Advancements in Proxy Re-Encryption: Defining Security for wider Applications

Elizabeth (Ela) Lee

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

Information Security Group
School of Mathematics and Information Security
Royal Holloway, University of London

2020

Declaration

These doctoral studies were conducted under the supervision of Professor Keith Martin (primary supervisor) and Dr Elizabeth Quaglia (advisor).

The work presented in this thesis is the result of original research I conducted, in collaboration with others, whilst enrolled in the School of Mathematics and Information Security 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.

Ela Lee

Acknowledgements

To start at the beginning, I must thank Stefanie Gerke who not only encouraged me to apply to the CDT the first time, but took special effort to convince me to reapply a second time. Without your encouragement, I would not have considered myself good enough to attempt this. Secondly, I would like to thank my supervisor, Keith M. Martin, for his support, guidance, and patience, particularly in improving my writing. Thank you to my advisor, Elizabeth Quaglia, and also to Martin Albrecht and Kenny Paterson, who provided invaluable advice to me during these studies.

I am very grateful to the other CDT students for the friendships and for extending my knowledge of cyber security outside my own bubble, with a special thanks to Gregory Fenn for forcing us to socialise. Thank you to Claire Hudson for all her administrative support - I'm sure we'd all be much worse off without her! Thank you to the EPSRC whose generous funding allowed me to attend workshops, conferences and additional training which certainly made me a better and more well-rounded researcher. I can truthfully say that I have an appreciation for the field as a whole. I am also grateful to the inspiring and encouraging cryptographers I met at conferences and summer schools, for making me feel part of a wider community, putting up with my novice questions, helping me see the potential of my own work and suggesting future directions. Special thanks goes to Katriel Cohn-Gordon on this latter point, as Chapter V (my second publication) came from a question he asked during one of the inter-CDT workshops.

To my husband, Rob. I definitely wasn't expecting to become a "Mrs" before becoming a "Dr"! Thank you for your love and support in all its forms, including but not limited to: proofreading, providing meals, encouraging me to keep pursuing my hobbies, calming me down that time I convinced myself the main proof in my paper was wrong a few hours before the submission deadline, providing snacks, and supporting me financially so that I could keep focusing on finishing my thesis after my funding ran out. I don't have the eloquence to express how thankful I am both to you and for you. I hope I never forget how wonderful you are.

Thank you to the community at St Elizabeth's, and for all those who have given me their support and prayed for me during this endeavour. Thank you for your friendship, for your prayers, for teaching me the value of community, and for reminding me that my worth does not come from my research.

Finally, to my grandparents, Mary Lee and Conway, who sadly passed away whilst I was undertaking this PhD. It has become increasingly apparent to me that, for whatever combination of nature and nurture makes a person the way they are, this thesis would not exist without you. Thank you both so much, for everything.

"*Glory be to Him whose power, working in us, can do infinitely more than we can ask or imagine.*" ~ Ephesians 3:20

Abstract

In cryptographic schemes that use keys, security of the scheme as a whole depends on the security of the key. Our area of interest is in applications where the keys used to encrypt files need to be changed, particularly when those files are stored by third parties. Proxy Re-Encryption (PRE), originally introduced by Blaze, Bleumer, and Strauss in 1998, allows a proxy to transform ciphertexts encrypted under one public key into an encryption of the same message under a new public key without learning the message. This is achieved by sending the proxy an update token created using the current secret key and new public key.

The original motivation cited for PRE was email forwarding, as a means of Alice sharing her encrypted emails with Bob. Access revocation, the other obvious application of PRE, was not considered, meaning many PRE schemes are unsuitable as a mechanism for mitigating key compromise. In this thesis, we take a fresh look at PRE, particularly its suitability in enforcing cryptographic access control and as a key rotation mechanism to enforce key life cycles.

In our first contribution, we consider a malicious proxy that seeks to perform unauthorised re-encryptions. We propose a number of security definitions concerning an adversary's ability to perform a re-encryption not initiated by the client. One of these definitions is authenticated re-encryption, which allows the identity of the party who initiated the re-encryption to be verified, which is useful for verifying that access has been granted legitimately.

Our second contribution is to define post-compromise security for PRE. PRE schemes meeting this definition can be used for access revocation and key expiry. We create the strongest definition to date, whereby compromise of the old secret key, old ciphertext and update token cannot distinguish re-encrypted ciphertexts. We provide an efficient post-compromise secure PRE scheme using lattice-based cryptography.

Finally, we investigate PRE security with adaptive key corruptions. Most work on PRE considers selective key corruptions, where key compromise happens before adversaries learn any challenges. Adaptive security, on the other hand, allows adversaries to corrupt keys at any point, as long as they do not corrupt any key that can be used to directly decrypt challenge ciphertexts. Existing work shows how adaptive security can be reached from selective security, but usually relies to some extent on needing to guess which keys will be corrupted in advance. This approach leads to a large security loss as the guess must be correct. We achieve tighter bounds by taking a different approach which extends observations already made in the literature to prove adaptive security at a much smaller loss, using previously undefined properties of PRE schemes.

Contents

Chapter I

Introduction

Contents

In this Chapter

This chapter introduces the context of this thesis and our motivation for studying Proxy Re-Encryption (PRE). We begin by discussing how use of cloud services has become increasingly popular, and how cryptography can be used as a mechanism for ensuring that clients still have control over their data when using cloud services. We then discuss how security is defined in cryptography, motivating our work on creating stronger, application-driven notions of security for PRE.

I: 1 Introduction to the Cloud

In the past, companies would have acquired and maintained their own infrastructure and software which would be physically located in sites they controlled. However, modern times have seen a huge increase in using cloud services, where infrastructure and other services are maintained externally by a third party. For example, cloud storage has become increasingly popular in recent years, evolving from acting as a source of backup data to becoming the default storage for many applications and systems. An extension of this is the rise of popular media streaming platforms such as Netflix and Spotify, allowing clients to subscribe to on-demand access for media files as opposed to storing them locally. This has increased the number for devices with limited in-built storage as long as they have an Internet connection.

Using cloud services allows clients to reduce their infrastructure and maintenance costs and only pay for the services they use. Whilst use of the term 'the cloud' was popularised by Amazon's Elastic Compute Cloud (EC2) in 2006, concepts of time-sharing and outsourcing computation have been around since the 1960s [Wil18]. In this thesis, we take "the cloud" to mean a server controlled by a third party, which a *client* is using for outsourcing purposes.

There are various types of cloud service, the most common of which are Software-as-a-Service (SaaS)[1], Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS) [NK16]. Forecasts predict that cloud usage will only continue to grow; the 2016 Harvey Nash / KPMG CIO Survey [NK16] found that the number of Chief Information Officers (CIOs) planning significant investment in cloud services for the following three years went up for all services – from 31% to 49% for SaaS, 25% to 39% for PaaS, and 20% to 37% for IaaS. There has also been a huge increase in individuals' use of cloud services. In particular, the use of online streaming services has led to a decline in sales of DVDs [YCPC18], with streaming and downloading services taking over physical sales in early 2017 [Swe17]. There is also increasing prevalence of cloud storage being the default over local storage, as can be seen in multiple smartphones, including Google Pixel phones, offering "unlimited storage" for photos because they automatically store photos in the cloud [Sav16]. This has clear advantages for end users including not needing to worry about storage limitations of their smartphone, and in being able to access remotely stored files when they get a new smartphone.

Whilst using the cloud has a number of practical advantages, it also comes with risks, some of which are poorly understood. One result of cloud backups by default in smartphones is that often, everyday users are unaware that data created on their phones is backed up in the cloud, as was the case with the 2014 iCloud data hack [Coo17]. One important

[1]which includes Storage-as-a-Service

factor that an organisation should take into account when using the cloud is that if an organisation chooses to outsource data storage and processing, this does not necessarily shift their responsibility for ensuring the data is adequately protected, meaning they still have liability for data breaches. For example, the General Data Protection Regulation (GDPR) in the EU [GDP16] indicates how customer personal data should be managed, and failure to comply can result in fines of up to 10 million euros or up to 2% of an organisation's global turnover, whichever is higher.

The most obvious concern with using the cloud for storage is the need to make sure that the stored data does not get lost, as satirised in Figure I.1. Solutions clarifying the

I MIGRATED OUR NORTHERN DATA CENTER TO THE CLOUD.
BUT THE CLOUD STOPPED WORKING AND I CAN'T FIND THE PHONE NUMBER FOR OUR CLOUD GUY.
SO . . . WHATEVER.
YOU LOST OUR DATA CENTER?
THAT'S ONE WAY TO LOOK AT IT.
Dilbert.com DilbertCartoonist@gmail.com
7-5-13 ©2013 Scott Adams, Inc.

Figure I.1: Outsourcing storage means trusting that the cloud will not lose the data.

way client data is handled include creating legislation over how data is handled, Cloud Service Providers (CSPs) publicising their data policies and terms and conditions of using the service, and making customers aware of data breaches. We will now explore these ideas further and point out some shortcomings with respect to how well they mitigate the risks associated with using cloud services.

Limitations of relying on policy. Whilst CSPs set out terms and conditions which state how client data will be used, this may not provide full confidence. An infamous example justifying this is Dropbox's terms and conditions not accurately conveying how client data is handled, as pointed out by security researcher Christopher Soghoian [Leo11]. On 12th April 2011, the Dropbox help site stated that *"Dropbox employees aren't able to access user files... they only have access to file metadata... All files stored on Dropbox servers are encrypted (AES-256) and are inaccessible without your account password."* This led customers to believe their files were encrypted in such a way that Dropbox did not have the means to decrypt them. However, Soghoian pointed out that this assertion was contradicted by another claim by Dropbox – that if a client attempts to upload a file which has previously been uploaded, then this would be detected and the second upload cancelled to save space. On 23rd April 2011, after Soghoian pointed out this discrepancy, Dropbox's policy changed to say that "*Dropbox employees are* ***prohibited*** *from viewing the content of files you store in your Dropbox account, and are only permitted to view file metadata... we have a small number of employees who must be able to access user data for the reasons stated in our privacy policy*" [emphasis added]. This policy wording change indicates that Dropbox always had the capability to access client data, despite what the policy originally said. This is a far cry from the privacy expectations that Dropbox appeared to convey at the start of 2011.

A further problem with relying on legal mechanisms alone is that policies are usually written by the CSP and not by clients, particularly when those clients are small businesses and individual users. Clients only have the choice to either agree or disagree with all terms and conditions, as opposed to having fine-grained control over what they deem acceptable. This is true both when clients initially agree, and when a CSP's data-handling policies change. Consenting to changes in a policy has the further issue that clients must also factor in the cost of transitioning to another service or starting a private infrastructure if they disagree with the changes. This may impact their decision to agree to new terms and end up with them agreeing to terms they would object to given a fresh choice. It is therefore unwise for a client to rely on CSP policies alone to ensure appropriate controls over their data.

Awareness of data breaches. Organisations have not always been transparent in their handling of customer data, and do not always disclose when there has been a data breach in good time, if at all. In 2012, Dropbox reported that user email addresses had been stolen but failed to mention until 2016 that passwords had also been stolen [BBC16]. In 2016, Uber suffered a data breach after which they paid money to the hackers to delete the stolen data and attempted to hide the incident from regulators [BBC18]. This demonstrates how the existence of regulation does not go far enough to control how data is handled. It is also worth noting that when organisations elect to pay hackers to delete stolen data, it is impossible to know whether the hackers actually did delete the data. Even if legislation is upheld by the provider, in order to disclose a breach they must first be aware of it. This lack of transparency in failure to disclose breaches creates a problem where clients are unable to respond to data breaches and mitigate damage. In summary, whilst regulation can act as an incentive for appropriate data controls, additional measures are necessary to implement control of clients' data.

The most pressing concern is that the cloud has the potential capability to perform malicious actions. Aiming for security in this setting covers scenarios where the cloud is hacked by malicious parties, when the cloud itself deliberately performs malicious actions, or where incompetence leads to malicious consequences. It would be preferable to control precisely what a malicious cloud service provider can do with data that has been uploaded by a client. More specifically, when the client also outsources some additional processing over that data we would like a mechanism that controls what actions the server is and is not capable of performing. In the ideal world, we would like to ensure that, as much as possible, the client maintains control over their data when using cloud services.

I: 2 Cryptography and the Cloud

Earlier, we noted how legal mechanisms may not be enough to ensure clients have adequate control over data outsourced to the cloud[2]. We now discuss how cryptography can offer clients more control over their data, including how it provides a formal means of both defining and proving security.

Cryptography addresses a number of aspects of information security including the provision of confidentiality, authentication and integrity. For example:

- *Encryption* keeps messages confidential from parties who do not know the secret key necessary to decrypt.
- *Cryptographic authentication mechanisms* provide a means of users verifying where or who some piece of data originated from.
- *Data integrity* enables us to ensure that data has not been altered.

If a client wants to ensure the confidentiality of externally stored files, one solution is to encrypt them prior to storage. This means that neither the cloud nor any other malicious parties attempting to hack into the cloud should be able to read the contents of the files.

[2]From this point on, we will be agnostic as to which Cloud Service Provider (CSP) is being used and will therefore use the term 'cloud'.

Since keys are almost always much shorter than files, it is reasonable to assume that a client who lacks the capacity to store files locally will at least be able to store cryptographic keys, so this solution is appropriate for the majority of cases where storage is outsourced.

Whilst this simple solution is adequate for the basic storage scenario, the cloud is increasingly being used for more than just storage. To this end, there are a number of cryptographic primitives specifically designed for the scenario where functionality is also outsourced to the cloud. These primitives aim to provide some level of security in this setting. For example:

- *Searchable encryption* [ABC+05, BBO07, SPS14, CJJ+14] allows clients to retrieve files associated with certain keywords, without the plaintext files or keywords being leaked to the cloud. It also seeks to verify that the cloud has retrieved all relevant files as opposed to only retrieving some subset of them.
- *Verifiable computation*, formalised in [ABG+97, GGP10], checks that a third party has performed a computation correctly, without the client needing to perform the whole computation locally.
- *Order-preserving encryption* [AKSX04, BCLO09, BCO11, TYM14] permits range queries over encrypted data, designed for encrypted databases.

However, as the priority in creating some of these primitives was functionality as opposed to security, the security guarantees offered can sometimes be lacking or poorly understood. For example, published material on order-preserving encryption first appeared in 2004 by Agrawal et al [AKSX04], and was proven secure in a comparatively weak model where the adversary has no prior information about the plaintext and can neither encrypt nor decrypt values of their choosing. These assumptions are unrealistic in many real-world scenarios, and make security barely comparable to that of standard encryption. There have been subsequent attempts to create stronger security definitions for order-preserving encryption [BCO11], but it is arguable that such definitions must either sacrifice how well they preserve the order, or the how well they model realistic attacks. Whilst it can be argued that encryption with additional functionality involves an inevitable trade-off with security, it is useful to know precisely what kind of security such schemes have.

