of DEN and k_2 and k_3 are the maximum number of queries to oracles $\mathcal{O}\mathsf{sign}$ and $\mathcal{O}\mathsf{coercesign}$ per signer. Therefore

$$\left| \Pr[S_2] - \Pr[S_3] \right| \leq 2 \cdot k_1 \cdot (1 + k_2 + k_3) \cdot \mathsf{negl}(\lambda)_{\mathsf{IND\text{-}EXP}}$$

where k_1 is the number of queries to oracles $\mathcal{O}\mathsf{reg}$.

In Game 3, the fake transcript is identical to the real transcript. Also signatures output by $\mathcal{O}\mathsf{coercesign}$ are always generated by algorithm Sign. Therefore, inputs to the adversary $\mathcal{A}$ are independent of β and so $Pr[S_3] = 1/2$.

We now have that

$$\left|\Pr[S_0] - 1/2\right| \leq 2 \cdot k_1 \cdot (1 + k_2) \cdot (\mathsf{negl}_{\mathsf{IND\text{-}EXP}} + \mathsf{negl}_{\mathsf{IND\text{-}CPA}}) + 2 \cdot k_1 \cdot k_3 \cdot \mathsf{negl}_{\mathsf{IND\text{-}EXP}}$$

and conclude that the advantage of the adversary in Game 0 is negligible as required. □

Lemma 8 (Strong Soundness)**.** CR.SIG *satisfies* strong soundness *if* H_1 *is a random oracle and deniable encryption scheme* DEN *is* IND-CPA *secure.*

Proof. Let $\mathcal{A}$ be an adversary that succeeds in the strong soundness against scheme CR.SIG that makes k_1 and k_2 queries to oracles $\mathcal{O}\mathsf{sign}$ and $\mathcal{O}\mathsf{reg}$ respectively. Then we can construct an adversary $\mathcal{B}$ that succeeds in the IND-CPA for multiple encryptions experiment for deniable encryption scheme DEN. We describe $\mathcal{B}$ in Figure 5.18.

First note that inputs to $\mathcal{A}$ are distributed identically to the strong soundness experiment for the CR.SIG construction. That is, pp and pk_A are honestly and identically generated. Furthermore, sk_A is not known but is never used. For all signers other than $\hat{id}$, oracle $\mathcal{O}\mathsf{reg}$ is identically distributed. For signer $\hat{id}$, sk_{SIG} and pk_{SIG} are generated identically. Ciphertext c is an encryption of $s^*_{\beta^*}$ under pk_A, where β^* is the bit chosen in the IND-CPA experiment. As both s^*_1, s^*_0 are chosen identically to s, ciphertext c is perfectly simulated. The second part of the secret key is not known, but is not used. The fake transcript is generated as normal using the fake key and the deniable encryption explanation algorithm. Therefore, $\mathcal{B}$ simulates oracle $\mathcal{O}\mathsf{reg}$ to $\mathcal{A}$. For all signers other than $\hat{id}$, oracle $\mathcal{O}\mathsf{sign}$ is also identically distributed. For signer $\hat{id}$, σ_1 is an encryption of $H_1(m||s^*_{\beta'})$ under pk_A, where β^* is the bit chosen in the IND-CPA experiment. This is consistent with oracle $\mathcal{O}\mathsf{reg}$ because $s^*_{\beta^*}$, and σ_2 and the fake transcript are generated identically and $\mathcal{B}$ simulated oracle $\mathcal{O}\mathsf{sign}$ to $\mathcal{A}$. If oracle $\mathcal{O}\mathsf{corrupt}$ is input $\hat{id}$, then $\mathcal{B}$ will abort. Oracles $\mathcal{O}\mathsf{fakecoerce}$ and $\mathcal{O}\mathsf{fakesign}$ are trivially simulated. Finally, the hash oracle H_1 is distributed identically to a random oracle.

For $\mathcal{A}$ to be successful, algorithm CR.Authenticate must return 1. That is, $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, \sigma^*_1) = H_1(m^*||\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, c))$. Therefore, $(m^*||\mathsf{DEN.Dec}$

$\mathcal{B}^{\mathcal{O}\mathsf{enc}}(pp_{\mathsf{DEN}}, pk_{\mathsf{DEN}})$

$q \leftarrow 0;\ q^* \leftarrow_\$ [k_2];\ \mathbf{pk_S} \leftarrow ();\ \mathbf{sk_S} \leftarrow ();\ \mathsf{trans} \leftarrow ();\ \mathsf{trans_{fake}} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset;\ \mathsf{crcL} \leftarrow \emptyset;\ \mathsf{corL} \leftarrow \emptyset;\ \mathcal{Q} \leftarrow \emptyset;\ \mathsf{hashL} \leftarrow \emptyset$
$pp_{\mathsf{SIG}} \leftarrow \mathsf{SIG.Setup}(1^\lambda);\ pp \leftarrow (pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}});\ pk_A \leftarrow pk_{\mathsf{DEN}}$
$(id^*, m^*, (\sigma_1^*, \sigma_2^*)) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{fakecoerce},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{fakesign},H_1}(pp, pk_A)$
if $id^* \neq \hat{id}$: $\mathcal{B}$ aborts
if $(m^*||s_1^*, \cdot) \in \mathsf{hashL}$ **return** 1 **else return** 0

$\mathcal{O}\mathsf{reg}_{(pp,pk_A,\mathbf{pk_S},\mathbf{sk_S},\mathsf{L},\mathsf{trans},\mathsf{trans_{fake}})}(id)$

parse pp as $(pp_{\mathsf{DEN}}, pp_{\mathsf{SIG}})$
if $id \in \mathsf{L}$ **return** $\perp$; $\mathsf{L} \leftarrow \mathsf{L} \cup \{id\};\ q \leftarrow q+1$
if $q = q^*$: $\hat{id} \leftarrow id$
  $(pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}) \leftarrow \mathsf{SIG.KGen}(pp_{\mathsf{SIG}}; r_{\mathsf{SIG}});\ s_0^* \leftarrow_\$ \mathsf{M};\ s_1^* \leftarrow_\$ \mathsf{M};\ s' \leftarrow_\$ \mathsf{M}$
  $c \leftarrow \mathcal{O}\mathsf{enc}(s_0^*, s_1^*);\ (\mathbf{pk_S}[id], \mathbf{sk_S}[id]) \leftarrow ((pk_{\mathsf{SIG}}, c), (sk_{\mathsf{SIG}}, \perp))$
  $r_c' \leftarrow \mathsf{DEN.Exp}(pp_{\mathsf{DEN}}, pk_A, c, s');\ \mathsf{trans_{fake}}[id] \leftarrow \{(r_{\mathsf{SIG}}, pk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, s', r_c', c)\}$
  return $\mathbf{pk_S}[id]$
else As usual.

$\mathcal{O}\mathsf{sign}_{(pp,pk_A,\mathbf{sk_S},\mathsf{L},\mathcal{Q},\mathsf{trans},\mathsf{trans_{fake}})}(id, m)$

if $id = \hat{id}$
  $\sigma_1 \leftarrow \mathcal{O}\mathsf{enc}(H_1(m||s_0^*), H_1(m||s_1^*));\ \sigma_2 \leftarrow \mathsf{SIG.Sign}(pp_{\mathsf{SIG}}, sk_{\mathsf{SIG}}, H_2(m||\sigma_1));\ \sigma \leftarrow (\sigma_1, \sigma_2)$
  $\mathsf{trans_{fake}}[id] \leftarrow \mathsf{CR.FakeTrans}(pp, pk_A, \mathsf{trans}[id], \mathsf{trans_{fake}}[id])$
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(id, m)\}$
  return σ
else As normal.

$\mathcal{O}\mathsf{corrupt}_{(\mathbf{sk_S},\mathsf{L},\mathsf{corL},\mathsf{crcL},\mathsf{trans})}(id)$

if $id = \hat{id}$ $\mathcal{B}$ aborts
else As normal.

$\mathcal{O}\mathsf{FAKEcoerce}/\mathcal{O}\mathsf{fakesign}$

As normal.

$H_1(X)$

if $(X, Y) \in \mathsf{hashL}$ **return** Y
else $Y \leftarrow_\$ \{0,1\}^\lambda;\ \mathsf{hashL} \leftarrow \mathsf{hashL} \cup \{(X, Y)\}$
  return Y

Figure 5.18: Adversary $\mathcal{B}$ that breaks the IND-CPA security of DEN in the proof of Lemma 8.

$(pp_{\mathsf{DEN}}, sk_A, c), \cdot)$ is included in set hashL. By definition, $\mathsf{DEN.Dec}(pp_{\mathsf{DEN}}, sk_A, c)$ is equal to s_0^* or s_1^*. As such, hashL includes $m^*||s_0^*$ or $m^*||s_1^*$. As $(id^*, m^*) \notin \mathcal{Q}$, neither is queried to the signing oracle. As such, adversary $\mathcal{A}$ must have queried H_1 directly. All inputs to the adversary are independent of $s_{1-\beta^*}^*$. Therefore, except for the negligible probability of the adversary guessing $s_{1-\beta^*}^*$, they must have queried $m^*||s_{\beta^*}^*$ to the hash oracle. Therefore, β^* is successfully guessed in the IND-CPA game. The probability that

$\mathcal{B}$ outputs 1 where $\beta^* = 1$ is greater than $(1 - \mathsf{negl}(\lambda))\epsilon$ where ϵ is non-negligible and factor $(1 - \mathsf{negl}(\lambda))$ is due to the negligible chance that $\mathcal{A}$ succeeds without querying $m^*||s_1^*$ to the hash oracle. The probability that $\mathcal{B}$ outputs 1 given $\beta^* = 0$ is negligible as all inputs to $\mathcal{A}$ are independent of s_1^* and, therefore, $\mathcal{A}$ inputs this to the hash oracle with negligible probability. As such, the IND-CPA multiple encryptions advantage is $\geq k_2 \cdot ((1 - \mathsf{negl}(\lambda)) \cdot \epsilon - \mathsf{negl}(\lambda))$, which is non-negligible. By contradiction, our CR.SIG construction satisfies strong soundness. □

5.5.2 Comparison with Existing Work

In [59], an embedded secret signature scheme construction is presented with an accompanying security model. The security model in [59] differs from our notion of coercion-resistance, capturing a notion more similar to our definition of receipt-freeness. The key difference is that, in the former definition, the adversary can only demand a secret key of a coerced signer, rather than full transcripts. The construction presented in [59], however, is similar to our coercion-resistant construction, and provides identical efficiency in terms of the sizes of signatures and computation during signing, verification and authentication. That being said, our construction can be distinguished in two ways. Firstly, the construction in [59] assumes that the authenticator and signer can privately share a secret during key generation. We do not require this assumption and, indeed, our syntax models key generation as non-interactive. Instead, in our construction, signers choose a secret randomly and encrypt this under the authenticator's public key with deniable encryption, including this encryption as part of their public key. Secondly, in [59], a deniably encrypted ciphertext is used as the randomness for the standard signature. In our constructions, we include an encryption of the signer's secret in the message signed by the standard signature scheme. Thus, our construction does not 'embed' the warning in the signature. Rather, it includes the warning as an additional part of the signature.

We note that, in [94], a very efficient construction for an embedded secret signature scheme is also given. In fact, this construction is more efficient that both our receipt-free and coercion-resistant constructions. However, it does not come with an accompanying security model, and, in fact, does not consider an attacker that demands a signer's secret key. Our constructions, on the other hand, are accompanied with rigorous proofs under suitable security models.

5.6 Receipt-Freeness and Deniable Encryption

We discussed in the introduction of this chapter that deniability is closely linked to incoercibility and can be used as a tool to achieve incoercibility. When designing incoercible signature constructions, deniable encryption was a natural choice because it provides signers with a tool that can be used to sign messages and then later lie about the contents of the signature. We now ask the question: is deniable encryption necessary to achieve receipt-freeness?

We show that, given a receipt-free incoercible signature scheme, we can construct a *partial* deniable encryption scheme, which is a deniable encryption scheme for a message space $\{0, 1\}$. Moreover, a partial deniable encryption scheme can only explain one of two messages (without loss of generality, we assume that only $m = 0$ can be explained). That is, given a ciphertext c, $\mathsf{DEN.Exp}$ can only generate randomness such that c appears to encrypt 0, regardless of the message it encrypts.

We define a partial deniable encryption scheme and then show that we can build a partial deniable encryption scheme from a receipt-free incoercible signature scheme. Finally, we show that our partial deniable encryption scheme construction is secure. That is, it satisfies correctness, $\mathsf{IND\text{-}CPA}$ security and $\mathsf{IND\text{-}EXP}$ security. Therefore, we answer our question by demonstrating that a variant of deniable encryption is necessary to build a receipt-free construction. We leave as an open question whether partial deniable encryption schemes can be built more efficiently than standard deniable encryption schemes, leading to efficiency improvements for receipt-free constructions.

5.6.1 Partial Deniable Encryption

We adapt the definition of public-key sender-deniable encryption (Definition 9 [34, 114]) such that the message space is $\{0, 1\}$ and algorithm $\mathsf{PDEN.Exp}$ no longer takes as input a message, because the only message that can be explained is 0.

Definition 66 (Partial Deniable Encryption Scheme)**.** *A* partial deniable encryption scheme PDEN *is a tuple of polynomial time algorithms* ($\mathsf{PDEN.Setup}, \mathsf{PDEN.KGen}, \mathsf{PDEN.Enc}$, $\mathsf{PDEN.Dec}, \mathsf{PDEN.Exp}$) *such that:*

$\mathsf{PDEN.Setup}(1^\lambda)$ On input of security parameter 1^λ, algorithm $\mathsf{PDEN.Setup}$ outputs public parameters pp_{PDEN}. We assume that pp_{PDEN} defines the message space $\mathsf{M}_{\mathsf{PDEN}} = \{0,1\}$.

$\mathsf{PDEN.KGen}(pp_{\mathsf{PDEN}})$ On input of public parameters pp_{PDEN}, algorithm $\mathsf{PDEN.KGen}$ outputs a key pair $(pk_{\mathsf{PDEN}}, sk_{\mathsf{PDEN}})$ where pk_{PDEN} is the public encryption key and sk_{PDEN} is the secret decryption key.

$\mathsf{PDEN.Enc}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, m)$ On input of public parameters pp_{PDEN}, public key pk_{PDEN} and message $m \in \mathsf{M}_{\mathsf{PDEN}}$, algorithm $\mathsf{PDEN.Enc}$ outputs a ciphertext c.

$\mathsf{PDEN.Dec}(pp_{\mathsf{PDEN}}, sk_{\mathsf{PDEN}}, c)$ On input of public parameters pp_{PDEN}, secret key sk_{PDEN} and ciphertext c, algorithm $\mathsf{PDEN.Dec}$ outputs a message m.

$\mathsf{PDEN.Exp}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, c)$ On input of public parameters pp_{PDEN}, public key pk_{PDEN}, and ciphertext c, algorithm $\mathsf{PDEN.Exp}$ outputs a string u.

We modify definitions of correctness, IND-CPA and IND-EXP to the partial deniability setting. In particular, the IND-CPA experiment does not provide the adversary with access to an encryption oracle because the only possible messages are 0 and 1. Rather, the adversary is provided with an encryption of a bit β where β is the bit that the adversary tries to guess in the IND-CPA experiment. Furthermore, in the IND-EXP experiment, the adversary does not have access to an oracle that provides the adversary with a ciphertext and randomness for a message chosen by the adversary because only the message 0 can be explained. Therefore, the adversary is simply provided with a ciphertext and randomness (either real or generated by algorithm $\mathsf{PDEN.Exp}$, depending on a bit β) for the message 0.

Definition 67 (Correctness)**.** *A partial deniable encryption scheme* PDEN *satisfies* correctness *if, for any message* $m \in \mathsf{M}_{\mathsf{PDEN}}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l} pp_{\mathsf{PDEN}} \leftarrow\!\$\, \mathsf{PDEN.Setup}(1^\lambda); \\ (pk_{\mathsf{PDEN}}, sk_{\mathsf{PDEN}}) \leftarrow\!\$\, \mathsf{PDEN.KGen}(pp_{\mathsf{PDEN}}); \\ c \leftarrow\!\$\, \mathsf{PDEN.Enc}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, m) \end{array} : \mathsf{PDEN.Dec}(pp_{\mathsf{PDEN}}, sk_{\mathsf{PDEN}}, c) \coloneqq m \right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Definition 68 (IND-CPA)**.** *A partial deniable encryption scheme* PDEN *satisfies* IND-CPA *security if, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}CPA},0}_{\mathsf{PDEN},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}CPA},1}_{\mathsf{PDEN},\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

$\mathsf{Exp}^{\mathsf{IND\text{-}CPA},\beta}_{\mathsf{PDEN},\mathcal{A}}(\lambda)$	$\mathsf{Exp}^{\mathsf{IND\text{-}EXP},\beta}_{\mathsf{PDEN},\mathcal{A}}(\lambda)$
$pp_{\mathsf{PDEN}} \leftarrow\!\!\$\ \mathsf{PDEN.Setup}(1^\lambda)$	$pp_{\mathsf{PDEN}} \leftarrow\!\!\$\ \mathsf{PDEN.Setup}(1^\lambda)$
$(pk_{\mathsf{PDEN}}, sk_{\mathsf{PDEN}}) \leftarrow\!\!\$\ \mathsf{PDEN.KGen}(pp_{\mathsf{PDEN}})$	$(pk_{\mathsf{PDEN}}, sk_{\mathsf{PDEN}}) \leftarrow\!\!\$\ \mathsf{PDEN.KGen}(pp_{\mathsf{PDEN}})$
$c^* \leftarrow\!\!\$\ \mathsf{PDEN.Enc}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, \beta)$	$c^* \leftarrow\!\!\$\ \mathsf{PDEN.Enc}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, 0; r_0)$;
$\beta' \leftarrow\!\!\$\ \mathcal{A}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, c^*)$	$r_1 \leftarrow\!\!\$\ \mathsf{PDEN.Exp}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, c^*)$;
return β'	$\beta' \leftarrow\!\!\$\ \mathcal{A}(pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}, c, u_\beta)$
	return β'

Figure 5.19: The IND-CPA and IND-EXP experiments for partial deniable encryption scheme PDEN.

where $\mathsf{Exp}^{\mathsf{IND\text{-}CPA},\beta}_{\mathsf{PDEN},\mathcal{A}}(\lambda)$ *is the experiment defined in Figure 5.19 for* $\beta \in \{0,1\}$.

Definition 69 (IND-EXP)**.** *A partial deniable encryption scheme* PDEN *satisfies* IND-EXP *security if, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}EXP},0}_{\mathsf{PDEN},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}EXP},1}_{\mathsf{PDEN},\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{IND\text{-}EXP},\beta}_{\mathsf{PDEN},\mathcal{A}}(\lambda)$ *is the experiment defined in Figure 5.19 for* $\beta \in \{0,1\}$.

5.6.2 Constructing a Partial Deniable Encryption Scheme

We show that, given an incoercible signature scheme INC-SIG that satisfies receipt-freeness, we can build a secure partial deniable encryption scheme PDEN. That is, we show that PDEN, defined formally in Figure 5.20, satisfies correctness, IND-CPA security and IND-EXP security, if INC-SIG is a receipt-free incoercible signature scheme. We give the PDEN construction in Figure 5.20.

Our PDEN construction can be described as follows. We define the public parameters to be equal to the public parameters for the incoercible signature scheme INC-SIG. Then, the key pair $(pk_{\mathsf{PDEN}}, sk_{\mathsf{PDEN}})$ is generated by algorithm PDEN.KGen and is equal to the key pair of the authenticator for the INC-SIG scheme. To encrypt a message, algorithm PDEN.Enc runs algorithm INC-SIG.SKGen to output a signer key pair and then produces a signature for scheme INC-SIG by running algorithm INC-SIG.Sign (if message $m = 0$) or INC-SIG.FakeSign (if message $m = 1$). Then, a real ($m = 0$) or a fake ($m = 1$) transcript is generated for the signer and the transcript is output as the ciphertext. To decrypt, algorithm PDEN.Dec runs INC-SIG.Authenticate, which returns 1 if the transcript (i.e., ciphertext) is real, and 0 otherwise. Therefore, if INC-SIG.Authenticate returns 1 (resp.,

0), the ciphertext encrypts message 0 (resp., 1). Finally, algorithm PDEN.Exp returns the randomness used to generate the signer key pair and the signature produced in algorithm PDEN.Enc.

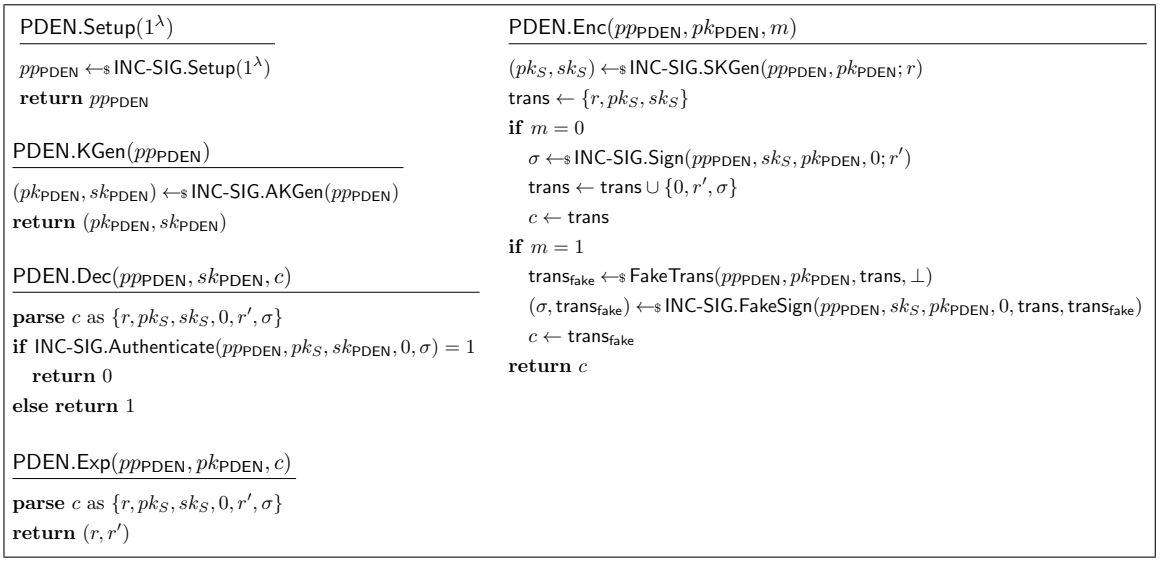

Figure 5.20: A partial deniable encryption scheme from a receipt-free incoercible signature scheme.

Theorem 18. PDEN *satisfies* correctness *if incoercible signature scheme* INC-SIG *is complete and sound.*

Proof. We first show that, if message $m = 0$, algorithm PDEN.Dec returns 0. In the correctness experiment, ciphertext c is distributed as follows: $(pk_S, sk_S) \leftarrow\!\!\$$ INC-SIG.SKGen(pp_{PDEN}, $pk_{\mathsf{PDEN}}; r$); trans $\leftarrow \{r, pk_S, sk_S\}$; $\sigma \leftarrow\!\!\$$ INC-SIG.Sign($pp_{\mathsf{PDEN}}, sk_S, pk_{\mathsf{PDEN}}, 0; r'$); trans $\leftarrow$ trans$\cup\{0, r', \sigma\}$; $c \leftarrow$ trans. By completeness of INC-SIG, INC-SIG.Authenticate(pp_{PDEN}, pk_S, $sk_{\mathsf{PDEN}}, 0, \sigma) = 0$ with at most negligible probability. Therefore, algorithm PDEN.Dec returns 0 with overwhelming probability.

We now show that, if $m = 1$, algorithm PDEN.Dec returns 1. Ciphertext c is distributed as follows: $(pk_S, sk_S) \leftarrow\!\!\$$ INC-SIG.SKGen($pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}; r$); trans $\leftarrow \{r, pk_S, sk_S\}$; $\mathsf{trans}_{\mathsf{fake}} \leftarrow\!\!\$$ FakeTrans($pp_{\mathsf{PDEN}}, pk_{\mathsf{PDEN}}$, trans, $\perp$); $(\sigma, \mathsf{trans}_{\mathsf{fake}}) \leftarrow\!\!\$$ INC-SIG.FakeSign(pp_{PDEN}, $sk_S, pk_{\mathsf{PDEN}}, 0$, trans, $\mathsf{trans}_{\mathsf{fake}}$); $c \leftarrow \mathsf{trans}_{\mathsf{fake}}$. By soundness of INC-SIG, INC-SIG.Authenticate($pp_{\mathsf{PDEN}}, pk_S, sk_{\mathsf{PDEN}}, 0, \sigma) = 1$ with at most negligible probability. Therefore, algorithm PDEN.Dec returns 1 with overwhelming probability. □

Theorem 19. PDEN *satisfies* IND-CPA *security if incoercible signature scheme* INC-SIG *satisfies* RF-IND *security.*

$\mathcal{B}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{coercesign}}(pp, pk_A)$	$\mathcal{B}_2^{\mathcal{O}\mathsf{coerce}}(st)$
for any id, $pk_S \leftarrow\!\!\$\ \mathcal{O}\mathsf{reg}(id)$	**parse** st as $(pp, pk_A, id, pk_S, \sigma)$
$\sigma \leftarrow\!\!\$\ \mathcal{O}\mathsf{coercesign}(id, 0)$	$\mathsf{trans} \leftarrow \mathcal{O}\mathsf{coerce}(id)$
return $st = (pp, pk_A, id, pk_S, \sigma)$	$\beta' \leftarrow\!\!\$\ \mathcal{A}(pp, pk_A, \mathsf{trans})$
	return β'

Figure 5.21: Adversary $\mathcal{B}$ that breaks the RF-IND security of scheme INC-SIG in the proof of Theorem 19.

Proof. Let $\mathcal{A}$ be an adversary that succeeds in the IND-CPA experiment for scheme PDEN defined in Figure 5.20. We show that we can construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that succeeds in the RF-IND experiment against scheme INC-SIG. We present $\mathcal{B}$ in Figure 5.21. It is clear that inputs to $\mathcal{A}$ are distributed identically to the IND-CPA experiment. When $\beta = 0$, trans is distributed identically to the encryption of 0 for the partial deniable encryption scheme PDEN. When $\beta = 1$, trans is distributed identically to the encryption of 1 in the partial deniable encryption scheme PDEN. Therefore, if $\mathcal{A}$ successfully guesses β in the IND-CPA experiment then $\mathcal{A}'$ successfully guesses β in the RF-IND experiment. □

Theorem 20. PDEN *satisfies* IND-EXP *security.*

Proof. Indistinguishability of explanation is perfectly satisfied by the PDEN construction. When 0 is encrypted, r and r' is the only randomness chosen and is output in the ciphertext. Algorithm PDEN.Exp returns this randomness, and inputs to the adversary are independent of β in the IND-EXP security experiment. □

5.7 Concluding Remarks

In this chapter, we introduced and defined incoercible signatures, and presented an accompanying security model. Specifically, our security model captures a strong notion of incoercibility, coercion-resistance, and we contributed an incoercible signature scheme construction that provably satisfies coercion-resistance. Additionally, our security model captures receipt-freeness. Though receipt-freeness is a weaker notion of security than coercion-resistance, we note that receipt-freeness may be sufficient in some application scenarios, for example, if the attacker can be assumed to interact with signers only after a protocol run is complete. Moreover, our receipt-freeness construction is more efficient than our coercion-resistant construction, demonstrating that efficiency/security trade-offs are

possible. We conclude this chapter by demonstrating that a variant of deniable encryption is necessary to construct incoercible signature schemes.

In this setting of digital signature schemes, we formally showed that coercion-resistance is stronger than receipt-freeness. By contrast, in Chapter 4 we found that, for the setting of e-voting, the relationship between receipt-freeness and coercion-resistance is not clear. Therefore, we can conclude that these complex notions of privacy are easier to reason about in a more fundamental setting. An interesting area of future work is to explore the 'gap' between receipt-freeness and coercion-resistance for incoercible signatures. Recalling that, for e-voting schemes, there exists several receipt-freeness definitions that capture different attacker models, we can explore whether similar variants of receipt-freeness and coercion-resistance exist (and can be achieved) for incoercible signatures. Potentially, studying this problem could aid understanding of the receipt-freeness and coercion-resistance notions in the context of fundamental cryptographic primitives such as digital signatures. The lessons learned from such a study could be carried forward to the study of complex cryptographic protocols such as e-voting.

Chapter 6

Report and Trace Ring Signatures

Contents

In this chapter we introduce report and trace ring signature schemes, balancing the desire for signer anonymity with the ability to report malicious behaviour and subsequently revoke anonymity. We contribute a formal security model for report and trace ring signatures, and present a construction of a report and trace ring signature scheme, proving its security and analysing its efficiency, comparing with the state of the art in the accountable ring signatures literature. We show the relation between report and trace and other ring signatures with tracing functionality. In particular, we formally establish relations amongst linkable, link and traceable, accountable, and report and trace ring signatures.

6.1 Introduction

In this chapter, we direct our attention to the topic of balancing privacy with other security goals. In particular, we consider how *data source privacy*, whereby the source of the data is secret, can be balanced with traceability, the property that the source of the data can be identified. We choose these properties as they appear to be contradictory. Indeed, how can a protocol preserve the privacy of the data source *and* identify the data source? One way in which these conflicting aims can be achieved is to preserve the privacy of a data source until an event occurs (e.g., malicious behaviour). After this event occurs, the data source is traced and identified. We consider how to achieve this balance in the context of ring signatures.

In a standard ring signature, signers generate key pairs and select a set of users, known as the ring, within which the signer's identity is hidden. The signer can generate publicly verifiably signatures, and anyone can tell that the signature is generated by a member of the ring, but not which member of the ring. Therefore, ring signatures provide a guarantee of data source privacy. Yet, if a signer acts maliciously, it may be desirable to provide a guarantee of traceability, which means that the privacy of the data source (i.e., the signer of the message) must be revoked. For this reason, ring signatures are a suitable setting in which to explore the relationship between data source privacy and traceability. In the ring signature literature, data source privacy is captured by an anonymity requirement [19], which states that the signer is anonymous within the ring. In this chapter, we explore how ring signatures can achieve the conflicting goals of of anonymity and traceability.

The solution to achieving the conflicting aims of anonymity and traceability is *accountable* ring signatures [130, 31], which define a designated tracer who can identify signers. Accountable ring signatures retain the versatility of ring signatures, allowing signers to generate their keys and select the anonymity ring, and additionally allow signer anonymity to be revoked by the designated tracer. In practice, to begin the tracing process, the designated tracer in an accountable ring signature will often receive a report of malicious behaviour from a reporter. That is, a protocol user may view a malicious message that is accompanied by a ring signature. The user may send a message to the tracer, alerting them to the malicious message, and the tracer can then revoke the signer's anonymity. However, the reporter is outside the scope of the syntax and security model of accountable

ring signatures. Consequently, it is implicit that the tracer must be trusted not to revoke anonymity without first receiving a report. Moreover, by omitting the role of the reporter from the security model, it is not possible to make any formal statements about the privacy of the reporter.

To address this, we introduce a new type of ring signature, which we call a *report and trace* ring signature. The underlying idea of report and trace is that a designated tracer can revoke anonymity of a signer if and only if a report of malicious behaviour is made by a user. In other words, a user reports a malicious message to the tracer, and the tracer must receive a report to revoke anonymity of the signer. Accordingly, report and trace achieves the balance between anonymity and traceability of accountable ring signatures, ensuring that the anonymity of a signer is preserved until tracing is complete. Additionally, report and trace incorporates a reporting system that preserves the anonymity of the signer *and* the reporter.

6.1.1 Our Contributions

We define syntax and a security model for report and trace (R&T) ring signatures. We demonstrate feasibility of report and trace by presenting a construction of an R&T ring signature scheme, and analyse the efficiency of our construction. We then extend our construction to support multiple reporters. Finally, we establish a hierarchy between linkable, link and traceable, accountable, and report and trace ring signature schemes. Here, we summarise our contributions.

Defining report and trace ring signatures. Report and trace ring signatures are an extension of a standard ring signature. Recall that, in a standard ring signature, the identity of a signer within a ring is not revealed. Ring signatures ensure that a signer cannot be identified; any ring member is equally likely to have produced a signature. R&T ring signatures extend this notion, allowing a signer to be identified if, and only if, an anonymous report is made to a designated tracer, who then traces the signer. To this end, our syntax defines a reporting user who provides the tracer with a reporter token, recovered from a signature, and a designated tracer who uses the reporter token to revoke anonymity of the signer.

Our security model extends the generic definitions of correctness, anonymity and unforgeability for ring signatures defined in [19] to capture ring signatures with a report and trace functionality. Furthermore, we define *traceability*, adapting the security properties of an accountable ring signature to our setting. Intuitively, traceability requires that the signer of a message can be correctly identified by the designated tracer. We capture this intuition via the following three requirements. *Trace correctness* ensures that an honestly generated report and trace will always identify the signer. *Non-frameability* requires that a non-signer cannot be identified as the signer, and *trace soundness* captures the requirement that the signer identified by the report and trace mechanism is unique. We complete our security model with a new definition of *reporter anonymity* for report and trace ring signatures, which captures the requirement that the reporter is anonymous even *after* tracing is complete.

Achieving report and trace. We present a generic R&T ring signature construction that relies on standard cryptographic primitives, namely, public-key encryption, non-interactive zero-knowledge proofs and a signature of knowledge. We prove that our construction is correct, anonymous, unforgeable, traceable and reporter anonymous, and our proofs of security rely on standard notions of security for the cryptographic primitives used in our construction.

Briefly, in our construction, the signer provably encrypts their identity under the tracer's public key for a public-key encryption scheme and then encrypts the resulting ciphertext using a one-time key-pair for a public-key encryption scheme. Additionally, the signer provably encrypts the one-time decryption key (which we call the reporter token) to all potential reporters. Then, the reporter decrypts their token, and the tracer requires the reporter token to recover the signer's identity. Our construction is based on the accountable ring signature of [31] in which the signer provable encrypts their identity under the tracer's public key and the tracer can revoke the signer's anonymity by decrypting the resulting ciphertext. We choose this construction due to its efficiency and because its security relies upon standard, well-understood cryptographic hardness assumptions (namely, the decisional Diffie-Hellman assumption). Furthermore, this approach allows us to clearly demonstrate the additional cost of reporting.

We analyse the efficiency of our construction, summarising the computational and commu-

nication costs associated with signing, reporting and tracing for our scheme. We provide an instantiation of our construction, which demonstrates that it can be implemented efficiently. In fact, for the cryptographic primitives we select, our construction performs favourably to the accountable ring signature of [31], and the additional cost of reporting is small.

Further results. We modify our construction to support multiple reporters (§6.5) using threshold publicly verifiable secret sharing [117]. We provide each potential reporter with a share of the reporter token, and a threshold of shares are required to recover the reporter token. We also present an efficiency analysis for our multiple reporter construction.

Finally, we consider R&T ring signatures in the context of related functionalities from the ring signature literature, namely linkable [98, 99] (LINK), link and traceable [68, 74] (L&T) and accountable [31, 130] (ACC) ring signatures. Specifically, we compare these functionalities and show formally how they relate to each other. In order to aid this analysis, we provide generic syntax that captures these three functionalities, and cast existing definitions of linkability, link and traceability and accountability into our syntax. We establish the following relations amongst these notions:

$$\mathsf{R\&T} \Rightarrow \mathsf{ACC} \Rightarrow \mathsf{L\&T} \Rightarrow \mathsf{LINK}.$$

6.2 Chapter Preliminaries

In this chapter, we introduce a new primitive, an R&T ring signature, and provide a way to achieve it. We are also interested in placing this primitive in the context of related schemes. Consequently, we present related work and highlight the advantages that our new primitive brings.

6.2.1 Related Work

Group signatures. Group signatures were introduced in [40], and the first security models were presented in [10, 15]. Group signatures require a group manager that determines the members of the group and issues key pairs to group members. Signers are

anonymous within the group, but the group manager can learn the identity of signers and revoke anonymity. Thus, group signatures rely on a trusted group manager to achieve the conflicting aims of anonymity and traceability. Several variants of group signatures have been proposed to limit trust in the group manager and protect the anonymity of non-malicious signers. For example, accountable tracing signatures [92] require that the group manager produce a proof of correct tracing and, if tracing occurred, a proof denying tracing cannot be produced. Traceable signatures [90] define a designated authority that can trace all signatures produced by a particular signer if the group manager first provides the authority with a tracing token related to that signer. In this way, the anonymity of non-malicious signers is protected; the group manager need not revoke anonymity of *all* signers to determine which signatures were produced by the signer in question. Furthermore, group signatures with message dependent opening (MDO signatures) [115] allow the group manager to revoke the anonymity of all signers that produced a signature for a particular message if and only if a reporter first produces a report related to that message. Our report and trace ring signature provides a similar distributed tracing function, but, in our setting, the report is attached to a signature rather than a message. Additionally, MDO signatures define the reporter to be a fixed entity with a secret key generated during setup. Report and trace ring signatures, on the other hand, model reporters as system users, and our security model ensures anonymity of the reporter. Finally, we note that report and trace is a variation of a *ring* signature and, as such, does not rely upon a trusted group manager to issue key pairs to users and allows users to select their anonymity ring.