Our scenario of interest is when a client has encrypted their files prior to storing them in the cloud, and the client later wishes to change the key which the file is encrypted under.

This could be for a number of reasons such as satisfying compliance directives or to enforce access control policies. For the former, NIST recommends regular key rotation [BBB$^+$12], as does the Payment Card Industry Data Security Standard [PCI18] and the Open Web Application Security Project (OWASP) [OWA18]. For the latter, an organisation can choose which of its employees can access specific files by having those files encrypted under those keys. Should access need to be granted or revoked from someone, the files need to be re-encrypted to a new key accordingly (we shall elaborate on this in Chapter IV). One trivial solution has the client download, decrypt, encrypt using the new key, then re-upload the file. However, this can be very expensive, particularly for modern applications involving large databases, or if the client has limited processing capability (as is the case with smartphones), or where bandwidth is expensive. A preferable solution would involve utilising the cloud's often advanced computational capacity, as opposed to placing the entire computational burden on the client.

Proxy Re-Encryption (PRE), introduced in [BBS98], aims to solve this problem. It allows a ciphertext encrypted using one key to be transformed into an encryption under another key without revealing either the plaintext or the decryption keys to the proxy. This is achieved by the client computing an *update token* which it sends to the cloud or 'proxy', enabling it to perform the transformation on the client's behalf. In other words, PRE allows clients who store encrypted files in the cloud to outsource re-encryption to new keys to the cloud without compromising either the secret keys or the underlying data.

I: 3 A Brief Introduction to Computational Security

One of the main advantages that cryptographic solutions offer over the alternative solutions discussed in Section I: 1 is that the security of cryptographic schemes is defined formally and in such a way that it can be proven. In this section, we discuss how security in cryptography is defined. We explain what formal security is, why it is necessary, and some basic techniques for proving security.

Historically, cryptography was mainly used by privileged entities such as governments and militaries. Its development was kept secret, and was generally considered more as a craft requiring creativity than a scientific discipline requiring scrutiny, peer review and formal proofs. As a result, history is littered with examples of poor cryptography some of which had a significant impact. For example, Mary, Queen of Scots was executed for participating in the Babington plot after her letters were intercepted and successfully decrypted. The code used was a type of substitution cipher, and therefore prone to simple statistical analysis [Sin00]. Another example is the breaking of the Enigma machine cipher used by the Nazis in the Second World War, which is thought to have shortened the war by two years [Kee10]. Whilst the developers believed their cipher to be unbreakable, it had features that enabled successful cryptanalysis, such as letters never mapping to themselves when encrypted [Jon13].

As a result of these failings, one of the important developments in modern cryptography is *formal security.* Formal security provides a means of specifying what 'security' means for a cryptographic primitive, and provides a mechanism for proving that a particular construction meets that definition. As written by Jonathan Katz and Yehuda Lindell in *'Introduction to Modern Cryptography'* [KL14]:

"...formal definitions of security are an essential first step in the design of any cryptographic primitive or protocol".

Definitions are a powerful tool for helping us understand precisely what attacks a cryptographic primitive protects against, and what attacks it does not. Definitions in cryptography model precise attacks and scenarios, with the overall aim of modelling real-world attacks. They take a high-level idea such as message confidentiality, and translate it into terms that give precision as to what a cryptographic scheme protects against from which a mathematical proof of security can be created. In Phillip Rogaway's paper *'On the Role of Definitions in and Beyond Cryptography'* [Rog04], he argues that it is definitions that tell us what a cryptographic primitive achieves, rather than specific algorithms or software. A comprehensive overview of the importance and usefulness of definitions in cryptography can be found in [Rog04].

Computational security is a formal approach to security which was first published by Goldwasser and Micali in 1984 [GM84] when they defined message privacy – the basic notion which encryption schemes are expected to meet. In this framework, security is defined in terms of a *challenger* running a *game* (or *experiment*) against an *adversary.* This adversary is assumed to be a Probabilistic Polynomial-Time (PPT) algorithm that the challenger calls on during the game which tries to correctly answer the challenge issued by the challenger. If a PPT adversary exists that can win the game significantly more often than would be expected by randomly guessing the correct answer, then the scheme is not considered secure. If there are no PPT adversaries that can win the game with a better strategy than guessing, then the scheme is considered secure.

The experiment run by the challenger describes precisely what information the adversary has access to and what their goal is. In computational security, cryptographic schemes are proven secure using an assumption which states that a particular problem is hard. We use a *reduction* to demonstrate how breaking the security of the scheme can be used to break the underlying assumption. We provide a more detailed discussion on how computational security works in Section II: 3 after we have presented the necessary preliminaries in Chapter II.

I: 4 Discussion on Security Definitions

Definitions not only allow us to assess the security of a particular scheme, but they can be compared to each other to determine which is stronger. For example, security against random-plaintext attacks (where two random messages are encrypted) is weaker than security against chosen-plaintext attacks (when the adversary gets to choose two specific messages). Therefore, if a scheme is not secure against random-plaintext attacks, then it is not secure against chosen-plaintext attacks.

Definitions can also be used to form basic observations that teach us the minimum requirements a scheme needs for the definition to be met. For example, for a public-key encryption scheme to be Indistinguishable against Chosen Plaintext Attacks (IND-CPA-secure), the encryption algorithm must introduce some randomness to the ciphertext to prevent encryptions of the same message always being the same. We shall elaborate on this in Section II: 3.2. Therefore, when constructing a scheme that is IND-CPA-secure, cryptographers know that some random sampling in the encryption algorithm is necessary.

I: 4.1 Creating a definition

There are several aspects to consider in the process of creating a formal security definition:

1. *The high-level (informal) idea.*

For example, “An eavesdropper should not be able to read encrypted messages”. It is good practice to give an informal definition together with the formal one, to give the reader an intuition of what the definition means.

2. *Defining what the attacker’s goal is.*

   Giving a precise idea of what constitutes a ‘win’ in the game. For an encryption scheme, does the attacker need to recover the full message, or do they just need to determine which of two messages has been encrypted? In practice definitions are often stronger than may seem necessary or realistic. This is partly because historically, people have been mistaken as to what an attacker can realistically do[3], and partly because if a scheme can be proven to meet a stronger definition then this is clearly preferable from a theoretical standpoint.

3. *Defining the advantage.*

   In the computational model, it is technically possible for an adversary to output a random guess and win the game. We are therefore more concerned with the advantage of winning the game over guessing. How the advantage is defined depends on the game. For example, if the adversary needs to choose between one of two options as in an indistinguishability game, then a random guess will be correct with probability $\frac{1}{2}$. The adversary’s advantage of winning an indistinguishability game is therefore defined as $\left[\left|(\text{probability of winning}) - \frac{1}{2}\right|\right]$. In other games where the adversary must output a specific value, such as creating a forged signature, the advantage is closer to the probability of winning the game at all as opposed to having to adjust for the possibility of a random guess being successful.

4. *What can the attacker do?*

   We must define what information is available to the adversary, including whether the challenge is in response to any input from the adversary or generated independently. For example, for plaintext indistinguishability, does the attacker know which two messages could have been encrypted? For an encryption scheme that is IND-CPA-secure,

[3]During the Pacific War, American intelligence intercepted Japanese communications which indicated that “AF” was under threat. Whilst they suspected that “AF” referred to Midway, they could not confirm this through cryptanalysis alone and the higher-ups wanted confirmation before agreeing to take action, as they believed Midway to be an unlikely target. Intelligence proceeded to send an encrypted message to Midway, asking them to send a plaintext message saying that their supplies were low. This message was intercepted by the Japanese who then sent an encrypted message indicating that supplies were low at “AF” – thus American intelligence was able to deduce that “AF” did indeed refer to Midway [Wea00]

the attacker gets to decide those messages.

A particular point of interest is whether to write the theoretically strongest possible definition, or to settle for a definition appropriate for a specific scenario. For example, the challenger in an IND-CCA-secure game gives the attacker a decryption oracle which enables them to decrypt any ciphertext *apart from* the challenge ciphertext. Whilst this is clearly stronger than IND-CPA-secure security, it requires extra cryptographic measures to be in place which may be more than is necessary for the basic encryption scenario where the eavesdropper has no means of decrypting any ciphertexts.

I: 4.2 What makes a new definition useful?

We now discuss our beliefs on what needs to be considered when creating a useful security definition in cryptography. The art of devising relevant definitions in cryptography requires the following aspects to be considered:

The definition should model an actual threat. Motivations for a new definition are highly important. A new definition is unlikely to be taken seriously unless there is a clear application for a construction that meets it, and this threat is not covered by existing definitions.

There should be a construction that meets the definition. Whilst definitions can provide a gold standard for a cryptographic primitive to meet, not being able to provide a scheme that meets it limits the usefulness of the definition – it is easy to create a notion of security which is very strong in theory, but impossible to meet in reality. Providing a tangible construction connects theory and practice[4]. It is also useful to indicate how such a scheme might be created by demonstrating results around the definition that convey necessary elements or properties that any construction meeting the definition must have, such as the need for certain algorithms to be probabilistic. This helps other cryptographers to create constructions that meet the definition. In other words, definitions should facilitate *composability* – where we can take a modular approach both to the design of constructions and the proof that these constructions meet the given definition [Rog04].

The definition should be as simple as possible. Definitions which are overly complicated or appear substantially different from similar definitions will be harder to understand and

[4]Note however that just because a construction exists, does not mean this construction will be efficient enough to be considered practical for real-life implementation.

are therefore less likely to be picked up by the cryptographic community and contribute to further advancements. Complex definitions may also result in complicated proofs, which are more likely to contain (potentially unnoticed) errors. Creating a definition which has a similar structure to related definitions is helpful both in making the two easier to compare, and in making it possible for proof techniques used for the related definition to be useful in proofs for the new one. Whilst it is often inevitable that stronger definitions are more complicated than weaker ones (for example, if the adversary is given additional oracles), it is important to keep the complexity to a minimum.

Many of the contributions in this thesis involve creating new security definitions. These are the principles we used for these contributions. In each case, we describe a construction that provably meets the definition. Many of our new definitions builds on existing ones, and for those that differ we seek to be as intuitive as possible.

I: 5 Motivation for studying PRE

In this thesis, we investigate the extent to which cryptography can enable clients outsourcing to a third party such as the cloud to retain control over their data, focusing on the cryptographic primitive of PRE. Recall that PRE is a cryptographic primitive that allows a ciphertext encrypted using one key to be transformed to an encryption under a new key without revealing either the plaintext or the decryption keys to the proxy.

Whilst out primary motivation is investigating how cryptography in the cloud can enable clients to have greater control over their data, our motivations for exploring PRE in particular are twofold. Firstly, security in most cryptographic schemes often comes down to how well the keys are managed. Therefore, the ability to change keys without losing security, and the opportunities presented by this, are worth further exploration. Secondly, unlike some other cryptographic schemes, the problem of changing keys already has a trivial solution – namely the client downloading, locally decrypting and encrypting using the new key, and then re-uploading the file. In comparison, other cryptographic primitives such as multi-party computation (where two or more parties wish to compute a function over their combined inputs without revealing the inputs to any other party) have no trivial solution outside cryptography. PRE schemes that are useful in practice therefore should have some advantage over the trivial solution. For example, the email forwarding application is justified by pointing out that re-encryption does not require Alice to be online once the server has the update token. Other practical benefits that a PRE scheme should have in order to be practically useful include it being cheaper for clients, either computationally or in terms of the bandwidth used. This caveat means that building a PRE scheme using some of the more intensive cryptographic tools (e.g. obfuscation) limits its usability, and so constructing a scheme that is useful in practice presents an interesting challenge. We shall elaborate on this when we look at existing PRE schemes in Section III: 4.

The original motivation given for PRE in [BBS98] was *email forwarding* – if Alice is going on holiday and wants to forward her emails to her secretary, Bob, without giving him her secret key, then she can have her email server re-encrypt all her emails to Bob's public key by generating an update token for the server to use during this time. In Section I: 2, we discussed other situations where it is beneficial to change the key that a ciphertext has been encrypted under, namely *key rotation*, and *enforcing cryptographic access control*. An initial exploration reveals that PRE in practice is not always appropriate for these applications, both in terms of how security is defined, and in constructions. We do not go into more detail here, as further explanation requires some preliminary understanding of PRE. We therefore revisit our overall motivations in Section III: 5, and describe the contributions of each chapter in their respective introductions.

I: 6 Overview of Contributions & Roadmap

In this thesis, we investigate how well PRE is suited to key expiry and access revocation. We do this by examining both existing security definitions and constructions, and proposing new alternatives. We also seek to model adversaries that are stronger than in existing models. We examine the existing security definitions for PRE and note where they are lacking with respect to applications where keys become compromised. We then fill this gap by giving new security definitions, and present constructions that provably meet those definitions.

In Chapter II we describe general preliminaries for mathematics and cryptography, before describing preliminaries specific to PRE in Chapter III. We then move on to our main contributions. In Chapter IV we explore definitions that prevent the proxy from performing unauthorised re-encryptions. In Chapter V we define Post-Compromise Security for PRE, as a means of identifying when a scheme is appropriate for key rotation and access revocation. In Chapter VI we look at strengthening our definitions to the adaptive key corruption model, and how to prove security in this model. Finally, we offer concluding remarks in Chapter VII.

Chapter II

Cryptography Preliminaries

Contents

In this Chapter

We introduce notation and preliminaries for general cryptography, and define some standard cryptographic schemes. We then discuss formal security in more detail, giving explicit security games and formally defining the concepts used to define something as secure in the computational model, before stating the standard assumptions used in computational proofs. Finally, we present some constructions of cryptographic primitives which are built on later in this thesis in the construction of PRE schemes.

II: 1 Maths and Notation

Here we discuss general notation, but note that more specific notation is defined in the relevant sections.

We denote by $\mathbb{Z}_n$ the set of integers modulo n, and by $\mathbb{Z}_n^*$ the set $\{1, 2, \ldots, n-1\}$. For a set $\mathcal{X}$, sampling an element x uniformly at random from $\mathcal{X}$ is denoted $x \overset{\$}{\leftarrow} \mathcal{X}$. We say a distribution D is (B, δ)-*bounded* if $\Pr\left[|x| > B : x \overset{\$}{\leftarrow} D\right] \leq \delta$. We use $x \overset{\$}{\leftarrow} D$ to denote that the element x is sampled according to distribution D.

We sometimes abuse this notation by writing $y \overset{\$}{\leftarrow} \mathsf{Alg}(x)$ to explicitly convey that the algorithm Alg is probabilistic and therefore generates some randomness in creating output y. We write $y \leftarrow \mathsf{Alg}(x)$ when Alg is deterministic, and $y \overset{(\$)}{\leftarrow} \mathsf{Alg}(x)$ for the general case where the function could either be probabilistic or deterministic, depending on the specific scheme.

II: 2 Basic cryptography definitions

Definitions in cryptography use a *security parameter*, $\lambda \in \mathbb{N}$ which is usually represented in unary as 1^λ. The security parameter measures how 'hard' it is to break the security of the cryptosystem, and influences parameter choices in cryptographic algorithms.

There are two main types of encryption – symmetric and public-key (also called asymmetric). We begin by defining both, as we shall discuss both symmetric and public-key PRE schemes.

Definition II.1. *A* Symmetric Encryption (SE) *scheme consists of the following algorithms:*

- $\mathsf{Setup}(1^\lambda) \overset{(\$)}{\rightarrow}$ params*: Outputs a set of public parameters* params*, including the message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$.*

  Note that params *is input to every subsequent algorithm, but we omit it for compactness of notation. We often omit the* Setup *algorithm from security games for the same reason.*
- $\mathsf{KeyGen}(1^\lambda) \overset{\$}{\rightarrow} k$*: A probabilistic algorithm that returns a symmetric key, k.*
- $\mathsf{Enc}(k, m) \overset{\$}{\rightarrow} c$*: Encrypts a message, m, using a key, k, producing a ciphertext, c. In this thesis we assume that* Enc *is probabilistic.*
- $\mathsf{Dec}(k, c) \rightarrow m' \cup \perp$*: A deterministic algorithm that takes a ciphertext c and produces either an element of the message space m' or an error symbol $\perp$.*

A symmetric encryption scheme is correct *if for all $m \in \mathcal{M}$ and all $k \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$:*

$$\mathsf{Dec}(k, \mathsf{Enc}(k, m)) \rightarrow m.$$

In a symmetric encryption system, the same key is used for both encryption and decryption, meaning both the sender and receiver of the message must already share a secret key k. In contrast, a public-key encryption system uses one key for encryption, and another for decryption. Since the encryption key cannot be used to decrypt, it does not need to be kept secret and can be publicly broadcast and hence is referred to as the *public key.*

Definition II.2. *A* Public-Key Encryption (PKE) *scheme consists of the following algorithms:*

- $\mathsf{Setup}(1^\lambda) \overset{(\$)}{\to} \mathrm{params}$*: Outputs a set of public parameters* params*, including the message space* $\mathcal{M}$ *and ciphertext space* $\mathcal{C}$*.*

  As before, params *is input to every subsequent algorithm, but we omit it for compactness of notation. We often omit the* Setup *algorithm for the same reason.*
- $\mathsf{KeyGen}(1^\lambda) \overset{\$}{\to} (\mathsf{pk}, \mathsf{sk})$*: A probabilistic algorithm that generates a public-private key pair.*
- $\mathsf{Enc}(\mathsf{pk}, m) \overset{\$}{\to} c$*: An algorithm that encrypts a message* m *using a public key* pk*, producing a ciphertext* c*. In this thesis we assume that* Enc *is probabilistic.*
- $\mathsf{Dec}(\mathsf{sk}, c) \to m' \cup \perp$*: A deterministic algorithm that decrypts a ciphertext* c *using secret key* sk *to produce either an element of the message space* m' *or an error symbol* $\perp$*.*

A PKE scheme is correct *if for all* $m \in \mathcal{M}$ *and all* $(\mathsf{pk}, \mathsf{sk}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$*:*

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) \to m.$$

Public-key cryptosystems enable two parties to communicate securely even if they have had no previous interaction with which to share a key. This separation between keys is useful for scenarios where the sender and receiver do not have a mutual trust relationship, as is the case when sharing a symmetric key as this gives the power to decrypt as well as encrypt messages. We shall see this when we explore security notions for PRE.

Another useful cryptographic primitive which we discuss in this thesis is *digital signature schemes.* These allow one party to sign a message, and another to verify the signature. This is also a public-key primitive, as the signing key must be kept secret so that only the relevant party can create signatures, but the verification key is public so that anyone with the verification key can verify the signature.

Definition II.3. *A* digital signature scheme *consists of the following algorithms:*

- $\mathsf{Setup}(1^\lambda) \overset{(\$)}{\to}$ params*: Outputs a set of public parameters* params*.*
- $\mathsf{KeyGen}(1^\lambda) \overset{\$}{\to} (\mathsf{svk}, \mathsf{ssk})$*: A probabilistic algorithm that generates a signature signing key* ssk *and a signature verification key* svk*.*
- $\mathsf{Sign}(\mathsf{ssk}, m) \overset{\$}{\to} \sigma$*: Signs a message* m *using a signing key* ssk*, producing a signature* σ*. We assume that* Sign *is probabilistic.*
- $\mathsf{Verify}(\mathsf{svk}, \sigma, m) \to \{0,1\}$*: A deterministic algorithm that verifies a signature* σ *over a message* m*, returning a bit.*

A digital scheme is correct *if for all* $m \in \mathcal{M}$ *and all* $(\mathsf{svk}, \mathsf{ssk}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$*:*

$$\mathsf{Verify}(\mathsf{svk}, \mathsf{Sign}(\mathsf{ssk}, m), m) \to 1.$$

Whilst encryption is designed for message confidentiality, signatures are designed for authentication. Consequently, what it means for these primitives to be ‘secure’ is different. We shall discuss the security of encryption schemes and digital signature schemes explicitly in Section II: 3.2.

The last primitive that we use in this thesis is a *(cryptographic) hash function.* For brevity, we explicitly define a secure cryptographic hash function as opposed to defining security separately.

Definition II.4. *A* secure cryptographic hash function (with output length l) *is a deterministic function* $h : \{0,1\}^* \to \{0,1\}^l$ *that takes an arbitrary-length input* x *and outputs a string* $h(x) = y$ *of length* l *such that:*

- *Given* y*, no* PPT *adversary can find* x *(preimage resistance).*
- *Given* x, y*, no* PPT *adversary can find an* $x' \neq x$ *such that* $h(x') = y$ *(second preimage resistance).*
- *No* PPT *adversary can find any* x, x' *such that* $h(x) = h(x')$ *(collision resistance).*

II: 3 Formal security

Recall that computational security is defined according to a challenger running an experiment (also called a game) against an adversary, $\mathcal{A}$, assumed to be a Probabilistic Polynomial-Time (PPT) algorithm. The size of $\mathcal{A}$ is given according to a security parameter, λ, usually represented in unary as 1^λ. The adversary can either be single-stage or multi-stage ($\mathcal{A} = (\mathcal{A}_0, \ldots, \mathcal{A}_n)$) and in the latter case is assumed to pass on a *state* to future stages conveying its current knowledge. The adversary is assumed to know the encryption algorithm being used, in keeping with Kerckhoffs' principle [Ker83], and other details such as the public parameters used by the cryptographic scheme being assessed.

The challenger may give some additional information to the adversary by providing *oracles*, where the adversary can choose inputs and receive the corresponding outputs. These functions may rely on secrets only known to the challenger, and thereby give the adversary a means of learning information computed by honest parties during normal activity of the cryptosystem, and potentially infer information about the underlying secrets. Sometimes, oracles carry *restrictions* on what the adversary can learn, generally to prevent the adversary from gaining a trivial win. For example, the challenge oracle in a chosen-plaintext indistinguishability game checks that the input messages are the same size ($|m_0| \overset{?}{=} |m_1|$), as a huge size discrepancy will result in different sized ciphertexts, making it simple for the adversary to know which message was encrypted. Some oracles can be queried only once, and we convey this using a state called maintained by the challenger. We use the notation $\mathcal{A}^{\mathsf{O_f}}$ to mean that adversary $\mathcal{A}$ can access oracle $\mathsf{O_f}$.

Adversarial *advantage* is a crucial concept in defining computational security. Informally, we define the advantage that an adversary has in winning the game to be the probability that the adversary wins, minus the probability that a random guess would win the game. The way in which advantage is determined depends on the game. In this thesis, we define advantage implicitly in our security definitions. Cryptographic schemes are considered secure if no adversary has a significant advantage in winning the game. We use the following to define whether an advantage is significant.

Definition II.5. *A function f is* negligible *if for every polynomial $p()$ there exists an N*

such that for all integers $\lambda > N$:

$$f(\lambda) < \frac{1}{p(\lambda)}.$$

We denote negligible functions as $\mathsf{negl}(\lambda)$.

If all adversaries can only win the security game with negligible advantage, then the cryptosystem meets the definition associated with that game. The concept of *overwhelming probability* compliments negligible probability. Informally, it means that the probability of losing is negligible.

Definition II.6. *Given a security parameter λ, we say that the probability p of success is* negligible *if there exists a negligible function $\mathsf{negl}(\lambda)$ such that $p \leq \mathsf{negl}(\lambda)$. We say the probability of success is* overwhelming *if $p \geq 1 - \mathsf{negl}(\lambda)$.*

II: 3.1 Mathematical assumptions

In computational security, a proof leverages a well-known mathematical assumption taken from number theory. Whilst unproven, these assumptions are well-studied and generally considered reliable by the cryptographic community. Proofs of security usually give a *reduction* which demonstrates that an adversary who can break the security of a construction would also be able to break the mathematical assumption.

The main assumptions used in the constructions described in this thesis are *Diffie-Hellman* and *ring-Learning With Errors*, which we now introduce.

II: 3.1.1 Diffie-Hellman

Diffie-Hellman assumptions underpin the security of ElGamal-based PRE schemes, which make up the majority of the literature, including the original PRE scheme [BBS98] and many subsequent schemes such as [LV08b, ABH09, CWYD10].

The Decisional Diffie-Hellman (DDH) problem. Let p be a prime and let g be a generator of the group $\mathbb{Z}_p$. Given a tuple (g, g^a, g^b, g^c) where $a, b \xleftarrow{\$} \mathbb{Z}_p^*$, determine whether $c \xleftarrow{\$} \mathbb{Z}_p^*$ or $c = ab$. The *DDH assumption* states that there is no PPT algorithm that can solve the DDH problem.

The Computational Diffie-Hellman (CDH) problem. Let p be a prime and let g be a generator of the group $\mathbb{Z}_p$. Given a tuple (g, g^a, g^b) where $a, b \xleftarrow{\$} \mathbb{Z}_p^*$, compute g^{ab}. The *CDH assumption* states that there is no PPT algorithm that can solve the CDH problem.

Lemma II.1. *If there exists a* PPT *adversary* $\mathcal{A}$ *that can solve the CDH problem, then there exists another* PPT *adversary* $\mathcal{B}$ *that can solve the DDH problem. In other words the hardness of the DDH problem implies that the CDH problem is also hard:*

$$\textit{DDH assumption} \implies \textit{CDH assumption.}$$

Proof. We show that if the CDH assumption is false, then one can solve the DDH problem. Given (g, g^a, g^b, g^c), $\mathcal{B}$ can call $\mathcal{A}(g, g^a, g^b) \to g^{ab}$ and compare this to g^c. $\square$

II: 3.1.2 RLWE

We now present notation and preliminaries relevant for lattice-based cryptography, which is built on the Learning With Errors (LWE) problem (or ring-LWE (RLWE) problem). We let $q \in \mathbb{Z}$ denote some modulus and n denote a ring dimension. We represent the set of integers modulo q as $\mathbb{Z}_q = \{\lfloor -q/2 \rfloor, \ldots, 0, \ldots, \lfloor q/2 \rfloor\}$. In lattice-based cryptography, computations are performed over power-of-two cyclotomic rings of the form $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle x^n + 1 \rangle$ where n is a power of two. We denote the discrete Gaussian distribution over $\mathbb{Z}_q$ as χ_σ. The distribution χ_σ has its support restricted to $\mathbb{Z}_q$ and a probability mass function proportional to that of a Gaussian distribution with variance σ^2. We sometimes write χ_e where we use a subscript e to denote an "error" distribution, but the underlying variance is still denoted by σ.

We can sample a polynomial $s \xleftarrow{\$} \chi_e$ by sampling each of the coefficients of s according to the distribution χ_e. We denote the uniform distribution over $\mathcal{R}_q$ as $\mathcal{U}_q$, and use the shorthand $(a, b) \xleftarrow{\$} \mathcal{U}_q \times \mathcal{U}_q$ to mean that $a \xleftarrow{\$} \mathcal{U}_q$, $b \xleftarrow{\$} \mathcal{U}_q$. The *RLWE*$_{n,q,\chi_e}(s)$ *distribution* outputs values of the form $(a, b = as + e) \in \mathcal{R}_q \times \mathcal{R}_q$ where $a \xleftarrow{\$} \mathcal{U}_q$, $e \xleftarrow{\$} \chi_e$.

The (normal form) ring-LWE (RLWE) problem. Let $s \xleftarrow{\$} \chi_e$. Distinguish whether a given oracle outputs samples from $\mathrm{RLWE}_{n,q,\chi_e}(s)$ or uniform elements in $\mathcal{R}_q \times \mathcal{R}_q$.

The *RLWE*$_{n,q,\chi_e}$ *assumption* states that no PPT algorithm can solve the $\mathrm{RLWE}_{n,q,\chi_e}$ problem with a non-negligible advantage[1]. In this thesis we focus on the RLWE problem

[1]Note that the RLWE assumption is built from other assumptions in lattice-based cryptography,

as opposed to other standard problems in lattice-based cryptography as it is the underlying assumption used in the most relevant existing work and produces more practical constructions.

II: 3.2 Security in Public-Key Encryption

We now present the basic confidentiality game for PKE schemes. Informally, it states that an adversary cannot determine the underlying plaintext given a ciphertext, even when the adversary chooses the two messages that could be encrypted.

Definition II.7. *Consider the following security game:*

```
$\underline{\mathsf{IND\text{-}CPA}^{\mathcal{PKE}}_{\mathcal{A}}(1^\lambda)}$
$b \xleftarrow{\$} \{0,1\}$
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$(m_0, m_1, \mathit{state}) \xleftarrow{(\$)} \mathcal{A}_0(1^\lambda, \mathsf{pk})$
$\mathbf{if}\ |m_0| \neq |m_1| : \mathbf{return}\ \bot$
$c_0^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_0)$
$c_1^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1)$
$b' \xleftarrow{(\$)} \mathcal{A}_1(1^\lambda, \mathit{state}, c_b^*)$
$\mathbf{return}\ (b' = b)$
```

A PKE scheme $\mathcal{PKE}$ is said to be ϵ-Indistinguishable against Chosen Plaintext Attacks (ϵ-IND-CPA-secure) *if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:*

$$\Pr\left[\mathsf{IND\text{-}CPA}^{\mathcal{PKE}}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon.$$

If ϵ is negligible as determined by the security parameter λ then we say $\mathcal{PKE}$ is Indistinguishable against Chosen Plaintext Attacks (IND-CPA-secure).