Ring signatures. Ring signatures were first formally defined in [112] and a security model for ring signatures was presented in [19]. Following this, numerous variations of ring signatures have appeared (see, for example, [129] for a survey of some of these variations). Specifically, a number of ring signature variants offer some notion of traceability. For instance, linkable [99] and traceable [68] (which we call link and traceable ring signatures in this chapter to avoid confusion with the generic term traceability) ring signatures provide limited tracing functionality, allowing two signatures generated by the same signer to be linked. In particular, linkable ring signatures can be used to determine whether two signatures are created by the same signer, without revealing the identity of the signer, and link and traceable ring signatures allow the linking of ring signatures created by the same signer with respect to the same 'tag', and additionally reveal the identity of the signer. Moreover, accountable ring signatures, introduced in [130] and formalised in [31], allow

revocation of signer anonymity by a designated tracer and are, as a result, most closely related to our work. In fact, report and trace ring signatures can be viewed as an extension of accountable ring signatures, where the role of the tracer is distributed and the reporter is modelled as an anonymous system user.

Reporting systems. A closely related line of work is purpose-built reporting systems [4, 95, 111]. Analogously to our work, these systems allow a user to report another user and subsequently allow revocation of anonymity by a designated tracer. However, unlike our report and trace scheme, these systems are stand-alone reporting systems. Specifically, their design allows a user to identify an individual that has, for example, harassed or assaulted the user, hiding the identity of the accused and reporter until a threshold of reports related to the accused are submitted, at which point a tracer reveals the identity of the accused and the reporter(s). We note that, critically, these systems require a *threshold* of reports to revoke anonymity of the accused. This design decision empowers reporters, allowing them to submit accusations with the confidence that they will remain anonymous unless (or until) a number of other reporters have come forward. Finally, in [125], a report and trace scheme was introduced in the context of end-to-end encrypted messaging. In such systems, a message receiver can report a malicious message to a designated tracer, and the tracer can revoke anonymity of the sender. The tracer learns nothing about the sender unless a report is provided by the recipient of that message, and the identity of the reporter is revealed only to the tracer, albeit the reporter's identity is known to the tracer before tracing is complete.

6.2.2 Contextualising Report and Trace

Anonymity is one of the main security requirements of ring signatures. In fact, we require that R&T ring signatures satisfy (signer) anonymity *and* reporter anonymity. Similarly, other primitives with identity revocation functionality from the literature [4, 95, 111, 125, 90, 92, 115, 31, 130] have related anonymity requirements. Here, we discuss the anonymity guarantees of our construction and compare with related primitives. We summarise our findings in Table 6.1.

Revoking anonymity of the accused. All primitives with tracing functionality discussed so far [4, 95, 111, 125, 90, 92, 115, 31, 130] hide the identity of the accused (i.e., the signer in an R&T ring signature schemes) until tracing is complete, at which point, anonymity of the accused is revoked. We note that [4, 95, 111, 125] reveal the identity of the accused only to the tracer. However, for accountable ring signatures schemes [130, 31], group signature variants [90, 92] and our R&T ring signature, anonymity of the accused is *publicly* revoked to allow for public verification of the tracing process. Accordingly, the tracer is *accountable* for their actions and can only (provably) revoke the anonymity of a real accused user.

Entities revoking anonymity. To complete tracing, every primitive we consider [4, 95, 111, 125, 90, 92, 115, 31, 130] requires a designated tracer. In some systems, e.g., [4, 95, 130], the tracer is distributed. Whilst our R&T ring signature construction, and our multiple reporter construction, model the tracer as a single entity, we remark that we can also distribute the tracer, thus distributing trust amongst a set of tracers. Trust in the tracer can be further reduced by requiring a reporter. Our R&T ring signature and [125, 4, 95, 111, 115] define a reporter such that the reporter and tracer must cooperate to revoke anonymity. Additionally, purpose-built tracing systems [4, 95, 111] require a *threshold* of reports to trigger the tracing process. We provide both options: our R&T ring signature construction requires a single report; our multiple reporter construction requires a threshold of reports.

Anonymity of the reporter. Our (single reporter) R&T scheme ensures that the reporter is anonymous even *after* tracing. This is not true in the context of MDO signatures [115], where the reporter is a fixed, publicly-known, entity. Also, for end-to-end encrypted messaging [125], the tracer learns the identity of the reporter *before* starting the tracing process. Moreover, purpose-build reporting systems [4, 95, 111] intentionally reveal the identity of reporters after tracing. Recall that reporting systems allow reporters to communicate the identity of an accused person (e.g., a person accused of assault or illegal activity). Therefore, to follow up on allegations, revealing the reporter's identity is necessary. As the tracer in our R&T scheme does not require the identity of the reporter to follow up on an allegation (in fact, the allegation is that the message signed by the accused is malicious, and the message is public), we can protect the reporter's anonymity even after tracing. This empowers reporters to report malicious signers without fear of

identity exposure.

Integrated functionality Finally, we highlight that, in comparison to purpose-built reporting systems [4, 95, 111], our R&T scheme has integrated functionality. That is, our R&T scheme is a ring signature scheme with a report and trace function. Similarly, several primitives build a tracing function atop a group or ring signature [90, 92, 115, 31, 130], and traceable end-to-end encrypted messaging [125] incorporates a tracing function into an end-to-end encrypted messaging scheme.

	Publicly verifiable tracing	**Entities revoking anonymity**	**Reporter Anonymity**	**Integrated functionality**
Group signature variants [90, 92]	✓	Tracer	N/A	Signature
Group signature with message dependent opening [115]	✗	Reporter Tracer	✗	Signature
Accountable ring signatures [31, 130]	✓	Tracer	N/A	Signature
Traceable E2E encrypted messaging [125]	✗	Reporter Tracer	✗	Encryption
Reporting systems [4, 95, 111]	✗	Reporter (threshold) Tracer	✓*	None
R&T **ring signatures** (This work)	✓	Reporter Tracer	✓	Signature
R&T **ring signatures (multiple reporters)** (This work)	✓	Reporter (threshold) Tracer	✗	Signature

Table 6.1: Contextualising R&T ring signatures. $*$ denotes anonymity only holds until tracing is complete.

6.2.3 Application of Report and Trace

We illustrate the usefulness of report and trace by describing a potential application. Consider a forum platform and a set of registered users that can post messages to the forum. Users may wish to post messages anonymously, while also providing a signature proving that they are a registered user. Moreover, if a user posts a malicious message, the platform may wish to hold the signer accountable. Certainly, standard group and ring signature facilitate the ability of a user to sign a message anonymously. Furthermore, group signatures and accountable ring signatures balance anonymity and traceability. However, we believe that R&T ring signatures provide a unique solution to this scenario. Firstly, R&T ring signatures (and group signatures with message dependent opening) do not rely solely on a designated tracer to revoke anonymity and, as such, provide additional protection for the signer's identity above that provided by accountable ring signatures and group signatures. Moreover, distributing the tracing function in a busy forum scenario reduces the burden on the tracer to check for malicious messages. Indeed, the tracer need only

check messages for which the tracer receives a report. Additionally, our R&T signature allows the tracer to revoke anonymity only for the reported signature. That is, the signer preserves their anonymity with respect to all other signatures and no other signer who posts the same message will be de-anonymised. In our forum scenario, it may not be desirable to revoke anonymity for all signatures produced by the signer of a single malicious message. Moreover, it may be the case that a signed message is malicious in the context of which it is reported, but may be entirely innocuous in a different context. Consequently, R&T ring signatures are more appropriate than traceable signatures or MDO signatures for this setting. Finally, R&T ring signatures retain the versatility of ring signatures and define the reporter to be a system user, which can foster a sense of community responsibility for content posted on the forum, and provide a unique guarantee of anonymity for the reporter which can empower users to report malicious behaviour without fear of repercussions.

6.3 Syntax and Security Model for Report and Trace

We introduce the syntax of a *report and trace* (R&T) ring signature scheme and accompanying security model. Alongside a set of users an R&T ring signature scheme involves the following entities. A *reporter* produces a report. Within our syntax and security model, reporters are ring members, though this need not be the case. For example, a signer could potentially define a set of reporters that is independent of the ring. A *designated tracer*, denoted T, revokes the signer's anonymity if the tracer received a report for the signature in question. Anybody can verify the correctness of the report and trace by running a public verification algorithm. Formally, we define an R&T ring signature in Definition 70.

Definition 70 (R&T ring signature)**.** *An* R&T ring signature scheme *is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{T.KGen}, \mathsf{U.KGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Report}, \mathsf{Trace}, \mathsf{VerTrace})$ *such that*

$\mathsf{Setup}(1^\lambda)$ On input of security parameter 1^λ, algorithm Setup outputs public parameters pp. We assume that pp includes the message space M, the randomness space Rand and the key spaces PK and SK.

$\mathsf{T.KGen}(pp)$ On input of public parameters pp, algorithm $\mathsf{T.KGen}$ outputs a tracer key pair $(pk_\mathsf{T}, sk_\mathsf{T})$ where pk_T is the tracer's public key and sk_T is the tracer's secret key. We write that $pk_\mathsf{T} \leftarrow\!\!\$\ \mathsf{T.KGen}(pp; sk_\mathsf{T})$.

$\mathsf{U.KGen}(pp)$ On input of public parameters pp, algorithm $\mathsf{U.KGen}$ outputs a user key pair $(pk_{\mathsf{S}}, sk_{\mathsf{S}})$ where pk_{S} is the user's public key and sk_{S} is the user's secret key. We write that $pk_{\mathsf{S}} \leftarrow\!\!\$\, \mathsf{U.KGen}(pp; sk_{\mathsf{S}})$.

$\mathsf{Sign}(pp, sk_{\mathsf{S}}, pk_{\mathsf{T}}, m, R)$ On input of public parameters pp, user secret key sk_{S}, tracer public key pk_{T}, message m and ring R, algorithm Sign outputs a signature σ.

$\mathsf{Verify}(pp, pk_{\mathsf{T}}, m, R, \sigma)$ On input of public parameters pp, tracer public key pk_{T}, message m, ring R and signature σ, algorithm Verify outputs 1 if σ is a valid signature on m with respect to R, and 0 otherwise.

$\mathsf{Report}(pp, pk_{\mathsf{T}}, sk_{\mathsf{S}}, m, R, \sigma)$ On input of public parameters pp, tracer public key pk_{T}, user secret key sk_{S}, message m, ring R and signature σ, algorithm Report outputs a reporter token $\mathtt{Rep}$.

$\mathsf{Trace}(pp, sk_{\mathsf{T}}, m, R, \sigma, \mathtt{Rep})$ On input of public parameters pp, tracer secret key sk_{T}, message m, ring R, signature σ and reporter token $\mathtt{Rep}$, algorithm Trace outputs user public key pk_{S}, trace information $\mathtt{Tr}$ that includes the reporter token, and a proof of correct trace ρ_t.

$\mathsf{VerTrace}(pp, pk_{\mathsf{T}}, m, R, \sigma, pk_{\mathsf{S}}, \mathtt{Tr}, \rho_t)$ On input of public parameters pp, tracer public key pk_{T}, message m, ring R, signature σ, user public key pk_{S}, trace information $\mathtt{Tr}$ and tracer proof ρ_t, algorithm $\mathsf{VerTrace}$ outputs 1 if the trace is valid, and 0 otherwise.

We define correctness for our syntax as the property that honestly generated signatures are verifiable.

Definition 71 (Correctness)**.** *An* R&T *ring signature satisfies* correctness *if, for any* $n = \mathsf{poly}(\lambda)$*,* $j \in [n]$ *and message* $m \in \mathsf{M}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\begin{array}{l} pp \leftarrow\!\!\$\, \mathsf{Setup}(1^\lambda); \\ (pk_{\mathsf{T}}, sk_{\mathsf{T}}) \leftarrow\!\!\$\, \mathsf{T.KGen}(pp); \\ \textbf{for } i = 1,\ldots,n : (pk_{\mathsf{S}_i}, sk_{\mathsf{S}_i}) \leftarrow\!\!\$\, \mathsf{U.KGen}(pp); \\ R \leftarrow \{pk_{\mathsf{S}_1}, \ldots, pk_{\mathsf{S}_n}\}; \\ \sigma \leftarrow\!\!\$\, \mathsf{Sign}(pp, sk_{\mathsf{S}_j}, pk_{\mathsf{T}}, m, R) \end{array} : \mathsf{Verify}(pp, pk_{\mathsf{T}}, m, R, \sigma) := 1\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

6.3.1 Security Model

We present a security model for our syntax that incorporates accepted security properties from the ring signature literature. Firstly, our security model incorporates notions of

anonymity and unforgeability, which are fundamental security properties for any ring signature scheme (cf. Chapter 2). Indeed, we extend well-established definitions of anonymity and unforgeability for standard ring signature schemes, presented in [19], to our setting. Then, we cast the security requirements of an accountable ring signature into our syntax. Namely, we define traceability, which extends notions of trace correctness, non-frameability and tracing soundness for accountable ring signatures, defined in [31], to the report and trace setting. Finally, we present a definition of reporter anonymity, a new security property for our setting.

Corruption strategies. Our security model considers entities (i.e., users, reporters and tracers) that are either honest, corrupt, or under the control of an attacker. In detail, honest entities do not provide an attacker with secret keys and follow the protocol. Corrupt entities generate their keys honestly, but may later provide the attacker with their secret keys. Following this, the attacker can act on behalf of corrupt entities. Finally, the attacker can generate keys on behalf of controlled entities. An attacker that has keys of users, reporters or tracers can generate signatures, reports or traces respectively.

Oracles. In Figure 6.1, we define a number of oracles for our security experiments. In detail, our oracles operate as follows.

$\mathcal{O}\mathsf{reg}_{(pp,\mathsf{L},\mathbf{pk}_\mathsf{S},\mathbf{sk}_\mathsf{S})}(id)$ registers a user. That is, $\mathcal{O}\mathsf{reg}$ runs algorithm $\mathsf{U.KGen}$ and outputs user id's public key. Additionally, the user is added to a list of registered users L, and the user's public and secret credentials are respectively stored in vectors $\mathbf{pk}_\mathsf{S}$ and $\mathbf{sk}_\mathsf{S}$.

$\mathcal{O}\mathsf{corrupt}_{(\mathsf{L},\mathsf{corL},\mathbf{sk}_\mathsf{S})}(id)$ corrupts a user, outputting the user's secret credential and adding the user to a list of corrupt users corL.

$\mathcal{O}\mathsf{sign}_{(pp,\mathsf{L},\mathcal{Q}\mathsf{sign},\mathbf{pk}_\mathsf{S},\mathbf{sk}_\mathsf{S})}(id, pk_\mathsf{T}, m, R)$ runs algorithm Sign to produce a signature on behalf of a registered user id. The oracle outputs the signature and updates a set $\mathcal{Q}\mathsf{sign}$ to include a tuple that consists of the tracer's public key, user's public key, message, ring and signature.

$\mathcal{O}\mathsf{report}_{(pp,\mathsf{L},\mathcal{Q}\mathsf{rep},\mathbf{pk}_\mathsf{S},\mathbf{sk}_\mathsf{S})}(id, m, R, \sigma)$ runs algorithm Report, producing a reporter token for a signature, on behalf of a user id, if id's public key is in the ring R. Oracle $\mathcal{O}\mathsf{report}$

outputs the reporter token and, additionally, updates set $\mathcal{Q}\mathsf{rep}$ to track the fact that a reporter token is produced. That is, set $\mathcal{Q}\mathsf{rep}$ is updated to include the reporter's public key, and the message, signature and ring reported.

$\mathcal{O}\mathsf{trace}_{(pp,sk_{\mathsf{T}},\mathcal{Q}\mathsf{trace})}(m, R, \sigma, \mathtt{Rep})$ traces the signer of a message m by running algorithm Trace to produce and output the signer's public credential pk_{S}, trace information $\mathtt{Tr}$, and proof of correct tracing ρ_t. A set $\mathcal{Q}\mathsf{trace}$ is updated to include the tuple consisting of the message, ring and signature, thereby tracking that signature σ has been queried to the oracle.

$\mathcal{O}\mathsf{reg}_{(pp,\mathsf{L},\mathbf{pk}_{\mathsf{S}},\mathbf{sk}_{\mathsf{S}})}(id)$

$(\mathbf{pk}_{\mathsf{S}}[id], \mathbf{sk}_{\mathsf{S}}[id]) \leftarrow\!\!\$\ \mathsf{U.KGen}(pp)$
$\mathsf{L} \leftarrow \mathsf{L} \cup \{\mathbf{pk}_{\mathsf{S}}[id]\}$
return $\mathbf{pk}_{\mathsf{S}}[id]$

$\mathcal{O}\mathsf{corrupt}_{(\mathsf{L},\mathsf{corL},\mathbf{pk}_{\mathsf{S}},\mathbf{sk}_{\mathsf{S}})}(id)$

if $\mathbf{pk}_{\mathsf{S}}[id] \notin \mathsf{L}$ **return** $\perp$
$\mathsf{corL} \leftarrow \mathsf{corL} \cup \{\mathbf{pk}_{\mathsf{S}}[id]\}$
return $\mathbf{sk}_{\mathsf{S}}[id]$

$\mathcal{O}\mathsf{sign}_{(pp,\mathsf{L},\mathcal{Q}\mathsf{sign},\mathbf{pk}_{\mathsf{S}},\mathbf{sk}_{\mathsf{S}})}(id, pk_{\mathsf{T}}, m, R)$

if $\mathbf{pk}_{\mathsf{S}}[id] \notin \mathsf{L}$ **return** $\perp$
$\sigma \leftarrow\!\!\$\ \mathsf{Sign}(pp, \mathbf{sk}_{\mathsf{S}}[id], pk_{\mathsf{T}}, m, R \cup \{\mathbf{pk}_{\mathsf{S}}[id]\})$
$\mathcal{Q}\mathsf{sign} \leftarrow \mathcal{Q}\mathsf{sign} \cup \{(pk_{\mathsf{T}}, \mathbf{pk}_{\mathsf{S}}[id], m, R, \sigma)\}$
return σ

$\mathcal{O}\mathsf{report}_{(pp,\mathsf{L},\mathcal{Q}\mathsf{rep},\mathbf{pk}_{\mathsf{S}},\mathbf{sk}_{\mathsf{S}})}(id, m, R, \sigma)$

if $\mathbf{pk}_{\mathsf{S}}[id] \notin R \ \vee\ \mathbf{pk}_{\mathsf{S}}[id] \notin \mathsf{L}$ **return** $\perp$
$\mathtt{Rep} \leftarrow\!\!\$\ \mathsf{Report}(pp, pk_{\mathsf{T}}, \mathbf{sk}_{\mathsf{S}}[id], m, R, \sigma)$
$\mathcal{Q}\mathsf{rep} \leftarrow \mathcal{Q}\mathsf{rep} \cup \{(\mathbf{pk}_{\mathsf{S}}[id], m, R, \sigma)\}$
return $\mathtt{Rep}$

$\mathcal{O}\mathsf{trace}_{(pp,sk_{\mathsf{T}},\mathcal{Q}\mathsf{trace})}(m, R, \sigma, \mathtt{Rep})$

$(pk_{\mathsf{S}}, \mathtt{Tr}, \rho_t) \leftarrow\!\!\$\ \mathsf{Trace}(pp, sk_{\mathsf{T}}, m, R, \sigma, \mathtt{Rep})$
$\mathcal{Q}\mathsf{trace} \leftarrow \mathcal{Q}\mathsf{trace} \cup \{(m, R, \sigma)\}$
return $(pk_{\mathsf{S}}, \mathtt{Tr}, \rho_t)$

Figure 6.1: Oracles used in the experiments for anonymity, unforgeability, traceability and reporter anonymity of an R&T ring signature scheme.

With these oracles, we now present our security model for R&T ring signatures.

Anonymity. We extend anonymity against adversarially chosen keys for a standard ring signature to the report and trace signature scheme setting. Our anonymity experiment is described as follows. As in the standard definition, the adversary in the anonymity experiment can obtain the public and secret keys of any signer by querying oracles $\mathcal{O}\mathsf{reg}$ and $\mathcal{O}\mathsf{corrupt}$ respectively, and can obtain signatures via oracle $\mathcal{O}\mathsf{sign}$. The adversary outputs two potential honest signers, denoted id_0 and id_1, and is provided with a challenge signature generated for one of the two signers that is dependent on a bit β chosen for the security experiment. In addition, our anonymity experiment provides the adversary with access to oracles $\mathcal{O}\mathsf{report}$ and $\mathcal{O}\mathsf{trace}$ that return reporter tokens and traces respectively. We require that the adversary does not query the challenge signature to $\mathcal{O}\mathsf{report}$ and $\mathcal{O}\mathsf{trace}$. Otherwise, the adversary trivially identifies the signer. For similar reasons, we assume that the tracer is honest in our anonymity experiment. An R&T ring signature scheme is anonymous with respect to adversarially chosen keys if the adversary can guess a bit β with probability only negligibly more than 1/2.

Definition 72 (Anonymity)**.** *An* R&T *ring signature satisfies* anonymity with respect to adversarially generated keys *if, for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* negl *such that,*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{anon},0}_{\mathsf{R\&T},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{anon},1}_{\mathsf{R\&T},\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{anon},\beta}_{\mathsf{R\&T},\mathcal{A}}(\lambda)$ *is the experiment defined in Figure 6.2 for* $\beta \in \{0, 1\}$*.*

Unforgeability. We adapt unforgeability for a standard ring signature to our syntax. Thus, we require that the adversary cannot output a valid signature on behalf of a ring of honest users, where the signature is not the output of the signing oracle. As with anonymity, the adversary has access to a number of oracles defined in Figure 6.1. To model the addition of a tracer in our syntax, our unforgeability definition allows the adversary to control the tracer and, on the condition that the adversary also corrupts a reporter or queries oracle $\mathcal{O}\mathsf{report}$, the adversary can trace the identity of any signer. To satisfy unforgeability, we require that an adversary can output a forgery on behalf of an honest ring with negligible probability.

Definition 73 (Unforgeability)**.** *An* R&T *ring signature scheme satisfies* unforgeability *if, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\mathsf{Exp}^{\mathsf{unfor}}_{\mathsf{R\&T},\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{unfor}}_{\mathsf{R\&T},\mathcal{A}}(\lambda)$ *is the experiment defined in Figure 6.2.*

Traceability. R&T signatures must satisfy *traceability*. Traceability comprises three conditions: trace correctness, non-frameability and trace soundness, which are adapted from the security model for an accountable ring signature presented in [31]. *Trace correctness* requires that an honestly generated signature must be traceable to the correct signer. We capture trace correctness in an experiment that requires an honestly generated report and trace for an honestly generated signature to verify. *Non-frameability* captures the requirement that an honest user cannot be accused of producing a ring signature that they did not produce. To this end, our non-frameability definition requires that the adversary, with control of the tracer and a subset of users, cannot output a valid trace such that the trace identifies a non-signer. Finally, *trace soundness* stipulates that the signer identified

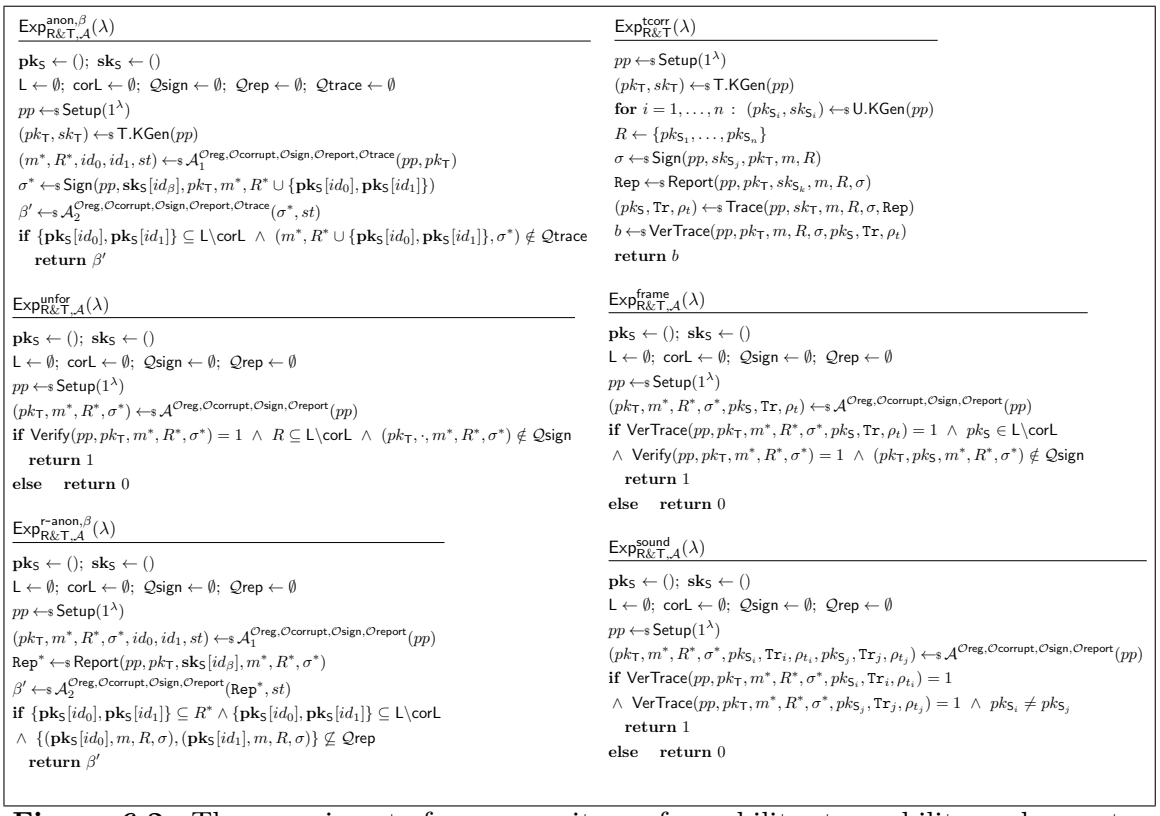

Figure 6.2: The experiments for anonymity, unforgeability, traceability and reporter anonymity of an R&T ring signature scheme where oracles used in the experiments are defined in Figure 6.1.

by the report and trace mechanism is unique. Formally, trace soundness considers an adversary that controls the tracer and can corrupt and control users and reporters. Trace soundness requires that the adversary cannot output two valid traces that identify two different signers for the same message.

Definition 74 (Traceability)**.** *An* R&T *ring signature satisfies* traceability *if the following conditions are satisfied.*

1. Trace correctness: *for any* $n = \mathsf{poly}(\lambda)$*,* $j \in [n]$*,* $k \in [n]$ *where* $j \neq k$*, and message* $m \in \mathsf{M}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\mathsf{Exp}^{\mathsf{tcorr}}_{\mathsf{R\&T}}(\lambda) = 1\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

2. Non-frameability; *for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\mathsf{Exp}^{\mathsf{frame}}_{\mathsf{R\&T},\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,.$$

3. Trace soundness: *for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl

such that,

$$\Pr\left[\mathsf{Exp}^{\mathsf{sound}}_{\mathsf{R\&T},\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{tcorr}}_{\mathsf{R\&T}}(\lambda)$, $\mathsf{Exp}^{\mathsf{frame}}_{\mathsf{R\&T},\mathcal{A}}(\lambda)$ *and* $\mathsf{Exp}^{\mathsf{sound}}_{\mathsf{R\&T},\mathcal{A}}(\lambda)$ *are the experiments defined in Figure 6.2.*

Reporter anonymity. We define *reporter anonymity*, a new property that requires that a report does not reveal the ring member that produced it. We formally define reporter anonymity as the property that an adversary, when provided with a report for a signature, cannot determine which of two potential reporters produced the report. We capture the reporter anonymity requirement in an indistinguishability experiment that is inspired by the anonymity experiment for R&T ring signatures. That is, we describe an experiment in which an adversary outputs two possible reporters and receives a report that is generated by one of the reporters (dependent on a bit β chosen for the security experiment). Through oracle access, the adversary can corrupt and register users, and generate signatures and reports on behalf of users. Additionally, the adversary controls the tracer. We require that the adversary does not corrupt either of the potential reporters and does not obtain a report through access to oracles. Otherwise, it is trivial for the adversary to locally compute a report and compare with the challenge report returned in the experiment. To satisfy reporter anonymity, it must be the case that the adversary can guess β with probability only negligibly more than 1/2.

Definition 75 (Reporter anonymity)**.** *An* R&T *ring signature satisfies* reporter anonymity *if, for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* negl *such that,*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{r\text{-}anon},0}_{\mathsf{R\&T},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{r\text{-}anon},1}_{\mathsf{R\&T},\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{r\text{-}anon},\beta}_{\mathsf{R\&T},\mathcal{A}}(\lambda)$ *is the experiment defined in Figure 6.2 for* $\beta \in \{0, 1\}$*.*

6.4 A Report and Trace Ring Signature Construction

We now show that our report and trace notion is achievable. That is, we present an R&T ring signature construction, formally defined in Figure 6.3. Our construction requires a one-way function f, a public-key encryption scheme PKE, a zero-knowledge proof system NIZK, and a signature of knowledge SOK.

The idea behind our construction is as follows. The tracer and users each obtain a key pair for a PKE scheme. The signer generates a fresh key pair for a PKE scheme, and the freshly generated decryption key (known as the reporter token in our construction) is encrypted to all members of the ring using a PKE scheme. The signer then uses a double layer of encryption to encrypt their public identity, which is generated via one-way function f. That is, the signer encrypts their public identity under the public key of the tracer, and encrypts the resulting ciphertext under the freshly generated encryption key. In this way, a reporter *and* the tracer are required to recover the signer's identity. Indeed, any ring member can decrypt the reporter token, and the tracer requires the reporter token, along with their own decryption key, to remove the double-layer of encryption and revoke anonymity of the signer. Our construction additionally employs NIZK proofs and an SOK to ensure that operations are performed correctly, i.e., that the signer encrypts the correct public identity and reporter token, and that the reporter and tracer identify the correct signer.

Our construction is similar to the construction in [31], which provides an efficient accountable ring signature scheme that allows a designated tracer to revoke signer anonymity. In [31], the signer uses a PKE scheme to encrypt their public identity under the tracer's public key, and the tracer recovers the signer's identity by decrypting the ciphertext. This construction also relies on an SOK that allows the signer to prove that they have encrypted a public identity for which they know a corresponding secret, and a NIZK proof such that the tracer can prove correct decryption, i.e., that they traced the correct signer. Our R&T construction differs from [31] in the following way: we require the encryption of a token to a set of reporters and provide a NIZK proof of correct encryption. Additionally, we use a double-layer of encryption, which is crucial to ensuring that the tracer cannot decrypt the signer's identity without a reporter token.

In this section, we proceed as follows. First, we present our construction and then prove its security. We then provide a concrete instantiation of our construction and conclude by discussing the efficiency of our generic construction and instantiation.

6.4.1 Description of Our Construction

We now describe the details of our construction. A trusted third party runs Setup, performing setup for the PKE, NIZK and SOK schemes. We assume that the public parameters generated for each scheme defines the public/secret key, randomness and message spaces (which we denote respectively as PK, SK, Rand and M) as appropriate. $\mathsf{T.KGen}$ generates a tracer key pair for a PKE scheme, and $\mathsf{U.KGen}$ is run to generate user key pairs. In particular, users generate a signing/verification key pair $(pk_{\mathsf{RS}}, sk_{\mathsf{RS}})$ using one-way function f, and a key pair $(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}})$ for a PKE scheme.

To sign a message m with respect to a ring R, the signer runs algorithm Sign. The signer generates a fresh key pair $(pk_{\mathsf{Sign}}, sk_{\mathsf{Sign}})$ for a PKE scheme and encrypts the reporter token sk_{Sign} to each member of the ring (i.e., encrypts sk_{Sign} under the public encryption key of each ring member). The signer proves that each PKE ciphertext encrypts the reporter token sk_{Sign} associated with pk_{Sign}, which is included in the signature. That is, the signer provides a NIZK proof for the following relation:

$$\mathcal{R}_{\mathsf{Enc}} = \left\{ \begin{matrix} (pp,(pk_{\mathsf{Sign}},(pk_{\mathsf{PKE}_1},\ldots,pk_{\mathsf{PKE}_{|R|}}),\mathbf{c_1}),(r_{1,1},\ldots,r_{1,|R|},sk_{\mathsf{Sign}})) \ : \\ pk_{\mathsf{Sign}} \coloneqq \mathsf{PKE.KGen}(pp_{\mathsf{PKE}};sk_{\mathsf{Sign}}) \wedge \left\{ \forall\ i \in 1,\ldots,|R| \ :\ c_{1,i} \coloneqq \mathsf{PKE.Enc}(pp_{\mathsf{PKE}},pk_{\mathsf{PKE}_i},sk_{\mathsf{Sign}};r_{1,i}) \right\} \end{matrix} \right\} \tag{6.1}$$

Then, the signer's verification key pk_{RS} is encrypted under the tracer's public key, resulting in ciphertext c_2, which is then encrypted under the freshly generated public key pk_{Sign}, giving ciphertext c_3. Finally, the signer produces a signature of knowledge, which proves that c_3 encrypts a verification key in the ring such that the signer knows the associated signing key. The signature of knowledge is associated with the following relation:

$$\mathcal{R}_{\mathsf{SOK}} = \left\{ \begin{matrix} (pp,(pk_{\mathsf{T}},pk_{\mathsf{Sign}},R,c_3),(r_2,r_3,sk_{\mathsf{RS}}),m) \ :\ c_3 \coloneqq \mathsf{PKE.Enc}(pp_{\mathsf{PKE}},pk_{\mathsf{Sign}},c_2;r_3) \\ \wedge\ c_2 \coloneqq \mathsf{PKE.Enc}(pp_{\mathsf{PKE}},pk_{\mathsf{T}},pk;r_2) \wedge pk \coloneqq f(sk_{\mathsf{RS}}) \in R \end{matrix} \right\} \tag{6.2}$$

To report a message, a member of the ring runs Report to decrypt the reporter token. The reporter additionally provides a proof of correct decryption, without revealing which

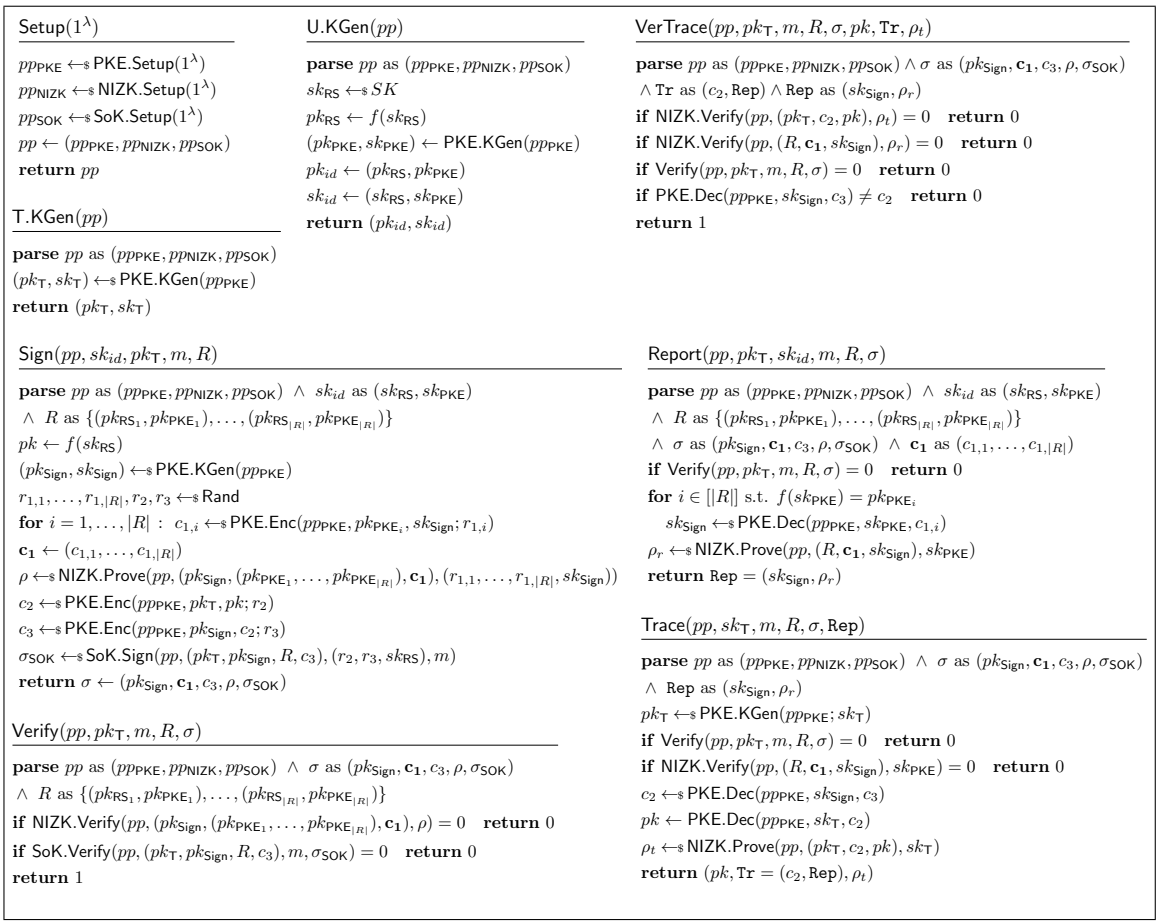

Figure 6.3: Our report and trace ring signature construction.

member of the ring decrypted the token. This is given by the following relation:

$$\mathcal{R}_{\mathsf{Dec}_r} = \left\{ \begin{array}{c} (pp,(R,\mathbf{c_1},sk_{\mathsf{Sign}}),sk_{\mathsf{PKE}}) \ : \ sk_{\mathsf{Sign}} \coloneqq \mathsf{PKE.Dec}(pp_{\mathsf{PKE}},sk_{\mathsf{PKE}},c_{1,i}) \\ \wedge\ c_{1,i} \in \mathbf{c_1} \ \wedge\ pk_{\mathsf{PKE}} \coloneqq \mathsf{PKE.KGen}(pp_{\mathsf{PKE}};sk_{\mathsf{PKE}}) \in R \end{array} \right\} \tag{6.3}$$

On receipt of a report, the tracer runs Trace to decrypt ciphertexts c_3 and c_2, thus revealing the signer's verification key. As sk_{Sign} is included in the report, anyone can decrypt c_3, hence checking correct decryption directly. As such, the tracer need only prove correct decryption of c_2, which is given by the following relation:

$$\mathcal{R}_{\mathsf{Dec}_t} = \left\{ \begin{array}{c} (pp,(pk_{\mathsf{T}},c_2,pk_{\mathsf{RS}}),sk_{\mathsf{T}}) \ : \ pk_{\mathsf{RS}} \coloneqq \mathsf{PKE.Dec}(pp_{\mathsf{PKE}},sk_{\mathsf{T}},c_2) \\ \wedge\ pk_{\mathsf{T}} \coloneqq \mathsf{PKE.KGen}(pp_{\mathsf{PKE}};sk_{\mathsf{T}}) \end{array} \right\} \tag{6.4}$$

Our construction additionally provides a public signing verification algorithm Verify, which ensures that the signer provides an encryption of their own public key, enabling tracing if the message is malicious. Moreover, a public trace verification algorithm VerTrace ensures that the correct signer is traced.