Here the advantage ϵ is the additional probability of a win over a random guess (which happens with probability $\frac{1}{2}$). This is standard for indistinguishability games as stated in Section I: 4. The definition we give here is the single-challenge game, as opposed to the multi-challenge game where the adversary has access to a left-or-right oracle which it can call multiple times on pairs of messages to obtain an encryption of either m_0 or

namely the LWE problem and Shortest Integer Solution (SIS) problem. If we take $\sigma \geq \omega(\log n)$ and $\sigma/q = 1/\mathrm{poly}(n)$, then $\mathrm{RLWE}_{n,q,\chi_e}$ problem has been shown to be at least as hard as solving standard worst-case lattice problems over ideal lattices up to polynomial approximation factors using quantum algorithms. See [LPR10] for full details.

m_1, as determined by b. Observe that because this is a public-key scheme, the adversary has the ability to encrypt both m_0 and m_1 locally. This implies that encryption schemes which meet this definition cannot have a deterministic encryption function – otherwise, the adversary could locally encrypt m_0 and m_1 and compare the resulting ciphertexts with the challenge ciphertext c_b^* to win the game. In this thesis we generally assume that encryption schemes are at least IND-CPA-secure, hence we assume Enc is probabilistic.

The assumption of algorithms being probabilistic or deterministic can influence how security games are written, and in particular what information is assumed to lead to a trivial win if given to the adversary. This idea of specific algorithms needing to be probabilistic is one we will revisit in this thesis.

We now present another, stronger, definition for message confidentiality.

Definition II.8. *Consider the following security game:*

```
$\mathsf{IND\text{-}CCA}_{\mathcal{A}}^{\mathcal{PKE}}(1^\lambda)$
$b \xleftarrow{\$} \{0,1\}$
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$(m_0, m_1, state) \xleftarrow{(\$)} \mathcal{A}_0(1^\lambda, \mathsf{pk})$
if $|m_0| \neq |m_1|$ : return $\perp$
$c_0^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, m_0)$
$c_1^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, m_1)$
$b' \xleftarrow{(\$)} \mathcal{A}_1^{\mathsf{O_{Dec}}}(1^\lambda, state, c_b^*)$
return $(b' = b)$

$\mathsf{O_{Dec}}(c) \to m$
if $c = c_b^*$ : return $\perp$
$m \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
return $m$
```

A PKE scheme $\mathcal{PKE}$ is said to be ϵ-Indistinguishable against Chosen Ciphertext Attacks (ϵ-IND-CCA-secure) *if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:*

$$\Pr\left[\mathsf{IND\text{-}CCA}_{\mathcal{A}}^{\mathcal{PKE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon.$$

If ϵ is negligible as determined by the security parameter λ then we say $\mathcal{PKE}$ is Indistinguishable against Chosen Plaintext Attacks (IND-CPA-secure).

In $\mathsf{IND\text{-}CCA}_{\mathcal{A}}^{\mathcal{PKE}}(1^\lambda)$, $\mathcal{A}_1$ has access to a decryption oracle $\mathsf{O_{Dec}}$, which carries the restriction that it cannot be queried on the challenge ciphertext c_b^*. By definition of a correct PKE scheme, being able to decrypt the challenge ciphertext will enable the adversary to win the game. Therefore, for CCA games to have any meaning over CPA games, we need to make this restriction. These types of restrictions are more common for powerful oracles such as decryption. We note that CCA security is strictly stronger than CPA security, as the adversary has access to more information.

We now move on from message confidentiality to give a different security definition for encryption schemes. *Key privacy* (also known as *key anonymity*) was first introduced by Bellare et al [BBDP01] for public-key encryption. Informally it means that a ciphertext does not reveal which key was used to encrypt the message. In other words, given a ciphertext, the receiver of the ciphertext remains anonymous.

Definition II.9. *Consider the following security game:*

```
$\mathsf{KeyPriv}_{\mathcal{A}}^{\mathcal{PKE}}(1^\lambda)$
$b \overset{\$}{\leftarrow} \{0,1\}$
$(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$
$(m, state) \overset{(\$)}{\leftarrow} \mathcal{A}_0(1^\lambda, \mathsf{pk}_0, \mathsf{pk}_1)$
$c_0^* \leftarrow \mathsf{Enc}(\mathsf{pk}_0, m)$
$c_1^* \leftarrow \mathsf{Enc}(\mathsf{pk}_1, m)$
$b' \overset{(\$)}{\leftarrow} \mathcal{A}_1(1^\lambda, state, c_b^*)$
return $(b' = b)$
```

A Public-Key Encryption (PKE) scheme $\mathcal{PKE}$ is said to be ϵ-key private *(or* ϵ-key anonymous*) if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:*

$$\Pr\left[\mathsf{KeyPriv}_{\mathcal{A}}^{\mathcal{PKE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon.$$

If ϵ is negligible as determined by the security parameter λ then we say $\mathcal{PKE}$ is key private.

As in $\mathsf{IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PKE}}(1^\lambda)$, the adversary has the ability to locally encrypt the message as it is given both public keys, so key privacy also requires the encryption function to be probabilistic.

Note that for $\mathsf{IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PKE}}(1^\lambda)$, $\mathsf{IND\text{-}CCA}_{\mathcal{A}}^{\mathcal{PKE}}(1^\lambda)$ and $\mathsf{KeyPriv}_{\mathcal{A}}^{\mathcal{PKE}}(1^\lambda)$, equivalent formulations for symmetric schemes can be achieved by giving the adversaries access to an encryption oracle $\mathsf{O}_{\mathsf{Enc}}(m)$, where the adversary inputs a message m, and receives a ciphertext $c \overset{\$}{\leftarrow} \mathsf{Enc}(k, m)$ (for $\mathsf{KeyPriv}_{\mathcal{A}}^{\mathcal{PKE}}(1^\lambda)$, the oracle also takes an index $i \in \{0,1\}$ as input so the adversary can specify which key it would like the message to be encrypted under). The encryption oracle is necessary as, without knowledge of the symmetric key, the adversary cannot learn any ciphertexts under keys generated by the challenger, aside from the challenge ciphertext.

II: 3.3 Security of digital signature schemes

The security of digital signature schemes is based on the inability of an attacker to create a valid signature on a message. In the following security game, the adversary has access to a signing oracle $\mathsf{O}_{\mathsf{Sign}}$ which will sign any message the adversary gives it. This necessitates the winning condition that the adversary cannot submit a signature on a message that it previously sent to the signing oracle. The challenger therefore maintains a list, $\mathcal{Q}$, of messages sent to $\mathsf{O}_{\mathsf{Sign}}$ for this purpose.

Definition II.10. *Consider the following security game:*

```
$\mathsf{SigForge}^{\Pi}_{\mathcal{A}}(1^\lambda)$
$\mathcal{Q} \leftarrow \emptyset$
$(\mathsf{ssk}, \mathsf{svk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$(\sigma, m) \xleftarrow{(\$)} \mathcal{A}^{\mathsf{O}_{\mathsf{Sign}}}(1^\lambda, \mathsf{svk})$
if $(m \in \mathcal{Q}) \vee (\mathsf{Verify}(\mathsf{svk}, \sigma, m) \neq 1)$ :
  return $0$
else :
  return $1$
```

```
$\mathsf{O}_{\mathsf{Sign}}(m) \xrightarrow{(\$)} \sigma$
$\sigma \xleftarrow{(\$)} \mathsf{Sign}(\mathsf{ssk}, m)$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$
return $\sigma$
```

A digital signature scheme Π *is said to be* ϵ-existentially unforgeable under an adaptive chosen-message attack *if for all* PPT *adversaries* $\mathcal{A}$*:*

$$\Pr\left[\mathsf{SigForge}^{\Pi}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \epsilon.$$

If ϵ *is negligible as determined by the security parameter* λ *then we say* Π *is* existentially unforgeable under an adaptive chosen-message attack.

II: 4 Some cryptographic schemes

In this section, we present examples of established cryptographic schemes. We focus on those which form the underlying bases of the PRE schemes discussed in this thesis.

II: 4.1 The ElGamal encryption scheme

We describe the ElGamal encryption scheme in Figure II.1.

```
$\mathsf{Setup}(1^\lambda) \to (p, g)$
    Select a prime $p$
    Select $g$, a generator of $\mathbb{Z}_p$
    params $= (p, g)$
    **return** params

$\mathsf{KeyGen}(1^\lambda) \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$
    $x \xleftarrow{\$} \mathbb{Z}_p$
    $(\mathsf{pk}, \mathsf{sk}) = (g^x, x)$
    **return** $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Enc}(\mathsf{pk}, m) \xrightarrow{\$} c$
    $y \xleftarrow{\$} \mathbb{Z}_p$
    $c_0 = g^y$
    $c_1 = \mathsf{pk}^y \cdot m$
    **return** $c = (c_0, c_1)$

$\mathsf{Dec}(\mathsf{sk}, c) \to m'$
    $(c_0, c_1) \leftarrow c$
    $s = c_0^{\mathsf{sk}}$
    $m' = s^{-1} \cdot c_1$
    **return** $m'$
```

Figure II.1: The ElGamal encryption scheme [ElG86]. All operations are performed modulo p.

We can demonstrate correctness as decryption derives $s = (g^y)^x = g^{yx}$, meaning that for a ciphertext $c = (c_0, c_1) = (g^y, (g^x)^y \cdot m)$:

$$\begin{aligned} m' &= s^{-1} \cdot (g^x)^y \cdot m \\ &= g^{-yx} \cdot g^{yx} \cdot m \\ &= m. \end{aligned}$$

Security can be proven via the DDH assumption. Informally, for a DDH challenge g^a, a^b, g^c, one can set the public key as g^a and challenge ciphertext as $c_b^* = (g^b, g^c \cdot m_b)$. Then if there is a PPT adversary that is able to determine b, then this implies that $c = ab$. See [ElG86] for full details.

II: 4.2 The BV encryption scheme

We now describe a PKE scheme based on lattices. Whilst the construction we present in Figure II.2 is based on [BV11], the version we present is the one presented in [PRSV17], which is defined over rings and comes from the work of Lyubashevsky, Peikert and Regev [LPR10, LPR13] and Micciancio [Mic10].

The security parameter is interpreted as a "root Hermite factor", δ, as defined in [LP11, GN08]. The ciphertext space is determined by the ring $\mathcal{R}_q := \mathbb{Z}_q[x]/\langle x^n + 1\rangle$, where q is the ciphertext modulus and n is the ring dimension. Errors are sampled from a B_e-bounded discrete Gaussian error distribution χ_e with distribution parameter σ_e. Each coefficient in the polynomials is represented in the range $\{-\lfloor \frac{q}{2} \rfloor, \dots, \lfloor \frac{q}{2} \rfloor\}$. The message space is defined as $\mathcal{M} := \{0, 1, ..., p-1\}^n$.

```
$\mathsf{Setup}(1^\lambda) \to (q, n, p, \sigma_e)$
Select ciphertext modulus $q \in \mathbb{N}$
Select ring dimension $n \in \mathbb{N}$
Select discrete Gaussian error distribution $\chi_e$
Select plaintext modulus $p \geq 2$
params = $(q, n, \chi_e, p)$
return params
```

```
$\mathsf{KeyGen}(1^\lambda) \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$
$a \xleftarrow{\$} \mathcal{U}_q$
$s, e \xleftarrow{\$} \chi_e$
$b := a \cdot s + pe$
$\mathsf{pk} = (a, b), \mathsf{sk} = s$
return $(\mathsf{pk}, \mathsf{sk})$
```

```
$\mathsf{Enc}(\mathsf{pk}, m) \xrightarrow{\$} c$
$(a, b) \leftarrow \mathsf{pk}$
$v, e_0, e_1 \xleftarrow{\$} \chi_e$
$c_0 := b \cdot v + pe_0 + m$
$c_1 := a \cdot v + pe_1$
return $c = (c_0, c_1)$
```

```
$\mathsf{Dec}(\mathsf{sk}, c) \to m'$
$(c_0, c_1) \leftarrow c$
$m' := c_0 - \mathsf{sk} \cdot c_1 \mod p$
return $m'$
```

Figure II.2: The BV encryption scheme defined over rings [BV11, PRSV17]. For more detail on parameter selection, see [PRSV17].

We can demonstrate correctness, since given a ciphertext $c = (c_0, c_1) = (b \cdot v + pe_0 + m, a \cdot v + pe_1)$:

$$\begin{aligned}
c_0 - s \cdot c_1 &= b \cdot v + pe_0 + m - s(a \cdot v + pe_1) \\
&= (a \cdot s + pe) \cdot v + pe_0 + m - s(a \cdot v + pe_1) \\
&= m + p(e \cdot v + e_0 - s \cdot e_1) \\
&= m \mod p.
\end{aligned}$$

Security follows from the RLWE assumption. Informally, the RLWE public keys can be replaced with uniform values in $\mathcal{R}_q$, which in turn means that ciphertexts can be replaced with uniform values in $\mathcal{R}_q$. See [BV11] for full details.

II: 4.3 The ElGamal digital signature scheme

We present the ElGamal signature scheme in Figure II.3.

```
$\mathsf{Setup}(1^\lambda) \rightarrow (p, g, h)$
Select a prime $p$
Select $g$, a generator of $\mathbb{Z}_p$
Select $h$, a hash function
params $= (p, g, h)$
return params
```

```
$\mathsf{KeyGen}(1^\lambda) \xrightarrow{\$} (\mathsf{svk}, \mathsf{ssk})$
$\mathbf{x} \xleftarrow{\$} \mathbb{Z}_p$
$(\mathsf{svk}, \mathsf{ssk}) = (g^{\mathbf{x}}, \mathbf{x})$
return $(\mathsf{svk}, \mathsf{ssk})$
```

```
$\mathsf{Sign}(\mathsf{ssk}, m) \xrightarrow{\$} \sigma$
$k \xleftarrow{\$} \mathbb{Z}_p$
$r = g^k$
$s = (h(m) - \mathsf{ssk} \cdot r)k^{-1} \mod p - 1$
$\sigma = (r, s)$
return $\sigma$
```

```
$\mathsf{Verify}(\mathsf{svk}, \sigma, m) \rightarrow \{0, 1\}$
$(\sigma_0, \sigma_1) \leftarrow \sigma$
if $g^{h(m)} = \mathsf{svk}^{\sigma_0} \cdot {\sigma_0}^{\sigma_1}$
  return 1
else :
  return 0
```

Figure II.3: The ElGamal signature scheme [ElG86]. All operations are performed modulo p unless stated otherwise.

We can demonstrate correctness, since given a signature $\sigma = (r, s) = (g^k, (h(m) - \mathbf{x} \cdot r)k^{-1})$:

$$\begin{aligned} g^{h(m)} &= (g^{\mathbf{x}})^r \cdot (g^k)^{(h(m)-\mathbf{x}r)k^{-1}} \\ &= g^{\mathbf{x}r} \cdot g^{h(m)-\mathbf{x}r} \\ &= g^{h(m)}. \end{aligned}$$

Proving unforgeability is similar to proving that ElGamal encryption is IND-CPA-secure, following from the DDH assumption. See [ElG86] for full details.

Chapter III

Proxy Re-Encryption

Contents

In this Chapter

Here we introduce the language and setting of PRE before defining it formally. We present the existing definitions of security for PRE schemes, as well as outlining some existing constructions based on the encryption schemes described in Section II: 4. In light of this information, we describe our thesis motivations and contributions in more detail. In particular, we use existing work to highlight how existing PRE schemes are unsuitable for some obvious applications as discussed in Section I: 5, setting the scene for the contributions presented in this thesis.

III: 1 Introduction to Proxy Re-Encryption

So far we have defined some basic cryptographic primitives, given some standard definitions of security for these primitives and described some well-known constructions which we build on in this thesis. We have also given an idea of how standard assumptions can be used to prove that specific schemes meet given definitions, demonstrating how computational security can be proven, and how it allows definitions to be compared as discussed in Section I: 4.

In this section, we introduce PRE more formally, focusing on public-key PRE. Informally, PRE allows a ciphertext encrypted under one key pk_i to be transformed into an encryption of the same message that can be correctly decrypted using a different secret key sk_j, without needing to reveal either the secret keys or the message to the party performing the re-encryption. This is done via the client generating an *update token* using the current secret key, sk_i, and new public key, pk_j, which is sent to the proxy, who uses it to perform the re-encryption.

III: 1.1 Terminology

We now introduce terminology for PRE that will be used in this thesis.

- *Proxy.* The party who performs the re-encryption operation. The proxy is never directly given either the message m nor any secret keys.
- *Ciphertext under* pk_i. A ciphertext c originally created by encrypting a message, m, that currently decrypts to m using sk_i, possibly having been re-encrypted a number of times.
- *New key / target key.* The key pk_j used to produce an update token $\Delta_{i,j}$.
- *Old key / source key.* The key pk_i where sk_i was used to produce an update token $\Delta_{i,j}$.
- *Underlying message.* The message $m \leftarrow \mathsf{Dec}(\mathsf{sk}_i, c)$ where c is an encryption under pk_i.
- *Fresh ciphertext.* A ciphertext created using the encryption algorithm Enc.
- *Re-encrypted ciphertext.* A ciphertext created by the re-encryption algorithm ReEnc.
- *Level.* The number of times a ciphertext has been or appears to have been re-encrypted.
- *Re-encryption initiator.* The party who computes the update token $\Delta_{i,j}$ This party must be in possession of sk_i.

Use of these terms can be seen visually in Figure III.1.

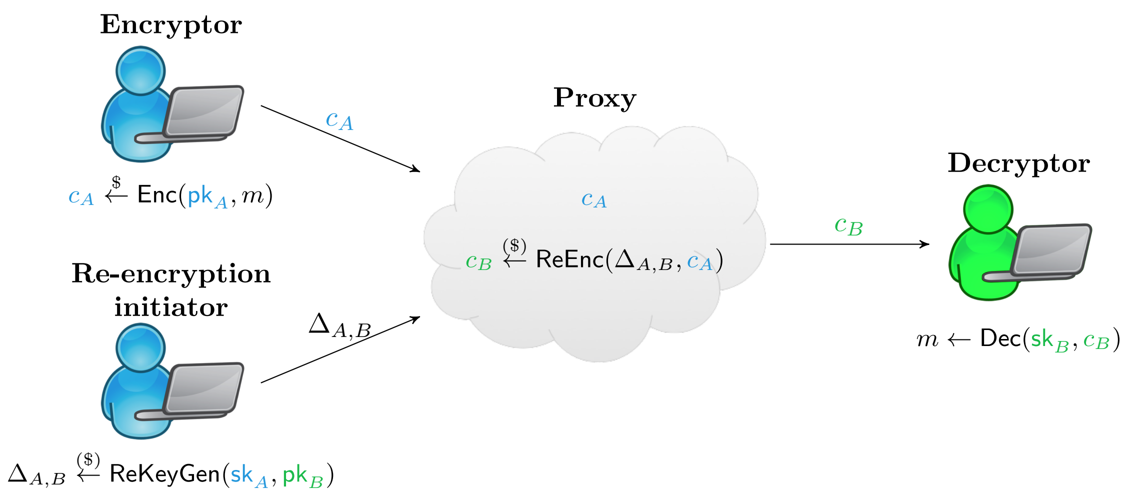
Encryptor
$c_A \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_A, m)$
c_A
Re-encryption initiator
$\Delta_{A,B}$
$\Delta_{A,B} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\mathsf{sk}_A, \mathsf{pk}_B)$
Proxy
c_A
$c_B \xleftarrow{(\$)} \mathsf{ReEnc}(\Delta_{A,B}, c_A)$
c_B
Decryptor
$m \leftarrow \mathsf{Dec}(\mathsf{sk}_B, c_B)$

Figure III.1: The normal PRE operation. sk_A is the source secret key, pk_B is the target public key. c_A is a fresh ciphertext, c_B is a re-encrypted ciphertext, both of which have m as the underlying message. Note that it is not necessary for the encryptor and re-encryption initiator to be the same party, as long as both know sk_A.

III: 2 PRE definitions

The first PRE scheme [BBS98] was defined assuming that both the current secret key sk_i and the target secret key sk_j are known to the party generating the update token needed for the proxy to perform the re-encryption. Subsequent work where update tokens require both secret keys to be known have considered this a symmetric form of PRE, which we now define.

Definition III.1. *A* symmetric PRE *scheme consists of the following algorithms:*

- $\mathsf{Setup}(1^\lambda) \xrightarrow{(\$)} \mathsf{params}$*: Outputs a set of public parameters* params, *including the message space,* $\mathcal{M}$*, ciphertext space,* $\mathcal{C}$*, and token space,* $\mathcal{T}$*.*

  Note that params *is input to every subsequent algorithm, but we omit it for compactness of notation. We often omit the* Setup *algorithm for the same reason.*
- $\mathsf{KeyGen}(1^\lambda) \xrightarrow{\$} k$*: A probabilistic algorithm that generates a symmetric key,* k*.*
- $\mathsf{Enc}(k, m) \xrightarrow{\$} c$*: A probabilistic algorithm that encrypts a message* m *using a key* k*, producing a ciphertext,* c*.*
- $\mathsf{Dec}(k, c) \rightarrow m' \cup \perp$*: A deterministic algorithm that decrypts a ciphertext* c *to produce either an element of the message space,* m'*, or an error symbol,* $\perp$*.*
- $\mathsf{ReKeyGen}(k_i, k_j) \xrightarrow{(\$)} \Delta_{i,j}$*: An algorithm that is either probabilistic or deterministic*

that takes two keys k_i and k_j and outputs an update token $\Delta_{i,j}$.

- $\mathsf{ReEnc}(\Delta_{i,j}, c) \overset{(\$)}{\rightarrow} c'$*: An algorithm that is either probabilistic or deterministic that takes a ciphertext, c, and update token, $\Delta_{i,j}$, and outputs a re-encrypted ciphertext, c'.*

A symmetric PRE scheme is correct *if for all $m \in \mathcal{M}, k \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$:*

$$\mathsf{Dec}(k, \mathsf{Enc}(k, m)) \rightarrow m$$

and if for all $c \in \mathcal{C}$ such that $\mathsf{Dec}(k_i, c) \rightarrow m'$:

$$\mathsf{Dec}(k_j, \mathsf{ReEnc}(\Delta_{i,j}, c)) \rightarrow m'$$

where $k_i, k_j, \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$ and $\Delta_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(k_i, k_j)$.

We remark that definitions in the literature often have an additional condition for ReKeyGen to return $\perp$ if $i = j$. We note that we can generalise updatable encryption [BLMR13, LT18] and key rotation [EPRS17] as types of symmetric PRE where re-encryptions are sequential, always updating from k_i to k_{i+1}. Updatable encryption also has an explicit goal of updating all components of a ciphertext upon re-encryption as opposed to, for example, just the ciphertext header. A full definition of updatable encryption schemes is given in Appendix A. We sometimes use the term '*re-encryption primitive*' as a general term to describe cryptographic primitives that re-encrypt ciphertexts from one key to another.

More recent work on PRE is in the public-key setting, including the contributions presented in this thesis. Recall that in PKE, anyone with the public key can encrypt the ciphertext but only someone with the private key can decrypt it. For public-key PRE schemes, this rationale is retained as the current secret key is needed to create an update token, but only the public key of the target key pair is needed.

Definition III.2. *A* (public-key) Proxy Re-Encryption (PRE) *scheme consists of the following algorithms:*

- $\mathsf{Setup}(1^\lambda) \overset{(\$)}{\rightarrow}$ params*: Outputs a set of public parameters* params*, including the message space, $\mathcal{M}$, ciphertext space, $\mathcal{C}$, and token space, $\mathcal{T}$.*

  Note that params *is input to every subsequent algorithm, but we omit it for compactness* *of notation. We often omit the* Setup *algorithm from security games for the same reason.*

- $\mathsf{KeyGen}(1^\lambda) \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$*: A probabilistic algorithm that generates a public-private key pair.*
- $\mathsf{Enc}(\mathsf{pk}, m) \xrightarrow{\$} c$*: A probabilistic algorithm that encrypts a message, m, using a public key,* pk*, producing a ciphertext, c.*
- $\mathsf{Dec}(\mathsf{sk}, c) \rightarrow m' \cup \perp$*: A deterministic algorithm that decrypts a ciphertext c to produce either an element of the message space, m', or an error symbol, $\perp$.*
- $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) \xrightarrow{(\$)} \Delta_{i,j}$*: An algorithm that is either probabilistic or deterministic that takes a secret key,* sk_i*, and public key,* pk_j*, and outputs an* update token*, $\Delta_{i,j}$.*
- $\mathsf{ReEnc}(\Delta_{i,j}, c) \xrightarrow{(\$)} c'$*: An algorithm that is either probabilistic or deterministic that takes a ciphertext, c, and update token, $\Delta_{i,j}$, and outputs a re-encrypted ciphertext, c'.*

A PRE scheme is correct *if for all $m \in \mathcal{M}$, $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$:*

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) \rightarrow m$$

and if for all $c \in \mathcal{C}$ such that $\mathsf{Dec}(\mathsf{sk}_i, c) \rightarrow m'$:

$$\mathsf{Dec}(\mathsf{sk}_j, \mathsf{ReEnc}(\Delta_{i,j}, c)) \rightarrow m'$$

where $(\mathsf{pk}_i, \mathsf{sk}_i), (\mathsf{pk}_j, \mathsf{sk}_j) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$ and $\Delta_{i,j} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$.

PRE schemes sometimes have a *correctness bound*, L – the number of times a ciphertext can be re-encrypted and still decrypt correctly. We shall see this in more detail in Chapters V and VI which utilise PRE schemes based on lattices.

Definition III.3. *For a PRE scheme $\mathcal{PRE} = (\mathcal{PRE}.\mathsf{Setup}, \mathcal{PRE}.\mathsf{KeyGen}, \mathcal{PRE}.\mathsf{Enc}, \mathcal{PRE}.\mathsf{Dec}, \mathcal{PRE}.\mathsf{ReKeyGen}, \mathcal{PRE}.\mathsf{ReEnc})$, we define the* underlying encryption scheme *as the encryption scheme given by $(\mathcal{PRE}.\mathsf{Setup}, \mathcal{PRE}.\mathsf{KeyGen}, \mathcal{PRE}.\mathsf{Enc}, \mathcal{PRE}.\mathsf{Dec})$. This applies to both symmetric and public-key PRE schemes.*

Note that, in this thesis, we assume that PRE schemes are public-key unless explicitly stated otherwise.

III: 2.1 Additional PRE properties

There are some important properties that distinguish PRE schemes which are not considered security definitions but sometimes affect how security is defined. We now define these before we present any security definitions for PRE. Note that the following definitions apply to both symmetric and public-key PRE

Definition III.4. *A PRE scheme is* bidirectional *if an update token $\Delta_{i,j}$ can be used to re-encrypt ciphertexts under pk_j (k_j) to pk_i (k_i).*
If the scheme is not bidirectional, we say it is unidirectional.

This description of directionality is taken from existing work [AFGH05, CH07, LCLS09, EPRS17]. Note that bidirectional schemes must be symmetric as, by definition, the secret key is required to re-encrypt ciphertexts under that key to another key. However, not all symmetric schemes are bidirectional, as any public-key unidirectional PRE scheme can be easily converted into a symmetric one. Directionality is important in that it affects applications of the PRE scheme, and in some cases is used to restrict the actions of the adversary during security games, meaning that it plays an important part in which security models should be applied to a particular PRE scheme.

Definition III.5. *A PRE scheme $\mathcal{PRE}$ is* single-hop *if a fresh ciphertext can be re-encrypted at most once. $\mathcal{PRE}$ is* multi-hop *if it permits multiple re-encryptions.*

Whether a scheme is single or multi-hop affects the syntax of the definitions. In the literature, single-hop PRE schemes are often defined differently, so that ciphertexts are *levelled* and have different forms – fresh encryptions are level 0 whilst re-encrypted ciphertexts are level 1. Typically ReEnc in single-hop schemes changes the format of the ciphertext, and there are two decryption algorithms $\mathsf{Dec}_0, \mathsf{Dec}_1$, one for each level. We shall see this when we give examples of existing schemes in Section III: 4.1. In some definitions of a PRE scheme [AFGH05, FKKP19], levelling is considered explicitly by having encryption take an additional input, ℓ, at which to produce the ciphertext. As we focus on multi-hop PRE schemes in this thesis, we assume PRE schemes are multi-hop unless explicitly stated otherwise.

Definition III.6. *A PRE scheme has* transparency *if ciphertexts have the same format regardless of how many times they have been re-encrypted.*

Transparency is so-named because the involvement of the proxy is essentially hidden.

We note that number of hops, directionality and transparency are separate aspects that do not necessarily affect one another.

III: 3 Basic PRE security

Now that we have defined PRE schemes formally and described some notable properties, we can move on to formally defining the security of PRE schemes. We begin by explaining a necessary winning condition in all experiments for PRE security.