6.4.2 Security of Our Construction

We prove that our construction satisfies correctness, anonymity, unforgeability, traceability and reporter anonymity as defined in Section 6.3. Here, we provide formal proofs for each of these security properties.

Theorem 21. *Our construction satisfies* correctness *if public-key encryption scheme* PKE *is correct, non-interactive zero-knowledge proof system* NIZK *is complete, and signature of knowledge* SOK *is correct.*

Proof. Consider a signature $\sigma = (pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}})$ output by algorithm Sign with respect to a message $m \in \mathsf{M}$ and ring R that consists of public keys generated by algorithm U.KGen. Let pp be the output of algorithm Setup and let $(pk_{\mathsf{T}}, sk_{\mathsf{T}})$ be the output of algorithm T.KGen. Moreover, let $pk = f(sk_{\mathsf{RS}})$ where $(pk, \cdot) \in R$ and $sk_{\mathsf{RS}} \leftarrow\!\!\$\, SK$ is used to sign message m.

By definition of correctness, our construction is correct if $\mathsf{Verify}(pp, pk_{\mathsf{T}}, m, R, \sigma)$ outputs 1 with overwhelming probability. Assume that algorithm Verify does not return 1. Then, it must be the case that $\mathsf{NIZK.Verify}(pp, (pk_{\mathsf{Sign}}, (pk_{\mathsf{PKE}_1}, \ldots, pk_{\mathsf{PKE}_{|R|}}), \mathbf{c_1}), \rho) = 0$ or $\mathsf{SoK.Verify}(pp, (pk_{\mathsf{T}}, pk_{\mathsf{Sign}}, R, c_3), m, \sigma_{\mathsf{SOK}}) = 0$. We consider these two possibilities in turn. We conclude that, if all building blocks are correct, our construction satisfies correctness.

First, assume that $\mathsf{NIZK.Verify}(pp, (pk_{\mathsf{Sign}}, (pk_{\mathsf{PKE}_1}, \ldots, pk_{\mathsf{PKE}_{|R|}}), \mathbf{c_1}), \rho) = 0$. By assumption, the PKE scheme satisfies correctness and so $\mathbf{c_1}$ encrypts sk_{Sign}, which is cryptographically linked to pk_{Sign}. Then, the inputs to algorithm NIZK.Verify are correctly generated and, by completeness of the NIZK scheme, algorithm NIZK.Verify returns 1 with overwhelming probability.

Now, assume that $\mathsf{SoK.Verify}(pp, (pk_{\mathsf{T}}, pk_{\mathsf{Sign}}, R, c_3), m, \sigma_{\mathsf{SOK}}) = 0$. As before, by assumption of correctness of the PKE scheme, ciphertexts c_2 and c_3 encrypt pk and c_2 respectively. Then, the inputs to algorithm SoK.Sign are generated correctly and, by correctness of the SOK scheme, algorithm SoK.Verify returns 1 with overwhelming probability. □

Theorem 22. *Our construction satisfies* anonymity *if public-key encryption scheme* PKE *satisfies* IND-CPA*, non-interactive zero-knowledge proof system* NIZK *satisfies zero-*

knowledge and knowledge extractability, and signature of knowledge SOK *satisfies simulatability and extractability.*

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary in the anonymity experiment that makes at most k_1 queries to oracle $\mathcal{O}$report and at most k_2 queries to oracle $\mathcal{O}$trace. We proceed through a series of game hops that we show are indistinguishable to the adversary. In the final game, the view of the adversary will be identical for $\beta = 0$ and $\beta = 1$. That is, in our game hops, we change the view of the adversary such that, in the final game, the adversary receives challenge signatures for signer id_0, regardless of the bit β. In our first three game hops, we take steps to simulate all NIZK proofs and signatures of knowledge generated in the experiment. We then define two game hops in which we extract the reporter token and trace (i.e., the signer's identity), rather than decrypting ciphertexts. We do this such that, in the final game hop, we can always encrypt the public key of signer id_0 and can rely on the IND-CPA property of the PKE scheme, rather than the IND-CCA property, to prove that an adversary cannot distinguish this change. We define Game 0 as the anonymity experiment with β chosen randomly. Let S_i denote the event that adversary $\mathcal{A}$ correctly guesses β in Game i.

Game 1 is identical to Game 0 except that, when running algorithm Setup, we replace algorithm SoK.Setup with algorithm SimSOK.Setup and algorithm NIZK.Setup with algorithm SimNIZK.Setup. We provide the adjusted experiment in Figure 6.4. This is a superficial change that does not affect the distribution of inputs to the adversary. Therefore,

$$\Big|\Pr[S_0] - \Pr[S_1]\Big| = 0.$$

$\mathsf{Exp}^{\mathsf{anon},\beta}_{\mathsf{R\&T},\mathcal{A}}(\lambda)$

$\mathbf{pk}_\mathsf{S} \leftarrow (); \ \mathbf{sk}_\mathsf{S} \leftarrow (); \ \mathsf{L} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{rep} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{trace} \leftarrow \emptyset$
$pp_\mathsf{PKE} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda); \ (pp_\mathsf{NIZK}, \tau_\mathsf{NIZK}) \leftarrow\!\!\$\ \mathsf{NIZK.SimSetup}(1^\lambda); \ (pp_\mathsf{SOK}, \tau_\mathsf{SOK}) \leftarrow\!\!\$\ \mathsf{SoK.SimSetup}(1^\lambda)$
$pp \leftarrow (pp_\mathsf{PKE}, pp_\mathsf{NIZK}, pp_\mathsf{SOK}); \ (pk_\mathsf{T}, sk_\mathsf{T}) \leftarrow\!\!\$\ \mathsf{T.KGen}(pp)$
$(m^*, R^*, id_0, id_1, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report},\mathcal{O}\mathsf{trace}}(pp, pk_\mathsf{T})$
$\sigma^* \leftarrow\!\!\$\ \mathsf{Sign}(pp, \mathbf{sk}_\mathsf{S}[id_\beta], pk_\mathsf{T}, m^*, R^* \cup \{\mathbf{pk}_\mathsf{S}[id_0], \mathbf{pk}_\mathsf{S}[id_1]\})$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report},\mathcal{O}\mathsf{trace}}(\sigma^*, st)$
if $\{\mathbf{pk}_\mathsf{S}[id_0], \mathbf{pk}_\mathsf{S}[id_1]\} \subseteq \mathsf{L}\backslash\mathsf{corL} \ \wedge \ (m^*, R^* \cup \{\mathbf{pk}_\mathsf{S}[id_0], \mathbf{pk}_\mathsf{S}[id_1]\}, \sigma^*) \notin \mathcal{Q}\mathsf{trace}$ **return** β'

Figure 6.4: Modified anonymity experiment for Game 1 in the proof of Theorem 22.

Game 2 is identical to Game 1 except for the following change. For queries to oracle $\mathcal{O}$sign

and the challenge signature returned to $\mathcal{A}$, we replace algorithm $\mathsf{SoK.Sign}$ with algorithm $\mathsf{SoK.SimSign}$. Then, generating the signature of knowledge does not require a witness. We define oracle $\mathcal{O}\mathsf{sign}$ and the adjusted experiment in Figure 6.5.

$\mathsf{Exp}^{\mathsf{anon},\beta}_{\mathsf{R\&T},\mathcal{A}}(\lambda)$

$\mathbf{pk_S} \leftarrow (); \ \mathbf{sk_S} \leftarrow (); \ \mathsf{L} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{rep} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{trace} \leftarrow \emptyset$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda); \ (pp_{\mathsf{NIZK}}, \tau_{\mathsf{NIZK}}) \leftarrow\!\!\$\ \mathsf{NIZK.SimSetup}(1^\lambda); \ (pp_{\mathsf{SOK}}, \tau_{\mathsf{SOK}}) \leftarrow\!\!\$\ \mathsf{SoK.SimSetup}(1^\lambda)$
$pp \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}}); \ (pk_{\mathsf{T}}, sk_{\mathsf{T}}) \leftarrow\!\!\$\ \mathsf{T.KGen}(pp)$
$(m^*, R^*, id_0, id_1, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report},\mathcal{O}\mathsf{trace}}(pp, pk_{\mathsf{T}})$
$\mathbf{parse}\ R^*\ \text{as}\ \{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\}$
$R' \leftarrow R^* \cup \{(pk_{\mathsf{RS}_{id_0}}, pk_{\mathsf{PKE}_{id_0}}), (pk_{\mathsf{RS}_{id_1}}, pk_{\mathsf{PKE}_{id_1}})\}; \ pk_\beta \leftarrow\!\!\$\ f(sk_{\mathsf{RS}_{id_\beta}})$
$(pk_{\mathsf{Sign}}, sk_{\mathsf{Sign}}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}}); \ r_{1,1}, \ldots, r_{1,|R'|}, r_2, r_3 \leftarrow\!\!\$\ \mathsf{Rand}$
$\mathbf{for}\ i = 1, \ldots, |R'| \ : \ c_{1,i} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}_i}, sk_{\mathsf{Sign}}; r_{1,i})$
$\mathbf{c_1} \leftarrow (c_{1,1}, \ldots, c_{1,|R'|})$
$\rho \leftarrow\!\!\$\ \mathsf{NIZK.Prove}(pp, (pk_{\mathsf{Sign}}, (pk_{\mathsf{PKE}_1}, \ldots, pk_{\mathsf{PKE}_{|R'|}}), \mathbf{c_1}), (r_{1,1}, \ldots, r_{1,|R'|}, sk_{\mathsf{Sign}}))$
$c_2 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{T}}, pk_\beta; r_2); \ c_3 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{Sign}}, c_2; r_3)$
$\sigma_{\mathsf{SOK}} \leftarrow\!\!\$\ \mathsf{SoK.SimSign}(pp, \tau_{\mathsf{SOK}}, (pk_{\mathsf{T}}, pk_{\mathsf{Sign}}, R', c_3), m^*); \ \sigma^* \leftarrow\!\!\$\ (pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}})$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report},\mathcal{O}\mathsf{trace}}(\sigma^*, st)$
$\mathbf{if}\ \{\mathbf{pk_S}[id_0], \mathbf{pk_S}[id_1]\} \subseteq \mathsf{L}\backslash\mathsf{corL} \ \wedge \ (m^*, R^*, \sigma^*) \notin \mathcal{Q}\mathsf{trace} \quad \mathbf{return}\ \beta'$

$\mathcal{O}\mathsf{sign}_{(pp,\tau_{\mathsf{SOK}},\mathsf{L},\mathcal{Q}\mathsf{sign},\mathbf{pk_S},\mathbf{sk_S})}(id, pk_{\mathsf{T}}, m, R)$

$\mathbf{if}\ \mathbf{pk_S}[id] \notin \mathsf{L} \quad \mathbf{return}\ \perp$
$\mathbf{parse}\ R\ \text{as}\ \{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\}$
$(pk_{\mathsf{Sign}}, sk_{\mathsf{Sign}}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}}); \ r_{1,1}, \ldots, r_{1,|R|}, r_2, r_3 \leftarrow\!\!\$\ \mathsf{Rand}$
$\mathbf{for}\ i = 1, \ldots, |R| \ : \ c_{1,i} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}_i}, sk_{\mathsf{Sign}}; r_{1,i})$
$\mathbf{c_1} \leftarrow (c_{1,1}, \ldots, c_{1,|R'|})$
$\rho \leftarrow\!\!\$\ \mathsf{NIZK.Prove}(pp, (pk_{\mathsf{Sign}}, (pk_{\mathsf{PKE}_1}, \ldots, pk_{\mathsf{PKE}_{|R|}}), \mathbf{c_1}), (r_{1,1}, \ldots, r_{1,|R|}, sk_{\mathsf{Sign}}))$
$c_2 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{T}}, \mathbf{pk_S}[id]; r_2); \ c_3 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{Sign}}, c_2; r_3)$
$\sigma_{\mathsf{SOK}} \leftarrow\!\!\$\ \mathsf{SoK.SimSign}(pp, \tau_{\mathsf{SOK}}, (pk_{\mathsf{T}}, pk_{\mathsf{Sign}}, R, c_3), m); \ \sigma \leftarrow\!\!\$\ (pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}})$
$\mathcal{Q}\mathsf{sign} \leftarrow \mathcal{Q}\mathsf{sign} \cup \{(pk_{\mathsf{T}}, \mathbf{pk_S}[id], m, R, \sigma)\}; \ \mathbf{return}\ \sigma$

Figure 6.5: Oracle $\mathcal{O}\mathsf{sign}$ and the adjusted experiment used in Game 2 in the proof of Theorem 22.

We show that Game 1 and Game 2 are indistinguishable if the SOK scheme satisfies simulatability. We give a distinguishing algorithm $\mathcal{D}_1$ in Figure 6.6. $\mathcal{D}_1$ aims to guess a bit β^* in the simulatability experiment and is given access to a signing oracle $\mathcal{O}\mathsf{SoK}$. On input of a statement, witness and message, oracle $\mathcal{O}\mathsf{SoK}$ returns, depending on bit β^*, either a signature of knowledge generated by algorithm $\mathsf{SoK.Sign}$, or a simulated signature of knowledge generated by algorithm $\mathsf{SoK.SimSign}$. We show that, when $\beta^* = 0$, inputs to $\mathcal{A}$ are identical to Game 1 and, when $\beta^* = 1$, inputs to $\mathcal{A}$ are identical to Game 1. Note that pp and pk_{T} input to $\mathcal{A}$ are honestly generated and are identical in Games 1 and 2. If $\beta^* = 0$, $\mathcal{D}_1$ is provided with a real signature of knowledge as in Game 1. If $\beta^* = 1$, the

signature of knowledge is simulated as in Game 2. Therefore,

$$|\Pr[S_1] - \Pr[S_2]| \leq \mathsf{negl}_{\mathsf{SOK\text{-}sim}}$$

where $\mathsf{negl}_{\mathsf{SOK\text{-}sim}}$ is the advantage in breaking the simulatability property of the SOK scheme.

$\mathcal{D}_1^{\mathcal{O}\mathsf{SoK}}(pp_{\mathsf{SOK}})$

$\beta \leftarrow\$ \{0,1\}$; $\mathbf{pk}_{\mathsf{S}} \leftarrow ()$; $\mathbf{sk}_{\mathsf{S}} \leftarrow ()$; $\mathsf{L} \leftarrow \emptyset$; $\mathsf{corL} \leftarrow \emptyset$; $\mathcal{Q}\mathsf{sign} \leftarrow \emptyset$; $\mathcal{Q}\mathsf{rep} \leftarrow \emptyset$; $\mathcal{Q}\mathsf{trace} \leftarrow \emptyset$
$pp_{\mathsf{PKE}} \leftarrow\$ \mathsf{PKE.Setup}(1^\lambda)$; $(pp_{\mathsf{NIZK}}, \tau_{\mathsf{NIZK}}) \leftarrow\$ \mathsf{NIZK.SimSetup}(1^\lambda)$; $pp \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}})$
$(pk_{\mathsf{T}}, sk_{\mathsf{T}}) \leftarrow\$ \mathsf{T.KGen}(pp)$
$(m^*, R^*, id_0, id_1, st) \leftarrow\$ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report},\mathcal{O}\mathsf{trace}}(pp, pk_{\mathsf{T}})$
parse R^* as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\}$
$R' \leftarrow R^* \cup \{(pk_{\mathsf{RS}_{id_0}}, pk_{\mathsf{PKE}_{id_0}}), (pk_{\mathsf{RS}_{id_1}}, pk_{\mathsf{PKE}_{id_1}})\}$; $pk_\beta \leftarrow\$ f(sk_{\mathsf{RS}_{id_\beta}})$
$(pk_{\mathsf{Sign}}, sk_{\mathsf{Sign}}) \leftarrow\$ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}})$; $r_{1,1}, \ldots, r_{1,|R'|}, r_2, r_3 \leftarrow\$ \mathsf{Rand}$
for $i = 1, \ldots, |R'|$: $c_{1,i} \leftarrow\$ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}_i}, sk_{\mathsf{Sign}}; r_{1,i})$
$\mathbf{c_1} \leftarrow (c_{1,1}, \ldots, c_{1,|R'|})$
$\rho \leftarrow\$ \mathsf{NIZK.Prove}(pp, (pk_{\mathsf{Sign}}, (pk_{\mathsf{PKE}_1}, \ldots, pk_{\mathsf{PKE}_{|R'|}}), \mathbf{c_1}), (r_{1,1}, \ldots, r_{1,|R'|}, sk_{\mathsf{Sign}}))$
$c_2 \leftarrow\$ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{T}}, pk_\beta; r_2)$; $c_3 \leftarrow\$ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{Sign}}, c_2; r_3)$
$\sigma_{\mathsf{SOK}} \leftarrow\$ \mathcal{O}\mathsf{SoK}((pk_{\mathsf{T}}, pk_{\mathsf{Sign}}, R', c_3), (r_2, r_3, sk_{\mathsf{RS}_{id_\beta}}), m^*)$; $\sigma^* \leftarrow (pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}})$
$\beta' \leftarrow\$ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report},\mathcal{O}\mathsf{trace}}(\sigma^*, st)$
if $\{\mathbf{pk}_{\mathsf{S}}[id_0], \mathbf{pk}_{\mathsf{S}}[id_1]\} \subseteq \mathsf{L} \backslash \mathsf{corL} \ \wedge \ (m^*, R^*, \sigma^*) \notin \mathcal{Q}\mathsf{trace} \ \wedge \ \beta' = \beta$ **return** 1

$\mathcal{O}\mathsf{sign}_{(pp,\mathsf{L},\mathcal{Q}\mathsf{sign},\mathbf{pk}_{\mathsf{S}},\mathbf{sk}_{\mathsf{S}})}(id, pk_{\mathsf{T}}, m, R)$

if $\mathbf{pk}_{\mathsf{S}}[id] \notin \mathsf{L}$ **return** $\perp$
parse R as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\} \ \wedge \ pp$ as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}})$
parse $\mathbf{pk}_{\mathsf{S}}[id]$ as $(sk_{\mathsf{RS}}, sk_{\mathsf{PKE}})$; $pk \leftarrow\$ f(sk_{\mathsf{RS}})$
$(pk_{\mathsf{Sign}}, sk_{\mathsf{Sign}}) \leftarrow\$ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}})$; $r_{1,1}, \ldots, r_{1,|R|}, r_2, r_3 \leftarrow\$ \mathsf{Rand}$
for $i = 1, \ldots, |R|$: $c_{1,i} \leftarrow\$ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}_i}, sk_{\mathsf{Sign}}; r_{1,i})$
$\mathbf{c_1} \leftarrow (c_{1,1}, \ldots, c_{1,|R|})$
$\rho \leftarrow\$ \mathsf{NIZK.Prove}(pp, (pk_{\mathsf{Sign}}, (pk_{\mathsf{PKE}_1}, \ldots, pk_{\mathsf{PKE}_{|R|}}), \mathbf{c_1}), (r_{1,1}, \ldots, r_{1,|R'|}, sk_{\mathsf{Sign}}))$
$c_2 \leftarrow\$ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{T}}, pk; r_2)$; $c_3 \leftarrow\$ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{Sign}}, c_2; r_3)$
$\sigma_{\mathsf{SOK}} \leftarrow\$ \mathcal{O}\mathsf{SoK}((pk_{\mathsf{T}}, pk_{\mathsf{Sign}}, R, c_3), (r_2, r_3, sk_{\mathsf{RS}}), m)$; $\sigma \leftarrow\$ (pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}})$
$\mathcal{Q}\mathsf{sign} \leftarrow \mathcal{Q}\mathsf{sign} \cup \{(pk_{\mathsf{T}}, \mathbf{pk}_{\mathsf{S}}[id], m, R, \sigma)\}$; **return** σ

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{report}/\mathcal{O}\mathsf{trace}$

As in Game 0.

Figure 6.6: Distinguisher $\mathcal{D}_1$ that distinguishes Game 1 and Game 2 in the proof of Theorem 22.

Game 3 is identical to Game 2 except for the following change, For queries to oracles $\mathcal{O}\mathsf{report}$ and $\mathcal{O}\mathsf{trace}$, and the challenge signature returned to adversary $\mathcal{A}$, we replace algorithm

NIZK.Prove with algorithm NIZK.SimProve. We define oracles $\mathcal{O}$report and $\mathcal{O}$trace and the adjusted experiment in Figure 6.7.

$\mathsf{Exp}^{\mathsf{anon},\beta}_{\mathsf{R\&T},\mathcal{A}}(\lambda)$

$\mathbf{pk}_\mathsf{S} \leftarrow ();\ \mathbf{sk}_\mathsf{S} \leftarrow ();\ \mathsf{L} \leftarrow \emptyset;\ \mathsf{corL} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{rep} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{trace} \leftarrow \emptyset$
$pp_\mathsf{PKE} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda);\ (pp_\mathsf{NIZK}, \tau_\mathsf{NIZK}) \leftarrow\!\!\$\ \mathsf{NIZK.SimSetup}(1^\lambda);\ (pp_\mathsf{SOK}, \tau_\mathsf{SOK}) \leftarrow\!\!\$\ \mathsf{SoK.SimSetup}(1^\lambda)$
$pp \leftarrow (pp_\mathsf{PKE}, pp_\mathsf{NIZK}, pp_\mathsf{SOK});\ (pk_\mathsf{T}, sk_\mathsf{T}) \leftarrow\!\!\$\ \mathsf{T.KGen}(pp)$
$(m^*, R^*, id_0, id_1, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report},\mathcal{O}\mathsf{trace}}(pp, pk_\mathsf{T})$
parse R^* as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\}$
$R' \leftarrow R^* \cup \{(pk_{\mathsf{RS}_{id_0}}, pk_{\mathsf{PKE}_{id_0}}), (pk_{\mathsf{RS}_{id_1}}, pk_{\mathsf{PKE}_{id_1}})\};\ pk_\beta \leftarrow\!\!\$\ f(sk_{\mathsf{RS}_{id_\beta}})$
$(pk_\mathsf{Sign}, sk_\mathsf{Sign}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_\mathsf{PKE});\ r_{1,1}, \ldots, r_{1,|R'|}, r_2, r_3 \leftarrow\!\!\$\ \mathsf{Rand}$
for $i = 1, \ldots, |R'|$: $c_{1,i} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_\mathsf{PKE}, pk_{\mathsf{PKE}_i}, sk_\mathsf{Sign}; r_{1,i})$
$\mathbf{c_1} \leftarrow (c_{1,1}, \ldots, c_{1,|R'|})$
$\rho \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}(pp, \tau_\mathsf{NIZK}, (pk_\mathsf{Sign}, (pk_{\mathsf{PKE}_1}, \ldots, pk_{\mathsf{PKE}_{|R'|}}), \mathbf{c_1}))$
$c_2 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_\mathsf{PKE}, pk_\mathsf{T}, pk_\beta; r_2);\ c_3 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_\mathsf{PKE}, pk_\mathsf{Sign}, c_2; r_3)$
$\sigma_\mathsf{SOK} \leftarrow\!\!\$\ \mathsf{SoK.SimSign}(pp, \tau_\mathsf{SOK}, (pk_\mathsf{T}, pk_\mathsf{Sign}, R', c_3), m^*);\ \sigma^* \leftarrow\!\!\$\ (pk_\mathsf{Sign}, \mathbf{c_1}, c_3, \rho, \sigma_\mathsf{SOK})$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report},\mathcal{O}\mathsf{trace}}(\sigma^*, st)$
if $\{\mathbf{pk}_\mathsf{S}[id_0], \mathbf{pk}_\mathsf{S}[id_1]\} \subseteq \mathsf{L} \backslash \mathsf{corL}\ \wedge\ (m^*, R^*, \sigma^*) \notin \mathcal{Q}\mathsf{trace}$ **return** β'

$\mathcal{O}\mathsf{report}_{(pp,\tau_\mathsf{NIZK},\mathsf{L},\mathcal{Q}\mathsf{rep},\mathbf{pk}_\mathsf{S},\mathbf{sk}_\mathsf{S})}(id, m, R, \sigma)$

if $\mathbf{pk}_\mathsf{S}[id] \notin R\ \vee\ \mathbf{pk}_\mathsf{S}[id] \notin \mathsf{L}$ **return** $\perp$
parse pp as $(pp_\mathsf{PKE}, pp_\mathsf{NIZK}, pp_\mathsf{SOK})\ \wedge\ \mathbf{sk}_\mathsf{S}[id]$ as $(sk_\mathsf{RS}, sk_\mathsf{PKE})$
$\wedge\ R$ as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\}$
$\wedge\ \sigma$ as $(pk_\mathsf{Sign}, \mathbf{c_1}, c_3, \rho, \sigma_\mathsf{SOK}) \wedge \mathbf{c_1}$ as $(c_{1,1}, \ldots, c_{1,|R|})$
if $\mathsf{Verify}(pp, pk_\mathsf{T}, m, R, \sigma) = 0$ **return** 0
for $i \in [|R|]$ s.t. $f(sk_\mathsf{PKE}) = pk_{\mathsf{PKE}_i}$
  $sk_\mathsf{Sign} \leftarrow\!\!\$\ \mathsf{PKE.Dec}(pp_\mathsf{PKE}, sk_\mathsf{PKE}, c_{1,i})$
$\rho_r \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}(pp, \tau_\mathsf{NIZK}, (R, \mathbf{c_1}, sk_\mathsf{Sign}))$
$\mathcal{Q}\mathsf{rep} \leftarrow \mathcal{Q}\mathsf{rep} \cup \{(\mathbf{pk}_\mathsf{S}[id], m, R, \sigma)\}$
return $\mathtt{Rep} = (sk_\mathsf{Sign}, \rho_r)$

$\mathcal{O}\mathsf{trace}_{(pp,\tau_\mathsf{NIZK},sk_\mathsf{T},\mathcal{Q}\mathsf{trace})}(m, R, \sigma, \mathtt{Rep})$

parse pp as $(pp_\mathsf{PKE}, pp_\mathsf{NIZK}, pp_\mathsf{SOK})\ \wedge\ \sigma$ as $(pk_\mathsf{Sign}, \mathbf{c_1}, c_3, \rho, \sigma_\mathsf{SOK})$
$\wedge\ \mathtt{Rep}$ as $(sk_\mathsf{Sign}, \rho_r)$
$pk_\mathsf{T} \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_\mathsf{PKE}; sk_\mathsf{T})$
if $\mathsf{Verify}(pp, pk_\mathsf{T}, m, R, \sigma) = 0$ **return** 0
if $\mathsf{NIZK.Verify}(pp, (R, \mathbf{c_1}, sk_\mathsf{Sign}), sk_\mathsf{PKE}) = 0$ **return** 0
$c_2 \leftarrow\!\!\$\ \mathsf{PKE.Dec}(pp_\mathsf{PKE}, sk_\mathsf{Sign}, c_3)$
$pk \leftarrow\!\!\$\ \mathsf{PKE.Dec}(pp_\mathsf{PKE}, sk_\mathsf{T}, c_2)$
$\rho_t \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}(pp, \tau_\mathsf{NIZK}, (pk_\mathsf{T}, c_2, pk))$
$\mathcal{Q}\mathsf{trace} \leftarrow \mathcal{Q}\mathsf{trace} \cup \{(m, R, \sigma)\}$
return $(pk, \mathtt{Tr} = (c_2, \mathtt{Rep}), \rho_t)$

Figure 6.7: Oracles $\mathcal{O}$report and $\mathcal{O}$trace, and the adjusted experiment used in Game 3 in the proof of Theorem 22.

We show that Game 2 and Game 3 are indistinguishable if the NIZK scheme satisfies zero-knowledge. We give a distinguishing algorithm $\mathcal{D}_2$ in Figure 6.8. $\mathcal{D}_2$ aims to guess a bit β^* in the zero-knowledge experiment and has access to a proving oracle $\mathcal{O}$Prove. On input of a statement and a witness, oracle $\mathcal{O}$Prove returns either a proof output by algorithm NIZK.Prove or the output of algorithm NIZK.SimProve. We show that, when $\beta^* = 0$, inputs to $\mathcal{A}$ are identical to Game 1 and, when $\beta^* = 1$, inputs to $\mathcal{A}$ are identical to Game 1. Note that pp and pk_T input to $\mathcal{A}$ are honestly generated and are identical in Games 1 and 2. When $\beta^* = 0$, $\mathcal{D}_2$ is provided with a real NIZK proof as in Game 2. If $\beta^* = 1$, the proof is simulated as in Game 3. Therefore,

$$|\Pr[S_2] - \Pr[S_3]| \leq \mathsf{negl}_{\mathsf{NIZK\text{-}zk}}$$

where $\mathsf{negl}_{\mathsf{NIZK\text{-}zk}}$ is the advantage in breaking the zero-knowledge property of the NIZK scheme.

$\mathcal{D}_2^{\mathcal{O}\mathsf{Prove}}(pp_{\mathsf{NIZK}})$

$\mathbf{pk}_{\mathsf{S}} \leftarrow ();\ \mathbf{sk}_{\mathsf{S}} \leftarrow ();\ \mathsf{L} \leftarrow \emptyset;\ \mathsf{corL} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{rep} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{trace} \leftarrow \emptyset$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda);\ (pp_{\mathsf{SOK}}, \tau_{\mathsf{SOK}}) \leftarrow\!\!\$\ \mathsf{SoK.SimSetup}(1^\lambda)$
$pp \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}});\ (pk_{\mathsf{T}}, sk_{\mathsf{T}}) \leftarrow\!\!\$\ \mathsf{T.KGen}(pp)$
$(m^*, R^*, id_0, id_1, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report},\mathcal{O}\mathsf{trace}}(pp, pk_{\mathsf{T}})$
parse R^* as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\}$
$R' \leftarrow R^* \cup \{(pk_{\mathsf{RS}_{id_0}}, pk_{\mathsf{PKE}_{id_0}}), (pk_{\mathsf{RS}_{id_1}}, pk_{\mathsf{PKE}_{id_1}})\};\ pk_\beta \leftarrow\!\!\$\ f(sk_{\mathsf{RS}_{id_\beta}})$
$(pk_{\mathsf{Sign}}, sk_{\mathsf{Sign}}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}});\ r_{1,1}, \ldots, r_{1,|R'|}, r_2, r_3 \leftarrow\!\!\$\ \mathsf{Rand}$
for $i = 1, \ldots, |R'|$: $c_{1,i} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}_i}, sk_{\mathsf{Sign}}; r_{1,i})$
$\mathbf{c_1} \leftarrow (c_{1,1}, \ldots, c_{1,|R'|})$
$\rho \leftarrow\!\!\$\ \mathcal{O}\mathsf{Prove}((pk_{\mathsf{Sign}}, (pk_{\mathsf{PKE}_1}, \ldots, pk_{\mathsf{PKE}_{|R'|}}), \mathbf{c_1}), (r_{1,1}, \ldots, r_{1,|R'|}, sk_{\mathsf{Sign}}))$
$c_2 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{T}}, pk_\beta; r_2);\ c_3 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{Sign}}, c_2; r_3)$
$\sigma_{\mathsf{SOK}} \leftarrow\!\!\$\ \mathsf{SoK.SimSign}(pp, \tau_{\mathsf{SOK}}, (pk_{\mathsf{T}}, pk_{\mathsf{Sign}}, R', c_3), m^*);\ \sigma^* \leftarrow\!\!\$\ (pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}})$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report},\mathcal{O}\mathsf{trace}}(\sigma^*, st)$
if $\{\mathbf{pk}_{\mathsf{S}}[id_0], \mathbf{pk}_{\mathsf{S}}[id_1]\} \subseteq \mathsf{L} \backslash \mathsf{corL} \ \wedge\ (m^*, R^*, \sigma^*) \notin \mathcal{Q}\mathsf{trace}$ **return** β'

$\mathcal{O}\mathsf{report}_{(pp,\mathsf{L},\mathcal{Q}\mathsf{rep},\mathbf{pk}_{\mathsf{S}},\mathbf{sk}_{\mathsf{S}})}(id, m, R, \sigma)$

if $\mathbf{pk}_{\mathsf{S}}[id] \notin R \ \vee\ \mathbf{pk}_{\mathsf{S}}[id] \notin \mathsf{L}$ **return** $\perp$
$\mathtt{Rep} \leftarrow\!\!\$\ \mathsf{Report}(pp, pk_{\mathsf{T}}, \mathbf{sk}_{\mathsf{S}}[id], m, R, \sigma)$
parse pp as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}}) \ \wedge\ \mathbf{sk}_{\mathsf{S}}[id]$ as $(sk_{\mathsf{RS}}, sk_{\mathsf{PKE}})$
$\wedge\ R$ as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\}$
$\wedge\ \sigma$ as $(pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}}) \wedge \mathbf{c_1}$ as $(c_{1,1}, \ldots, c_{1,|R|})$
if $\mathsf{Verify}(pp, pk_{\mathsf{T}}, m, R, \sigma) = 0$ **return** 0
for $i \in 1, \ldots, |R|$ s.t. $f(sk_{\mathsf{PKE}}) = pk_{\mathsf{PKE}_i}$
  $sk_{\mathsf{Sign}} \leftarrow\!\!\$\ \mathsf{PKE.Dec}(pp_{\mathsf{PKE}}, sk_{\mathsf{PKE}}, c_{1,i})$
$\rho_r \leftarrow\!\!\$\ \mathcal{O}\mathsf{Prove}((R, \mathbf{c_1}, sk_{\mathsf{Sign}}), sk_{\mathsf{PKE}})$
$\mathcal{Q}\mathsf{rep} \leftarrow \mathcal{Q}\mathsf{rep} \cup \{(\mathbf{pk}_{\mathsf{S}}[id], m, R, \sigma)\}$
return $\mathtt{Rep} = (sk_{\mathsf{Sign}}, \rho_r)$

$\mathcal{O}\mathsf{trace}_{(pp,sk_{\mathsf{T}},\mathcal{Q}\mathsf{trace})}(m, R, \sigma, \mathtt{Rep})$

parse pp as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}}) \ \wedge\ \sigma$ as $(pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}})$
$\wedge\ \mathtt{Rep}$ as $(sk_{\mathsf{Sign}}, \rho_r)$
$pk_{\mathsf{T}} \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}}; sk_{\mathsf{T}})$
if $\mathsf{Verify}(pp, pk_{\mathsf{T}}, m, R, \sigma) = 0$ **return** 0
if $\mathsf{NIZK.Verify}(pp, (R, \mathbf{c_1}, sk_{\mathsf{Sign}}), sk_{\mathsf{PKE}}) = 0$ **return** 0
$c_2 \leftarrow\!\!\$\ \mathsf{PKE.Dec}(pp_{\mathsf{PKE}}, sk_{\mathsf{Sign}}, c_3)$
$pk \leftarrow\!\!\$\ \mathsf{PKE.Dec}(pp_{\mathsf{PKE}}, sk_{\mathsf{T}}, c_2)$
$\rho_t \leftarrow\!\!\$\ \mathcal{O}\mathsf{Prove}((pk_{\mathsf{T}}, c_2, pk), sk_{\mathsf{T}})$
$(pk_{\mathsf{S}}, \mathtt{Tr}, \rho_t) \leftarrow\!\!\$\ \mathsf{Trace}(pp, sk_{\mathsf{T}}, m, R, \sigma, \mathtt{Rep})$
$\mathcal{Q}\mathsf{trace} \leftarrow \mathcal{Q}\mathsf{trace} \cup \{(m, R, \sigma)\}$
return $(pk, \mathtt{Tr} = (c_2, \mathtt{Rep}), \rho_t)$

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}$

As in Game 0.