III: 3.1 Trivial wins and Re-encryption Graphs

Recall that in $\mathsf{IND\text{-}CCA}_{\mathcal{A}}^{\mathcal{PKE}}(1^\lambda)$ (Definition II.8) we had a winning condition that the adversary cannot use the decryption oracle $\mathsf{O}_{\mathsf{Dec}}$ on the challenge ciphertext c_b^*, and that this is enforced in $\mathsf{O}_{\mathsf{Dec}}$.

Security in PRE usually also requires a winning condition to prevent a trivial win. Most security games in PRE consider corrupted keys. To model the functionality of PRE, security games usually give the adversary access to oracles $\mathsf{O}_{\mathsf{ReKeyGen}}$ and $\mathsf{O}_{\mathsf{ReEnc}}$ that create update tokens and re-encrypt ciphertexts respectively. This means that even though the challenge ciphertext is created under one key, it can be re-encrypted to other keys. This leads to the necessary winning condition that the adversary cannot have corrupted a secret key under which a challenge exists. In other words, the key the challenge was originally created under cannot be corrupted, and neither can any key that the challenge ciphertext could (or already has) been re-encrypted to. We refer to this as the *trivial win condition.*

This is a necessary condition, as by the definition of a correct PRE scheme, the adversary will be able to decrypt the challenge ciphertexts under corrupted keys. As such, any meaningful security definition will hinge on the assumption that such corruptions have not happened. Clearly, the trivial win condition is minimal in that there is no weaker condition that can avoid such trivial wins.

A good visualisation technique for demonstrating which queries are permitted by the experiment is the use of *re-encryption graphs.* These are generally defined as follows:

- a vertex v_i is added for each keypair $(\mathsf{pk}_i, \mathsf{sk}_i)$ (or symmetric key k_i in the symmetric case) generated by the challenger,

- an edge $e_{i,j}$ is added if the adversary learns an update token $\Delta_{i,j}$.

The precise definition of the graph will be given in the experiments, as most experiments also add an edge when certain re-encryptions are made. In general, we work with unidirectional schemes meaning that edges are unidirectional, which we denote as $\overrightarrow{e}_{i,j}$. We generally refer to this graph as a *Directed Re-encryption Graph (DRG)*. In some formulations of PRE security definitions, the re-encryption graph is used in to enforce the trivial win condition. An example of how the trivial win condition is represented using a DRG is given in Figure III.2. We shall see this use of the re-encryption graph in Chapter VI.

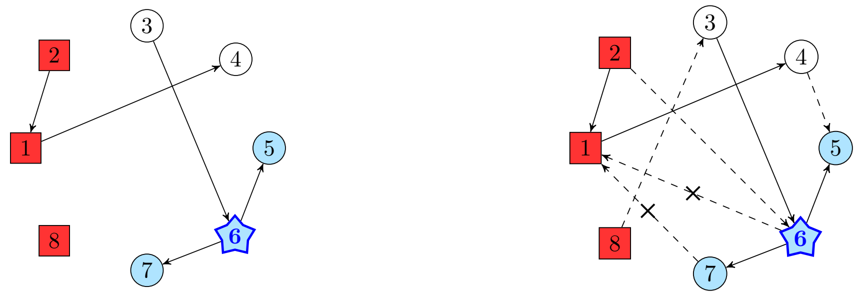

Figure III.2: An example $\mathcal{DRG}$. If a challenge is called under pk_6 ($\mathbf{v_6}$) then pk_6 is considered a **challenge key**. Other keys that are reachable from the initial challenge key (v_5,v_7) are therefore also considered challenge keys. Queries that create edges from challenge keys to corrupted (square) ones (v_1,v_2,v_8) will result in a trivial win. The dashed lines on the graph on the right demonstrate queries the adversary can and cannot make, assuming that queries resulting the graph on the left have already been made.

Re-encryption graphs can reflect applications. For example, key rotation, where keys are updated sequentially from pk_i to pk_{i+1} will result in a *chain* (depicted in Figure III.3), whereas some hierarchical access control policies will result in a *tree* (depicted in Figure III.4). Recall that there exist other re-encryption primitives that focus specifically on key rotation as an application [LT18, EPRS17]. This can be considered as restricting oracle queries in such a way that only chains are considered. Some other work, including [PRSV17, FKKP19], only considers re-encryption graphs that are *acyclic*. Typically, works where re-encryption graphs are restricted contain security proofs that apply only to acyclic graphs and cannot necessarily be extended to the general case. We explore this in detail in Chapter VI. As we do not rely on these techniques, in this thesis we do not assume that re-encryption graphs are acyclic.

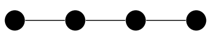
Figure III.3: A chain.

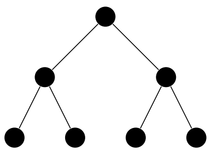
Figure III.4: A tree.

As a final note, our security games differ from common formulations, where the adversary does not explicitly pass any update tokens to the oracle and instead uses the appropriate indexes, and then the oracle generates the token. One of the goals of this work is to create definitions that are applicable for all re-encryption schemes, and thus we allow for the possibility that $\mathsf{ReKeyGen}$ is a probabilistic algorithm, meaning that $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$ has more than one possible output. We therefore allow the adversary to call $\mathsf{O}_{\mathsf{ReEnc}}$ by either giving an explicit token, or just indices for the appropriate keys (setting $\Delta_{i,j} = \perp$). This has the caveat that the token must have been honestly generated, and the challenger maintains a list $\mathcal{T}_{\mathsf{honest}}$ for this purpose. If the token is not honestly generated, then the oracle generates an honest update token and uses this instead. This caveat does not make our definitions weaker than others in the literature, as the common formulation is for $\mathsf{O}_{\mathsf{ReEnc}}$ to only take inputs (i, j) which the challenger uses to look up an honestly-generated update token. We also note that our syntax is still applicable for PRE schemes where $\mathsf{ReKeyGen}$ is deterministic, as in this case $\mathsf{O}_{\mathsf{ReKeyGen}}(i, j)$ will always return the same value.

III: 3.2 Chosen Plaintext Attacks

We begin by giving some definitions for message confidentiality for PRE schemes. Similarly to PKE, the basic definition of security for PRE is indistinguishability against chosen plaintext attacks. The intuition behind this definition is that, in addition to encryption masking the message, the re-encryption process should not reveal anything about the underlying message. This means confidentiality remains intact despite the proxy performing re-encryptions. As mentioned in Section III: 3.1, we need to model the functionality of re-encryption and therefore provide the adversary with more oracles than $\mathsf{IND\text{-}CPA}^{\mathcal{PKE}}_{\mathcal{A}}(1^{\lambda})$ (for PKE schemes), specifically $\mathsf{O}_{\mathsf{ReKeyGen}}$ and $\mathsf{O}_{\mathsf{ReEnc}}$.

Definition III.7. *Consider the following security game:*

```
$\mathsf{PRE\text{-}IND\text{-}CPA}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$
$\mathcal{K}_{\mathsf{corrupt}}, \mathcal{T}_{\mathsf{honest}} = \emptyset$
$\kappa := 0, \mathsf{called} := \mathsf{false}$
$b \xleftarrow{\$} \{0,1\}$
$state \xleftarrow{(\$)} \mathcal{A}_0^{\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Corrupt}}}(1^\lambda)$
$b' \xleftarrow{(\$)} \mathcal{A}_1^{\mathsf{O}_{\mathsf{ReKeyGen}}, \mathsf{O}^{\mathsf{CPA}}_{\mathsf{ReEnc}}, \mathsf{Chal}^{\mathsf{CPA}}_{\mathsf{Enc}}}(1^\lambda, state)$
return $(b' = b)$
```

```
$\mathsf{O}_{\mathsf{KeyGen}}(1^\lambda) \xrightarrow{\$} \mathsf{pk}_\kappa$
$\kappa = \kappa + 1$
$(\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
return $\mathsf{pk}_\kappa$
```

```
$\mathsf{O}_{\mathsf{Corrupt}}(i) \rightarrow \mathsf{sk}_i$
$\mathcal{K}_{\mathsf{corrupt}}.\mathsf{add}\{\mathsf{sk}_i\}$
return $\mathsf{sk}_i$
```

```
$\mathsf{O}_{\mathsf{ReKeyGen}}(i,j) \xrightarrow{(\$)} \Delta_{i,j}$
if $(\mathsf{sk}_i \notin \mathcal{K}_{\mathsf{corrupt}}) \wedge (\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}})$ :
  return $\perp$
$\boxed{\textbf{if } (\mathsf{sk}_j \notin \mathcal{K}_{\mathsf{corrupt}}) \wedge (\mathsf{sk}_i \in \mathcal{K}_{\mathsf{corrupt}}) :}$
  $\boxed{\textbf{return } \perp}$
$\Delta_{i,j} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$\mathcal{T}_{\mathsf{honest}}.\mathsf{add}\{i, j, \Delta_{i,j}\}$
return $\Delta_{i,j}$
```

```
$\mathsf{O}^{\mathsf{CPA}}_{\mathsf{ReEnc}}(i, j, [\Delta_{i,j}], c) \xrightarrow{(\$)} c'$
if $(\mathsf{sk}_i \notin \mathcal{K}_{\mathsf{corrupt}}) \wedge (\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}})$ :
  return $\perp$
$\boxed{\textbf{if } (\mathsf{sk}_j \notin \mathcal{K}_{\mathsf{corrupt}}) \wedge (\mathsf{sk}_i \in \mathcal{K}_{\mathsf{corrupt}}) :}$
  $\boxed{\textbf{return } \perp}$
if $((i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}})$ :
  $\Delta_{i,j} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$c' \xleftarrow{(\$)} \mathsf{ReEnc}(\Delta_{i,j}, c)$
return $c'$
```

```
$\mathsf{Chal}^{\mathsf{CPA}}_{\mathsf{Enc}}(i, m_0, m_1) \xrightarrow{\$} c^*_b$
if $(\mathsf{called} = \mathsf{true}) \vee (|m_0| \neq |m_1|)$ :
  return $\perp$
if $\mathsf{sk}_i \in \mathcal{K}_{\mathsf{corrupt}}$ : return $\perp$
$c^*_0 \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m_0)$
$c^*_1 \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m_1)$
$\mathsf{called} \leftarrow \mathsf{true}$
return $c^*_b$
```

where $\boxed{\textit{boxed conditions}}$ *apply only to bidirectional PRE schemes, and values in squared brackets [] are optional arguments.*

A Proxy Re-Encryption (PRE) scheme, $\mathcal{PRE}$, is said to be (selectively) ϵ-Indistinguishable against Chosen Plaintext Attacks (ϵ-PRE-IND-CPA-secure) *if for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*:*

$$\Pr\left[\mathsf{PRE\text{-}IND\text{-}CPA}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon.$$

If ϵ is negligible as determined by the security parameter λ then we say $\mathcal{PRE}$ is Indistinguishable against Chosen Plaintext Attacks (PRE-IND-CPA-secure)*.*

Here the trivial win condition is enforced by having $\mathsf{O}_{\mathsf{ReKeyGen}}$ and $\mathsf{O}_{\mathsf{ReEnc}}$ return $\perp$ if queried on (i, j) where sk_j is a corrupted key and sk_i is not. Here we have assumed that $\mathcal{PRE}$ is unidirectional. Note that in the literature this definition is referred to as being IND-CPA-secure, but we opt for PRE-IND-CPA-secure to make the distinction between security for PRE and PKE schemes. Clearly, the underlying encryption scheme must be IND-CPA-secure in order for the PRE scheme to be PRE-IND-CPA-secure.

Graphic representations of the queries permitted by the PRE-IND-CPA game are given in Figure III.5 and Figure III.6. Observe that key compromise for bidirectional schemes corresponds to proving that confidentiality is maintained when the set of corrupted keys shares no connection whatsoever with the challenge keys. This is clearly much weaker than unidirectional security.

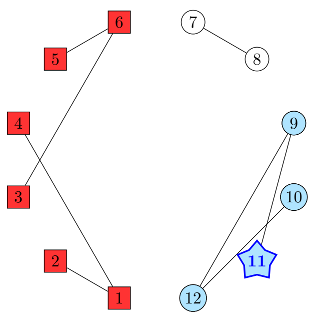

Figure III.5: For bidirectional PRE schemes, PRE-IND-CPA forbids all queries between uncorrupted and corrupted keys.

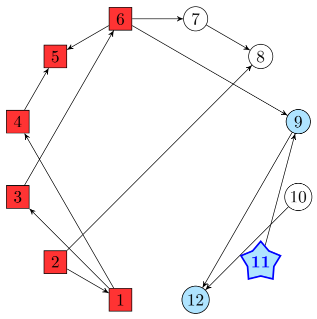

Figure III.6: For unidirectional PRE schemes, PRE-IND-CPA forbids queries from uncorrupted to corrupted keys but allows the reverse.

III: 3.3 Honest Re-Encryption Attacks

A development in PRE security introduced in 2017 was the notion of security against *honest re-encryption attacks* [Coh17]. Recall that the PRE-IND-CPA experiment $\mathsf{PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ forbids all re-encryption queries from uncorrupted to corrupted keys. Honest re-encryption attacks are more powerful in that they permit re-encryptions from uncorrupted to corrupted keys for non-challenge ciphertexts, as long as these ciphertexts were honestly (oracle) generated. For this reason, the adversary also receives an encryption oracle $\mathsf{O}_{\mathsf{Enc}}$, and the challenger maintains a list $\mathcal{C}_{\mathsf{honest}}$ of honest ciphertexts and a list $\mathcal{C}_{\mathsf{chal}}$ of challenge ciphertexts.