$\mathcal{O}\mathsf{sign}$

As in Game 2.

Figure 6.8: Distinguisher $\mathcal{D}_2$ that distinguishes Game 2 and Game 3 in the proof of Theorem 22.

Game 4 requires a further change to $\mathcal{O}\mathsf{report}$. Rather than decrypting a ciphertext from $\mathbf{c_1}$, we run algorithm $\mathsf{NIZK.Extract}$ to output the witness $(r_{1,1}, \ldots, r_{1,|R|}, sk_{\mathsf{Sign}})$ from proof of correct encryption ρ. We define oracle $\mathcal{O}\mathsf{report}$ in Figure 6.9. If the NIZK scheme satisfies knowledge extractability then, with overwhelming probability, the witness output by algorithm $\mathsf{NIZK.Extract}$ is identical to the input to algorithm $\mathsf{PKE.Enc}$ to generate vector of ciphertexts $\mathbf{c_1}$. As such, the view of the adversary is identical following this game hop, unless the NIZK scheme does not satisfy knowledge extractability. Therefore,

$$|\Pr[S_3] - \Pr[S_4]| \leq k_1 \cdot \mathsf{negl}_{\mathsf{NIZK\text{-}Ext}}$$

where $\mathsf{negl}_{\mathsf{NIZK\text{-}Ext}}$ is the probability that the NIZK scheme does not satisfy knowledge extractability.

$\mathcal{O}\mathsf{report}_{(pp,\tau_{\mathsf{NIZK}},\mathsf{L},\mathcal{Q}\mathsf{rep},\mathbf{pk}_{\mathsf{S}},\mathbf{sk}_{\mathsf{S}})}(id, m, R, \sigma)$

if $\mathbf{pk}_{\mathsf{S}}[id] \notin R \ \vee \ \mathbf{pk}_{\mathsf{S}}[id] \notin \mathsf{L}$ **return** $\perp$
parse pp as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}}) \ \wedge \ \mathbf{sk}_{\mathsf{S}}[id]$ as $(sk_{\mathsf{RS}}, sk_{\mathsf{PKE}})$
$\wedge \ R$ as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\} \ \wedge \ \sigma$ as $(pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}}) \ \wedge \ \mathbf{c_1}$ as $(c_{1,1}, \ldots, c_{1,|R|})$
if $\mathsf{Verify}(pp, pk_{\mathsf{T}}, m, R, \sigma) = 0$ **return** 0
$(r_{1,1}, \ldots, r_{1,|R|}, sk_{\mathsf{Sign}}) \leftarrow\!\!\$ \ \mathsf{NIZK.Extract}(pp, \tau_{\mathsf{NIZK}}, (pk_{\mathsf{Sign}}, (pk_{\mathsf{PKE}_1}, \ldots, pk_{\mathsf{PKE}_{|R|}}), \mathbf{c_1}))$
$\rho_r \leftarrow\!\!\$ \ \mathsf{NIZK.SimProve}(pp, \tau_{\mathsf{NIZK}}, (R, \mathbf{c_1}, sk_{\mathsf{Sign}}))$
$\mathcal{Q}\mathsf{rep} \leftarrow \mathcal{Q}\mathsf{rep} \cup \{(\mathbf{pk}_{\mathsf{S}}[id], m, R, \sigma)\}$
return $\mathtt{Rep} = (sk_{\mathsf{Sign}}, \rho_r)$

Figure 6.9: Oracle $\mathcal{O}\mathsf{report}$ used in Game 4 in the proof of Theorem 22.

Game 5 requires a further change to $\mathcal{O}\mathsf{trace}$. Rather than decrypting ciphertexts c_2 and c_3, we run algorithm $\mathsf{SoK.Extract}$ to output witness $(r_2, r_3, sk_{\mathsf{RS}})$ and obtain the identity of the signer. We define oracle $\mathcal{O}\mathsf{trace}$ in Figure 6.10. If the SOK scheme satisfies extractability then, with overwhelming probability, the witness output by algorithm $\mathsf{SoK.Extract}$ is identical to the input to algorithm $\mathsf{PKE.Enc}$ to generate ciphertext c_3. As such, the view of the adversary is identical following this game hop, unless the SOK scheme does not satisfy extractability. Therefore,

$$|\Pr[S_4] - \Pr[S_5]| \leq k_2 \cdot \mathsf{negl}_{\mathsf{SOK\text{-}Ext}}$$

where $\mathsf{negl}_{\mathsf{SOK\text{-}Ext}}$ is the probability that the SOK scheme does not satisfy extractability.

$\mathcal{O}\mathsf{trace}_{(pp,\tau_{\mathsf{NIZK}},\tau_{\mathsf{SOK}},sk_{\mathsf{T}},\mathcal{Q}\mathsf{trace})}(m, R, \sigma, \mathtt{Rep})$

parse pp as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}}) \ \wedge \ \sigma$ as $(pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}}) \ \wedge \ \mathtt{Rep}$ as $(sk_{\mathsf{Sign}}, \rho_r)$
$pk_{\mathsf{T}} \leftarrow\!\!\$ \ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}}; sk_{\mathsf{T}})$
if $\mathsf{Verify}(pp, pk_{\mathsf{T}}, m, R, \sigma) = 0$ **return** 0
if $\mathsf{NIZK.Verify}(pp, (R, \mathbf{c_1}, sk_{\mathsf{Sign}}), sk_{\mathsf{PKE}}) = 0$ **return** 0
$(r_2, r_3, sk_{\mathsf{RS}}) \leftarrow\!\!\$ \ \mathsf{SoK.Extract}(pp, \tau_{\mathsf{SOK}}, (pk_{\mathsf{T}}, pk_{\mathsf{sign}}, R, c_3), m, \sigma_{\mathsf{SOK}})$
$pk \leftarrow\!\!\$ \ f(sk_{\mathsf{RS}}); \ \rho_t \leftarrow\!\!\$ \ \mathsf{NIZK.SimProve}(pp_{\mathsf{NIZK}}, \tau_{\mathsf{NIZK}}, (pk_{\mathsf{T}}, c_2, pk))$
$\mathcal{Q}\mathsf{trace} \leftarrow \mathcal{Q}\mathsf{trace} \cup \{(m, R, \sigma)\}$; **return** $(pk, \mathtt{Tr} = (c_2, \mathtt{Rep}), \rho_t)$

Figure 6.10: Oracle $\mathcal{O}\mathsf{trace}$ used in Game 5 in the proof of Theorem 22.

Game 6 requires a change to the challenge signature returned to $\mathcal{A}$ when $\beta = 1$. Rather than encrypting the public key of signer id_1 when $\beta = 1$, we encrypt the public key of signer id_0. We show that Game 5 and Game 6 are indistinguishable if the PKE scheme satisfies the $\mathsf{IND\text{-}CPA}$ property. We define the modified anonymity experiment in Figure 6.11.

We give a distinguishing algorithm $\mathcal{D}_3$ in Figure 6.12. $\mathcal{D}_3$ aims to guess a bit β^* in the $\mathsf{IND\text{-}CPA}$ experiment and is given access to an encryption oracle $\mathcal{O}\mathsf{enc}$. When $\beta^* = 0$,

$\mathsf{Exp}^{\mathsf{anon},\beta}_{\mathsf{R\&T},\mathcal{A}}(\lambda)$

$\mathbf{pk}_{\mathsf{S}} \leftarrow (); \ \mathbf{sk}_{\mathsf{S}} \leftarrow (); \ \mathsf{L} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{rep} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{trace} \leftarrow \emptyset$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda); \ (pp_{\mathsf{NIZK}}, \tau_{\mathsf{NIZK}}) \leftarrow\!\!\$\ \mathsf{NIZK.SimSetup}(1^\lambda); \ (pp_{\mathsf{SOK}}, \tau_{\mathsf{SOK}}) \leftarrow\!\!\$\ \mathsf{SoK.SimSetup}(1^\lambda)$
$pp \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}}); \ (pk_{\mathsf{T}}, sk_{\mathsf{T}}) \leftarrow\!\!\$\ \mathsf{T.KGen}(pp)$
$(m^*, R^*, id_0, id_1, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report},\mathcal{O}\mathsf{trace}}(pp, pk_{\mathsf{T}})$
parse R^* as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\}$
$R' \leftarrow R^* \cup \{(pk_{\mathsf{RS}_{id_0}}, pk_{\mathsf{PKE}_{id_0}}), (pk_{\mathsf{RS}_{id_1}}, pk_{\mathsf{PKE}_{id_1}})\}; \ pk_0 \leftarrow\!\!\$\ f(sk_{\mathsf{RS}_{id_0}})$
$(pk_{\mathsf{Sign}}, sk_{\mathsf{Sign}}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}}); \ r_{1,1}, \ldots, r_{1,|R'|}, r_2, r_3 \leftarrow\!\!\$\ \mathsf{Rand}$
for $i = 1, \ldots, |R'|$: $c_{1,i} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}_i}, sk_{\mathsf{Sign}}; r_{1,i})$
$\mathbf{c_1} \leftarrow (c_{1,1}, \ldots, c_{1,|R'|})$
$\rho \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}(pp, \tau_{\mathsf{NIZK}}, (pk_{\mathsf{Sign}}, (pk_{\mathsf{PKE}_1}, \ldots, pk_{\mathsf{PKE}_{|R'|}}), \mathbf{c_1}))$
$c_2 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{T}}, pk_0; r_2); \ c_3 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{Sign}}, c_2; r_3)$
$\sigma_{\mathsf{SOK}} \leftarrow\!\!\$\ \mathsf{SoK.SimSign}(pp, \tau_{\mathsf{SOK}}, (pk_{\mathsf{T}}, pk_{\mathsf{Sign}}, R', c_3), m^*); \ \sigma^* \leftarrow\!\!\$\ (pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}})$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report},\mathcal{O}\mathsf{trace}}(\sigma^*, st)$
if $\{\mathbf{pk}_{\mathsf{S}}[id_0], \mathbf{pk}_{\mathsf{S}}[id_1]\} \subseteq \mathsf{L} \backslash \mathsf{corL} \ \wedge \ (m^*, R^*, \sigma^*) \notin \mathcal{Q}\mathsf{trace}$ **return** β'

Figure 6.11: The modified experiment for Game 6 in the proof of Theorem 22.

inputs to $\mathcal{A}$ are identical to Game 5 and, when $\beta^* = 1$ inputs to $\mathcal{A}$ are identical to Game 6. Indeed, pp and pk_{T} input to $\mathcal{A}$ are identical in Games 5 and 6, and sk_{T} is not known but is never used. If $\beta = 1$ and $\beta^* = 0$, $\mathcal{D}_5$ is input an encryption of pk_1 as in Game 5. If $\beta = 1$ and $\beta^* = 1$, $\mathcal{D}_5$ is input an encryption of pk_0 as in Game 6. Therefore,

$$|\Pr[S_5] - \Pr[S_6]| \leq \mathsf{negl}_{\mathsf{IND\text{-}CPA}}$$

where $\mathsf{negl}_{\mathsf{IND\text{-}CPA}}$ is the advantage in breaking the IND-CPA property of the PKE scheme.

In Game 6, the inputs to $\mathcal{A}$ are identical for $\beta = 0$ and $\beta = 1$. Therefore, $\Pr[S_6] = 1/2$. We now have that

$$|\Pr[S_0] - 1/2| \leq \mathsf{negl}_{\mathsf{SOK\text{-}Sim}} + \mathsf{negl}_{\mathsf{NIZK\text{-}zk}} + k_1 \cdot \mathsf{negl}_{\mathsf{NIZK\text{-}Ext}} + k_2 \cdot \mathsf{negl}_{\mathsf{SOK\text{-}Ext}} + \mathsf{negl}_{\mathsf{IND\text{-}CPA}}$$

and conclude that the advantage of the adversary in Game 0 is negligible as required. □

Theorem 23. *Our construction satisfies* unforgeability *if f is a one-way function and signature of knowledge* SOK *satisfies extractability.*

Proof. Let $\mathcal{A}$ be an adversary that succeeds in the unforgeability experiment. Then, $\mathcal{A}$ can

$\mathcal{D}_3^{\mathcal{O}\mathsf{enc}}(pp_{\mathsf{PKE}}, pk_{\mathsf{T}})$

$\mathbf{pk}_{\mathsf{S}} \leftarrow (); \ \mathbf{sk}_{\mathsf{S}} \leftarrow (); \ \mathsf{L} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{rep} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{trace} \leftarrow \emptyset$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda); \ (pp_{\mathsf{NIZK}}, \tau_{\mathsf{NIZK}}) \leftarrow\!\!\$\ \mathsf{NIZK.SimSetup}(1^\lambda); \ (pp_{\mathsf{SOK}}, \tau_{\mathsf{SOK}}) \leftarrow\!\!\$\ \mathsf{SoK.SimSetup}(1^\lambda)$
$pp \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}})$
$(m^*, R^*, id_0, id_1, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report},\mathcal{O}\mathsf{trace}}(pp, pk_{\mathsf{T}})$
parse R^* as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\}$
$R' \leftarrow R^* \cup \{(pk_{\mathsf{RS}_{id_0}}, pk_{\mathsf{PKE}_{id_0}}), (pk_{\mathsf{RS}_{id_1}}, pk_{\mathsf{PKE}_{id_1}})\}; \ pk_0 \leftarrow\!\!\$\ f(sk_{\mathsf{RS}_{id_0}})$
$(pk_{\mathsf{Sign}}, sk_{\mathsf{Sign}}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}}); \ r_{1,1}, \ldots, r_{1,|R'|}, r_2, r_3 \leftarrow\!\!\$\ \mathsf{Rand}$
for $i = 1, \ldots, |R'| \ : \ c_{1,i} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}_i}, sk_{\mathsf{Sign}}; r_{1,i})$
$\mathbf{c_1} \leftarrow (c_{1,1}, \ldots, c_{1,|R'|})$
$\rho \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}(pp, \tau_{\mathsf{NIZK}}, (pk_{\mathsf{Sign}}, (pk_{\mathsf{PKE}_1}, \ldots, pk_{\mathsf{PKE}_{|R'|}}), \mathbf{c_1}))$
if $\beta = 0 \ : \ c_2 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_0; r_2)$
if $\beta = 1 \ : \ c_2 \leftarrow\!\!\$\ \mathcal{O}\mathsf{enc}(pk_1, pk_0)$
$c_3 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{Sign}}, c_2; r_3)$
$\sigma_{\mathsf{SOK}} \leftarrow\!\!\$\ \mathsf{SoK.SimSign}(pp, \tau_{\mathsf{SOK}}, (pk_{\mathsf{T}}, pk_{\mathsf{Sign}}, R', c_3), m^*); \ \sigma^* \leftarrow\!\!\$\ (pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}})$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report},\mathcal{O}\mathsf{trace}}(\sigma^*, st)$
if $\{\mathbf{pk}_{\mathsf{S}}[id_0], \mathbf{pk}_{\mathsf{S}}[id_1]\} \subseteq \mathsf{L} \backslash \mathsf{corL} \ \wedge \ (m^*, R^*, \sigma^*) \notin \mathcal{Q}\mathsf{trace}$ **return** β'

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}$	$\mathcal{O}\mathsf{sign}$	$\mathcal{O}\mathsf{report}$	$\mathcal{O}\mathsf{trace}$
As in Game 0	As in Game 2	As in Game 4	As in Game 5

Figure 6.12: Distinguisher $\mathcal{D}_3$ that distinguishes between Game 5 and Game 6 in the proof of Theorem 22.

output a tuple $(pk_{\mathsf{T}}, m^*, R^*, \sigma^*)$ such that $\mathsf{Verify}(pp, pk_{\mathsf{T}}, m^*, R^*, \sigma^*) = 1$ where the ring is honest and the tuple output is not queried to $\mathcal{O}\mathsf{sign}$. Hence, $\mathcal{A}$ can output a valid signature for an honest ring member. On behalf of an honest ring member, $\mathcal{A}$ can perform all steps of Sign *except* produce signature of knowledge σ_{SOK}. For this, $\mathcal{A}$ must obtain the secret credential of the signer, or compute a signature of knowledge without a valid witness.

Let Cred denote the event that $\mathcal{A}$ can obtain the secret credential of the signer and let $\mathsf{Success}$ denote the event that the unforgeability experiment returns 1. Note that,

$$\begin{aligned}\Pr\left[\mathsf{Exp}^{\mathsf{unfor}}_{\mathsf{R\&T},\mathcal{A}}(\lambda) = 1\right] &= \Pr[\mathsf{Success} \wedge \mathsf{Cred}] + \Pr\left[\mathsf{Success} \wedge \overline{\mathsf{Cred}}\right] \\ &\leq \Pr[\mathsf{Cred}] + \Pr\left[\mathsf{Success} \wedge \overline{\mathsf{Cred}}\right]\end{aligned}$$

We first note that $\Pr[\mathsf{Cred}] \leq \mathsf{negl}(\lambda)$. Indeed, this is equivalent to breaking the one-wayness property of function f.

We now show that $\Pr\left[\mathsf{Success} \wedge \overline{\mathsf{Cred}}\right] \leq \mathsf{negl}(\lambda)$. We show that we can construct an adversary $\mathcal{B}$ that succeeds in the extractability experiment for the SOK scheme. We present $\mathcal{B}$ in Figure 6.13 that has access to a signing oracle that, on input of a statement, witness and message, returns a signature generated by algorithm SoK.SimSign. It is clear that inputs to $\mathcal{A}$ are distributed identically to the extractability experiment because public parameters are generated identically, and oracles are distributed identically. We now show that $\mathcal{B}$ is successful in the extractability experiment. That is, we show that $((pk_{\mathsf{T}}, pk_{\mathsf{Sign}}, R, c_3), m, \sigma_{\mathsf{SOK}})$ is a valid statement/message/signature tuple for which a witness can be extracted that has not been input to oracle $\mathcal{O}$. By assumption, algorithm Verify returns 1. Then, by definition, algorithm $\mathsf{SoK.Verify}(pp, (pk_{\mathsf{T}}, pk_{\mathsf{Sign}}, R, c_3), m, \sigma_{\mathsf{SOK}}) = 1$. Moreover, $(pk_{\mathsf{T}}, \mathbf{pk_S}[id], m, R, \sigma) \notin \mathcal{Q}\mathsf{sign}$ and, therefore, $((pk_{\mathsf{T}}, pk_{\mathsf{Sign}}, R, c_3), m, \sigma_{\mathsf{SOK}})$ is not input to oracle $\mathcal{O}$. Therefore, $\Pr\left[\mathsf{Success} \wedge \overline{\mathsf{Cred}}\right] \leq \mathsf{negl}(\lambda)$ as required and the result holds. □

$\mathcal{B}^{\mathcal{O}}(pp_{\mathsf{SOK}})$

$\mathbf{pk_S} \leftarrow (); \ \mathbf{sk_S} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{rep} \leftarrow \emptyset$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$pp_{\mathsf{NIZK}} \leftarrow\!\!\$\ \mathsf{NIZK.Setup}(1^\lambda)$
$pp \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}})$
$(pk_{\mathsf{T}}, m^*, R^*, \sigma^*) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{reg}, \mathcal{O}\mathsf{corrupt}, \mathcal{O}\mathsf{sign}, \mathcal{O}\mathsf{report}}(pp)$
parse σ^* as $(pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}})$
if $\mathsf{Verify}(pp, pk_{\mathsf{T}}, m^*, R^*, \sigma^*) = 1 \ \wedge \ R^* \subseteq \mathsf{L} \backslash \mathsf{corL}$
$\wedge \ (pk_{\mathsf{T}}, \cdot, m^*, R^*, \sigma^*) \notin \mathcal{Q}\mathsf{sign}$
  return $((pk_{\mathsf{T}}, pk_{\mathsf{Sign}}, R, c_3), m, \sigma_{\mathsf{SOK}})$

$\mathcal{O}\mathsf{sign}_{(pp, \mathsf{L}, \mathcal{Q}\mathsf{sign}, \mathbf{pk_S}, \mathbf{sk_S})}(id, pk_{\mathsf{T}}, m, R)$

if $\mathbf{pk_S}[id] \notin \mathsf{L}$ **return** $\perp$
parse pp as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}}) \ \wedge \ sk_{id}$ as $(sk_{\mathsf{RS}}, sk_{\mathsf{PKE}})$
$\wedge \ R$ as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\}$
$pk \leftarrow\!\!\$\ f(sk_{\mathsf{RS}})$
$(pk_{\mathsf{Sign}}, sk_{\mathsf{Sign}}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(pp_{\mathsf{PKE}})$
$r_{1,1}, \ldots, r_{1,|R|}, r_2, r_3 \leftarrow\!\!\$\ \mathsf{Rand}$
for $i = 1, \ldots, |R| \ : \ c_{1,i} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}_i}, sk_{\mathsf{Sign}}; r_{1,i})$
$\mathbf{c_1} \leftarrow (c_{1,1}, \ldots, c_{1,|R|})$
$\rho \leftarrow\!\!\$\ \mathsf{NIZK.Prove}(pp, (pk_{\mathsf{Sign}}, (pk_{\mathsf{PKE}_1}, \ldots, pk_{\mathsf{PKE}_{|R|}}), \mathbf{c_1}), (r_{1,1}, \ldots, r_{1,|R|}, sk_{\mathsf{Sign}}))$
$c_2 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{T}}, pk; r_2)$
$c_3 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{Sign}}, c_2; r_3)$
$\sigma_{\mathsf{SOK}} \leftarrow\!\!\$\ \mathcal{O}((pk_{\mathsf{T}}, pk_{\mathsf{Sign}}, R, c_3), (r_2, r_3, sk_{\mathsf{RS}}), m)$
$\sigma \leftarrow (pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}})$
$\mathcal{Q}\mathsf{sign} \leftarrow \mathcal{Q}\mathsf{sign} \cup \{(pk_{\mathsf{T}}, \mathbf{pk_S}[id], m, R, \sigma)\}$
return σ

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{sign}/\mathcal{O}\mathsf{report}/\mathcal{O}\mathsf{trace}$

As normal.

Figure 6.13: Adversary $\mathcal{B}$ that succeeds in the extractability property of signature of knowledge SOK.

Theorem 24. *Our construction satisfies* traceability. *That is:*

1. *Our construction satisfies* tracing correctness *if public-key encryption scheme* PKE *is correct, non-interactive zero-knowledge proof system* NIZK *is complete, and signature of knowledge* SOK *is correct.*

2. *Our construction satisfies* non-frameability *if f is a one-way function, public-key encryption scheme* PKE *is correct, non-interactive zero-knowledge proof system* NIZK *satisfies soundness, and signature of knowledge* SOK *satisfies extractability.*

3. *Our construction satisfies* soundness *if public-key encryption scheme* PKE *is correct and non-interactive zero-knowledge proof system* NIZK *satisfies soundness.*

Proof. We show that all three properties of traceability are satisfied.

Tracing correctness: Consider a signature $\sigma = (pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}})$ output by algorithm Sign with respect to a message $m \in \mathsf{M}$ and ring R that consists of public keys generated by algorithm $\mathsf{U.KGen}$. Let pp be the output of algorithm Setup and let $(pk_{\mathsf{T}}, sk_{\mathsf{T}})$ be the output of algorithm $\mathsf{T.KGen}$. Let $pk = f(sk_{\mathsf{RS}})$ where $(pk, \cdot) \in R$ and sk_{RS} is an element of SK and is used to sign message m. Moreover, let $\mathtt{Rep} = (sk_{\mathsf{Sign}}, \rho_r)$ and $(pk, \mathtt{Tr} = (c_2, \mathtt{Rep}), \rho_t)$ be the outputs of algorithms Report and Trace respectively.

By definition of tracing correctness, our construction satisfies tracing correctness if $\mathsf{VerTrace}(pp, pk_{\mathsf{T}}, m, R, \sigma, pk, \mathtt{Tr}, \rho_t)$ outputs 1 with overwhelming probability. Assume that algorithm $\mathsf{VerTrace}$ does not return 1. Then, it must be the case that $\mathsf{NIZK.Verify}(pp, (pk_{\mathsf{T}}, c_2, pk), \rho_t) = 0$, $\mathsf{NIZK.Verify}(pp, (R, \mathbf{c_1}, sk_{\mathsf{Sign}}), \rho_r) = 0$, $\mathsf{Verify}(pp, pk_{\mathsf{T}}, m, R, \sigma) = 0$ or $\mathsf{PKE.Dec}(pp_{\mathsf{PKE}}, sk_{\mathsf{Sign}}, c_3) \neq c_2$. We consider these possibilities in turn. We conclude that, if all building blocks are correct, our construction satisfies tracing correctness.

First, recall that, by correctness of the construction, algorithm Verify returns 1 with overwhelming probability unless one of the building blocks do not satisfy correctness. Second, as in our proof of correctness, algorithm $\mathsf{NIZK.Verify}$ returns 1 with overwhelming probability if the NIZK scheme satisfies completeness and the PKE scheme is correct. Finally, by correctness of the PKE scheme, $c_2 = \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{T}}, c_3)$ with overwhelming probability, and the result holds.

Non-frameability: Let $\mathcal{A}$ be an adversary that succeeds in the non-frameability experiment. Then $\mathcal{A}$ can output a tuple $(pk_{\mathsf{T}}, m^*, R^*, \sigma^*, pk_{\mathsf{RS}}, \mathtt{Tr} = (c_2, \mathtt{Rep}), \rho_t)$ such that algorithm $\mathsf{VerTrace}$ output 1 where it is assumed that pk_{RS} is honest and the tuple output by $\mathcal{A}$ is not queried to $\mathcal{O}\mathsf{sign}$. Futhermore, algorithm Verify must return 1. That is, to succeed, $\mathcal{A}$ must output a valid signature, which may be generated by $\mathcal{A}$ on behalf of an honest or corrupt ring member. Let Forge denote the event that $\mathcal{A}$ outputs a valid signature on behalf of an honest ring member and let $\mathsf{Success}$ denote the event that the non-frameability experiment returns 1. Note that,

$$\begin{aligned}\Pr\left[\mathsf{Exp}^{\mathsf{frame}}_{\mathsf{R\&T},\mathcal{A}}(\lambda)=1\right] &= \Pr\left[\mathsf{Success}\wedge\mathsf{Forge}\right]+\Pr\left[\mathsf{Success}\wedge\overline{\mathsf{Forge}}\right]\\ &\leq \Pr\left[\mathsf{Forge}\right]+\Pr\left[\mathsf{Success}\wedge\overline{\mathsf{Forge}}\right]\end{aligned}$$

By unforgeability of our construction, e.g., one-wayness of function f and extractability of the SOK, $\mathcal{A}$ can output a valid signature on behalf of an honest ring member with negligible probability. Therefore, $\Pr\left[\mathsf{Forge}\right]\leq\mathsf{negl}(\lambda)$.

We now show that $\Pr\left[\mathsf{Success}\wedge\overline{\mathsf{Forge}}\right]\leq\mathsf{negl}(\lambda)$. That is, we assume that $\mathcal{A}$ produces a signature $\sigma^*=(pk_{\mathsf{Sign}},\mathbf{c_1},c_3,\rho,\sigma_{\mathsf{SOK}})$ for a corrupt ring member pk for message m^* and ring R^* such that $\mathsf{Verify}(pp,pk_{\mathsf{T}},m^*,R^*,\sigma^*)=1$. Then $\mathcal{A}$ must construct a report and trace that identifies a different, honest, ring member pk' as the signer. $\mathcal{A}$ must construct proofs ρ_r and ρ_t such that algorithm $\mathsf{NIZK.Verify}$ returns 1. We show that, if the public-key encryption scheme is correct, we can construct an adversary $\mathcal{B}$ that succeeds in the soundness experiment for the NIZK scheme. We present $\mathcal{B}$ in Figure 6.14. It is clear that inputs to $\mathcal{A}$ are distributed identically to the non-frameability experiment because public parameters are generated identically. We now show that $\mathcal{B}$ is successful in the soundness experiment for the NIZK scheme. Tuple $((pk_{\mathsf{T}},c_2,pk),\rho_t)$ is a valid statement/proof pair because algorithm Verify returns 1. Therefore, $\Pr\left[\mathsf{Success}\wedge\overline{\mathsf{Forge}}\right]\leq\mathsf{negl}(\lambda)$ as required and the result holds.

$\mathcal{B}(pp_{\mathsf{NIZK}})$

$\mathbf{pk}_{\mathsf{S}}\leftarrow();\ \mathbf{sk}_{\mathsf{S}}\leftarrow();\ \mathsf{L}\leftarrow\emptyset;\ \mathsf{corL}\leftarrow\emptyset;\ \mathcal{Q}\mathsf{sign}\leftarrow\emptyset;\ \mathcal{Q}\mathsf{rep}\leftarrow\emptyset$
$pp_{\mathsf{PKE}}\leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda);\ pp_{\mathsf{SOK}}\leftarrow\!\!\$\ \mathsf{SoK.Setup}(1^\lambda);\ pp\leftarrow(pp_{\mathsf{PKE}},pp_{\mathsf{NIZK}},pp_{\mathsf{SOK}})$
$(pk_{\mathsf{T}},m^*,R^*,\sigma^*,pk_{\mathsf{S}},\mathtt{Tr},\rho_t)\leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report}}(pp)$
if $\mathsf{VerTrace}(pp,pk_{\mathsf{T}},m^*,R^*,\sigma^*,pk,\mathtt{Tr},\rho_t)=1\ \wedge\ pk_{\mathsf{S}}\in\mathsf{L}\backslash\mathsf{corL}\ \wedge\ \mathsf{Verify}(pp,pk_{\mathsf{T}},m^*,R^*,\sigma^*)=1$
$\wedge\ (pk_{\mathsf{T}},pk_{\mathsf{S}},m^*,R^*,\sigma^*)\notin\mathcal{Q}\mathsf{sign}$
  parse $\mathtt{Tr}$ as $(c_2,\mathtt{Rep})$
  return $((pk_{\mathsf{T}},c_2,pk_{\mathsf{S}}),\rho_t)$

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{report}/\mathcal{O}\mathsf{trace}$

As normal.

Figure 6.14: Adversary $\mathcal{B}$ that breaks the non-frameability security of the NIZK scheme in the proof of Theorem 24.

Soundness: Let $\mathcal{A}$ be an adversary that succeeds in the soundness experiment. Then, $\mathcal{A}$

can output a tuple $(pk_{\mathsf{T}}, m^*, R^*, \sigma^*, pk_i, \mathtt{Tr}_i, \rho_{t_i}, pk_j, \mathtt{Tr}_j, \rho_{t_j})$ such that algorithm $\mathsf{VerTrace}(pp, pk_{\mathsf{T}}, m^*, R^*, \sigma^*, pk_k, \mathtt{Tr}_k, \rho_{t,k}) = 1$ for $k \in \{0,1\}$ and $pk_i \neq pk_j$. As $\mathcal{A}$ can corrupt all parties, we can assume that one tuple $(pk_{\mathsf{RS}}, \mathtt{Tr}, \rho_t)$ generated by $\mathcal{A}$ identifies the signer and VerTrace returns 1 by correctness of the building blocks. Without loss of generality, we assume that tuple $(pk_i, \mathtt{Tr}_i, \rho_{t_i})$ identifies the signer. Then, $\mathcal{A}$ must generate a second tuple $(pk_{\mathsf{RS}_j}, \mathtt{Tr}_j, \rho_{t_j})$ where $i \neq j$ that identifies a user that did not sign message m^*. By correctness of the PKE scheme, $\mathtt{Tr}_j = \mathtt{Tr}_i$. Otherwise, ciphertext c_3 does not decrypt correctly as required by algorithm VerTrace. As such, we require that $\mathcal{A}$ must output proofs ρ_{r_j} and ρ_{t_j} such that algorithm NIZK.Verify returns 1.

We show that we can construct an adversary $\mathcal{B}$ that succeeds in the soundness experiment for the NIZK scheme. We present $\mathcal{B}$ in Figure 6.15. It is clear that inputs to $\mathcal{A}$ are distributed identically to the soundness experiment because public parameters are generated identically. We now show that $\mathcal{B}$ is successful in the soundness experiment. That is, $((pk_{\mathsf{T}}, c_{2,j}, pk_{\mathsf{S}_j}), \rho_{t_j})$ is a valid statement/proof pair. This holds because algorithm Verify returns 1. Therefore, $\mathcal{B}$ succeeds in the soundness experiment and, by contradiction, the result holds. □

$\mathcal{B}(pp_{\mathsf{NIZK}})$	$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{report}/\mathcal{O}\mathsf{trace}$
$\mathbf{pk}_{\mathsf{S}} \leftarrow (); \ \mathbf{sk}_{\mathsf{S}} \leftarrow ()$	As normal.
$\mathsf{L} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{rep} \leftarrow \emptyset$	
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$	
$pp_{\mathsf{SOK}} \leftarrow\!\!\$\ \mathsf{SoK.Setup}(1^\lambda)$	
$pp \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}})$	
$(pk_{\mathsf{T}}, m^*, R^*, \sigma^*, pk_{\mathsf{S}_i}, \mathtt{Tr}_i, \rho_{t_i}, pk_{\mathsf{S}_j}, \mathtt{Tr}_j, \rho_{t_j}) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report}}(pp)$	
if $pk_{\mathsf{S}_i} \neq pk_{\mathsf{S}_j} \ \wedge \ \mathsf{VerTrace}(pp, pk_{\mathsf{T}}, m^*, R^*, \sigma^*, pk_{\mathsf{S}_i}, \mathtt{Tr}_i, \rho_{t_i}) = 1$	
$\wedge \ \mathsf{VerTrace}(pp, pk_{\mathsf{T}}, m^*, R^*, \sigma^*, pk_{\mathsf{S}_j}, \mathtt{Tr}_j, \rho_{t_j}) = 1$	
parse $\mathtt{Tr}_j$ as $(c_{2,j}, \mathsf{Rep}_j)$	
return $((pk_{\mathsf{T}}, c_{2,j}, pk_{\mathsf{S}_j}), \rho_{t_j})$	

Figure 6.15: Adversary $\mathcal{B}$ that breaks the soundness security of the NIZK scheme in the proof of Theorem 24.

Theorem 25. *Our construction satisfies* reporter anonymity *if non-interactive zero-knowledge proof system* NIZK *satisfies zero-knowledge and knowledge extractability.*

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary in the reporter anonymity experiment that makes at most k queries to oracle $\mathcal{O}\mathsf{report}$. We proceed through a series of game hops that we show are indistinguishable to the adversary. In the final game, the view of $\mathcal{A}$ will be identical for $\beta = 0$ and $\beta = 1$. We define Game 0 as the reporter anonymity experiment with β chosen randomly and let S_i denote the event that $\mathcal{A}$ correctly guesses β in Game i.