Definition III.8. *Consider the following security game (shaded parts highlight the differences from* $\mathsf{PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$*):*

```
$\mathsf{IND\text{-}HRA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$
$\mathcal{K}_{\mathsf{corrupt}}, \mathcal{T}_{\mathsf{honest}}, \mathcal{C}_{\mathsf{honest}}, \mathcal{C}_{\mathsf{chal}} = \emptyset$
$\kappa := 0, \mathsf{called} \leftarrow \mathsf{false}$
$b \xleftarrow{\$} \{0,1\}$
$state \xleftarrow{(\$)} \mathcal{A}_0^{\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Corrupt}}}(1^\lambda)$
$b' \xleftarrow{(\$)} \mathcal{A}_1^{\mathsf{O}_{\mathsf{Enc}}, \mathsf{O}_{\mathsf{ReKeyGen}}, \mathsf{O}_{\mathsf{ReEnc}}^{\mathsf{HRA}}, \mathsf{Chal}_{\mathsf{Enc}}^{\mathsf{HRA}}}(1^\lambda, state)$
**return** $(b' = b)$
```

```
$\mathsf{O}_{\mathsf{KeyGen}}(1^\lambda) \xrightarrow{\$} \mathsf{pk}$
$\kappa = \kappa + 1$
$(\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
**return** $\mathsf{pk}_\kappa$
```

```
$\mathsf{O}_{\mathsf{Corrupt}}(i) \rightarrow \mathsf{sk}_i$
$\mathcal{K}_{\mathsf{corrupt}}.\mathsf{add}\{\mathsf{sk}_i\}$
**return** $\mathsf{sk}_i$
```

```
$\mathsf{O}_{\mathsf{Enc}}(i, m) \xrightarrow{\$} c$
$c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m)$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(i, c)\}$
**return** $c$
```

```
$\mathsf{O}_{\mathsf{ReKeyGen}}(i, j) \xrightarrow{(\$)} \Delta_{i,j}$
**if** $(\mathsf{sk}_i \notin \mathcal{K}_{\mathsf{corrupt}}) \wedge (\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}})$ :
    **return** $\perp$
$\boxed{\textbf{if } (\mathsf{sk}_j \notin \mathcal{K}_{\mathsf{corrupt}}) \wedge (\mathsf{sk}_i \in \mathcal{K}_{\mathsf{corrupt}}) :}$
    $\boxed{\textbf{return } \perp}$
$\Delta_{i,j} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$\mathcal{T}_{\mathsf{honest}}.\mathsf{add}\{(i, j, \Delta_{i,j})\}$
**return** $\Delta_{i,j}$
```

```
$\mathsf{O}_{\mathsf{ReEnc}}^{\mathsf{HRA}}(i, j, [\Delta_{i,j}], c) \xrightarrow{(\$)} c'$
**if** $\big((i, c) \notin \mathcal{C}_{\mathsf{honest}}\big) \vee \big(((i, c) \in \mathcal{C}_{\mathsf{chal}}) \wedge (\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}})\big)$ :
    **return** $\perp$
**if** $\big((i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}\big)$ :
    $\Delta_{i,j} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$c' \xleftarrow{(\$)} \mathsf{ReEnc}(\Delta_{i,j}, c)$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(j, c')\}$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(j, c')\}$
**if** $(i, c) \in \mathcal{C}_{\mathsf{chal}}$ :
    $\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\{(j, c')\}$
**return** $c'$
```

```
$\mathsf{Chal}_{\mathsf{Enc}}^{\mathsf{HRA}}(i, m_0, m_1) \xrightarrow{\$} c_b^*$
**if** $(\mathsf{sk}_i \in \mathcal{K}_{\mathsf{corrupt}})$ : **return** $\perp$
**if** $(\mathsf{called} = \mathsf{true}) \vee (|m_0| \neq |m_1|)$ :
    **return** $\perp$
$c_0^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m_0)$
$c_1^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m_1)$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(i, c_b^*)\}$
$\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\{(i, c_b)\}$
$\mathsf{called} \leftarrow \mathsf{true}$
**return** $c_b^*$
```

where $\boxed{\textit{boxed conditions}}$ *apply to bidirectional schemes, and values in squared brackets* [] *are optional arguments.*

A Proxy Re-Encryption (PRE) scheme, $\mathcal{PRE}$*, is said to be* (selectively) ϵ-Indistinguishable against Honest Re-Encryption Attacks (ϵ-IND-HRA-secure) *if for all*

PPT *adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*:*

$$\Pr\left[\mathsf{IND\text{-}HRA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon.$$

If ϵ *is negligible as determined by the security parameter* λ *then we say* $\mathcal{PRE}$ *is* Indistinguishable against Honest Re-Encryption Attacks (IND-HRA-secure)*.*

Queries permitted in $\mathsf{IND\text{-}HRA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ are represented graphically in Figure III.7.

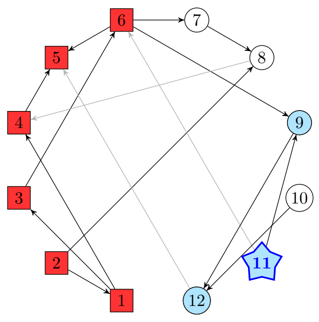

Figure III.7: Grey lines indicate re-encryptions of non-challenge ciphertexts that the adversary, $\mathcal{A}$, is permitted to make in $\mathsf{IND\text{-}HRA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$.

The definitions considered so far are in the *selective* key corruption model. In this setting, the adversary is in two stages ($\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$) where $\mathcal{A}_0$ corrupts keys, and $\mathcal{A}_1$ learns the challenge, update tokens and re-encryptions. *Adaptive* key corruption allows the adversary to corrupt keys even after the challenge has been learned, resulting in a single-stage adversary. Adaptive key corruption will be defined and explored explicitly in Chapter VI.

III: 4 Existing PRE Schemes

We now give some intuition behind constructions for PRE schemes, first by giving the original PRE scheme, then by describing a common mechanism for deploying PRE in practice.

III: 4.1 The BBS PRE scheme [BBS98]

We first note that while we have given a definition for public-key PRE in Definition III.2, the original scheme given in [BBS98] was formulated slightly differently as it is defined in the public-key setting but $\mathsf{ReKeyGen}$ takes both *secret* keys $\mathsf{sk}_i, \mathsf{sk}_j$ as input. The [BBS98] PRE scheme, BBS, is described in Figure III.8.

```
$\mathsf{Setup}(1^\lambda)$
    $p = 2q + 1$ where $p$ and $q$ are primes
    $g$ a generator of $\mathbb{Z}_p^*$
    return params $= (p, q, g)$
```

```
$\mathsf{KeyGen}(1^\lambda) \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$
    $a \xleftarrow{\$} \mathbb{Z}_p*$ must be coprime to $p - 1$
    $\mathsf{pk} = g^a \mod p$
    $\mathsf{sk} = a$
    return $(\mathsf{pk}, \mathsf{sk})$
```

```
$\mathsf{Enc}(\mathsf{pk}, m) \xrightarrow{\$} c$
    $k \xleftarrow{\$} \mathbb{Z}_{2q}^*$
    $c_0 = m \cdot g^k \mod p$
    $c_1 = (\mathsf{pk})^k \mod p$
    return $c = (c_0, c_1)$
```

```
$\mathsf{Dec}(\mathsf{sk}, c) \to m$
    $x = c_1^{\mathsf{sk}^{-1}} \mod p$
    $m' = c_0 \cdot x^{-1} \mod p$
    return $m'$
```

```
$\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_j) \to \Delta_{i,j}$
    $\Delta_{i,j} = (\mathsf{sk}_i^{-1}) \cdot \mathsf{sk}_j \mod p$
    return $\Delta_{i,j}$
```

```
$\mathsf{ReEnc}(\Delta_{i,j}, c) \to c'$
    $c_1' = c_1^{\Delta_{i,j}}$
    return $c' = (c_0, c_1')$
```

Figure III.8: The original PRE scheme, BBS [BBS98].

Let $\mathsf{sk}_i = a, \mathsf{sk}_j = b$. Then ciphertexts are of the form $c = (m \cdot g^k, g^{ak})$, and tokens are of the form $\Delta_{i,j} = b/a$. We therefore see that re-encrypted ciphertexts are of the form $c' = (m \cdot g^k, (g^{ak})^{b/a} = g^{bk})$, which is the same form as an encryption under pk_j, meaning BBS has transparency. This means correctness follows from correctness of the underlying encryption scheme:

$$\begin{aligned} m' &= c_0 \cdot (c_2^{a^{-1}})^{-1} \mod p \\ &= m \cdot g^k \cdot (g^{ak \cdot \frac{1}{a}})^{-1} \mod p \\ &= m \cdot g^k \cdot g^{-k} \mod p \\ &= m. \end{aligned}$$

Many PRE schemes are based on this original one. Unlike BBS however, they are defined as symmetric PRE schemes, despite relying on public-key-type primitives. In fact, most symmetric PRE schemes rely on public-key-type primitives. It is therefore important to note that, unlike regular encryption, it is not necessarily accurate to say that symmetric PRE schemes are more efficient than public-key ones. In fact, recent work [AMP19] suggests it is unavoidable for PRE to depend on public-key-like primitives. To some extent this is intuitive, as re-encryption relies on the underlying encryption scheme having some key-homomorphic properties which do not usually exist in symmetric encryption. The preference in choosing a symmetric PRE scheme over a public-key one is therefore more likely to depend on whether the re-encryption initiator in the application will know the target secret key.

III: 4.2 The Key Encapsulation Approach

A common approach to PRE is the *key encapsulation approach*, where the ciphertext header contains the (usually symmetric) data encryption key, k, encrypted using a key encryption key (which we assume to be public key), pk, from a PRE scheme. Typically, such schemes perform re-encryption by replacing pk — so the ciphertext header now contains the same k encrypted with the new key encryption key pk' and the body of the ciphertext remains unchanged. The effect of a re-encryption in the key encapsulation approach can be visualised as shown in Figure III.9. A full description is given in Figure III.10.

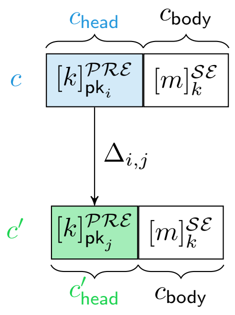

Figure III.9: A key change in the key encapsulation approach. $[a]^{\mathcal{PRE}}_{\mathsf{pk}}$ represents a PRE ciphertext that produces a when decrypted using the appropriate sk, and $[m]^{\mathcal{SE}}_k$ is a symmetric encryption of the message m using k.

The appeal of this approach is that it has the benefits of hybrid encryption. The key encapsulation approach is used widely, for example in Amazon's Key Management Service [Ama17]. Whilst these schemes are simple and easy to implement, they do not completely re-randomise a ciphertext during re-encryption. This creates the concern that key scraping attacks are possible, where any entity that learns k will be able to decrypt the ciphertext regardless of how many times it has been re-encrypted. Therefore, this approach is not suitable for key rotation, or where a malicious revoked user wants to retain access

$$
\boxed{
\begin{aligned}
&\mathcal{KEM}.\mathsf{KeyGen}(1^\lambda) : (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathcal{PRE}.\mathsf{KeyGen}(1^\lambda)\\
&\mathcal{KEM}.\mathsf{Enc}(\mathsf{pk}, m) : k \xleftarrow{\$} \mathcal{SE}.\mathsf{KeyGen}(1^\lambda),\\
&\qquad\qquad c_{\mathcal{PRE}} \xleftarrow{\$} \mathcal{PRE}.\mathsf{Enc}(\mathsf{pk}, k),\\
&\qquad\qquad c_{\mathcal{SE}} \xleftarrow{\$} \mathcal{SE}.\mathsf{Enc}(k, m).\\
&\qquad\qquad c := (c_{\mathcal{PRE}}, c_{\mathcal{SE}})\\
&\mathcal{KEM}.\mathsf{Dec}(\mathsf{sk}, c) : (c_{\mathcal{PRE}}, c_{\mathcal{SE}}) \leftarrow c\\
&\qquad\qquad k' \leftarrow \mathcal{PRE}.\mathsf{Dec}(\mathsf{sk}, c_{\mathcal{PRE}}),\\
&\qquad\qquad m' \leftarrow \mathcal{SE}.\mathsf{Dec}(k', c_{\mathcal{SE}})\\
&\mathcal{KEM}.\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) : \Delta_{i,j} \xleftarrow{(\$)} \mathcal{PRE}.\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)\\
&\mathcal{KEM}.\mathsf{ReEnc}(\Delta_{i,j}, c) : (c_{\mathcal{PRE}}, c_{\mathcal{SE}}) \leftarrow c,\\
&\qquad\qquad c'_{\mathcal{PRE}} \xleftarrow{(\$)} \mathcal{PRE}.\mathsf{ReEnc}(\Delta_{i,j}, c_{\mathcal{PRE}}),\\
&\qquad\qquad c' \leftarrow (c'_{\mathcal{PRE}}, c_{\mathcal{SE}})
\end{aligned}
}
$$

Figure III.10: A full description of the key encapsulation approach to PRE.

to the message. We note that a similar concern also applies to BBS, as the part of the ciphertext that contains the message does not change upon re-encryption.

III: 4.3 Other noteworthy PRE schemes

We now discuss some other existing PRE schemes which build on those described thus far, and give an overview of the properties met by existing constructions. Our goal here is to indicate how PRE schemes can be constructed so that they will have certain properties, as opposed to reviewing all related work. We instead discuss related work in detail in the most appropriate chapter for each topic. For now, we focus on a few prominent examples.

Unidirectionality has historically been a challenging property to attain. One approach to achieving unidirectionality is to use bilinear maps to create a single-hop scheme. For example, in [AFGH05] Ateniese at al. present a scheme based on ElGamal encryption, and achieve unidirectionality using a *bilinear pairing* $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. We give an overview here but refer to [AFGH05] for full details. The scheme has public parameters $\mathbb{G}_1, \mathbb{G}_2, e, g, Z = e(g, g)$ where g is a generator for $\mathbb{G}_1$ and Z is a generator for $\mathbb{G}_2$. For key pairs $(\mathsf{pk}_i, \mathsf{sk}_i) = (g^a, a)$, $(\mathsf{pk}_j, \mathsf{sk}_j) = (g^b, b)$:

- ciphertexts have the form $c = (\mathsf{pk}^k, m \cdot Z^k) = (g^{ak}, m \cdot Z^k)$

- update tokens have form $\Delta_{i,j} = \mathsf{pk}_j^{(\mathsf{sk}_i^{-1})} = g^{b/a}$
- re-encrypted ciphertexts have the form $c' = (e(c_0, \Delta_{i,j}), c_1) = (e(g^{ak}, g^{b/a}), m \cdot Z^k) = (Z^{bk}, m \cdot Z^k)$.

Decryption of fresh ciphertexts is the same as for BBS. Re-encrypted ciphertexts cannot be decrypted in the same way as fresh ciphertexts (at level 0), as components of the ciphertexts are now in $\mathbb{G}_2$ as opposed to $\mathbb{G}_1$. This means a second decryption function is necessary (as discussed in Section III: 2.1).

Ateniese at al. take advantage of this and also define a second encryption function that outputs ciphertexts of the same format of re-encrypted (level 1) ciphertexts by computing $c = (e(g^{ak}, g), m \cdot Z^k) = (Z^{ak}, m \cdot Z^k)$. These ciphertexts cannot be re-encrypted, and thus gives the encryptor a means of ensuring that PRE cannot be used to the underlying message m with further parties. A similar approach is taken in other unidirectional, single-hop PRE schemes [LV08b, ABH09].