Game 1 is identical to Game 0 except that we replace algorithm $\mathsf{NIZK.Setup}$ with algorithm $\mathsf{NIZK.SimSetup}$. We present the adjusted experiment in Figure 6.16. This is a superficial change that does not affect the distribution of inputs to the adversary. Therefore,

$$\Big|\Pr[S_0] - \Pr[S_1]\Big| = 0.$$

$\mathsf{Exp}^{\mathsf{r\text{-}anon},\beta}_{\mathsf{R\&T},\mathcal{A}}(\lambda)$

$\mathbf{pk}_{\mathsf{S}} \leftarrow ();\ \mathbf{sk}_{\mathsf{S}} \leftarrow ();\ \mathsf{L} \leftarrow \emptyset;\ \mathsf{corL} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{rep} \leftarrow \emptyset$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda);\ (pp_{\mathsf{NIZK}}, \tau_{\mathsf{NIZK}}) \leftarrow\!\!\$\ \mathsf{NIZK.SimSetup}(1^\lambda)$
$pp_{\mathsf{SOK}} \leftarrow\!\!\$\ \mathsf{SoK.Setup}(1^\lambda);\ pp \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}})$
$(pk_{\mathsf{T}}, m^*, R^*, \sigma^*, id_0, id_1, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report}}(pp)$
$\mathsf{Rep}^* \leftarrow\!\!\$\ \mathsf{Report}(pp, pk_{\mathsf{T}}, \mathbf{sk}_{\mathsf{S}}[id_\beta], m^*, R^*, \sigma^*)$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report}}(\mathsf{Rep}, st)$
if $\{\mathbf{pk}_{\mathsf{S}}[id_0], \mathbf{pk}_{\mathsf{S}}[id_1]\} \subseteq R^* \ \wedge\ \{\mathbf{pk}_{\mathsf{S}}[id_0], \mathbf{pk}_{\mathsf{S}}[id_1]\} \subseteq \mathsf{L}\backslash\mathsf{corL} \ \wedge\ \{(\mathbf{pk}_{\mathsf{S}}[id_0], m^*, R^*, \sigma^*), (\mathbf{pk}_{\mathsf{S}}[id_1], m^*, R^*, \sigma^*)\} \not\subseteq \mathcal{Q}\mathsf{rep}$
  return β'

Figure 6.16: The modified reporter anonymity experiment for Game 1 in the proof of Theorem 25.

Game 2 is identical to Game 1 except for the following change. For queries to $\mathcal{O}\mathsf{report}$ and the challenge report returned to $\mathcal{A}$, we replace algorithm $\mathsf{NIZK.Prove}$ with algorithm $\mathsf{NIZK.SimProve}$. In this way, generating a reporter proof of correct decryption does not require a witness. We define oracle $\mathcal{O}\mathsf{report}$ and the adjusted experiment in Figure 6.17.

$\mathsf{Exp}^{\mathsf{r\text{-}anon},\beta}_{\mathsf{R\&T},\mathcal{A}}(\lambda)$

$\mathbf{pk}_{\mathsf{S}} \leftarrow ();\ \mathbf{sk}_{\mathsf{S}} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset;\ \mathsf{corL} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{rep} \leftarrow \emptyset$
$pp_{\mathsf{PKE}} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$(pp_{\mathsf{NIZK}}, \tau_{\mathsf{NIZK}}) \leftarrow\!\!\$\ \mathsf{NIZK.SimSetup}(1^\lambda)$
$pp_{\mathsf{SOK}} \leftarrow\!\!\$\ \mathsf{SoK.Setup}(1^\lambda)$
$pp \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}})$
$(pk_{\mathsf{T}}, m^*, R^*, \sigma^*, id_0, id_1, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report}}(pp)$
parse R^* as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\}$
$\wedge\ \sigma^*$ as $(pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}}) \ \wedge\ \mathbf{c_1}$ as $(c_{1,1}, \ldots, c_{1,|R^*|})$
if $\mathsf{Verify}(pp, pk_{\mathsf{T}}, m^*, R^*, \sigma^*) = 0$ **return** 0
for $i \in [|R^*|]$ s.t. $pk_{\mathsf{PKE}_i} = \mathbf{pk}_{\mathsf{S}}[id_\beta]$
  $sk_{\mathsf{Sign}} \leftarrow\!\!\$\ \mathsf{PKE.Dec}(pp_{\mathsf{PKE}}, \mathbf{sk}_{\mathsf{S}}[id_\beta], c_{1,i})$
$\rho_r \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}(pp, \tau_{\mathsf{NIZK}}, (R^*, \mathbf{c_1}, sk_{\mathsf{Sign}}))$
$\mathsf{Rep}^* \leftarrow (sk_{\mathsf{Sign}}, \rho_r)$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report}}(\mathsf{Rep}^*, st)$
if $\{\mathbf{pk}_{\mathsf{S}}[id_0], \mathbf{pk}_{\mathsf{S}}[id_1]\} \subseteq R^* \ \wedge\ \{\mathbf{pk}_{\mathsf{S}}[id_0], \mathbf{pk}_{\mathsf{S}}[id_1]\} \subseteq \mathsf{L}\backslash\mathsf{corL}$
$\wedge\ \{(\mathbf{pk}_{\mathsf{S}}[id_0], m^*, R^*, \sigma^*), (\mathbf{pk}_{\mathsf{S}}[id_1], m^*, R^*, \sigma^*)\} \not\subseteq \mathcal{Q}\mathsf{rep}$
  return β'

$\mathcal{O}\mathsf{report}_{(pp, \tau_{\mathsf{NIZK}}, \mathsf{L}, \mathcal{Q}\mathsf{rep}, \mathbf{pk}_{\mathsf{S}}, \mathbf{sk}_{\mathsf{S}})}(id, m, R, \sigma)$

if $\mathbf{pk}_{\mathsf{S}}[id] \notin R \ \vee\ \mathbf{pk}_{\mathsf{S}}[id] \notin \mathsf{L}$ **return** $\perp$
parse R as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\}$
$\wedge\ \sigma$ as $(pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}}) \ \wedge\ \mathbf{c_1}$ as $(c_{1,1}, \ldots, c_{1,|R|})$
if $\mathsf{Verify}(pp, pk_{\mathsf{T}}, m, R, \sigma) = 0$ **return** 0
for $i \in [|R|]$ s.t. $pk_{\mathsf{PKE}_i} = \mathbf{pk}_{\mathsf{S}}[id]$
  $sk_{\mathsf{Sign}} \leftarrow\!\!\$\ \mathsf{PKE.Dec}(pp_{\mathsf{PKE}}, \mathbf{sk}_{\mathsf{S}}[id], c_{1,i})$
$\rho_r \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}(pp, \tau_{\mathsf{NIZK}}, (R, \mathbf{c_1}, sk_{\mathsf{Sign}}))$
$\mathsf{Rep} \leftarrow (sk_{\mathsf{Sign}}, \rho_r)$
$\mathcal{Q}\mathsf{rep} \leftarrow \mathcal{Q}\mathsf{rep} \cup \{(\mathbf{pk}_{\mathsf{S}}[id], m, R, \sigma)\}$
return Rep

Figure 6.17: Oracle $\mathcal{O}\mathsf{report}$ and the adjusted experiment used in Game 2 in the proof of Theorem 25.

We show that Game 1 and Game 2 are indistinguishable if the NIZK scheme satisfies zero-knowledge. We give a distinguishing algorithm $\mathcal{D}_1$ in Figure 6.18. $\mathcal{D}_1$ aims to guess

a bit β^* in the zero-knowledge experiment. We show that, when $\beta^* = 0$, inputs to $\mathcal{A}$ are identical to Game 1 and, when $\beta^* = 1$, inputs to $\mathcal{A}$ are identical to Game 2. Note that pp input to $\mathcal{A}$ is honestly generated and identical in both games. If $\beta^* = 0$, $\mathcal{D}_1$ is provided with a real proof as in Game 1. If $\beta^* = 1$, $\mathcal{D}_1$ is provided with a simulated proof. Therefore,

$$|\Pr[S_1] - \Pr[S_2]| \leq \mathsf{negl}_{\mathsf{NIZK\text{-}zk}}$$

where $\mathsf{negl}_{\mathsf{NIZK\text{-}zk}}$ is the advantage in breaking the zero-knowledge property of the NIZK scheme.

$\mathcal{D}_1^{\mathcal{O}\mathsf{Prove}}(pp_{\mathsf{NIZK}})$

$\beta \leftarrow\$ \{0,1\}$
$\mathbf{pk}_{\mathsf{S}} \leftarrow (); \ \mathbf{sk}_{\mathsf{S}} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{rep} \leftarrow \emptyset$
$pp_{\mathsf{PKE}} \leftarrow\$ \mathsf{PKE.Setup}(1^\lambda)$
$(pp_{\mathsf{NIZK}}, \tau_{\mathsf{NIZK}}) \leftarrow\$ \mathsf{NIZK.SimSetup}(1^\lambda)$
$pp_{\mathsf{SOK}} \leftarrow\$ \mathsf{SoK.Setup}(1^\lambda)$
$pp \leftarrow (pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}})$
$(pk_{\mathsf{T}}, m^*, R^*, \sigma^*, id_0, id_1, st) \leftarrow\$ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report}}(pp)$
parse R^* as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\}$
$\wedge \ \sigma^*$ as $(pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}}) \ \wedge \ \mathbf{c_1}$ as $(c_{1,1}, \ldots, c_{1,|R^*|})$
if $\mathsf{Verify}(pp, pk_{\mathsf{T}}, m^*, R^*, \sigma^*) = 0$ **return** 0
for $i \in [|R^*|]$ s.t. $pk_{\mathsf{PKE}_i} = \mathbf{pk}_{\mathsf{S}}[id_\beta]$
  $sk_{\mathsf{Sign}} \leftarrow\$ \mathsf{PKE.Dec}(pp_{\mathsf{PKE}}, \mathbf{sk}_{\mathsf{S}}[id_\beta], c_{1,i})$
$\rho_r \leftarrow\$ \mathcal{O}\mathsf{Prove}((R^*, \mathbf{c_1}, sk_{\mathsf{Sign}}), \mathbf{sk}_{\mathsf{S}}[id_\beta])$
$\mathsf{Rep}^* \leftarrow (sk_{\mathsf{Sign}}, \rho_r)$
$\beta' \leftarrow\$ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report}}(\mathsf{Rep}^*, st)$
if $\{\mathbf{pk}_{\mathsf{S}}[id_0], \mathbf{pk}_{\mathsf{S}}[id_1]\} \subseteq R^* \ \wedge \ \{\mathbf{pk}_{\mathsf{S}}[id_0], \mathbf{pk}_{\mathsf{S}}[id_1]\} \subseteq \mathsf{L} \backslash \mathsf{corL}$
$\wedge \ \{(\mathbf{pk}_{\mathsf{S}}[id_0], m^*, R^*, \sigma^*), (\mathbf{pk}_{\mathsf{S}}[id_1], m^*, R^*, \sigma^*)\} \not\subseteq \mathcal{Q}\mathsf{rep}$
  return β'

$\mathcal{O}\mathsf{report}_{(pp,\mathsf{L},\mathcal{Q}\mathsf{rep},\mathbf{pk}_{\mathsf{S}},\mathbf{sk}_{\mathsf{S}})}(id, m, R, \sigma)$

if $\mathbf{pk}_{\mathsf{S}}[id] \notin R \ \vee \ \mathbf{pk}_{\mathsf{S}}[id] \notin \mathsf{L}$ **return** $\perp$
parse R as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\}$
$\wedge \ \sigma$ as $(pk_{\mathsf{Sign}}, \mathbf{c_1}, c_3, \rho, \sigma_{\mathsf{SOK}}) \ \wedge \ \mathbf{c_1}$ as $(c_{1,1}, \ldots, c_{1,|R|})$
if $\mathsf{Verify}(pp, pk_{\mathsf{T}}, m, R, \sigma) = 0$ **return** 0
for $i \in [|R|]$ s.t. $pk_{\mathsf{PKE}_i} = \mathbf{pk}_{\mathsf{S}}[id]$
  $sk_{\mathsf{Sign}} \leftarrow\$ \mathsf{PKE.Dec}(pp_{\mathsf{PKE}}, \mathbf{sk}_{\mathsf{S}}[id], c_{1,i})$
$\rho_r \leftarrow\$ \mathcal{O}\mathsf{Prove}((R, \mathbf{c_1}, sk_{\mathsf{Sign}}), \mathbf{sk}_{\mathsf{S}}[id])$
$\mathsf{Rep} \leftarrow (sk_{\mathsf{Sign}}, \rho_r)$
$\mathcal{Q}\mathsf{rep} \leftarrow \mathcal{Q}\mathsf{rep} \cup \{(\mathbf{pk}_{\mathsf{S}}[id], m, R, \sigma)\}$
return Rep

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{sign}$

As in Game 0.

Figure 6.18: Distinguisher $\mathcal{D}_1$ that distinguishes Game 1 and Game 2 in the proof of Theorem 25.

Game 3 is identical to Game 2 except that, for queries to $\mathcal{O}$report and the challenge report output to $\mathcal{A}$ we run algorithm NIZK.Extract to output the witness $(r_{1,1}, \ldots, r_{1,|R|}, sk_{\mathsf{Sign}})$ from proof of correct encryption ρ, rather than decrypting a ciphertext from $\mathbf{c_1}$. We define oracle $\mathcal{O}$report and the adjusted experiment in Figure 6.19. If the NIZK scheme satisfies knowledge extractability then, with overwhelming probability, the witness output by algorithm NIZK.Extract is identical to the input to algorithm PKE.Enc to generate vector of ciphertexts $\mathbf{c_1}$. As such, the view of the adversary is identical following this game hop, unless the NIZK scheme does not satisfy knowledge extractability. Therefore,

$$|\Pr[S_2] - \Pr[S_3]| \leq k \cdot \mathsf{negl}_{\mathsf{NIZK\text{-}Ext}}$$

where $\mathsf{negl}_{\mathsf{NIZK\text{-}Ext}}$ is the probability that the NIZK scheme does not satisfy knowledge extractability.

$\mathsf{Exp}^{\mathsf{r\text{-}anon},\beta}_{\mathsf{R\&T},\mathcal{A}}(\lambda)$

$\mathbf{pk}_\mathsf{S} \leftarrow (); \ \mathbf{sk}_\mathsf{S} \leftarrow (); \ \mathsf{L} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{rep} \leftarrow \emptyset$
$pp_\mathsf{PKE} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda); \ (pp_\mathsf{NIZK}, \tau_\mathsf{NIZK}) \leftarrow\!\!\$\ \mathsf{NIZK.SimSetup}(1^\lambda)$
$pp_\mathsf{SOK} \leftarrow\!\!\$\ \mathsf{SoK.Setup}(1^\lambda); \ pp \leftarrow (pp_\mathsf{PKE}, pp_\mathsf{NIZK}, pp_\mathsf{SOK})$
$(pk_\mathsf{T}, m^*, R^*, \sigma^*, id_0, id_1, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report}}(pp)$
parse R^* as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\} \ \wedge \ \sigma^*$ as $(pk_\mathsf{Sign}, \mathbf{c_1}, c_3, \rho, \sigma_\mathsf{SOK}) \ \wedge \ \mathbf{c_1}$ as $(c_{1,1}, \ldots, c_{1,|R^*|})$
if $\mathsf{Verify}(pp, pk_\mathsf{T}, m^*, R^*, \sigma^*) = 0$ **return** 0
$(r_{1,1}, \ldots, r_{1,|R^*|}, sk_\mathsf{Sign}) \leftarrow\!\!\$\ \mathsf{NIZK.Extract}(pp_\mathsf{NIZK}, \tau_\mathsf{NIZK}, (pk_\mathsf{Sign}, (pk_{\mathsf{PKE}_1}, \ldots, pk_{\mathsf{PKE}_{|R^*|}}), \mathbf{c_1}), \rho)$
$\rho_r \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}(pp, \tau_\mathsf{NIZK}, (R^*, \mathbf{c_1}, sk_\mathsf{Sign}))$; $\mathsf{Rep}^* \leftarrow (sk_\mathsf{Sign}, \rho_r)$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report}}(\mathsf{Rep}^*, st)$
if $\{\mathbf{pk}_\mathsf{S}[id_0], \mathbf{pk}_\mathsf{S}[id_1]\} \subseteq R^* \ \wedge \ \{\mathbf{pk}_\mathsf{S}[id_0], \mathbf{pk}_\mathsf{S}[id_1]\} \subseteq \mathsf{L}\backslash\mathsf{corL} \ \wedge \ \{(\mathbf{pk}_\mathsf{S}[id_0], m^*, R^*, \sigma^*), (\mathbf{pk}_\mathsf{S}[id_1], m^*, R^*, \sigma^*)\} \not\subseteq \mathcal{Q}\mathsf{rep}$
return β'

$\mathcal{O}\mathsf{report}_{(pp,\tau_\mathsf{NIZK},\mathsf{L},\mathcal{Q}\mathsf{rep},\mathbf{pk}_\mathsf{S},\mathbf{sk}_\mathsf{S})}(id, m, R, \sigma)$

if $\mathbf{pk}_\mathsf{S}[id] \notin R \vee \mathbf{pk}_\mathsf{S}[id] \notin \mathsf{L}$ **return** $\perp$
parse R as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\} \ \wedge \ \sigma$ as $(pk_\mathsf{Sign}, \mathbf{c_1}, c_3, \rho, \sigma_\mathsf{SOK}) \ \wedge \ \mathbf{c_1}$ as $(c_{1,1}, \ldots, c_{1,|R|})$
if $\mathsf{Verify}(pp, pk_\mathsf{T}, m, R, \sigma) = 0$ **return** 0
$(r_{1,1}, \ldots, r_{1,|R|}, sk_\mathsf{Sign}) \leftarrow\!\!\$\ \mathsf{NIZK.Extract}(pp_\mathsf{NIZK}, \tau_\mathsf{NIZK}, (pk_\mathsf{Sign}, (pk_{\mathsf{PKE}_1}, \ldots, pk_{\mathsf{PKE}_{|R|}}), \mathbf{c_1}), \rho)$
$\rho_r \leftarrow\!\!\$\ \mathsf{NIZK.SimProve}(pp, \tau_\mathsf{NIZK}, (R, \mathbf{c_1}, sk_\mathsf{Sign}))$; $\mathsf{Rep} \leftarrow (sk_\mathsf{Sign}, \rho_r)$
$\mathcal{Q}\mathsf{rep} \leftarrow \mathcal{Q}\mathsf{rep} \cup \{(\mathbf{pk}_\mathsf{S}[id], m, R, \sigma)\}$
return Rep

Figure 6.19: Oracle $\mathcal{O}$report and the adjusted experiment used in Game 3 in the proof of Theorem 25.

In Game 3, the inputs to $\mathcal{A}$ are identical for $\beta = 0$ and $\beta = 1$. Therefore, $\Pr[S_3] = 1/2$. We now have that

$$|\Pr[S_0] - 1/2| \leq \mathsf{negl}_{\mathsf{NIZK\text{-}zk}} + k \cdot \mathsf{negl}_{\mathsf{Ext}}$$

and conclude that the advantage of the adversary in Game 0 is negligible as required. □

6.4.3 Instantiating Our Construction

Our construction can be instantiated with ElGamal encryption and the signature of knowledge of [31], modified to account for the double layer of encryption of the signer's identity. Additionally, we instantiate our NIZK protocols for signing, reporting and tracing with various Σ-protocols, and outline these below. For our choice of cryptographic primitives, our instantiation is secure. Indeed, our chosen schemes satisfy the security requirements for the construction to be secure. We now describe our instantiation and then outline its security.

Setup and key generation. Our scheme shares common setup parameters. That is, we let $pp_{\mathsf{PKE}} = pp_{\mathsf{NIZK}} = (\mathbb{G}, g, q)$ where $\mathbb{G}$ is a cyclic group of order q with generator g. Moreover, $pp_{\mathsf{SOK}} = (ek, ck)$ where ek is an ElGamal encryption key and ck is a key for the Pedersen commitment scheme. We define one-way function f to perform group exponentiation such that $f(x) = g^x$ for some $x \in \mathbb{Z}_q$. We write $pp = (pp_{\mathsf{PKE}}, pp_{\mathsf{SOK}})$ as the output of algorithm Setup. We write the key pair for the tracer as $(pk_{\mathsf{T}} = g^{sk_{\mathsf{T}}}, sk_{\mathsf{T}})$ and the key pair for the user as $(pk_{id}, sk_{id}) = ((g^{sk_{\mathsf{RS}}}, g^{sk_{\mathsf{PKE}}}), (sk_{\mathsf{RS}}, sk_{\mathsf{PKE}}))$ where $sk_{\mathsf{T}}, sk_{\mathsf{RS}}, sk_{\mathsf{PKE}} \in \mathbb{Z}_q$.

Sign. We use a standard ElGamal encryption scheme to generate ciphertext c_1 and to double-encrypt the signer's identity pk in algorithm Sign. That is, we define ciphertexts c_2 and c_3 as follows.

$$c_2 = (A_2, B_2) = (g^{r_2}, pk_{\mathsf{T}}^{r_2} \cdot pk) \tag{6.5}$$

$$c_3 = (A_2, A_3, B_3) = (g^{r_2}, g^{r_3}, pk_{\mathsf{Sign}}^{r_3} \cdot pk_{\mathsf{T}}^{r_2} \cdot pk) \tag{6.6}$$

To generate proof ρ for the relation in Equation 6.1, we use the Σ-protocol of [117], defined in Figure 6.20, which shows that c_1 encrypts the discrete log of a public element pk_{Sign} and can be transformed into a NIZK proof using the Fiat-Shamir transform [64].

$\mathcal{P}(pp_{\mathsf{PKE}}, (pk_{\mathsf{Sign}}, (pk_{\mathsf{PKE}_1}, \ldots, pk_{\mathsf{PKE}_{\lvert R\rvert}}, \mathbf{c_1}), (r_{1,1}, \ldots, r_{1,\lvert R\rvert}, sk_{\mathsf{Sign}}))$		$\mathcal{V}(pp_{\mathsf{PKE}}, (pk_{\mathsf{Sign}}, (pk_{\mathsf{PKE}_1}, \ldots, pk_{\mathsf{PKE}_{\lvert R\rvert}}, \mathbf{c_1}))$
parse pp_{PKE} as $(\mathbb{G}, g, q)$		**parse** pp_{PKE} as $(\mathbb{G}, g, q)$
parse $\mathbf{c_1}$ as $(c_{1,1}, \ldots, c_{1,\lvert R\rvert})$	$\xrightarrow{t_{h_1}, \ldots, t_{h_{\lvert R\rvert}}, t_{g_1} \ldots, t_{g_{\lvert R\rvert}}}$	**parse** $\mathbf{c_1}$ as $(c_{1,1}, \ldots, c_{1,\lvert R\rvert})$
for $i = 1, \ldots \lvert R\rvert$: **parse** $c_{1,i}$ as $(A_{1,i}, B_{1,i})$		**for** $i = 1, \ldots \lvert R\rvert$: **parse** $c_{1,i}$ as $(A_{1,i}, B_{1,i})$
for $i = 1, \ldots \lvert R\rvert$	$\xleftarrow{x_1, \ldots, x_{\lvert R\rvert} \leftarrow\$ \{0,1\}}$	**return** 1 iff **for** $i = 1, \ldots, \lvert R\rvert$
$w_i \leftarrow\$ \mathbb{Z}_q$; $t_{h_i} \leftarrow g^{w_i}$; $t_{g_i} = g^{pk_{\mathsf{PKE}_i}^{w_i}}$		$t_{h_i} = g^{z_i} A_{1,i}^{x_i}$
for $i = 1, \ldots, \lvert R\rvert$: $z_i = w_i - x_i r_{1,i}$	$\xrightarrow{z_1, \ldots, z_{\lvert R\rvert}}$	**if** $x_i = 0$: $t_{g_i} = g^{pk_{\mathsf{PKE}_i}^{z_i}}$
		if $x_i = 1$: $t_{g_i} = \dfrac{g^{B_{1,i} + pk_{\mathsf{PKE}_i}^{z_i}}}{pk_{\mathsf{Sign}}}$

Figure 6.20: A Σ-protocol for proving that ciphertext $\mathbf{c_1}$ encrypts a discrete log of pk_{Sign}.

Finally, we modify the SOK of [31] to generate σ_{SOK}, which proves that the signer knows sk_{RS} such that $pk_{\mathsf{RS}} = g^{sk_{\mathsf{RS}}}$ is an element of the ring. Our modification accounts for the double-layer of encryption used in our construction, rather than the single ElGamal

encryption required in the accountable ring signature construction of [31] and we obtain the Σ-protocol for Equation 6.2 in Figure 6.21. We note that this protocol, and the Σ-protocol of [31], relies on two additional Σ-protocols, one to prove that a commitment opens to a sequence of bits which contains a single 1, and a second to prove that a list of ElGamal ciphertexts contains an encryption of 1. These Σ protocols are required to prove that the signer knows the signing key corresponding to a verification key in the ring, without revealing their identity in the ring. We can use the additional Σ-protocols from [31] without any adjustments.

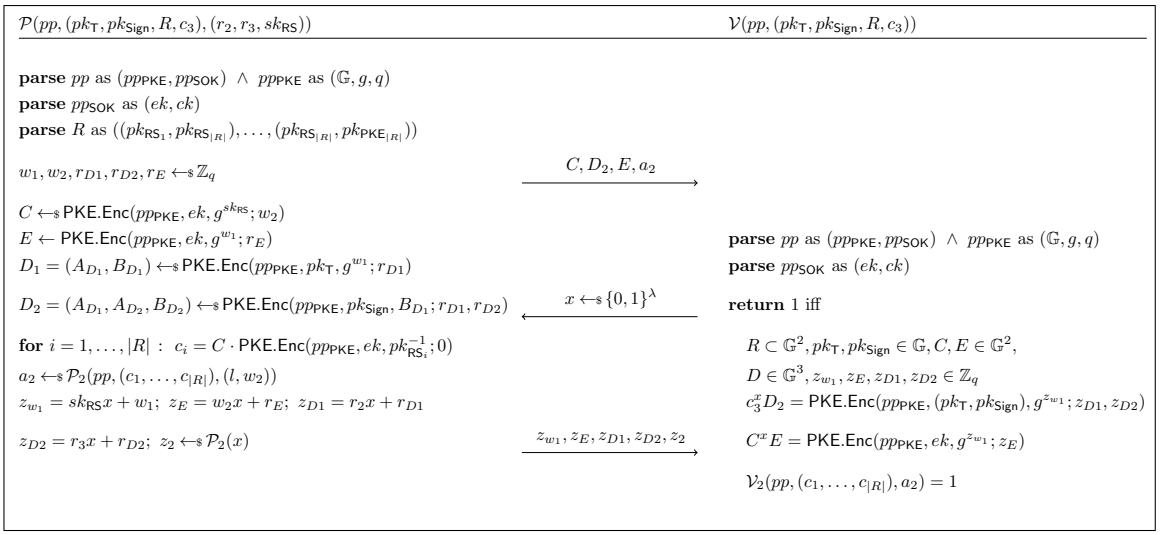

Figure 6.21: A modified Σ-protocol for $\mathcal{R}_{\mathsf{SOK}}$ from [31].

Report and trace. The reporter runs $\mathsf{PKE.Dec}$ to generate sk_{Sign}. For our instantiation, the reporter need not generate a proof of knowledge of correct decryption for the relation in Equation 6.3. In fact, as $\mathbb{G}$ is a cyclic group, $g^i = g^j$ if $i = j \mod q$. Therefore, pk_{Sign} is uniquely defined by $g^{sk_{\mathsf{Sign}}}$. The tracer (and any public verifier) can trivially check correct decryption of the reporter token by computing a single group exponentiation, i.e., check $g^{sk_{\mathsf{Sign}}} = pk_{\mathsf{Sign}}$ where sk_{Sign} is returned by the reporter and pk_{Sign} is included in the signature. Then, the tracer decrypts the signer's identity by computing

$$\frac{B_3}{A_2^{sk_{\mathsf{T}}} A_3^{sk_{\mathsf{Sign}}}}$$

for $c_3 = (A_2, A_3, B_3)$, as defined in Equation 6.6. The tracer can prove correct decryption of ciphertext c_2 using a standard Σ-protocol as outlined in Figure 6.22.

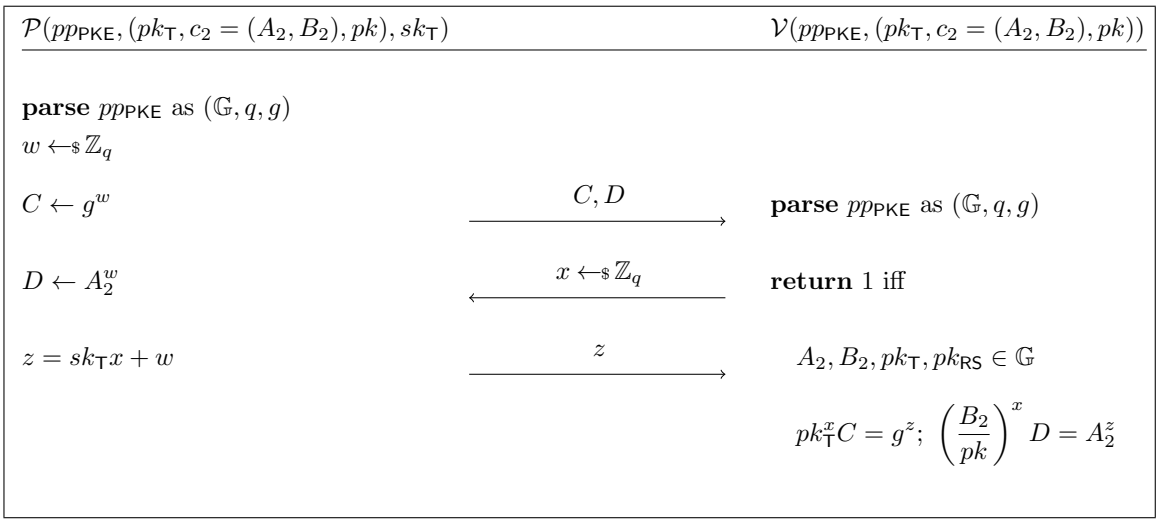

Figure 6.22: A Σ-protocol for proving correct decryption of an ElGamal ciphertext.

Security of our instantiation. Here we provide sketch proofs of the security of our instantiation, demonstrating that our choices of cryptographic primitives meet the requirements for security, that is, the proofs in Chapter 6.4.2 hold. We restrict ourselves to sketch proofs noting that details of the proofs are very similar to the proofs presented in [31], and we refer the reader to [31] for more details.

Lemma 9. *Function f , where $f(x) = g^x$ for some $x \in \mathbb{Z}_q$ and group $(\mathbb{G}, g, q)$, is a one-way function if the discrete logarithm assumption holds with respect to group $(\mathbb{G}, g, q)$.*

Sketch proof. Recall that the discrete log assumption holds relative to group $(\mathbb{G}, g, q)$ if an adversary, on input of $(\mathbb{G}, g, q)$ and a group element y, cannot output a discrete log x such that $y = g^x$, except with negligible probability. Let $\mathcal{A}$ be an adversary in the hard to invert property of one-way function f that outputs an element x' such that $f(x') = y$. Then we can use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ against the discrete logarithm assumption. Trivially, $\mathcal{B}$ can output x' and the result holds. □

Lemma 10. *The double-layer of encryption that defines ciphertext c_3 (Equation 6.6) is correct and satisfies* IND-CPA *security.*

Sketch proof. Correctness holds by direct verification. Moreover, if there exists an adversary $\mathcal{A}$ against the IND-CPA security of ciphertext c_3, then we can use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ against the IND-CPA property of the underlying standard ElGamal encryption scheme. By assumption, ElGamal satisfies IND-CPA security so, by contradiction, ciphertext c_3 is IND-CPA secure. □

Lemma 11. *The Σ-protocol in Figure 6.21 satisfies completeness, special honest verifier zero-knowledge, 2-special soundness and has quasi-unique responses. Moreover, applying the Fiat-Shamir transform to the Σ-protocol leads to an* SOK *in the ROM that is correct, simulatable and extractable.*

Sketch proof. This result follows directly from the security of the Σ-protocol of [31], including the security of the two additional Σ-protocols upon which it is based. We refer the reader to [31] for full details of the proofs. Briefly, completeness follows from direct verification, and correctness of the SOK follows from completeness of the Σ-protocol. For special honest verifier zero-knowledge, we define algorithm Sim to choose responses $z_{w1}, z_E, z_{D1}, z_{D2} \leftarrow\!\!\$\, \mathbb{Z}_q$ and choose $C \leftarrow\!\!\$\, \mathbb{G}^2$. Ciphertexts D_2 and E can be obtained from the verification equations. Then, by the DDH assumption, C output by Sim is indistinguishable from C when generated according to the Σ protocol if the DDH assumption holds with respect to group $(\mathbb{G}, g, q)$. Moreover, $z_{w1}, z_E, z_{D1}, z_{D2}$ are uniform and uniquely define D_2 and E when generated by $\mathcal{P}$ or Sim and the result holds. Simulatability of the SOK scheme follows. For 2-special soundness, we define two challenges x, x' and responses $z_{w1}, z_E, z_{D1}, z_{D2}$ and $z'_{w1}, z'_E, z'_{D1}, z'_{D2}$, such that the verifier returns 1 for each challenge/response pair. Then by computing r_2, r_3 and sk_{RS} from the first verification equation $c_3^{(x-x')} = \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, (pk_{\mathsf{T}}, pk_{\mathsf{Sign}}), g^{z_{w1}-z'_{w1}}; z_{D1} - z'_{D1}, z_{D2} - z'_{D2})$ and replacing in the second verification equation, it is clear that c_3 encrypts pk as required. Extractability of the SOK scheme follows. □

Lemma 12. *The Σ-protocol in Figure 6.20 satisfies completeness, special honest verifier zero-knowledge and 2-special soundness. Moreover, applying the Fiat-Shamir transform to the Σ-protocol leads to an* NIZK *proof in the ROM that satisfies completeness, zero-knowledge, soundness and knowledge extractability.*

Sketch proof. This result is similar to the proof of Lemma 11. In fact, completeness follows from direct verification and completeness of the NIZK proof follows. Special honest verifier zero knowledge holds because Sim can choose responses $z_1, \ldots, z_n \leftarrow\!\!\$\, \mathbb{Z}_q$ and selects challenges t_h and $t_{g_1}, \ldots, t_{g_n}$ based on the random responses and x. As such, the result holds and simulatability of the NIZK proof follows. Finally, special soundness holds as two challenges x, x' and responses $z_1, \ldots, z_n$ and $z'_1, \ldots, z'_n$ ensure that c_1 encrypts a discrete log as required. As before, soundness and knowledge extractability of the NIZK proof follows. □

Lemma 13. *The Σ-protocol in Figure 6.22 satisfies completeness, special honest verifier zero-knowledge, 2-special soundness and has unique responses. Moreover, applying the Fiat-Shamir transform to the Σ-protocol leads to an* NIZK *proof in the ROM that satisfies completeness, zero-knowledge, soundness and knowledge extractability.*

Sketch proof. The proof of this result follows is identical to the proof presented in [31], which presents a Σ-protocol for the same relation. Moreover, completeness of the NIZK proof follows from completeness of the Σ-protocol, zero-knowledge follows from special honest verifier zero-knowledge and soundness and knowledge extractability follows from special soundness. □

6.4.4 Efficiency of Our Construction

Here, we discuss the efficiency of our construction, showing that it incurs reasonable costs with respect to the functionality provided and that our proposed instantiation is practical. We also highlight the additional costs associated with our report and trace functionality by comparing with the accountable ring signature in [31]. We show that our construction compares favourably to the accountable ring signature construction of [31]: indeed, we require additional computation, but, considering the extra functionality provided by our R&T construction, we believe that this is reasonable. We summarise the computation and communication costs of our generic construction and instantiation in Tables 6.2 and 6.3, respectively. We now briefly describe the costs incurred by the signer, reporter, tracer and verifier.

Signer. The signer's computation costs are dominated by the SOK and PKE computations, both of which grow linearly in the size of the ring. In comparison to the accountable ring signature of [31], we see that encrypting a reporter token doubles the computation costs for the signer. Moreover, the size of the signature also increases, requiring the communication of a number of group and field elements that grow linearly in the size of the ring (whereas the accountable ring signature communicates a constant number of group elements).

		Accountable ring signature [31]	**Our R&T construction** (Fig.3)	**R&T with multiple reporters** (§5)
Sign	Comp.	1 PKE.Enc 1 SoK.Sign	$\|R\|+2$ PKE.Enc 1 NIZK.Prove ($\|R\|$ enc) 1 SoK.Sign	$\|R\|+2$ PKE.Enc $\|R\|$ NIZK.Prove (enc) 1 SoK.Sign $\|R\|$ public share gen
	Comm.	1 PKE ciphertext 1 SOK	$\|R\|+1$ PKE ciphertext 1 NIZK proof ($\|R\|$ enc) 1 SOK 1 element pk_{Sign}	$\|R\|+1$ PKE ciphertext $\|R\|$ NIZK proof (enc) 1 SOK $\|R\|$ PVSS public shares 1 element S
Verify	Comp.	1 SoK.Verify	1 SoK.Verify 1 NIZK.Verify ($\|R\|$ enc)	1 SoK.Verify $\|R\|$ NIZK.Verify (enc) 1 SS.Verify
	Comm.	N/A	N/A	N/A
Report	Comp.	N/A	1 PKE.Dec 1 NIZK.Prove(dec)	1 PKE.Dec 1 NIZK.Prove(dec)
	Comm.	N/A	1 token sk_{Sign} 1 NIZK proof (dec)	1 sub-token s_i 1 NIZK proof (dec)
Trace	Comp.	1 PKE.Dec 1 NIZK.Prove (dec)	2 PKE.Dec 1 NIZK.Prove (dec) 1 NIZK.Verify(dec)	2 PKE.Dec 1 NIZK.Prove (dec) t NIZK.Verify(dec) 1 SS.Combine
	Comm.	1 pk of signer 1 NIZK proof (dec)	1 pk of signer 1 token sk_{Sign} 1 PKE ciphertext 2 NIZK proof (dec)	1 pk of signer t sub-tokens $s_1, \ldots, s_t$ 1 PKE ciphertext $t+1$ NIZK proof (dec)
VerTrace	Comp.	1 NIZK.Verify (dec)	2 NIZK.Verify (dec) 1 PKE.Dec	$t+1$ NIZK.Verify (dec) 1 PKE.Dec 1 SS.Combine
	Comm.	N/A	N/A	N/A

Table 6.2: Computation (comp.) and communication (comm.) costs of generic constructions. We write (enc) and (dec) to indicate a proof of correct encryption or decryption respectively. For our multiple reporters construction, we provide costs relative to a ring of size $|R|$ and a threshold of t.

Reporter. The computation and communication costs incurred by the reporter, which is unique to our construction, are small. In fact, for our instantiation, the reporter need only perform a single decryption (i.e., 1 group exponentiation). As such, a report consists of a single field element. This demonstrates that, though our generic construction allows for the case where a reporter proves correct decryption, by basing our instantiation on cyclic group operations, we can provide an efficient instantiation in which the computation costs of the reporter are small and constant.

Tracer. The tracer's costs are small and compare favourably to accountable ring signatures. Indeed, computation of the trace requires a constant number of group exponentiations, calling for only 2 additional group exponentiation when compared to accountable ring

		Accountable ring signature [31]	**Our** R&T **instantiation** (§6.4.3)
Sign	Comp.	$4\|R\| + 14$	$8\|R\| + 19$
	Comm.	$14\mathbb{G} + (\|R\| + 7)\mathbb{Z}_q$	$(3\|R\| + 19)\mathbb{G} + (2\|R\| + 7)\mathbb{Z}_q$
Verify	Comp.	$3\|R\| + 19$	$6\|R\| + 23$
	Comm.	N/A	N/A
Report	Comp.	N/A	1
	Comm.	N/A	$1\mathbb{Z}_q$
Trace	Comp.	3	5
	Comm.	$3\mathbb{G} + 1\mathbb{Z}_q$	$5\mathbb{G} + 2\mathbb{Z}_q$
VerTrace	Comp.	4	6
	Comm.	N/A	N/A

Table 6.3: Computation (comp.) and communication (comm.) costs of instantiations. We present costs relative to a ring of size $|R|$. Our computation costs are given in terms of the number of group exponentiations required, and our communication costs are presented in terms of the number of group elements from $\mathbb{G}$ and field elements from $\mathbb{Z}_q$.

signatures. In particular, to verify correct decryption of the reporter token, the tracer need only perform a single group exponentiation, rather than verifying a NIZK proof (cf. §6.4.3). Furthermore, the size of a trace is constant and requires only 1 extra field element and 2 extra group elements, when compared to accountable ring signatures.

Verifier. As for signature generation, signature verification costs are dominated by the SOK and PKE scheme. Specifically, our PKE computations require computation costs similar to those required to verify the SOK, which means verification of an R&T ring signature incurs double the computational costs of an accountable ring signature. On the other hand, verification of the trace requires only 6 group exponentiations, only 2 more than accountable ring signatures.

Potential efficiency improvements. Our instantiation builds upon the accountable ring signature of [31]. It is possible that techniques from group signature literature (e.g., [90, 92, 115]) could be used to provide a more efficient construction. However, we opt to build upon the construction of [31] to clearly demonstrate the additional costs associated with reporting when reporting is incorporated into an accountable ring signature. Indeed, our instantiation presents worst-case efficiency results and can perform reasonably for a small ring. Here, we highlight potential modifications to our instantiation that can lead to efficiency improvements.

Our instantiation could use a broadcast encryption (BE) scheme [63] to encrypt the reporter token to all members of the ring. Potentially, by using a BE scheme that is based on bilinear maps (as, for example, [30] and subsequent works), our costs would be closer to those of the accountable ring signature construction [31]. Our reasons for not following this approach are twofold. First, in contrast with BE schemes based on bilinear maps, our construction does not require an interactive key generation protocol. In fact, in our construction, users generate their public/secret key pair without the need for a trusted key distributor that holds a master secret key. Accordingly, our instantiation retains the benefit of ring signatures with respect to non-interactive key generation. Second, NIZK proofs of correct encryption for schemes based on bilinear maps are currently unknown [109]. Our construction, on the other hand, requires Σ-protocols to prove correct encryption and decryption of the reporter token. Moreover, by relying on cyclic groups, our instantiation does not require a proof of correct decryption for the reporter token, and the tracer can efficiently verify correct decryption.

Furthermore, we note two modifications that lead to a more efficient, yet more limited, protocol. Firstly, during setup, the signer could choose a *static* set of possible reporters that share a secret key for a PKE scheme. Then, more simply, the reporter share is encrypted under the corresponding public key. Though this approach is more efficient, we opt to allow the signer to *dynamically* choose the reporter set, fostering a sense of user empowerment and capturing the functionality of a user posting entries in different fora. Secondly, if reporter anonymity is not a concern and a proof of correct decryption is required by the reporter, the reporter can produce a proof that indicates which reporter decrypted the token, which would decrease the computation costs incurred by the reporter. However, we opt to provide a construction that meets a strong security model, ensuring that reporters can produce reports without the concern of their identity being leaked.

6.5 Extending R&T to Multiple Reporters

In our construction, we assume that the reporter and tracer do not collude and, hence, if a reported message is not malicious, the tracer does not reveal the identity of the signer. That being said, we can further mitigate against a malicious reporter by requiring that the tracer receive *multiple* reports to trigger the tracing process. We describe an extension

of our construction to multiple reporters that requires a (t, n)-publicly verifiable secret sharing scheme PVSS, where $n = |R|$ is the size of the ring and t is the number of shares required to reconstruct the secret.

The PVSS scheme is used to generate $|R|$ shares of the reporter token, i.e., reporter token $s = (s_1, \ldots, s_{|R|})$. Each reporter share is encrypted under a ring member's public key for a PKE scheme, rather than encrypting a single reporter token to all members of the ring. This extension requires minimal changes to our construction, which we outline here.

$\mathsf{Sign}(pp, sk_{id}, pk_{\mathsf{T}}, m, R)$

parse pp as $(pp_{\mathsf{PKE}}, pp_{\mathsf{NIZK}}, pp_{\mathsf{SOK}})$ $\wedge$ sk_{id} as $(sk_{\mathsf{RS}}, sk_{\mathsf{PKE}})$
$\wedge$ R as $\{(pk_{\mathsf{RS}_1}, pk_{\mathsf{PKE}_1}), \ldots, (pk_{\mathsf{RS}_{|R|}}, pk_{\mathsf{PKE}_{|R|}})\}$
$pk \leftarrow\!\!\$\ f(sk_{\mathsf{RS}})$; $s \leftarrow\!\!\$\ SK_{\mathsf{PKE}}$; $r_{1,1}, \ldots, r_{1,|R|}, r_2, r_3 \leftarrow\!\!\$\ \mathsf{Rand}$
$(\{s_1, \ldots, s_{|R|}\}, \{S_1, \ldots, S_{|R|}\}, S) \leftarrow\!\!\$\ \mathsf{SS.Gen}(s, t, |R|)$
for $i = 1, \ldots, |R|$
  $c_{1,i} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{PKE}_i}, s_i; r_{1,i})$
  $\rho_i \leftarrow\!\!\$\ \mathsf{NIZK.Prove}(pp, (S_i, pk_{\mathsf{PKE}_i}, c_{1,i}), (r_{1,i}, s_i))$
  $\mathsf{Share}_i \leftarrow (S_i, c_{1,i}, \rho_{1,i})$
$c_2 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, pk_{\mathsf{T}}, pk; r_2)$; $c_3 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pp_{\mathsf{PKE}}, S, c_2; r_3)$
$\sigma_{\mathsf{SOK}} \leftarrow\!\!\$\ \mathsf{SoK.Sign}(pp, (pk_{\mathsf{T}}, S, R, c_3), (r_2, r_3, sk_{\mathsf{RS}}), m)$
return $\sigma \leftarrow (S, (\mathsf{Share}_1, \ldots, \mathsf{Share}_{|R|}), c_3, \sigma_{\mathsf{SOK}})$

Figure 6.23: Algorithm Sign for our R&T ring signature scheme with multiple reporters.

To sign a message, the signer uses the PVSS scheme to produce $|R|$ reporter sub-tokens. Each sub-token s_i is encrypted under the public key of a ring member and is accompanied with a NIZK proof of correct encryption and a public version of the sub-token, S_i. For clarity, we outline the changes to the signing algorithm in Figure 6.23. As before, our construction provides public verification algorithm Verify, which additionally requires that the verifier run algorithm SS.Verify to verify the secret sharing operations.

A ring member reports a message by decrypting their sub-token and sending the sub-token to the tracer, accompanied with a proof of correct decryption. Once the tracer has received a threshold number of sub-tokens, the tracer runs algorithm SS.Combine to recover the reporter token, then decrypts the signer's identity as per the original construction.

Anonymity of reporters. Recall that a feature of our single reporter construction is that the reporter is anonymous, even after tracing. For our multiple reporter construction, this is no longer the case. Each reporter is provided with a unique sub-token, and public

versions of each sub-token are published in order to verify correctness of the PVSS scheme. In this way, when a reporter produces a report, i.e., their sub-token, anyone can check which share (denoted Share) this belongs to. Thus, it is possible to determine the identity of the reporters in our multiple reporter construction. Despite this, our multiple reporter construction can provide limited protection for reporters. In fact, the tracer can keep reporter tokens secret, thereby protecting the identity of reporters, until tracing is complete. In this way, the identities of reporters are known to the tracer before tracing but are not made public until after tracing.

Efficiency of multiple reporters. Our construction with multiple reporters is less efficient than our generic construction with a single reporter, where the communication and computation costs are outlined in Table 6.2. Specifically, producing a signature requires the additional computation and communication of $|R|$ reporter sub-tokens, and the computation and communication costs of the tracer grow linearly in the size of the threshold. Moreover, the computation costs associated with signature and trace verification grow linearly in the size of the ring and threshold, respectively. However, these costs can be minimised. In particular, secret share generation and combination, for efficient PVSS schemes (e.g., [117]), simply requires the computation of group exponentiations and the addition of field elements respectively [117]. Additionally, the NIZK proofs associated with these extra costs can be instantiated with efficient primitives, as in our single reporter construction. That being said, we note that the size and computation costs of a report are identical to our single reporter construction, consisting of a single reporter sub-token, and requiring a single decryption of a PKE ciphertext.

6.6 A Comparison of Tracing Functions for Ring Signatures

As we have introduced a new primitive, we wish to understand how it relates to the existing literature. Within the ring signature literature, there exists many variants of ring signatures, several of which provide some tracing function. We believe it is useful to understand how these ring signature variants relate and, in particular, how our new primitive fits into the ring signature landscape. To this end, we compare report and trace with three tracing functions from the literature, namely linkable [98, 99], link and traceable [68, 74] and accountable [31, 130] ring signatures. We show that these functionalities are related.

Specifically, we formally prove that a secure accountable ring signature can be used to construct a secure link and traceable ring signature, which can, in turn, be used to construct a secure linkable ring signature. Finally, we show that a secure R&T ring signature can be used to construct a secure accountable ring signature. To prove these results, we first define generic syntax that captures each of the tracing functions that we consider and cast the security model for each tracing function into our syntax. We then formally prove the relations between the different tracing functions.

6.6.1 Syntax

We introduce the syntax of a designated tracer ring signature (DTRS) scheme and demonstrate how this syntax captures a range of functionalities. A DTRS scheme extends a standard ring signature scheme with algorithms that allow a designated tracer to generate a key pair and perform the tracing function. We note that, though R&T and accountable ring signatures *require* a designated tracer, linkable and link and traceable schemes, on the other hand, sometimes rely on a designated tracer [74, 98], but not always [68, 99]. In the latter case, the link/link and trace function is public. However, our syntax captures a designated tracer as this applies to the majority of functionalities we consider. Additionally, our syntax defines a public verification algorithm VerTrace, which allows anybody to check that the trace output by algorithm Trace is correct. Linkable and link and traceable ring signatures do not employ a verification algorithm, but we include it in our syntax for completeness. If the tracing mechanism is *not* verifiable, algorithm VerTrace can simply be defined to output 1.

Definition 76 (Designated tracer ring signature)**.** *A* designated tracer ring signature scheme DTRS *is a tuple of algorithms* (Setup, T.KGen, U.KGen, Sign, Verify, Trace, VerTrace) *such that*

Setup(1^λ) On input of security parameter 1^λ, algorithm Setup outputs public parameters pp. We assume that pp includes the message space M and the randomness space Rand.

T.KGen(pp) On input of public parameters pp, algorithm T.KGen outputs a tracer key pair $(pk_\mathsf{T}, sk_\mathsf{T})$ where pk_T is the tracer's public key and sk_T is the tracer's secret key. We write that $pk_\mathsf{T} \leftarrow_\$ \mathsf{T.KGen}(pp, sk_\mathsf{T})$.

$\mathsf{U.KGen}(pp)$ On input of public parameters pp algorithm $\mathsf{U.KGen}$ outputs a user key pair (pk_{id}, sk_{id}) where pk_{id} is the user's public key and sk_{id} is the user's secret key. We write that $pk_{id} \leftarrow\!\!\$\ \mathsf{U.KGen}(pp, sk_{id})$.

$\mathsf{Sign}(pp, sk_{id}, pk_{\mathsf{T}}, m, R)$ On input of public parameters pp, user secret key sk_{id}, tracer public key pk_{T}, message m and ring R, algorithm Sign outputs a signature σ.

$\mathsf{Verify}(pp, pk_{\mathsf{T}}, m, R, \sigma)$ On input of public parameters pp, tracer public key pk_{T}, message m, ring R and signature σ, algorithm Verify outputs 1 if σ is a valid signature on m with respect to R, and 0 otherwise.

$\mathsf{Trace}(pp, sk_{\mathsf{T}}, m, R, \sigma)$ On input of public parameters pp, tracer secret key sk_{T}, message m, ring R and signature σ, algorithm Trace outputs the trace $\mathtt{Tr}$, where the exact form of $\mathtt{Tr}$ is defined for each functionality we consider.

$\mathsf{VerTrace}(pp, pk_{\mathsf{T}}, m, R, \sigma, \mathtt{Tr})$ On input of public parameters pp, tracer public key pk_{T}, message m, ring R, signature σ and trace $\mathtt{Tr}$, algorithm $\mathsf{VerTrace}$ outputs 1 if the trace is valid, and 0 otherwise.

We define correctness for our syntax as the property that honestly generated signatures are verifiable.

Definition 77 (Correctness)**.** *A* DTRS *ring signature satisfies* correctness *if, for any* $n = \mathsf{poly}(\lambda)$*,* $j \in [n]$ *and message* $m \in \mathsf{M}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\begin{array}{l} pp \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda); \\ (pk_{\mathsf{T}}, sk_{\mathsf{T}}) \leftarrow\!\!\$\ \mathsf{T.KGen}(pp); \\ \textbf{for } i = 1,\ldots,n : (pk_{\mathsf{S}_i}, sk_{\mathsf{S}_i}) \leftarrow\!\!\$\ \mathsf{U.KGen}(pp); \\ R \leftarrow \{pk_{\mathsf{S}_1}, \ldots, pk_{\mathsf{S}_n}\}; \\ \sigma \leftarrow\!\!\$\ \mathsf{Sign}(pp, sk_{\mathsf{S}_j}, pk_{\mathsf{T}}, m, R) \end{array} : \mathsf{Verify}(pp, pk_{\mathsf{T}}, m, R, \sigma) := 1\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

We now demonstrate how our generic syntax can capture linkable, link and traceable and accountable ring signature schemes.

Linkable ring signatures. Linkable ring signatures (e.g., [6, 98, 99, 124]) allow two signatures produced by the same user to be linked. This does not reveal the identity of the signer; it only determines whether the same user produced both signatures. To this end,

algorithm Trace, from Definition 76, is defined as follows.

$$\mathsf{Trace}(pp, sk_{\mathsf{T}}, (m_1, m_2), R, (\sigma_1, \sigma_2)) \quad = \quad \begin{cases} 1 & : sk_{id_i} = sk_{id_j} \\ 0 & : sk_{id_i} \neq sk_{id_j} \end{cases} \tag{6.7}$$

where $\sigma_1 \leftarrow_{\$} \mathsf{Sign}(pp, sk_{id_i}, pk_{\mathsf{T}}, m_1, R)$ and $\sigma_2 \leftarrow_{\$} \mathsf{Sign}(pp, sk_{id_j}, pk_{\mathsf{T}}, m_2, R)$. That is, algorithm Trace outputs a bit to indicate whether two signatures are produced by the same user. It takes two message/signature pairs as input and outputs $\mathtt{Tr} = 1$ if both signatures are produced by the same user, and $\mathtt{Tr} = 0$ otherwise. Linkable ring signatures do not require a public VerTrace algorithm. As such, we define VerTrace to always output 1.

Link and traceable ring signatures. Link and traceable ring signatures (e.g., [67, 68, 74, 97]) link ring signatures produced by the same user, and also reveal the identity of the signer if two signatures are linked. Link and traceable ring signatures require that the two signatures must be produced with respect to the same 'tag', which could, for example, indicate an election and ensure that, if a user signs more than one vote in a particular election, their identity is revealed. For simplicity, we consider the tag to be included in the ring, which is also the case in [68]. Algorithm Trace is defined as

$$\mathsf{Trace}(pp, sk_{\mathsf{T}}, (m_1, m_2), R, (\sigma_1, \sigma_2)) \quad = \quad \begin{cases} 0 & : sk_{id_i} \neq sk_{id_j} \\ 1 & : \text{else if } m_1 = m_2 \\ pk_{id_i} & : \text{ otherwise} \end{cases} \tag{6.8}$$

where $\sigma_1 \leftarrow_{\$} \mathsf{Sign}(pp, sk_{id_i}, pk_{\mathsf{T}}, m_1, R)$ and $\sigma_2 \leftarrow_{\$} \mathsf{Sign}(pp, sk_{id_j}, pk_{\mathsf{T}}, m_2, R)$. Then, algorithm Trace not only determines whether two message/signature pairs are linked, but also outputs the identity of a signer if the signer produces two signatures with respect to different messages and the same ring. As with linkable ring signatures, link and traceable ring signatures do not require a public VerTrace algorithm.

Accountable ring signatures. Accountable ring signatures [31, 130] provide a tracing function whereby a designated tracer can reveal the identity of a signer, and can provide a proof of correct tracing. In comparison to linkable and link and traceable ring signatures, accountable ring signatures do not require two message/signature pairs to reveal an identity.

Then, algorithm Trace is defined as

$$\mathsf{Trace}(pp, sk_{\mathsf{T}}, m, R, \sigma) = (pk_{id}, \rho_t) \tag{6.9}$$

where pk_{id} is the identity of the signer and ρ_t is a proof that the tracer identified the signer correctly. Accountable ring signatures are also furnished with a public verification algorithm $\mathsf{VerTrace}$ that allows anyone to check that the trace is computed correctly.

6.6.2 Security Model

We present a security model that includes definitions of anonymity and unforgeability that are defined analogously to our security model for R&T ring signatures. We then define notions of traceability for linkable, link and traceable, and accountable ring signatures. Respectively, we call these notions linkability, link and traceability, and accountability. As in Chapter 6.3, our security model considers signers and tracers that are honest, corrupt, or under the control of the attacker.

We define the oracles for our security experiments in Figure 6.24. These oracles are identical to the oracles for our R&T security model, with the following two exceptions. Firstly, we omit oracle $\mathcal{O}\mathsf{report}$ as we no longer require an oracle that returns a report. Secondly, as algorithm Trace is defined for each tracing function, we define oracle $\mathcal{O}\mathsf{X\text{-}trace}$ for $\mathsf{X} = \{\mathsf{LINK}, \mathsf{L\&T}, \mathsf{ACC}\}$ that runs algorithm Trace as defined for each tracing function and outputs the result.

$\underline{\mathcal{O}\mathsf{reg}_{(pp,\mathsf{L},\mathbf{pk}_{\mathsf{S}},\mathbf{sk}_{\mathsf{S}})}(id)}$
$(\mathbf{pk}_{\mathsf{S}}[id], \mathbf{sk}_{\mathsf{S}}[id]) \leftarrow\!\!\$\ \mathsf{U.KGen}(pp)$
$\mathsf{L} \leftarrow \mathsf{L} \cup \{\mathbf{pk}_{\mathsf{S}}[id]\}$
return $\mathbf{pk}_{\mathsf{S}}[id]$

$\underline{\mathcal{O}\mathsf{corrupt}_{(\mathsf{L},\mathsf{corL},\mathbf{pk}_{\mathsf{S}},\mathbf{sk}_{\mathsf{S}})}(id)}$
if $\mathbf{pk}_{\mathsf{S}}[id] \notin \mathsf{L}$ **return** $\perp$
$\mathsf{corL} \leftarrow \mathsf{corL} \cup \{\mathbf{pk}_{\mathsf{S}}[id]\}$
return $\mathbf{sk}_{\mathsf{S}}[id]$

$\underline{\mathcal{O}\mathsf{sign}_{(pp,\mathsf{L},\mathcal{Q}\mathsf{sign},\mathbf{pk}_{\mathsf{S}},\mathbf{sk}_{\mathsf{S}})}(id, pk_{\mathsf{T}}, m, R)}$
if $\mathbf{pk}_{\mathsf{S}}[id] \notin \mathsf{L}$ **return** $\perp$
$\sigma \leftarrow\!\!\$\ \mathsf{Sign}(pp, \mathbf{sk}_{\mathsf{S}}[id], pk_{\mathsf{T}}, m, R \cup \{\mathbf{pk}_{\mathsf{S}}[id]\})$
$\mathcal{Q}\mathsf{sign} \leftarrow \mathcal{Q}\mathsf{sign} \cup \{(pk_{\mathsf{T}}, \mathbf{pk}_{\mathsf{S}}[id], m, R, \sigma)\}$
return σ

$\underline{\mathcal{O}\mathsf{LINK\text{-}trace}_{(pp,sk_{\mathsf{T}},\mathcal{Q}\mathsf{trace})}(m_1, m_2, R, \sigma_1, \sigma_2)}$
$b \leftarrow\!\!\$\ \mathsf{Trace}(pp, sk_{\mathsf{T}}, (m_1, m_2), R, (\sigma_1, \sigma_2))$
$\mathcal{Q}\mathsf{trace} \leftarrow \mathcal{Q}\mathsf{trace} \cup \{(m_1, R, \sigma_1), (m_2, R, \sigma_2)\}$
return b

$\underline{\mathcal{O}\mathsf{L\&T\text{-}trace}_{(pp,sk_{\mathsf{T}},\mathcal{Q}\mathsf{trace})}(m_1, m_2, R, \sigma_1, \sigma_2)}$
$b \leftarrow\!\!\$\ \mathsf{Trace}(pp, sk_{\mathsf{T}}, (m_1, m_2), R, (\sigma_1, \sigma_2))$
$\mathcal{Q}\mathsf{trace} \leftarrow \mathcal{Q}\mathsf{trace} \cup \{(m_1, R, \sigma_1), (m_2, R, \sigma_2)\}$
return b

$\underline{\mathcal{O}\mathsf{ACC\text{-}trace}_{(pp,sk_{\mathsf{T}},\mathcal{Q}\mathsf{trace})}(m, R, \sigma)}$
$(pk_{id}, \rho_t) \leftarrow\!\!\$\ \mathsf{Trace}(pp, sk_{\mathsf{T}}, m, R, \sigma)$
$\mathcal{Q}\mathsf{trace} \leftarrow \mathcal{Q}\mathsf{trace} \cup \{(m, R, \sigma)\}$
return (pk_{id}, ρ_t)

Figure 6.24: Oracles used in the experiments for anonymity, unforgeability and traceability of a DTRS ring signature scheme.

Anonymity. We define anonymity against adversarially generated keys, capturing an adversary that is provided with a challenge signature for one of two potential , honest, signers and cannot determine which signer produced the signature. We require that the adversary

does not query the challenge signature to oracle $\mathcal{O}\mathsf{X}\text{-}\mathsf{trace}$ for $\mathsf{X} = \{\mathsf{LINK}, \mathsf{L\&T}, \mathsf{ACC}\}$, thereby ensuring that the adversary does not obtain the identity of the signer via the tracing function. We define anonymity in Definition 78 below.

Definition 78 (DTRS-anonymity)**.** *A* DTRS *ring signature satisfies* anonymity with respect to adversarially generated keys *if, for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exist a negligible function* negl *such that,*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{X}\text{-}\mathsf{anon},0}_{\mathsf{DTRS},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{X}\text{-}\mathsf{anon},1}_{\mathsf{DTRS},\mathcal{A}}(\lambda)\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{X}\text{-}\mathsf{anon},\beta}_{\mathsf{DTRS},\mathcal{A}}(\lambda)$ *is the experiment defined in Figure 6.25 for* $\beta \in \{0,1\}$*, and* $\mathsf{X} \in \{\mathsf{LINK}, \mathsf{L\&T}, \mathsf{ACC}\}$*.*

$\mathsf{Exp}^{\mathsf{X}\text{-}\mathsf{anon}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)$

$\mathbf{pk}_\mathsf{S} \leftarrow (); \ \mathbf{sk}_\mathsf{S} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{trace} \leftarrow \emptyset$
$pp \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(pk_\mathsf{T}, sk_\mathsf{T}) \leftarrow\!\!\$\ \mathsf{T.KGen}(pp)$
$(m^*, R^*, id_0, id_1, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{X}\text{-}\mathsf{trace}}(pp, pk_\mathsf{T})$
$\sigma^* \leftarrow\!\!\$\ \mathsf{Sign}(pp, \mathbf{sk}_\mathsf{S}[id_\beta], pk_\mathsf{T}, m^*, R^* \cup \{\mathbf{pk}_\mathsf{S}[id_0], \mathbf{pk}_\mathsf{S}[id_1]\})$
$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{X}\text{-}\mathsf{trace}}(\sigma^*, st)$
if $\{\mathbf{pk}_\mathsf{S}[id_0], \mathbf{pk}_\mathsf{S}[id_1]\} \subseteq \mathsf{L}\backslash\mathsf{corL} \ \wedge \ (m^*, R^* \cup \{\mathbf{pk}_\mathsf{S}[id_0], \mathbf{pk}_\mathsf{S}[id_1]\}, \sigma^*) \notin \mathcal{Q}\mathsf{trace}$
  return β'

$\mathsf{Exp}^{\mathsf{unfor}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)$

$\mathbf{pk}_\mathsf{S} \leftarrow (); \ \mathbf{sk}_\mathsf{S} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{rep} \leftarrow \emptyset$
$pp \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(pk_\mathsf{T}, m^*, R^*, \sigma^*) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign}}(pp)$
if $\mathsf{Verify}(pp, pk_\mathsf{T}, m^*, R^*, \sigma^*) = 1 \ \wedge \ R \subseteq \mathsf{L}\backslash\mathsf{corL}$
$\wedge \ (pk_\mathsf{T}, \cdot, m^*, R^*, \sigma^*) \notin \mathcal{Q}\mathsf{sign}$ **return** 1
else return 0

Figure 6.25: The experiments for anonymity and unforgeability of a DTRS scheme where oracles used in the experiment are defined in Figure 6.24.

Unforgeability. Unforgeability for a DTRS scheme is defined identically to unforgeability for an R&T ring signature, with the exception that the adversary in the unforgeability experiment does not have access to an oracle $\mathcal{O}\mathsf{report}$.

Definition 79 (DTRS-unforgeability)**.** *A* DTRS *ring signature satisfies* unforgeability *if, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\mathsf{Exp}^{\mathsf{unfor}}_{\mathsf{DTRS},\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{unfor}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)$ *is the experiment defined in Figure 6.25.*

Linkability. We define linkability to incorporate three requirements: link correctness, non-frameability and linking soundness. In doing so, our definition of linkability captures the security model for linkable ring signatures with a designated linker that is presented

in [98], with the exception of a fourth condition. This condition requires that an attacker cannot determine the linkability of two signatures without access to the tracer's secret key. We omit this from our definition of linkability as it is captured by our anonymity definition for a DTRS signature scheme. We provide a brief overview of our linkability requirements and formally define linkability in Definition 80.

Link correctness requires that honestly generated signatures will only trace to the same signer if they are produced by the same signer. That is, in our link correctness experiment, we require that algorithm Trace outputs 1 if two signatures are produced by the same user, and outputs 0 otherwise. *Non-frameability* requires that an attacker cannot produce a message/signature pair on behalf of an honest signer such that algorithm Trace, on input of a second message/signature pair for an honest user, returns 1. To this end, our non-frameability condition, which is based on the non-frameability condition of [98] challenges the adversary, with control of the tracer and input of a signature produced on behalf of an honest user chosen by the adversary, to output a message/signature pair such that both signatures are deemed to originate from the same honest signer. Finally, *soundness* captures the condition that an attacker cannot produce two message/signature pairs on behalf of a single user such that algorithm Trace returns 0. Our soundness condition models an adversary that, with control of the tracer, outputs two message/signature pairs such that algorithm Trace returns 0. We require that the adversary can corrupt at most one user in the ring; otherwise the adversary can return two message/signature pairs signed by two different ring members and algorithm Trace trivially outputs 0.

Definition 80 (Linkability)**.** *A* DTRS *ring signature scheme satisfies* linkability *if algorithm* Trace *is defined as in Equation 6.7, algorithm* VerTrace *outputs 1, and the following conditions are satisfied.*

1. ***Link correctness****: for any* $n = \mathsf{poly}(\lambda)$*,* $i, j \in [n]$ *where* $i \neq j$*, and messages* $m_1, m_2 \in \mathsf{M}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\mathsf{Exp}^{\mathsf{LINKcorr}}_{\mathsf{DTRS}}(\lambda) = 1\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

2. ***Non-frameability****: for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\mathsf{Exp}^{\mathsf{LINKframe}}_{\mathsf{DTRS},\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,.$$

3. ***Soundness**: for any PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that,*

$$\Pr\left[\mathsf{Exp}^{\mathsf{LINKsound}}_{\mathsf{DTRS},\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{LINKcorr}}_{\mathsf{DTRS}}(\lambda)$, $\mathsf{Exp}^{\mathsf{LINKframe}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)$ *and* $\mathsf{Exp}^{\mathsf{LINKsound}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)$ *are the experiments defined in Figure 6.26.*

Link and traceability. A security model for link and traceable (public tracer) ring signatures is presented in [68] and adapted to designated tracing in [74]. Both models require that three properties hold: correctness, non-frameability (called exculpability in [68, 74] and non-slanderability in [97]), and a soundness condition (known as linkability or tag-linkability in the link and trace ring signature literature). We formally define link and traceability in Definition 81, capturing these three requirements.

We define *link and trace correctness* as the property that algorithm Trace returns 0 if two signatures input to the algorithm are produced by different signers. Otherwise, algorithm Trace returns 1 (resp., the signer's identity) if the signatures are generated with respect to the same message (resp., the messages are different and are generated by the same signer). Our *non-frameability* requirement is defined similarly to non-frameability for a linkable ring signature and captures exculpability, as defined in [98]. We require that an adversary, with control of the tracer and input of a signature produced on behalf of an honest user chosen by the adversary, cannot output a message/signature pair such that algorithm Trace returns the public key of the honest signer. *Soundness* is identical to the soundness condition for linkable ring signatures.

Definition 81 (Link and traceability)**.** *A* DTRS *ring signature scheme satisfies* link and traceability *if algorithm* Trace *is defined as in Equation 6.8, algorithm* $\mathsf{VerTrace}$ *outputs 1, and the following conditions are satisfied.*

1. ***Link and trace correctness**: for any $n = \mathsf{poly}(\lambda)$, $i, j \in [n]$ where $i \neq j$, and messages $m_1, m_2 \in \mathsf{M}$, there exists a negligible function* negl *such that,*

$$\Pr\left[\mathsf{Exp}^{\mathsf{L\&Tcorr}}_{\mathsf{DTRS}}(\lambda) = 1\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

2. ***Non-frameability**: for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible*

function negl *such that,*

$$\Pr\left[\mathsf{Exp}^{\mathsf{L\&Tframe}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)=1\right]\leq \mathsf{negl}(\lambda)\,.$$

3. ***Soundness****: for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\mathsf{Exp}^{\mathsf{LINKsound}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)=1\right]\leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{L\&Tcorr}}_{\mathsf{DTRS}}(\lambda)$*,* $\mathsf{Exp}^{\mathsf{L\&Tframe}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)$ *and* $\mathsf{Exp}^{\mathsf{LINKsound}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)$ *are the experiments defined in Figure 6.26*

Accountability. Accountable ring signatures are defined in [31] and a security framework is provided. As expected, it incorporates notions of correctness, non-frameability and soundness. Definition 82 adapts the security properties from [31] to our syntax and is very similar to the definition of traceability for R&T schemes, with the notable omission of a report mechanism. We recall that trace correctness is the basic requirement that any trace output by the designated trace must be valid; non-frameability requires that an honest user cannot be accused of producing a signature they did not produce; and soundness demands that it is possible to trace only *one* signer for any signature.

Definition 82 (Accountability)**.** *A* DTRS *scheme is* accountable *if algorithm* Trace *is defined as in Equation 6.9 and the following conditions are satisfied.*

1. ***Accountable trace correctness****: for any* $n=\mathsf{poly}(\lambda)$*,* $i\in[n]$ *and message* $m\in\mathsf{M}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\mathsf{Exp}^{\mathsf{ACCcorr}}_{\mathsf{DTRS}}(\lambda)=1\right]\geq 1-\mathsf{negl}(\lambda)\,.$$

2. ***Non-frameability****: for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\mathsf{Exp}^{\mathsf{ACCframe}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)=1\right]\leq \mathsf{negl}(\lambda)\,.$$

3. ***Soundness****: for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that,*

$$\Pr\left[\mathsf{Exp}^{\mathsf{ACCsound}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)=1\right]\leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{ACCcorr}}_{\mathsf{DTRS}}(\lambda)$, $\mathsf{Exp}^{\mathsf{ACCframe}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)$ *and* $\mathsf{Exp}^{\mathsf{ACCsound}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)$ *are the experiments defined in Figure 6.26.*

$\mathsf{Exp}^{\mathsf{LINKcorr}}_{\mathsf{DTRS}}(\lambda)$

$pp \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(pk_\mathsf{T}, sk_\mathsf{T}) \leftarrow\!\!\$\ \mathsf{T.KGen}(pp)$
for $i = 1, \ldots, n$: $(pk_{\mathsf{S}_i}, sk_{\mathsf{S}_i}) \leftarrow\!\!\$\ \mathsf{U.KGen}(pp)$
$R \leftarrow \{pk_{\mathsf{S}_1}, \ldots, pk_{\mathsf{S}_n}\}$
$\sigma_1 \leftarrow\!\!\$\ \mathsf{Sign}(pp, sk_{\mathsf{S}_i}, pk_\mathsf{T}, m_1, R)$
$\sigma_2 \leftarrow\!\!\$\ \mathsf{Sign}(pp, sk_{\mathsf{S}_j}, pk_\mathsf{T}, m_2, R)$
$\mathtt{Tr} \leftarrow\!\!\$\ \mathsf{Trace}(pp, sk_\mathsf{T}, (m_1, m_2), R, (\sigma_1, \sigma_2))$
if $(\mathtt{Tr} = 1 \ \wedge\ sk_{\mathsf{S}_i} = sk_{\mathsf{S}_j}) \ \vee\ (\mathtt{Tr} = 0 \ \wedge\ sk_{\mathsf{S}_i} \neq sk_{\mathsf{S}_j})$ **return** 1
else return 0

$\mathsf{Exp}^{\mathsf{L\&Tcorr}}_{\mathsf{DTRS}}(\lambda)$

$pp \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(pk_\mathsf{T}, sk_\mathsf{T}) \leftarrow\!\!\$\ \mathsf{T.KGen}(pp)$
for $i = 1, \ldots, n$: $(pk_{\mathsf{S}_i}, sk_{\mathsf{S}_i}) \leftarrow\!\!\$\ \mathsf{U.KGen}(pp)$
$R \leftarrow \{pk_{\mathsf{S}_1}, \ldots, pk_{\mathsf{S}_n}\}$
$\sigma_1 \leftarrow\!\!\$\ \mathsf{Sign}(pp, sk_{\mathsf{S}_i}, pk_\mathsf{T}, m_1, R)$
$\sigma_2 \leftarrow\!\!\$\ \mathsf{Sign}(pp, sk_{\mathsf{S}_j}, pk_\mathsf{T}, m_2, R)$
$\mathtt{Tr} \leftarrow\!\!\$\ \mathsf{Trace}(pp, sk_\mathsf{T}, (m_1, m_2), R, (\sigma_1, \sigma_2))$
if $(\mathtt{Tr} = 0 \ \wedge\ sk_{\mathsf{S}_i} \neq sk_{\mathsf{S}_j}) \ \vee\ (\mathtt{Tr} = 1 \ \wedge\ sk_{\mathsf{S}_i} = sk_{\mathsf{S}_j} \ \wedge\ m_1 = m_2)$
$\vee\ (\mathtt{Tr} = pk_{\mathsf{S}_i} \ \wedge\ sk_{\mathsf{S}_i} = sk_{\mathsf{S}_j} \ \wedge\ m_1 \neq m_2)$ **return** 0
else return 0

$\mathsf{Exp}^{\mathsf{ACCcorr}}_{\mathsf{DTRS}}(\lambda)$

$pp \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(pk_\mathsf{T}, sk_\mathsf{T}) \leftarrow\!\!\$\ \mathsf{T.KGen}(pp)$
for $i = 1, \ldots, n$: $(pk_{\mathsf{S}_i}, sk_{\mathsf{S}_i}) \leftarrow\!\!\$\ \mathsf{U.KGen}(pp)$
$R \leftarrow \{pk_{\mathsf{S}_1}, \ldots, pk_{\mathsf{S}_n}\}$
$\sigma \leftarrow\!\!\$\ \mathsf{Sign}(pp, sk_{\mathsf{S}_i}, pk_\mathsf{T}, m, R)$
$(pk_\mathsf{S}, \rho_t) \leftarrow\!\!\$\ \mathsf{Trace}(pp, sk_\mathsf{T}, m, R, \sigma)$
$b \leftarrow\!\!\$\ \mathsf{VerTrace}(pp, pk_\mathsf{T}, m, R, \sigma, (pk_\mathsf{S}, \rho_t))$
return b

$\mathsf{Exp}^{\mathsf{LINKframe}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)$

$\mathbf{pk}_\mathsf{S} \leftarrow ();\ \mathbf{sk}_\mathsf{S} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset;\ \mathsf{corL} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset$
$pp \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(sk_\mathsf{T}, pk_\mathsf{S}, m_1, R, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}^{\mathsf{reg}}, \mathcal{O}^{\mathsf{corrupt}}, \mathcal{O}^{\mathsf{sign}}}(pp)$
$pk_\mathsf{T} \leftarrow\!\!\$\ \mathsf{T.KGen}(pp, sk_\mathsf{T})$
$\sigma_1 \leftarrow\!\!\$\ \mathsf{Sign}(pp, sk_\mathsf{S}, pk_\mathsf{T}, m_1, R)$
$(m_2, \sigma_2) \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}^{\mathsf{reg}}, \mathcal{O}^{\mathsf{corrupt}}, \mathcal{O}^{\mathsf{sign}}}(\sigma_1, st)$
if $\mathsf{Trace}(pp, sk_\mathsf{T}, (m_1, m_2), R, (\sigma_1, \sigma_2)) = 1$
$\wedge\ \mathsf{Verify}(pp, pk_\mathsf{T}, m_2, R, \sigma_2) = 1$
$\wedge\ (pk_\mathsf{T}, pk_\mathsf{S}, m_2, R, \sigma_2) \notin \mathcal{Q}\mathsf{sign} \ \wedge\ pk_\mathsf{S} \in \mathsf{L} \setminus \mathsf{corL}$
return 1

$\mathsf{Exp}^{\mathsf{L\&Tframe}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)$

$\mathbf{pk}_\mathsf{S} \leftarrow ();\ \mathbf{sk}_\mathsf{S} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset;\ \mathsf{corL} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset$
$pp \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(sk_\mathsf{T}, pk_\mathsf{S}, m_1, R, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}^{\mathsf{reg}}, \mathcal{O}^{\mathsf{corrupt}}, \mathcal{O}^{\mathsf{sign}}}(pp)$
$pk_\mathsf{T} \leftarrow\!\!\$\ \mathsf{T.KGen}(pp, sk_\mathsf{T})$
$\sigma_1 \leftarrow\!\!\$\ \mathsf{Sign}(pp, sk_\mathsf{S}, pk_\mathsf{T}, m_1, R)$
$(m_2, \sigma_2) \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}^{\mathsf{reg}}, \mathcal{O}^{\mathsf{corrupt}}, \mathcal{O}^{\mathsf{sign}}}(\sigma_1, st)$
if $\big((\mathsf{Trace}(pp, sk_\mathsf{T}, (m_1, m_2), R, (\sigma_1, \sigma_2)) = pk_\mathsf{S} \ \wedge m_1 \neq m_2)$
$\vee\ (\mathsf{Trace}(pp, sk_\mathsf{T}, (m_1, m_2), R, (\sigma_1, \sigma_2)) = 1 \ \wedge\ m_1 = m_2)\big)$
$\wedge\ \mathsf{Verify}(pp, pk_\mathsf{T}, m_2, R, \sigma_2) = 1$
$\wedge\ (pk_\mathsf{T}, pk_\mathsf{S}, m_2, R, \sigma_2) \notin \mathcal{Q}\mathsf{sign} \ \wedge\ pk_\mathsf{S} \in \mathsf{L} \setminus \mathsf{corL}$
return 1

$\mathsf{Exp}^{\mathsf{ACCframe}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)$

$\mathbf{pk}_\mathsf{S} \leftarrow ();\ \mathbf{sk}_\mathsf{S} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset;\ \mathsf{corL} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset$
$pp \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(pk_\mathsf{T}, m^*, R^*, \sigma^*, pk_\mathsf{S}, \rho_t) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}^{\mathsf{reg}}, \mathcal{O}^{\mathsf{corrupt}}, \mathcal{O}^{\mathsf{sign}}}(pp)$
if $\mathsf{VerTrace}(pp, pk_\mathsf{T}, m^*, R^*, \sigma^*, (pk_\mathsf{S}, \rho_t)) = 1$
$\wedge\ \mathsf{Verify}(pp, pk_\mathsf{T}, m^*, R^*, \sigma^*) = 1$
$\wedge\ (pk_\mathsf{T}, \cdot, m^*, R^*, \sigma^*) \notin \mathcal{Q}\mathsf{Sign} \ \wedge\ pk_\mathsf{S} \in \mathsf{L} \setminus \mathsf{corL}$
return 1

$\mathsf{Exp}^{\mathsf{LINKsound}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)$

$\mathbf{pk}_\mathsf{S} \leftarrow ();\ \mathbf{sk}_\mathsf{S} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset;\ \mathsf{corL} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset$
$pp \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(sk_\mathsf{T}, m_1, m_2, R, \sigma_1, \sigma_2) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}^{\mathsf{reg}}, \mathcal{O}^{\mathsf{corrupt}}, \mathcal{O}^{\mathsf{sign}}}(pp)$
$pk_\mathsf{T} \leftarrow\!\!\$\ \mathsf{T.KGen}(pp, sk_\mathsf{T})$
if $\mathsf{Trace}(pp, sk_\mathsf{T}, (m_1, m_1), R, (\sigma_1, \sigma_2)) = 0 \ \wedge\ |R \cap \mathsf{corL}| \leq 1$
$\wedge\ \big(\text{ for } i \in (1, 2) : \mathsf{Verify}(pp, pk_\mathsf{T}, m_i, R, \sigma_i) = 1$
$\wedge\ (pk_\mathsf{T}, \cdot, m_i, R, \sigma_i) \notin \mathcal{Q}\mathsf{Sign}\big)$
return 1

$\mathsf{Exp}^{\mathsf{ACCsound}}_{\mathsf{DTRS},\mathcal{A}}(\lambda)$

$\mathbf{pk}_\mathsf{S} \leftarrow ();\ \mathbf{sk}_\mathsf{S} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset;\ \mathsf{corL} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset$
$pp \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(pk_\mathsf{T}, m^*, R^*, \sigma^*, (pk_{\mathsf{S}_i}, \rho_{t_i}), (pk_{\mathsf{S}_j}, \rho_{t_j})) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}^{\mathsf{reg}}, \mathcal{O}^{\mathsf{corrupt}}, \mathcal{O}^{\mathsf{sign}}}(pp)$
if $\mathsf{VerTrace}(pp, pk_\mathsf{T}, m^*, R^*, \sigma^*, (pk_{\mathsf{S}_i}, \rho_{t_i})) = 1$
$\wedge\ \mathsf{VerTrace}(pp, pk_\mathsf{T}, m^*, R^*, \sigma^*, (pk_{\mathsf{S}_j}, \rho_{t_j})) = 1 \ \wedge\ pk_{\mathsf{S}_i} \neq pk_{\mathsf{S}_j}$
return 1

Figure 6.26: The experiments for linkability, link and traceability and accountability of a DTRS scheme where oracles used in the experiment are defined in Figure 6.24.

6.6.3 Comparing Functionalities

In this section we use the security definitions presented in Chapter 6.6.2 to compare tracing functionalities. We show that accountability implies link and traceability, which implies linkability. Interestingly, this means that, even though the tracing function of accountable ring signatures reveals more information than the tracing function of link and traceable ring signatures, we can still construct a secure link and traceable ring signature from an accountable ring signature. This result holds because the additional information that is revealed in an accountable ring signature is known only to the designated tracer and is never publicly revealed. A similar argument holds for link and traceable ring signatures and linkable ring signatures.

Recall that, in our syntax, algorithm Trace is defined for each functionality. For clarity, in this section, we rename algorithm Trace to Link, LinkTr and Acc for linkable, link and traceable, and accountable ring signatures, respectively. In this section, we say that DTRS is a X secure signature scheme for $\mathsf{X} \in \{\mathsf{LINK}, \mathsf{L\&T}, \mathsf{ACC}\}$ if it satisfies X-anonymity, unforgeability and the relevant tracing notion.

Our analyses proceed as follows. We show that accountability implies link and traceability by first defining a modified designated tracer ring signature scheme that is assumed to be a secure accountable ring signature. We then demonstrate that the modified scheme is a secure link and traceable ring signature. We note that, in our analysis, we do not consider the unforgeability property as unforgeability is defined analogously for each tracing function. We then perform the same steps to demonstrate that link and traceability implies linkability and, finally, that report and trace implies accountability.

Accountability implies link and traceability. We define a designated tracer ring signature DTRS' to have algorithm LinkTr defined as in Equation 6.8 that internally runs algorithm Acc as defined in Equation 6.9.

Definition 83 (Ring signature DTRS')**.** *Let* $\mathsf{DTRS} = (\mathsf{Setup}, \mathsf{T.KGen}, \mathsf{U.KGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Acc}, \mathsf{VerTrace})$ *be a secure accountable ring signature. We define a modified scheme* $\mathsf{DTRS}' = (\mathsf{Setup}, \mathsf{T.KGen}, \mathsf{U.KGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{LinkTr}, \mathsf{VerTrace})$ *such that algorithm* LinkTr *is defined as follows.*

$\underline{\mathsf{LinkTr}(pp, sk_{\mathsf{T}}, (m_1, m_2), R, (\sigma_1, \sigma_2))}$

$(pk_{\mathsf{S}_1}, \rho_{t,1}) \leftarrow\!\!\$\ \mathsf{Acc}(pp, sk_{\mathsf{T}}, m_1, R, \sigma_1)$

$(pk_{\mathsf{S}_2}, \rho_{t,2}) \leftarrow\!\!\$\ \mathsf{Acc}(pp, sk_{\mathsf{T}}, m_2, R, \sigma_2)$

if $pk_{\mathsf{S}_1} \neq pk_{\mathsf{S}_2}$ **return** 0

if $pk_{\mathsf{S}_1} = pk_{\mathsf{S}_2} \wedge m_1 = m_2$ **return** 1

else return pk_{S_1}

We now prove that accountability implies link and traceability. We obtain Theorem 26, which we prove via a series of Lemmata.

Theorem 26 ($\mathsf{ACC} \Rightarrow \mathsf{L\&T}$)**.** *Let* DTRS *be a secure accountable ring signature scheme. Then we can construct a modified ring signature* DTRS′ *that is a secure link and traceable ring signature scheme.*

We prove Theorem 26 via a series of Lemmata.

Lemma 14. *Let* DTRS *be a ring signature that satisfies* ACC-anon*. Then* DTRS′ *satsifes* L&T-anon*.*

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary that succeeds in the L&T-anon experiment against scheme DTRS′. We show that we can construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that succeeds in the ACC-anon experiment against scheme DTRS. We present $\mathcal{B}$ in Figure 6.27. It is clear that inputs to $\mathcal{A}$ are distributed identically to the L&T-anon experiment. In fact, public parameters pp, tracer public key pk_{T} and challenge signature σ^* are generated identically. Moreover, all oracles are distributed identically. In particular, oracle $\mathcal{O}$L&T-trace is identical because $\mathcal{B}$ runs $\mathcal{O}$ACC-trace to obtain the identity of the signer of each message and subsequently returns the correct output to $\mathcal{A}$. We now show that $\mathcal{B}$ is successful in the ACC-anonymity experiment. By assumption, $\mathcal{A}$ outputs $\beta' = \beta$ without tracing the challenge signature or corrupting signers id_0 and id_1. That is, $\mathcal{A}$ correctly determines whether challenge signature σ^* is produced by signer id_0 or id_1. As σ^* is generated in the ACC-anonymity experiment, this means that, if $\mathcal{B}$ returns β' output by $\mathcal{A}$, then $\mathcal{B}$ correctly guesses β. Therefore, $\mathcal{B}$ succeeds in the ACC-anonymity experiment and, by contradiction, the result holds. □

Lemma 15. *Let* DTRS *be a ring signature that satisfies* ACCcorr*. Then* DTRS′ *satisfies* L&Tcorr*.*

$\underline{\mathcal{B}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{ACC\text{-}trace}}(pp, pk_\mathsf{T})}$

$\mathbf{pk}_\mathsf{S} \leftarrow (); \ \mathbf{sk}_\mathsf{S} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{trace} \leftarrow \emptyset$
$(m^*, R^*, id_0, id_1, st) \leftarrow\!\!\$ \ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{L\&T\text{-}trace}}(pp, pk_\mathsf{T})$
return $(m^*, R^*, id_0, id_1, st)$

$\underline{\mathcal{B}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{ACC\text{-}trace}}(\sigma^*, st)}$

$\beta' \leftarrow\!\!\$ \ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{L\&T\text{-}trace}}(\sigma^*, st)$
return β'

$\underline{\mathcal{O}\mathsf{L\&T\text{-}trace}_{(pp,sk_\mathsf{T},\mathcal{Q}\mathsf{trace})}(m_1, m_2, R, \sigma_1, \sigma_2)}$

$(pk_{\mathsf{S}_1}, \rho_{t,1}) \leftarrow\!\!\$ \ \mathcal{O}\mathsf{ACC\text{-}trace}(m_1, R, \sigma_1)$
$(pk_{\mathsf{S}_2}, \rho_{t,2}) \leftarrow\!\!\$ \ \mathcal{O}\mathsf{ACC\text{-}trace}(m_2, R, \sigma_2)$
$\mathcal{Q}\mathsf{trace} \leftarrow \mathcal{Q}\mathsf{trace} \cup \{(m_1, R, \sigma_1), (m_2, R, \sigma_2)\}$
if $pk_{\mathsf{S}_1} \neq pk_{\mathsf{S}_2}$ **return** 0
if $pk_{\mathsf{S}_1} = pk_{\mathsf{S}_2} \ \wedge \ m_1 = m_2$ **return** 1
else return pk_{S_1}

$\underline{\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{sign}}$

$\mathcal{B}$ calls the identical oracle in the ACC-anon experiment.

Figure 6.27: Adversary $\mathcal{B}$ that breaks the ACC-anon security of DTRS in the proof of Lemma 14.

Proof. Consider a trace $\mathtt{Tr}$ output by algorithm $\mathsf{LinkTr}(pp, sk_\mathsf{T}, (m_1, m_2), R, (\sigma_1, \sigma_2))$ that takes as input two signatures σ_1 and σ_2 for ring R and messages m_1 and m_2 respectively. Moreover, let sk_T be the tracer secret key generated by algorithm T.KGen and let pp be the public parameters generated by algorithm Setup. By definition, algorithm LinkTr returns 1, 0 or a signer public key pk_S. We consider these three possibilities in turn. First, assume that algorithm LinkTr returns 0. We assume that DTRS satisifes ACCcorr. Therefore, algorithm Acc returns the public key of the signer with overwhelming probability. As a result, if LinkTr returns 0, the signatures are produced by different signers as required. Similarly, if LinkTr returns 1 or pk_S, by the correctness property of DTRS, both signatures are signed by the same signer or for the same message, as required. We conclude that DTRS′ satisfies L&Tcorr. □

Lemma 16. *Let* DTRS *be a ring signature that satisfies* ACCframe*. Then* DTRS′ *satisfies* L&Tframe*.*

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary that succeeds in the L&T-frame experiment for scheme DTRS′. We show that we can construct an adversary $\mathcal{B}$ that succeeds in the ACC-frame experiment for scheme DTRS. We present $\mathcal{B}$ in Figure 6.28. It is clear that inputs to $\mathcal{A}$ are distributed identically to the L&T-frame experiment. In fact, public parameters pp and signature σ_1 are generated identically. Moreover, all oracles are distributed identically. We now show that $\mathcal{B}$ is successful in the ACC-frame experiment. That is, algorithms VerTrace and Verify return 1, the tuple output by adversary $\mathcal{B}$ is not queried to oracle $\mathcal{O}$sign and pk_S is an honest signer. We note that $\mathcal{B}$ honestly traces the correct signer of message m_2 (i.e., runs algorithm Trace). Then, by trace correctness of DTRS, algorithm VerTrace returns 1. By assumption, tuple $(pk_\mathsf{T}, pk_\mathsf{S}, m_2, R, \sigma_2)$ is not queried to oracle $\mathcal{O}$sign in the L&T-frame experiment and pk_S is honest. Moreover, algorithm Verify returns 1. Finally, by

assumption, $pk'_{\mathsf{S}} = pk_{\mathsf{S}}$ as $\mathcal{A}$ succeeds in the L&T-frame experiment. Therefore, $\mathcal{B}$ satisfies all conditions to succeed in the ACC-non-frameability experiment. We conclude that, by contradiction, the result holds. □

$\mathcal{B}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign}}(pp)$	$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{sign}$
$\mathbf{pk}_{\mathsf{S}} \leftarrow ();\ \mathbf{sk}_{\mathsf{S}} \leftarrow ()$	$\mathcal{B}$ calls the identical oracle
$\mathsf{L} \leftarrow \emptyset;\ \mathsf{corL} \leftarrow \emptyset;\ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset$	in the ACCframe experiment.
$(sk_{\mathsf{T}}, pk_{\mathsf{S}}, m_1, R, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign}}(pp)$	
$pk_{\mathsf{T}} \leftarrow\!\!\$\ \mathsf{T.KGen}(pp; sk_{\mathsf{T}})$	
$\sigma_1 \leftarrow\!\!\$\ \mathcal{O}\mathsf{sign}(id, pk_{\mathsf{T}}, m_1, R)$	
$(m_2, \sigma_2) \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign}}(\sigma_1, st)$	
$(pk'_{\mathsf{S}}, \rho_t) \leftarrow\!\!\$\ \mathsf{Acc}(pp, sk_{\mathsf{T}}, m, R, \sigma)$	
return $(pk_{\mathsf{T}}, m_2^*, R^*, \sigma_2^*, pk_{\mathsf{S}}, \rho_t)$	

Figure 6.28: Adversary $\mathcal{B}$ that breaks the ACCframe security of scheme DTRS in the proof of Lemma 16.

Lemma 17. *Let* DTRS *be a ring signature that satisfies* ACCframe *and* ACCcorr*. Then* DTRS′ *satisfies* LINKsound*.*

Proof. Let $\mathcal{A}$ be an adversary that succeeds in the L&T-sound experiment against scheme DTRS′. We show that we can construct an adversary $\mathcal{B}$ that succeeds in the ACC-frame experiment against scheme DTRS. We present $\mathcal{B}$ in Figure 6.29. It is clear that inputs to $\mathcal{A}$ are distributed identically to the L&T-sound experiment. In fact, public parameters pp are generated identically. Moreover, all oracles are distributed identically. We now show that $\mathcal{B}$ is successful in the ACC-frame experiment. That is, algorithm VerTrace and Verify return 1, the tuple output by adversary $\mathcal{B}$ is not queried to oracle $\mathcal{O}$sign and pk_{S} is an honest signer. By assumption, the signature output by $\mathcal{B}$ verifies (i.e., algorithm Verify returns 1) and the tuple $(pk_{\mathsf{T}}, pk_{\mathsf{S}_i}, m_i, R, \sigma_i) \notin \mathcal{Q}\mathsf{sign}$. Furthermore, the signer output by $\mathcal{B}$ is honest because adversary $\mathcal{A}$ outputs a ring that has, at most, 1 corrupt user. Indeed, in the L&T-sound experiment, adversary $\mathcal{A}$ must output one message/signature tuple that is a forgery (i.e., a tuple produced on behalf of an honest ring member that is valid) as, otherwise, by the ACCcorr property of DTRS, algorithm LinkTr returns 1. Finally, $\mathcal{B}$ honestly generates the proof of correct tracing, which verifies by the ACCcorr property of DTRS and $pk_{\mathsf{S}_1} \neq pk_{\mathsf{S}_2}$. Therefore, $\mathcal{B}$ satisfies all conditions to succeed in the ACC-non-frameability experiment. We conclude that, by contradiction, the result holds. □

$\mathcal{B}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign}}(pp)$

$\mathbf{pk}_{\mathsf{S}} \leftarrow ()$; $\mathbf{sk}_{\mathsf{S}} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset$; $\mathsf{corL} \leftarrow \emptyset$; $\mathcal{Q}\mathsf{sign} \leftarrow \emptyset$
$(sk_{\mathsf{T}}, m_1, m_2, R, \sigma_1, \sigma_2) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},}(pp)$
$pk_{\mathsf{T}} \leftarrow\!\!\$\ \mathsf{T.KGen}(pp; sk_{\mathsf{T}})$
$(pk_{\mathsf{S}_1}, \rho_{t,1}) \leftarrow\!\!\$\ \mathsf{Acc}(pp, sk_{\mathsf{T}}, m_1, R, \sigma_1)$
$(pk_{\mathsf{S}_2}, \rho_{t,2}) \leftarrow\!\!\$\ \mathsf{Acc}(pp, sk_{\mathsf{T}}, m_2, R, \sigma_2)$
return $(pk_{\mathsf{T}}, m_i^*, R^*, \sigma_i^*, pk_{\mathsf{S}_i}, \rho_{t,i})$ s.t. $pk_{\mathsf{S}_i} \in \mathsf{L} \setminus \mathsf{corL}$

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{sign}$

$\mathcal{B}$ calls the identical oracle in the ACC-frame experiment.

Figure 6.29: Adversary $\mathcal{B}$ that breaks the ACC-frame security of DTRS in the proof of Lemma 17.

Link and traceability implies linkability We now show that link and traceability implies linkability. This result is rather intuitive. In fact, if a tracing algorithm can link signatures and *additionally* trace a signer, then such an algorithm can also link signatures. Nonetheless, we prove this formally and obtain the formal result in Theorem 27, which we prove via a series of Lemmata. As with our previous results, we need not consider unforgeability in our analysis. Moreover, we need not consider the soundness requirement as it is defined identically for linkability and link and traceability. We first define a designated tracer ring signature DTRS′ that is a secure link and traceable ring signature, and show that it is also a secure linkable ring signature.

Definition 84 (Ring signature DTRS′)**.** *Let* DTRS = (Setup, T.KGen, U.KGen, Sign, LinkTr, VerTrace) *be a link and traceable ring signature that satisfies* L&T*-anonymity, unforgeability and link and traceability. We define a modified scheme* DTRS′ = (Setup, T.KGen, U.KGen, Sign, TraceToLink, VerTrace) *such that algorithm* TraceToLink *is defined as follows.*

$\mathsf{TraceToLink}(pp, sk_{\mathsf{T}}, m_1, m_2, R, \sigma_1, \sigma_2)$

$\mathtt{Tr} \leftarrow\!\!\$\ \mathsf{LinkTr}(pp, sk_{\mathsf{T}}, (m_1, m_2), R, (\sigma_1, \sigma_2))$
if $\mathtt{Tr} = 0$ **return** 0
else return 1

Theorem 27 (L&T ⇒ LINK)**.** *Let* DTRS *be a secure link and traceable ring signature scheme. Then we can construct a modified ring signature* DTRS′ *that is a secure linkable ring signature scheme.*

Lemma 18. *Let* DTRS *be a ring signature that satisfies* L&T-anon*. Then* DTRS′ *satisfies* LINK-anon*.*

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary that succeeds in the LINK-anon experiment against scheme DTRS′. We show that we can construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that

succeeds in the L&T-anon experiment against scheme DTRS. We present $\mathcal{B}$ in Figure 6.30. It is clear that inputs to $\mathcal{A}$ are distributed identically to the LINK-anon experiment. In fact, public parameters pp, tracer public key pk_{T} and signature σ^* are generated identically. Moreover, all oracles are distributed identically. In particular, oracle $\mathcal{O}$LINK-trace is identical because $\mathcal{B}$ runs $\mathcal{O}$L&T-trace to obtain the identity of the signer (if both signatures are produced by the same signer), and a bit otherwise. $\mathcal{B}$ can subsequently returns the correct output to $\mathcal{A}$. We now show that $\mathcal{B}$ is successful in the L&T-anon experiment. By assumption, $\mathcal{A}$ correctly guesses β without tracing the challenge signature or corrupting signers id_0 and id_1. That is, $\mathcal{A}$ correctly determines whether challenge signature σ^* is produced by signer id_0 or id_1. As σ is generated in the L&T-anon experiment, this means that, if $\mathcal{B}$ returns β' output by $\mathcal{A}$, then $\mathcal{B}$ correctly guesses β. Therefore, $\mathcal{B}$ succeeds in the L&T-anonymity experiment and, by contradiction, the result holds. □

$\mathcal{B}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{L\&T\text{-}trace}}(pp, pk_{\mathsf{T}})$

$\mathbf{pk}_{\mathsf{S}} \leftarrow ()$; $\mathbf{sk}_{\mathsf{S}} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset$; $\mathsf{corL} \leftarrow \emptyset$; $\mathcal{Q}\mathsf{sign} \leftarrow \emptyset$; $\mathcal{Q}\mathsf{trace} \leftarrow \emptyset$
$(m^*, R^*, id_0, id_1, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{LINK\text{-}trace}}(pp, pk_{\mathsf{T}})$
return $(m^*, R^*, id_0, id_1, st)$

$\mathcal{B}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{L\&T\text{-}trace}}(\sigma^*, st)$

$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{LINK\text{-}trace}}(\sigma^*, st)$
return β'

$\mathcal{O}\mathsf{LINK\text{-}trace}_{(pp,sk_{\mathsf{T}},\mathcal{Q}\mathsf{trace})}(m_1, m_2, R, \sigma_1, \sigma_2)$

$b \leftarrow\!\!\$\ \mathcal{O}\mathsf{L\&T\text{-}trace}(pp, sk_{\mathsf{T}}, (m_1, m_2), R, (\sigma_1, \sigma_2))$
$\mathcal{Q}\mathsf{trace} \leftarrow \mathcal{Q}\mathsf{trace} \cup \{(m_1, R, \sigma_1), (m_2, R, \sigma_2)\}$
if $b = 0$ **return** 0
else return 1

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{sign}$

$\mathcal{B}$ calls the identical oraclein the L&T-anon experiment.

Figure 6.30: Adversary $\mathcal{B}$ that breaks the L&T-anon security of DTRS in the proof of Lemma 18.

Lemma 19. *Let* DTRS *be a ring signature that satisfies* L&T-corr*. Then* DTRS′ *satisfies* LINK-corr*.*

Proof. Consider a trace Tr output by algorithm $\mathsf{TraceToLink}(pp, sk_{\mathsf{T}}, (m_1, m_2), R, (\sigma_1, \sigma_2))$ that takes as input two signatures σ_1 and σ_2 for ring R and messages m_1 and m_2 respectively. Moreover, let sk_{T} be the tracer secret key generated by algorithm T.KGen and let pp be parameters generated by algorithm Setup. By definition, algorithm TraceToLink returns 1 or 0. We consider these two possibilities in turn. First, assume that algorithm TraceToLink returns 0. Then, by definition, $pk_{\mathsf{S}_1} \neq pk_{\mathsf{S}_2}$. Indeed, we assume that DTRS satisifes L&Tcorr. Therefore, algorithm LinkTr returns 0 with overwhelming probability. As a result, if TraceToLink returns 0, the signatures are produced by different signers as required. Similarly, if TraceToLink returns 1, by the L&T-corr property of DTRS, both signatures are signed by the same signer, as required. We conclude that DTRS′ satisfies L&Tcorr. □

Lemma 20. *Let* DTRS *be a ring signature that satisfies* L&T-frame*. Then* DTRS′ *satisfies* LINK-frame*.*

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary that succeeds in the LINK-frame experiment against scheme DTRS′. We show that we can construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that succeeds in the L&T-frame experiment against scheme DTRS. We present $\mathcal{B}$ in Figure 6.31. It is clear that inputs to $\mathcal{A}$ are distributed identically to the LINK-frame experiment. In fact, public parameters pp and signature σ_1 are generated identically. Moreover, all oracles are distributed identically. We now show that $\mathcal{B}$ is successful in the L&T-frame experiment. That is, algorithm LinkTr returns the identity of the (honest) signer, and the signature σ_2 verifies and is not output by oracle $\mathcal{O}$sign. By assumption, tuple $(pk_{\mathsf{T}}, pk_{\mathsf{S}}, m_2, R, \sigma_2) \notin \mathcal{Q}\mathsf{sign}$, pk_{S} is honest, and the signature σ_2 verifies. Moreover, if algorithm TraceToLink outputs 1 in the LINK-frame experiment, signatures σ_1 and σ_2 are generated by the same signer. Therefore, in the L&T-frame experiment, algorithm LinkTr returns pk_{S} (if $m_1 \neq m_2$) or 1 (if $m_1 = m_2$). Therefore, $\mathcal{B}$ satisfies all conditions to succeed in the L&T-frame experiment. We conclude that, by contradiction, the result holds. □

$\mathcal{B}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign}}(pp)$

$\mathbf{pk}_{\mathsf{S}} \leftarrow ()$; $\mathbf{sk}_{\mathsf{S}} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset$; $\mathsf{corL} \leftarrow \emptyset$; $\mathcal{Q}\mathsf{sign} \leftarrow \emptyset$
$(sk_{\mathsf{T}}, pk_{\mathsf{S}}, m_1, R, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign}}(pp)$
return $(sk_{\mathsf{T}}, pk_{\mathsf{S}}, m_1, R, st)$

$\mathcal{B}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign}}(\sigma_1, st)$

$(m_2, \sigma_2) \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign}}(\sigma_1, st)$
return (m_2, σ_2)

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{sign}$

$\mathcal{B}$ calls the identical oracle
in the L&T-frame experiment.

Figure 6.31: Adversary $\mathcal{B}$ that breaks the L&T-frame security of DTRS in the proof of Lemma 20.

6.6.3.1 R&T implies Accountability

We conclude this section by showing that R&T implies accountability. To demonstrate this result, we first define a ring signature that is modified from any secure R&T ring signature.

Definition 85. *Let* DTRS = (Setup, T.KGen, U.KGen, Sign, Report, Trace, VerTrace) *be an* R&T *ring signature that satisfies anonymity, unforgeability and traceability. We define a modified scheme* DTRS′ = (Setup, T.KGen, U.KGen, Sign, VerTrace, RepToAcc, VerRepToAcc) *such that algorithms* RepToAcc *and* VerRepToAcc *are defined as follows.*

$\mathsf{RepToAcc}(pp, sk_{\mathsf{T}}, sk_{\mathsf{S}}, m, R, \sigma)$

$pk_{\mathsf{T}} \leftarrow\!\!\$\ \mathsf{T.KGen}(pp; sk_{\mathsf{T}})$

$\mathtt{Rep} \leftarrow\!\!\$\ \mathsf{Report}(pp, pk_{\mathsf{T}}, sk_{\mathsf{S}}, m, R, \sigma)$

$(pk_{\mathsf{S}}, \rho = (\mathtt{Tr}, \rho_t)) \leftarrow\!\!\$\ \mathsf{Trace}(pp, sk_{\mathsf{T}}, m, R, \sigma, \mathtt{Rep})$

return (pk_{S}, ρ)

$\mathsf{VerRepToAcc}(pp, pk_{\mathsf{T}}, m, R, \sigma, pk_{\mathsf{S}}, \rho)$

parse ρ as $(\mathtt{Tr}, \rho_t)$

$b \leftarrow\!\!\$\ \mathsf{VerTrace}(pp, pk_{\mathsf{T}}, m, R, \sigma, pk_{\mathsf{S}}, \mathtt{Tr}, \rho_t)$

return b

We now prove that DTRS′ is a secure accountable ring signature scheme. We obtain Theorem 28, which we prove via a series of Lemmata.

Theorem 28 (R&T ⇒ ACC)**.** *Let* DTRS *be a secure* R&T *ring signature scheme. Then* DTRS′ *is a secure* R&T *ring signature scheme and a secure accountable ring signature scheme.*

Lemma 21. *Let* DTRS *be a ring signature that satisfies anonymity for an* R&T *ring signature. Then we can construct a ring signature* DTRS′ *that satisfies* ACC-anon*.*

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary that succeeds in the ACC-anon experiment against scheme DTRS′. We show that we can construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that succeeds in the R&T anonymity experiment against scheme DTRS. We present $\mathcal{B}$ in Figure 6.32. It is clear that inputs to $\mathcal{A}$ are distributed identically to the ACC-anon experiment. In fact, public parameters pp, tracer public key pk_{T} and signature σ^* are generated identically. Moreover, all oracles are distributed identically. In particular, oracle $\mathcal{O}$ACC-trace returns the signer's identity and a proof of correct tracing as generated by algorithm RepToAcc for DTRS′. We now show that $\mathcal{B}$ is successful in the R&T anonymity experiment. By assumption, $\mathcal{A}$ correctly guesses β without tracing the challenge signature or corrupting signers id_0 and id_1. That is, $\mathcal{A}$ correctly determines whether challenge signature σ^* is produced by signer id_0 or id_1. As σ^* is generated by the challenger in the R&T anonymity experiment, this means that, if $\mathcal{B}$ returns β' output by $\mathcal{A}$, then $\mathcal{B}$ correctly guesses β. Therefore, $\mathcal{B}$ succeeds in the ACC-anon experiment and, by contradiction, the result holds. □

Lemma 22. *Let* DTRS *be a ring signature that satisfies tracing correctness for an* R&T *ring signature. Then we can construct a ring signature* DTRS′ *that satisfies* ACCcorr*.*

Proof. We show that DTRS′ satisfies ACCcorr. Consider a tuple (pk_{S}, ρ) output by algorithm RepToAcc that takes as input a signature σ for ring R and messages m. Moreover, let sk_{T} be the tracer secret key generated by algorithm T.KGen, let pp be parameters

$\mathcal{B}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report},\mathcal{O}\mathsf{trace}}(pp, pk_\mathsf{T})$

$\mathbf{pk}_\mathsf{S} \leftarrow (); \ \mathbf{sk}_\mathsf{S} \leftarrow ()$
$\mathsf{L} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{trace} \leftarrow \emptyset$
$(m^*, R^*, id_0, id_1, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{ACC\text{-}trace}}(pp, pk_\mathsf{T})$
return $(m^*, R^*, id_0, id_1, st)$

$\mathcal{B}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{trace}}(\sigma^*, st)$

$\beta' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{L\&T\text{-}trace}}(\sigma^*, st)$
return β'

$\mathcal{O}\mathsf{ACC\text{-}trace}_{(pp,sk_\mathsf{T},\mathcal{Q}\mathsf{trace})}(m, R, \sigma)$

$id \leftarrow\!\!\$\ R$
$\mathtt{Rep} \leftarrow\!\!\$\ \mathcal{O}\mathsf{report}(id, m, R, \sigma)$
$(pk_\mathsf{S}, \mathtt{Tr}, \rho_t) \leftarrow\!\!\$\ \mathcal{O}\mathsf{trace}(m, R, \sigma, \mathtt{Rep})$
$\mathcal{Q}\mathsf{trace} \leftarrow \mathcal{Q}\mathsf{trace} \cup \{(m, R, \sigma)\}$
return $(pk_\mathsf{S}, \rho = (\mathtt{Tr}, \rho_t))$

$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{sign}$

$\mathcal{B}$ calls the identical oracle
in the R&T anonymity experiment.

Figure 6.32: Adversary $\mathcal{B}$ that breaks the R&T anonymity security of DTRS in the proof of Lemma 21.

generated by algorithm Setup and let $sk_\mathsf{S} \in R$. By definition of correctness, DTRS′ satisfies ACCcorr if algorithm VerRepToAcc returns 1 with overwhelming probability. Assume that VerRepToAcc does not return 1. Then it must be the case that VerTrace does not return 1. By assumption, DTRS′ satisfies R&T correctness. Therefore, algorithm VerTrace returns 1 with overwhelming probability where all operations are correctly computed. Then, by contradiction, DTRS′ satisfies ACCcorr. □

Lemma 23. *Let* DTRS *be a ring signature that satisfies non-frameability for an* R&T *ring signature. Then we can construct a ring signature* DTRS′ *that satisfies* ACC-frame*.*

Proof. Let $\mathcal{A}$ be an adversary that succeeds in the ACC-frame experiment for scheme DTRS′. We show that we can construct an adversary $\mathcal{B}$ that succeeds in the R&T non-frameability experiment against scheme DTRS. We present $\mathcal{B}$ in Figure 6.33. It is clear that inputs to $\mathcal{A}$ are distributed identically to the ACC-non-frameability experiment. In fact, public parameters pp are generated identically and all oracles are distributed identically. We now show that $\mathcal{B}$ is successful in the R&T non-frameability experiment. That is, algorithms VerTrace and Verify return 1, pk_S is honest, and the tuple output by adversary $\mathcal{B}$ is not queried to $\mathcal{O}$sign. By assumption, tuple $(pk_\mathsf{T}, pk_\mathsf{S}, m^*, R^*, \sigma^*) \notin \mathcal{Q}\mathsf{sign}$, pk_S is honest, and the signature σ^* verifies. Moreover, algorithm VerRepToTrace returns 1 in the ACC-frame experiment so algorithm VerTrace returns 1 in the R&T non-frameability experiment. Therefore, $\mathcal{A}$ satisfies all conditions to succeed in the R&T non-frameability experiment. We conclude that, by contradiction, the result holds. □

Lemma 24. *Let* DTRS *be a ring signature that satisfies soundness for an* R&T *ring signature. Then we can construct a ring signature* DTRS′ *that satisfies* ACC-sound*.*

$\mathcal{B}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report}}(pp)$	$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{sign}$
$\mathbf{pk}_{\mathsf{S}} \leftarrow (); \ \mathbf{sk}_{\mathsf{S}} \leftarrow ()$	$\mathcal{B}$ calls the identical oracle
$\mathsf{L} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset$	in the R&T non-frameability experiment.
$(pk_{\mathsf{T}}, m^*, R^*, \sigma^*, pk_{\mathsf{S}}, \rho) \leftarrow\!\!\$ \ \mathcal{A}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign}}(pp)$	
parse ρ as $(\mathtt{Tr}, \rho_t)$	
return $(pk_{\mathsf{T}}, m^*, R^*, \sigma^*, pk_{\mathsf{S}}, \mathtt{Tr}, \rho_t)$	

Figure 6.33: Adversary $\mathcal{B}$ that breaks the R&T non-frameability security of DTRS′ in the proof of Lemma 23.

Proof. Let $\mathcal{A}$ be an adversary that succeeds in the ACC-sound experiment for scheme DTRS′. We show that we can construct an adversary $\mathcal{B}$ that succeeds in the R&T soundness experiment for scheme DTRS. We present $\mathcal{B}$ in Figure 6.34. It is clear that inputs to $\mathcal{A}$ are distributed identically to the ACC soundness experiment because public parameters pp are generated identically and all oracles are distributed identically. We now show that $\mathcal{B}$ is successful in the R&T non-frameability experiment. By assumption, $pk_{\mathsf{S}_i} \neq pk_{\mathsf{S}_j}$. Moreover, as algorithm VerRepToAcc outputs 1 in the accountable soundness experiment, algorithm VerTrace outputs 1 in the R&T soudnness experiment. Therefore, $\mathcal{A}$ satisfies all conditions to succeed in the R&T soundness experiment and, by contradiction, the result holds. □

$\mathcal{B}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},\mathcal{O}\mathsf{report}}(pp)$	$\mathcal{O}\mathsf{reg}/\mathcal{O}\mathsf{corrupt}/\mathcal{O}\mathsf{sign}$
$\mathbf{pk}_{\mathsf{S}} \leftarrow (); \ \mathbf{sk}_{\mathsf{S}} \leftarrow ()$	$\mathcal{B}$ calls the identical oracle
$\mathsf{L} \leftarrow \emptyset; \ \mathsf{corL} \leftarrow \emptyset; \ \mathcal{Q}\mathsf{sign} \leftarrow \emptyset$	in the R&T soundness experiment.
$(pk_{\mathsf{T}}, m^*, R^*, \sigma^*, (pk_{\mathsf{S}_i}, \rho_i), (pk_{\mathsf{S}_j}, \rho_j)) \leftarrow\!\!\$ \ \mathcal{A}^{\mathcal{O}\mathsf{reg},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{sign},}(pp)$	
parse ρ_i as $(\mathtt{Tr}_i, \rho_{t_i})$	
parse ρ_j as $(\mathtt{Tr}_j, \rho_{t_j})$	
return $(pk_{\mathsf{T}}, m^*, R^*, \sigma^*, pk_{\mathsf{S}_i}, \mathtt{Tr}_i, \rho_{t_i}, pk_{\mathsf{S}_j}, \mathtt{Tr}_j, \rho_{t_j})$	

Figure 6.34: Adversary $\mathcal{B}$ that breaks the R&T soundness security of DTRS in the proof of Lemma 24.

6.7 Concluding Remarks

In this chapter, we introduced and defined a new primitive: report and trace ring signatures, and presented an accompanying security model. Our new primitive protects the identity of the signer until tracing is complete and protects the identity of the reporter even after tracing. We presented a construction of an R&T ring signature scheme that satisfies our security model, and extended our construction to the multiple reporter setting. Additionally, we provided an instantiation of our single reporter construction and compared its efficiency

with accountable ring signatures [31]. This allowed us to discuss the additional costs associated with our report and trace functionality, and demonstrate that our proposed instantiation is practical. Finally, we analysed our new primitive in the context of ring signatures with tracing functions from the literature, demonstrating how these different tracing functions relate.

Our work in this chapter departs from previous chapters. Where the previous chapters in this thesis considered data privacy, this chapter considered data source privacy. Moreover, in this chapter we chose to consider the relationship between data source privacy and tracing requirements. The contributions in this chapter recognise that, though privacy is important, and, indeed, is the subject of this thesis, it is important to recognise that there are circumstances under which revocation of privacy is desirable and, perhaps, necessary. Moreover, we showed that, despite the tensions that arise between privacy and other security properties (i.e., traceability), it is possible to achieve a balance.

Chapter 7

Concluding Remarks

Contents

In this thesis we investigated formal definitions of data privacy and data source privacy within the context of three cryptographic protocols and primitives. More specifically, we explored data privacy for e-voting schemes and digital signature schemes, and we explored privacy of the data source with respect to ring signatures. Here, we provide an overview of the results in this thesis, and highlight potential opportunities for future work.

In the setting of e-voting, ballot secrecy is a fundamental notion of privacy that requires that a voter's vote remains secret throughout an election. Formal ballot secrecy definitions have been subject to careful analysis and rigorous definitions exist [22]. However, such definitions [22, 44, 48], which follow the popular BPRIV [22] approach to ballot secrecy, are limited in some respects. In particular, BPRIV [22] is difficult to extend to the malicious setting [48, 50]. In light of this limitation, we revisited the approach of [16] and presented three new definitions of ballot secrecy in **Chapter 3**. We presented a definition of ballot secrecy in the honest model (i.e., all election officials are assumed to be honest) for e-voting schemes with internal registration that allows adaptive voter corruption. Then, we defined ballot secrecy in the malicious ballot box setting and, finally, extended this definition to the (partially) malicious tallier setting. We showed that our definitions are satisfiable, presenting e-voting schemes that provably satisfy our various definitions.

Receipt-freeness and coercion-resistance are stronger notions of privacy for e-voting schemes that were, respectively, introduced in [17] and [88]. Unlike ballot secrecy, these notions are not as well-understood, and existing definitions have not been the subject of thorough analysis. We address this gap in the literature in **Chapter 4**. Accordingly, we systematically examined three receipt-freeness definitions from the literature [25, 38, 91]. Our analysis uncovered soundness issues with two of the definitions and completeness issues with all three. In the second part of Chapter 4, we introduced new formal definitions of receipt-freeness, with the aim of addressing weaknesses and limitations that we found in existing definitions. We analysed our new definitions and presented formal results that compare our new definitions and existing definitions. We concluded that our new definition is a step towards rigorous receipt-freeness definitions for e-voting. Nevertheless, e-voting protocols are complex and, as demonstrated in Chapters 3 and 4 of this thesis, defining privacy for such protocols is non-trivial. Therefore, we believe that further work is required in this exciting area of research and we hope that the work presented in this thesis will help to inform this area.

To further inform our understanding of receipt-freeness (and coercion-resistance) and to explore advanced notions of privacy in a different context, we switched focus in **Chapter 5** and considered receipt-freeness and coercion-resistance in the context of digital signatures. Specifically, we defined incoercible signatures, an extension of a standard digital signature scheme that allows a signer to indicate to a designated authority that a signature was generated under coercion. In our syntax, only the designated authority is alerted to the coercion attempt, and an attacker cannot determine whether a signer notified the authority of coercion even if the signer presents the attacker with a full transcript of their actions. That is to say, signatures that indicate coercion are indistinguishable from real signatures. We introduced a security model for our new primitive that captures notions of receipt-freeness and coercion-resistance. Then, we presented generic constructions of incoercible signature schemes that transform standard signature schemes and additionally require a sender-deniable encryption scheme. In Chapter 5, we demonstrated that receipt-freeness is closely linked to deniability. Indeed, we prove that a variant of deniable encryption is necessary to construct an incoercible signature scheme that satisfies our notion of receipt-freeness. We note that this reflects the established understanding of the relationship between deniable encryption and receipt-freeness in the e-voting space, specifically that deniable encryption is closely linked to weak notions of incoercibility (i.e., receipt-freeness) and can be used to construct receipt-free e-voting schemes [17]. Therefore, our results in

this chapter not only define new attack models for digital signatures, but also expand our understanding of complex notions of privacy for cryptographic protocols.

Finally, we considered privacy of the data source by examining anonymity in the context of accountable ring signatures in **Chapter 6**. Specifically, we extended accountable ring signatures, requiring that the tracer can only revoke signer anonymity if the tracer first receives a report from a ring member. We presented syntax and a security model for our new primitive, extending existing security notions for ring signatures to our setting. Additionally, our security model captures reporter anonymity, thereby ensuring that malicious message/signature pairs can be reported without repercussions for the reporter. We presented a generic construction and proved its security. Finally, to understand our new primitive and, specifically, how it relates to existing ring signatures, we formally related report and trace ring signatures with other ring signatures equipped with tracing functionality, namely, linkable, traceable and accountable ring signatures.

7.1 Future Work

We conclude this thesis by highlighting several potential directions for future work.

In Chapter 3 we presented definitions of ballot secrecy that capture malicious election officials. However, our definitions in the malicious setting only capture static corruption of voters. An interesting improvement to our work would be to extend our definitions to the adaptive setting, wherein an attacker can corrupt voters throughout an election. Moreover, our definition in the distributed and malicious tallier setting assumes that key material is generated honestly by talliers. A further possible extension to our work could capture an attacker that can generate key material for talliers.

In Chapter 4, we focused on analysing and presenting definitions of receipt-freeness in the honest model. Thus, we leave defining receipt-freeness in the malicious setting as an open problem. We believe that this problem can be addressed by building upon our definitions of ballot secrecy in the malicious setting presented in Chapter 3.5 (rather than building upon BPRIV [22], which is the approach we take in Chapter 4), whilst retaining the intuition of our receipt-freeness definitions that we detail in Chapter 4.6.

We also uncovered an interesting result in Chapter 4: we demonstrated that, in the context of e-voting, the relationship between receipt-freeness and coercion-resistance is not always linear. We believe that precisely defining the relationship between these properties is an interesting open problem that can be approached as follows. Recall that receipt-freeness definitions capture (variants of) vote-buying attacks, and coercion-resistance definitions additionally capture abstention, randomisation and simulation attacks (which we will collectively refer to as ARS attacks here). We suggest that, by formalising each of the aforementioned attacks as separate security properties, it may be possible to make formal statements about the compatibility of these attacks, which can clarify the relationship between receipt-freeness and coercion-resistance. For example, it may be possible to prove that e-voting schemes cannot simultaneously protect against a strong variant of vote buying attacks and ARS attacks. In this event, the relationship between receipt-freeness and coercion-resistance can be confirmed to be non-linear.

In Chapter 5, we present constructions of incoercible signature schemes that use deniable encryption, and prove that (partial) deniable encryption is necessary to achieve receipt-freeness. A useful direction for future research would be to determine whether it is possible to build partial deniable encryption schemes that are more efficient than standard deniable encryption schemes, which could lead to more efficient receipt-free incoercible signature constructions. A further interesting direction for future work could explore the 'gap' between receipt-freeness and coercion-resistance for incoercible signatures. Specifically, there may be attack strategies that lie between receipt-freeness and coercion-resistance. Identifying such strategies may also help lead to more efficient constructions of incoercible signature schemes.

We presented a generic construction of an R&T ring signature scheme in Chapter 6. A promising direction for future work would be to explore the use of broadcast encryption for our construction, which could lead to efficiency improvements. Finally, recall that report and trace ring signatures achieve a balance between anonymity and traceability. Many other cryptographic protocols also aim to balance these conflicting goals, as we outlined in Chapter 6.2.2. We believe that a broader study that explores the relationship between anonymity and traceability for cryptographic protocols is achievable and desirable, and believe that the work in Chapter 6 is a first step in this direction.

Bibliography

[1] B. Adida. *Advances in Cryptographic Voting Systems*. PhD thesis, Massachusetts Institute of Technology, 2006.

[2] B. Adida. Helios: Web-based open-audit voting. In *17th USENIX Security Symposium*, pages 335–348. USENIX Association, 2008.

[3] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques — Eurocrypt 2002*, pages 83–107. Springer, Berlin, Heidelberg, 2002.

[4] V. Arun, A. Kate, D. Garg, P. Druschel, and B. Bhattacharjee. Finding safety in numbers with secure allegation escrows. In *The Network and Distributed System Security Symposium – NDSS 2020*. Internet Society, 2020.

[5] G. Ateniese, B. Magri, and D. Venturi. Subversion-resilient signature schemes. In *Proceedings of the 22nd ACM Conference on Computer and Communications Security – CCS 2015*, pages 364–375. Association for Computing Machinery, 2015.

[6] M. H. Au, J. K. Liu, W. Susilo, and T. H. Yuen. Certificate based (linkable) ring signature. In *Information Security Practice and Experience – ISPEC 2007*, pages 79–92. Springer Berlin Heidelberg, 2007.

[7] A. Baskar, R. Ramanujam, and S. Suresh. Knowledge-based modelling of voting protocols. In *Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge – TARK 2007*, pages 62–71. Association for Computing Machinery, 2007.

[8] Belenios voting system. `https://www.belenios.org/index.html`.

[9] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Annual International Cryptology Conference – CRYPTO 1998*, pages 26–45, 1998.

[10] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques — Eurocrypt 2003*, pages 614–629. Springer, Berlin, Heidelberg, 2003.

[11] M. Bellare and S. K. Miner. A forward-secure digital signature scheme. In *Annual International Cryptology Conference – CRYPTO 1999*, pages 431–448. Springer, Berlin, Heidelberg, 1999.

[12] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security – CCS 1993*, pages 62–73. Association for Computing Machinery, 1993.

[13] M. Bellare and P. Rogaway. The game-playing technique. *International Association for Cryptographic Research (IACR) ePrint Archive: Report*, 331, 2004.

[14] M. Bellare and A. Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In *Annual International Cryptology Conference*, pages 519–536. Springer, Berlin, Heidelberg, 1999.

[15] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *Cryptographers' Track at the RSA Conference – CT-RSA 2005*, pages 136–153. Springer, Berlin, Heidelberg, 2005.

[16] J. Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Yale University, 1987.

[17] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing – STOC 1994*, pages 544–553. Association for Computing Machinery, 1994.

[18] J. Benaloh and M. Yung. Distributing the power of a government to enhance the privacy of votes. In *Proceedings of the 5th Annual ACM Symposium on Principles of Distributed Computing – PODC 1986*, pages 52–62. Association for Computing Machinery, 1986.

[19] A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Theory of Cryptography Conference – TCC 2006*, pages 60–79. Springer, Berlin, Heidelberg, 2006.

[20] R. Bendlin, J. B. Nielsen, P. S. Nordholt, and C. Orlandi. Lower and upper bounds for deniable public-key encryption. In *Annual International Conference on the Theory and Application of Cryptology and Information Security – ASIACRYPT 2011*, pages 125–142. Springer, Berlin, Heidelberg, 2011.

[21] D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi. A comprehensive analysis of game-based ballot privacy definitions. ePrint Report 2015/255, 2015.

[22] D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi. Sok: a comprehensive analysis of game-based ballot privacy definitions. In *IEEE Security and Privacy*, pages 499–516. IEEE, 2015.

[23] D. Bernhard, V. Cortier, O. Pereira, B. Smyth, and B. Warinschi. Adapting helios for provable ballot privacy. In *European Symposium on Research in Computer Security – ESORICS 2011*, pages 335–354. Springer, Berlin, Heidelberg, 2011.

[24] D. Bernhard, O. Kulyk, and M. Volkamer. Security proofs for participation privacy, receipt-freeness, ballot privacy, and verifiability against malicious bulletin board for the helios voting scheme. *International Association for Cryptographic Research (IACR) ePrint Archive: Report*, 431, 2016.

[25] D. Bernhard, O. Kulyk, and M. Volkamer. Security proofs for participation privacy, receipt-freeness and ballot privacy for the helios voting scheme. In *Proceedings on the 12th International Conference on Availability, Reliability and Security*, pages 1 – 10. Association for Computing Machinery, 2017.

[26] D. Bernhard, N. K. Nguyen, and B. Warinschi. Adaptive proofs have straightline extractors (in the random oracle model). In *International Conference on Applied Cryptography and Network Security*, pages 336–353. Springer, 2017.

[27] D. Bernhard, O. Pereira, and B. Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In *Annual International Conference on the Theory and Application of Cryptology and Information Security – ASIACRYPT 2012*, pages 626–643. Springer, Berlin, Heidelberg, 2012.

[28] D. Bernhard and B. Smyth. Ballot privacy and ballot independence coincide. In *European Symposium on Research in Computer Security – ESORICS 2013*, pages 463–480. Springer, Berlin, Heidelberg, 2013.

[29] D. Bernhard and B. Smyth. Ballot secrecy with malicious bulletin boards. *International Association for Cryptographic Research (IACR) ePrint Archive: Report*, 822, 2014.

[30] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Annual International Cryptology Conference – CRYPTO 2005*, pages 258–275. Springer, Berlin, Heidelberg, 2005.

[31] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, J. Groth, and C. Petit. Short accountable ring signatures based on ddh. In *European Symposium on Research in Computer Security – ESORICS 2015*, pages 243–265. Springer, Berlin, Heidelberg, 2015.

[32] K. Braunlich and R. Grimm. Formalization of receipt-freeness in the context of electronic voting. In *Proceedings of the 6th International Conference on Availability, Reliability and Security*, pages 119–126. IEEE, 2011.

[33] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd Conference on Foundations of Computer Science – FOCS 2001*, pages 136–145. IEEE, 2001.

[34] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In *Annual International Cryptology Conference – CRYPTO 1997*, pages 90–104. Springer, Berlin, Heidelberg, 1997.

[35] R. Canetti and R. Gennaro. Incoercible multi-party computation. In *Proceedings of the 37th Conference on Foundations of Computer Science – FOCS 1996*, pages 504–513. IEEE, 1996.

[36] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *Journal of the ACM (JACM)*, 51(4):557–594, 2004.

[37] R. Canetti, S. Park, and O. Poburinnaya. Fully deniable interactive encryption. In *Annual International Cryptology Conference – CRYPTO 2020*, pages 807–835. Springer, Berlin, Heidelberg, 2020.

[38] P. Chaidos, V. Cortier, G. Fuchsbauer, and D. Galindo. Beleniosrf: A non-interactive receipt-free electronic voting scheme. In *Proceedings of the 23rd ACM Conference on*

Computer and Communications Security – CCS 2016, pages 1614–1625. Association for Computing Machinery, 2016.

[39] M. Chase and A. Lysyanskaya. On signatures of knowledge. In *Annual International Cryptology Conference – CRYPTO 2006*, pages 78–96. Springer, Berlin, Heidelberg, 2006.

[40] D. Chaum and E. Van Heyst. Group signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques — Eurocrypt 1991*, pages 257–265. Springer, Berlin, Heidelberg, 1991.

[41] G. Chen, C. Wu, W. Han, X. Chen, H. Lee, and K. Kim. A new receipt-free voting scheme based on linkable ring signature for designated verifiers. In *International Conference on Embedded Software and Systems Symposia – ICESS 2008*, pages 18–23. IEEE, 2008.

[42] Civitas voting system. `www.cs.cornell.edu/projects/civitas/`.

[43] M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: toward a secure voting system. In *IEEE Security and Privacy*, pages 354–368. IEEE, 2008.

[44] V. Cortier, C. C. Dragan, F. Dupressoir, and B. Warinschi. Machine-checked proofs for electronic voting: Privacy and verifiability for belenios. In *31st Computer Security Foundations Symposium – CSF 2018*, pages 298–312. IEEE, 2018.

[45] V. Cortier, D. Galindo, S. Glondu, and M. Izabachène. Election verifiability for helios under weaker trust assumptions. In *European Symposium on Research in Computer Security – ESORICS 2014*, pages 327–344. Springer, Berlin, Heidelberg, 2014.

[46] V. Cortier, D. Galindo, R. Küsters, J. Müller, and T. Truderung. Sok: Verifiability notions for e-voting protocols. In *IEEE Security and Privacy*, pages 779–798. IEEE, 2016.

[47] V. Cortier and J. Lallemand. Voting: You cant have privacy without individual verifiability. In *Proceedings of the 25th ACM Conference on Computer and Communications Security – CCS 2018*, pages 53–66. Association for Computing Machinery, 2018.

[48] V. Cortier, J. Lallemand, and B. Warinschi. Fifty shades of ballot privacy: Privacy against a malicious board. *International Association for Cryptographic Research (IACR) ePrint Archive: Report*, 127, 2020.

[49] V. Cortier and B. Smyth. Attacking and fixing helios: An analysis of ballot secrecy. In *24th Computer Security Foundations Symposium – CSF 2011*, pages 297–311. IEEE, 2011.

[50] R. del Pino, V. Lyubashevsky, G. Neven, and G. Seiler. Practical quantum-safe voting from lattices. In *Proceedings of the 24th ACM Conference on Computer and Communications Security – CCS 2017*, pages 1565–1581. Association for Computing Machinery, 2017.

[51] S. Delaune, S. Kremer, and M. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *19th Computer Security Foundations Workshop – CSFW 2006*, pages 12–42. IEEE, 2006.

[52] S. Delaune, S. Kremer, and M. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.

[53] S. Delaune, S. Kremer, and M. D. Ryan. Receipt-freeness: Formal definition and fault attacks. In *Proceedings of the Workshop Frontiers in Electronic Elections – FEE 2005, Milan, Italy*, 2005.

[54] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Annual International Cryptology Conference – CRYPTO 1990*, pages 307–315. Springer, Berlin, Heidelberg, 1990.

[55] M. Di Raimondo and R. Gennaro. New approaches for deniable authentication. *Journal of cryptology*, 22(4):572–615, 2009.

[56] Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-insulated public key cryptosystems. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques — Eurocrypt 2002*, pages 65–82. Springer, Berlin, Heidelberg, 2002.

[57] Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong key-insulated signature schemes. In *International Workshop on Public Key Cryptography — PKC 2003*, pages 130–144. Springer, Berlin, Heidelberg, 2003.

[58] J. Dreier, P. Lafourcade, and Y. Lakhnech. A formal taxonomy of privacy in voting protocols. In *International Conference on Communications – ICC 2012*, pages 6710–6715. IEEE, 2012.

[59] K. Durnoga, J. Pomykała, and T. Trabszys. Digital signature with secretly embedded warning. *Control and Cybernetics*, 42(4):805–824, 2013.

[60] C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. *Journal of the ACM (JACM)*, 51(6):851–898, 2004.

[61] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Annual International Cryptology Conference – CRYPTO 1985*, pages 10–18. Springer, Berlin, Heidelberg, 1985.

[62] i-voting. `e-estonia.com/solutions/e-governance/i-voting/`.

[63] A. Fiat and M. Naor. Broadcast encryption. In *Annual International Cryptology Conference – CRYPTO 1994*, pages 480–491. Springer, Berlin, Heidelberg, 1994.

[64] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Annual International Cryptology Conference – CRYPTO 1986*, pages 186–194. Springer, Berlin, Heidelberg, 1986.

[65] A. Fraser and E. A. Quaglia. Protecting the privacy of voters: New definitions of ballot secrecy for e-voting. In *International Conference on Selected Areas in Cryptography – SAC 2020*. Springer, Berlin, Heidelberg, 2020.

[66] A. Fraser, E. A. Quaglia, and B. Smyth. A critique of game-based definitions of receipt-freeness for voting. In *International Conference on Provable Security – ProvSec 2019*, pages 189–205. Springer, Cham, 2019.

[67] E. Fujisaki. Sub-linear size traceable ring signatures without random oracles. In *Cryptographers' Track at the RSA Conference – CT-RSA 2011*, pages 393–415. Springer, Berlin, Heidelberg, 2011.

[68] E. Fujisaki and K. Suzuki. Traceable ring signature. In *International Workshop on Public Key Cryptography — PKC 2007*, pages 181–200. Springer, Berlin, Heidelberg, 2007.

[69] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing – STOC 1982*, pages 365–377. Association for Computing Machinery, 1982.

[70] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[71] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[72] J. Groth and M. Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques — Eurocrypt 2015*, pages 253–280. Springer, Berlin, Heidelberg, 2015.

[73] J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for np. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques — Eurocrypt 2006*, pages 339–358. Springer, Berlin, Heidelberg, 2006.

[74] K. Gu, X. Dong, and L. Wang. Efficient traceable ring signature scheme without pairings. *Advances in Mathematics of Communications*, 14:207, 2019.

[75] T. Haines and B. Smyth. Surveying definitions of coercion resistance. *International Association for Cryptographic Research (IACR) ePrint Archive: Report*, 822, 2019.

[76] J. Håstad, J. Jonsson, A. Juels, and M. Yung. Funkspiel schemes: an alternative to conventional tamper resistance. In *Proceedings of the 7th ACM Conference on Computer and Communications Security – CCS 2000*, pages 125–133. Association for Computing Machinery, 2000.

[77] Helios voting system. `heliosvoting.org/`.

[78] Helios faq. `https://vote.heliosvoting.org/faq`.

[79] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public key and signature systems. In *Proceedings of the 4th ACM Conference on Computer and Communications Security – CCS 1997*, pages 100–110. Association for Computing Machinery, 1997.

[80] M. Hirt. Receipt-free k-out-of-l voting based on elgamal encryption. In *Towards Trustworthy Elections: New Directions in Electronic Voting*, pages 64–82. Springer, 2010.

[81] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques — Eurocrypt 2000*, pages 539–556. Springer, Berlin, Heidelberg, 2000.

[82] G. Itkis. Cryptographic tamper evidence. In *Proceedings of the 10th ACM Conference on Computer and Communications Security – CCS 2003*, pages 355–364. Association for Computing Machinery, 2003.

[83] G. Itkis and L. Reyzin. Sibir: Signer-base intrusion-resilient signatures. In *Annual International Cryptology Conference – CRYPTO 2002*, pages 499–514. Springer, Berlin, Heidelberg, 2002.

[84] G. Itkis and P. Xie. Generalized key-evolving signature schemes or how to foil an armed adversary. In *International Conference on Applied Cryptography and Network Security – ACNS 2003*, pages 151–168. Springer, Berlin, Heidelberg, 2003.

[85] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques — Eurocrypt 1996*, pages 143–154. Springer, Berlin, Heidelberg, 1996.

[86] H. L. Jonker and E. P. de Vink. Formalising receipt-freeness. In *International Conference on Information Security – ISC 2006*, pages 476–488. Springer, Berlin, Heidelberg, 2006.

[87] H. L. Jonker and W. Pieters. Receipt-freeness as a special case of anonymity in epistemic logic. In *Proceedings of the IAVoSS Workshop On Trustworthy Elections – WOTE 2006*. Robinson College, Cambridge, 2006.

[88] A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *Towards Trustworthy Elections: New Directions in Electronic Voting*, pages 37–63. Springer, 2010.

[89] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. CRC press, 2014.

[90] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques — Eurocrypt 2004*, pages 571–589. Springer, Berlin, Heidelberg, 2004.

[91] A. Kiayias, T. Zacharias, and B. Zhang. End-to-end verifiable elections in the standard model. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques — Eurocrypt 2015*, pages 468–498. Springer, Berlin, Heidelberg, 2015.

[92] M. Kohlweiss and I. Miers. Accountable metadata-hiding escrow: A group signature case study. *Proceedings on Privacy Enhancing Technologies – PoPETs 2015*, 2015(2):206–221, 2015.

[93] O. Kulyk, V. Teague, and M. Volkamer. Extending helios towards private eligibility verifiability. In *International Conference on E-Voting and Identity*, pages 57–73. Springer, Cham, 2015.

[94] M. Kutyowski and P. Kubiak. Lightweight digital signature with secretly embedded warning. *Control and Cybernetics*, 42(4):825–827, 2013.

[95] B. Kuykendall, H. Krawczyk, and T. Rabin. Cryptography for #metoo. *Proceedings on Privacy Enhancing Technologies – PoPETs 2019*, 2019(3):409–429, 2019.

[96] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *International Conference on Information Security and Cryptology – ICISC 2003*, pages 245–258. Springer, Berlin, Heidelberg, 2004.

[97] J. K. Liu, M. H. Au, W. Susilo, and J. Zhou. Linkable ring signature with unconditional anonymity. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):157–165, 2014.

[98] J. K. Liu, W. Susilo, and D. S. Wong. Ring signature with designated linkability. In *1st International Workshop on Security – IWSEC 2006*, pages 104–119. Springer, Berlin, Heidelberg, 2006.

[99] J. K. Liu, V. K. Wei, and D. S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In *Australasian Conference on Information Security and Privacy*, pages 325–335. Springer, Berlin, Heidelberg, 2004.

[100] P. Locher and R. Haenni. Receipt-free remote electronic elections with everlasting privacy. *Annals of Telecommunications*, 71(7-8):323–336, 2016.

[101] T. Moran and M. Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *Annual International Cryptology Conference*, pages 373–392. Springer, Berlin, Heidelberg, 2006.

[102] D. Naccache, D. Pointcheval, and C. Tymen. Monotone signatures. In *International Conference on Financial Cryptography*, pages 305–318. Springer, Berlin, Heidelberg, 2002.

[103] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 427–437. Association for Computing Machinery, 1990.

[104] T. Okamoto. Receipt-free electronic voting schemes for large scale elections. In *International Workshop on Security Protocols*, pages 25–35. Springer, Berlin, Heidelberg, 1998.

[105] A. O'Neill, C. Peikert, and B. Waters. Bi-deniable public-key encryption. In *Annual International Cryptology Conference – CRYPTO 2011*, pages 525–542. Springer, Berlin, Heidelberg, 2011.

[106] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing – PODC 1991*, pages 51–59. Association for Computing Machinery, 1991.

[107] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual International Cryptology Conference*, pages 129–140. Springer, Berlin, Heidelberg, 1991.

[108] O. Pereira. Internet voting with helios. *Real-World Electronic Voting*, pages 277 – 308, 2016.

[109] D.-H. Phan, D. Pointcheval, S. F. Shahandashti, and M. Strefler. Adaptive cca broadcast encryption with constant-size secret keys and ciphertexts. *International Journal of Information Security*, 12(4):251–265, 2013.

[110] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Annual International Cryptology Conference*, pages 433–444. Springer, Berlin, Heidelberg, 1991.

[111] A. Rajan, L. Qin, D. W. Archer, D. Boneh, T. Lepoint, and M. Varia. Callisto: A cryptographic approach to detecting serial perpetrators of sexual misconduct. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*, pages 1–4. Association for Computing Machinery, 2018.

[112] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Annual International Conference on the Theory and Application of Cryptology and Information Security – ASIACRYPT 2001*, pages 552–565. Springer, Berlin, Heidelberg, 2001.

[113] S. Saeednia, S. Kremer, and O. Markowitch. An efficient strong designated verifier signature scheme. In *International Conference on Information Security and Cryptology – ICISC 2003*, pages 40–54. Springer, Berlin, Heidelberg, 2003.

[114] A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing – STOC 2014*, pages 475–484. Association for Computing Machinery, 2014.

[115] Y. Sakai, K. Emura, G. Hanaoka, Y. Kawai, T. Matsuda, and K. Omote. Group signatures with message-dependent opening. In *International Conference on Pairing-Based Cryptography*, pages 270–294. Springer, Berlin, Heidelberg, 2013.

[116] K. Sako and J. Kilian. Receipt-free mix-type voting scheme. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques — Eurocrypt 1995*, pages 393–403. Springer, Berlin, Heidelberg, 1995.

[117] B. Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Annual International Cryptology Conference – CRYPTO 1999*, pages 148–164. Springer, Berlin, Heidelberg, 1999.

[118] G. J. Simmons. The prisoners problem and the subliminal channel. In *Annual International Cryptology Conference – CRYPTO 1984*, pages 51–67. Springer, Berlin, Heidelberg, 1984.

[119] G. J. Simmons. Subliminal communication is easy using the dsa. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques — Eurocrypt 1994*, pages 218–232. Springer, Berlin, Heidelberg, 1994.

[120] B. Smyth. A foundation for secret, verifiable elections. *International Association for Cryptographic Research (IACR) ePrint Archive: Report*, 225, 2018.

[121] B. Smyth and D. Bernhard. Ballot secrecy and ballot independence: Definitions and relations. *International Association for Cryptographic Research (IACR) ePrint Archive: Report*, 235, 2013.

[122] B. Smyth, S. Frink, and M. R. Clarkson. Election verifiability: cryptographic definitions and an analysis of helios, helios-c, and jcj. *International Association for Cryptographic Research (IACR) ePrint Archive: Report*, 233, 2015.

[123] M. Tabatabaei, W. Jamroga, and P. Y. Ryan. Expressing receipt-freeness and coercion-resistance in logics of strategic ability: Preliminary attempt. In *Proceedings of the 1st International Workshop on AI for Privacy and Security*, pages 1–8. Association for Computing Machinery, 2016.

[124] P. P. Tsang and V. K. Wei. Short linkable ring signatures for e-voting, e-cash and attestation. In *International Conference on Information Security Practice and Experience*, pages 48–60. Springer, Berlin, Heidelberg, 2005.

[125] N. Tyagi, I. Miers, and T. Ristenpart. Traceback for end-to-end encrypted messaging. In *Proceedings of the 26th ACM Conference on Computer and Communications Security – CCS 2019*, pages 413–430. Association for Computing Machinery, 2019.

[126] Gov.uk how to vote. `https://www.gov.uk/how-to-vote`.

[127] D. Unruh and J. Müller-Quade. Universally composable incoercibility. In *Annual International Cryptology Conference – CRYPTO 2010*, pages 411–428. Springer, Berlin, Heidelberg, 2010.

[128] Transparency international how much is your vote worth. `https://www.transparency.org/en/news/how-much-is-your-vote-worth#`.

[129] L. Wang, G. Zhang, and C. Ma. A survey of ring signature. *Frontiers of Electrical and Electronic Engineering in China*, 3(1):10–19, 2008.

[130] S. Xu and M. Yung. Accountable ring signatures: A smart card approach. In *Smart Card Research and Advanced Applications VI*, pages 271–286. Springer, Boston, MA, 2004.

[131] A. Young and M. Yung. Kleptography: Using cryptography against cryptography. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques — Eurocrypt 1997*, pages 62–74. Springer, Berlin, Heidelberg, 1997.