Another approach to creating a PRE scheme that is both unidirectional and single-hop is using *obfuscation* [HRsV07, CCV12]. The downside of this approach is that known methods of achieving obfuscation are computationally intensive. Whilst schemes that rely on computationally intensive methods are theoretically sound, they incur a heavy overhead in terms of the computation needed to perform a re-encryption (from both the client and the proxy). This means the naive approach of downloading, re-encrypting locally and re-uploading may be preferable and, therefore, PRE schemes that rely on these measures are hard to justify over the trivial solution.

In 2007, Green and Ateniese created a unidirectional, multi-hop PRE scheme [GA07]. However, their construction works by appending new elements to the ciphertext upon re-encryption, meaning that decryption takes longer the more times a ciphertext has been re-encrypted. It also means that, similarly to the key encapsulation approach discussed in Section III: 4.2, the scheme is unsuitable for key rotation and access revocation as the part of the ciphertext containing the message does not change.

This justifies the desire for PRE schemes with transparency. Whilst it is hard to achieve this for schemes whose security relies on Diffie-Hellman, advancements in lattice-based cryptography have enabled the first constructions of multi-hop, unidirectional PRE schemes with transparency. The first generic description of a PRE scheme based on lattices was presented in Gentry's work [Gen09] which uses Fully Homomorphic Encryption (FHE). Since then, a number of concrete lattice-based PRE schemes have been proposed including [Kir14, NAL15, FL16, PRSV17]. We describe an explicit construction of a lattice-based PRE scheme in Chapter V.

In Table III.1 we give a summary of notable existing work. Note that this is a simplified overview of the contributions given by existing work that highlights the factors most notable for this thesis, and should not be taken as a complete guide to the contributions of each paper. The PRE schemes that we directly build on and compare in the body of our work are described in the appropriate chapters.

In Table III.1, key corruption adaptivity describes whether keys are generated as being corrupted (fixed), keys can be corrupted but only before challenges are issued (selective), or keys can be corrupted at any point (adaptive). Graph adaptivity refers to whether the pattern of re-encryption queries the adversary can make is predetermined. These are discussed fully in Chapter VI.

We also acknowledge work on defining security against Chosen Ciphertext Attacks (CCA) in PRE. This was first introduced in [CH07] for bidirectional, single-hop schemes. A definition of IND-CCA security for unidirectional schemes is given in [LV08b]. However since the focus of this chapter is to create definitions with particular applications in mind, for simplicity we restrict security to IND-CPA and IND-HRA-security. We do not consider this a significant weakening of security in comparison with existing practical schemes, as recent schemes which are both unidirectional and multi-hop such as [PRSV17, FKKP19] also do not consider IND-CCA security.

Paper	**Public-Key/ Symmetric**	**CPA/HRA[a]/ CCA**	**Assumed directionality**	**Hops**	**Assumption**	**Key corruption adaptivity**	**Graph adaptivity**
[BBS98]	Symmetric	CPA	Bidirectional	Multi	DDH	None	Fixed but maximal
[AFGH05]	Public-key	CPA	Unidirectional	Single	eDBDH	Fixed	Fixed but maximal
[LV08b]	Public-key	CCA	Unidirectional	Single	DBDH	Fixed	Fixed but maximal
[ABH09]	Public-key	CPA	Unidirectional	Single	EDBDH	Fixed	Adaptive
[FL16]	Public-key	CPA, CCA	Unidirectional	Single, Multi	LWE	Fixed	Adaptive
[PWA$^+$16]	Public-key	CPA,CCA	Unidirectional[b]	Single, Multi	LWE	Fixed	Adaptive
[Coh19]	Public-key	HRA	Unidirectional	Single	eDBDH	Fixed	Adaptive
[EPRS17]	Symmetric	CPA	Unidirectional	Multi	Non-specific[c]	Fixed/Selective	Adaptive
[PRSV17]	Public-key	CPA	Unidirectional	Multi	RLWE	Fixed	Adaptive[d]
[FKKP19]	Public-Key	CPA, HRA	Unidirectional	Multi	EDBDH, LWE	Selective, Adaptive	Fixed
[LT18]	Symmetric	HRA	None	Multi	DDH	Fixed/Selective	Adaptive[e]

Table III.1: A summary of adaptive behaviour in existing work.
When graph adaptivity is 'fixed but maximal', this means the adversary is given an update token between every pair of keys that does not break the trivial win condition.

[a]Indistinguishable against Honest Re-Encryption Attacks (Definition III.8)
[b]directionality not actually specified, but bidirectional schemes do not meet the specified security definition.
[c]Their construction, ReCrypt is a general construction built using a Key-Homomorphic Pseudo-Random Function, and so has no specific construction.
[d]Whilst their definition allows the adversary to adaptively choose re-encryption queries, their proof assumes the graph is fixed at the start of the game.
[e]Adaptive but graphs must be chains.

III: 5 Thesis motivations revisited

Recall that the original motivation given in [BBS98] was for email forwarding, where Alice wants to share her emails with Bob. Another obvious application of PRE is to enforce access control. For example, [ABH09] gives an overview of using PRE as a means of access control for distributed file systems. In this application, keys are associated with access rights and files are encrypted under keys according to an access control policy and the classification of the file. In practice, access control is a dynamic process where access rights change over time, such as when an employee gains higher security clearance, or when an employee leaves the company. For the latter, we can consider any keys in the former employee's possession to be corrupted, and it is good practice to re-encrypt any files under the corrupted key to fresh keys. In this way, former employees do not retain access to files. Access revocation can be considered similarly to key rotation.

Many PRE schemes in the literature, including [BBS98, ID03, AFGH05, LV08b], do not (fully) re-randomise a ciphertext upon re-encryption. In particular, often the component containing the message does not change. In fact, [AFGH05] calls this *original access* and postulates that there are applications of PRE where Alice still wants to be able to decrypt ciphertexts even after she has had them re-encrypted to Bob's key. Recall from the key encapsulation example that this means such schemes are unsuitable for applications involving access revocation or key expiry.

In this thesis, we re-examine PRE with regards to its suitability as a mechanism for enforcing cryptographic access control. We therefore are particularly interested in investigating unidirectionality and key corruption to create PRE schemes suitable for revocation as well as sharing. We also note that, as a third party, the proxy is untrusted. Most of the focus on the literature considers the proxy an adversary in the sense that message confidentiality should be maintained even when the proxy performs the re-encryption. Recall that searchable encryption, verifiable computation and order-preserving encryption all consider adversaries who are interested in more than just breaking confidentiality. Security beyond confidentiality is rarely considered in PRE. A particular concern presented by access control is that no unauthorised re-encryptions should be performed. This necessitates considering a more active malicious proxy who may try to create update tokens not provided by clients, to share files without authorisation. Whilst unidirectionality covers the idea that tokens from pk_i to pk_j cannot re-encrypt ciphertexts under pk_j to pk_i, we investigate malicious re-encryptions in a more general setting, separate to confidentiality. This involves investigating not only the corruption of keys but also the corruption of update tokens in greater detail.

Note that public-key PRE schemes can trivially be made into symmetric ones. We therefore focus on building public-key schemes, as these will cover a wider number of applications. In particular, as symmetric schemes require both secret keys to be known in order to create an update token, the re-encryption initiator must know both secret keys, which is not necessarily the case, depending on the application. Suppose that an employee wishes to share a confidential file with their employer. Because the employer is more senior, we can assume that they have access to more sensitive files, and so their secret key is not known to all their employees. Where access control is defined by such hierarchical structures, public-key schemes may be preferable. Recall that, as existing symmetric PRE schemes rely on public-key type primitives, symmetric PRE does not have the same efficiency benefits of regular symmetric encryption. Symmetric PRE also does not have the key management benefits of public-key cryptography. We therefore aim to create PRE solutions in the public-key setting.

We also note that access control as a main motivation necessitates that appropriate PRE schemes must be multi-hop, as access permissions may change more than once. There are other practical considerations to bear in mind. As PRE solves a problem that can be solved naively by re-encrypting locally, efficiency be considered in the construction of a PRE scheme. This means that some means of achieving strong security guarantees will render the resulting scheme too inefficient for realistic usage. For example, one could instantiate PRE using garbled circuits, where the update token is a garbled circuit that first decrypts under the old key and then encrypts under the new key. As garbled circuits are meant to mask any interim values calculated during evaluation, this will be secure. However, the act of creating the circuit without knowing the input ciphertext (otherwise re-encryption could be performed locally) is highly inefficient, especially in comparison to other approaches to PRE.

A summary of our motivations is as follows:

- To investigate modelling malicious actions that could be carried out by the proxy other than breaking confidentiality.
- To investigate key compromise more fully, especially the compromise of update tokens.
- To provide public-key constructions that are practical in the sense that:
  - *They are both unidirectional and multi-hop.*

    This means they can be used for key rotation and dynamic access control.
  - *Ciphertexts do not expand upon re-encryption.*

    This limit the resources needed to store and decrypt re-encrypted ciphertexts, making it a more attractive alternative to downloading and re-encrypting locally.
  - *They have transparency.*

    This means the resulting schemes can be used in applications where the act of re-encryption should be hidden.
  - *As much as possible, they do not rely on heavy parameter choices or heavy computations to achieve security.*

    We investigate beyond what is theoretically possible into constructions that could have reasonably efficient implementations.

Chapter IV

Controlling Re-Encryptions

Contents

In this Chapter

Recall from Section III: 5 that one application of PRE is as a mechanism for enforcing cryptographic access control over remotely stored files, where files are encrypted under specific keys according to an access control policy and keys are shared with users in accordance with their access rights. Re-encryption can signify a change in user access rights, particularly revocation or key expiry. In this chapter, we investigate security definitions for PRE with a particular focus on enforcing access control as an application. Our particular focus is on definitions that imply the proxy cannot perform a re-encryption unless that re-encryption was initiated by the client.

The contributions of this chapter were published at Latincrypt 2017 [Lee17a]. A full version of the published work is published on the Cryptology ePrint Archive [Lee17b].

IV: 1 Introduction

PRE schemes have clear applications to cryptographic access control by allowing outsourced data to be selectively shared to users via re-encryption to appropriate keys. For instance, if data is given a classification level and keys are shared with users according to an access control policy, then re-encryption can be used to enforce changes to the policy [AFGH05, PRSV17]. It is important to ensure that only parties with the current secret key can re-encrypt ciphertexts as, in this application, other 'unauthorised' re-encryptions signify files being shared without authorisation of the data owner(s). In other words, having a mechanism for ensuring that re-encryptions are authorised is desirable.

Most existing work on re-encryption is limited to the *honest-but-curious* model, where the adversary participates in the protocol honestly but tries to break message confidentiality. By extension, this means much existing work focuses only on the confidentiality of messages in re-encryption schemes, and not authorised re-encryption. An existing work of note which inspires our work is that of Everspaugh et al. on '*Key Rotation for Authenticated Encryption*' [EPRS17]. This goes beyond the honest-but-curious setting and extends symmetric authenticated encryption to the re-encryption setting ('key rotation' in their language) and introduces the concept of *ciphertext dependence*, where $\mathsf{ReKeyGen}$ takes an additional input from the ciphertext to be re-encrypted, and ReEnc can only correctly re-encrypt that ciphertext. We apply similar concepts to PRE and extend their work to creating stronger definitions that imply that re-encryptions not initiated by a party with the current secret key are not possible, subject to realistic assumptions. This models a malicious cloud service provider aiming to subvert the re-encryption process to leak sensitive data.

We break unauthorised re-encryptions down into two main security notions: the inability to re-encrypt a ciphertext to a key it has not been encrypted under before, and a stronger notion of unidirectionality which considers reversal attacks where adversaries may try to reverse a re-encryption by retaining information about prior ciphertexts and update tokens. We also consider an adaptation of Data Origin Authentication (DOA) (where the encryptor of a message can be verified), that allows parties to formally verify identity of the last party to re-encrypt a ciphertext. This is useful not only in preventing unauthorised re-encryptions, but in auditing changes to access control policy.

IV: 1.1 Contributions

The summary of contributions presented in this chapter is as follows.

Token robustness. We build on ciphertext dependence to define *token robustness* – a property that implies it is impossible for an adversary without an appropriate secret key to re-encrypt a ciphertext to a key it has not been encrypted under before. Our model assumes the adversary sees a number of legitimate update tokens, modelling a malicious proxy who honestly performs re-encryptions requested by the client, then attempts to perform a new re-encryption. We give a separating example which shows that token robustness is stronger than ciphertext dependence.

Formalised unidirectionality. The ability to re-encrypt back to the old key can grant access back to an unauthorised user or re-encrypt to an expired key. To model an adversary who attempts to re-encrypt a ciphertext to a key it has previously been encrypted under, we revisit the existing notion of unidirectionality. Recall that in a unidirectional PRE scheme, a re-encryption token can only be used to transform a ciphertext under pk_i to pk_j and not from pk_j to pk_i, otherwise the scheme is bidirectional. This existing, informal notion of unidirectionality does not consider reversal attacks where a server may retain some limited information about an old ciphertext and update token to reverse the re-encryption. This consideration is particularly important for ciphertext dependent and token robust schemes where update tokens can only re-encrypt specific ciphertexts, meaning that reversing that re-encryption should depend on that specific update token, as opposed to one generic, ciphertext-independent update token $\Delta_{i,j}$ permitting the re-encryption of any ciphertext under pk_j to pk_i.

Clearly, all re-encryption operations are trivially reversible if the adversary stores the entirety of a previous ciphertext. Whilst existing notions of unidirectionality implicitly consider the adversary who retains the update token, it is more realistic to assume that a server willing to retain the update token would be willing to retain any state of at least the same size. We therefore formally define reversal attacks with respect to the size of the state the server must retain in order to reverse a re-encryption. This allows us to define *maximal irreversibility* and *best-achievable unidirectionality* for token robust schemes. Using our stronger and more realistic model, we see that there exist schemes which claim to be unidirectional but are bidirectional in our more realistic model, and vice versa.

Ciphertext Origin Authentication. Finally, we revisit the notion of DOA for PRE. *DOA* is a cryptographic utility where the identity of the encryptor can be inferred from the ciphertext, thereby providing the recipient a means of verifying who sent the message. Typically, DOA assumes that the same party who created the message also encrypted it, hence tying the data creator's identity to the message underlying the ciphertext is sufficient, and this is the approach generally taken in constructions. In PRE, however, the re-encryption functionality means that the party who initialised re-encryption is not necessarily the same party that created the message. We thus introduce the notion of *Ciphertext Origin Authentication (COA)* to determine which party encrypted the message (or initiated the most recent re-encryption) and show how to fulfil this requirement in practice. We also discuss mandatory COA where the identity of the re-encryptor must be verified in order for the message to be recoverable.

Overall, our security model covers a wider range of attacks than prior definitions. We show that best-achievable unidirectionality can be met by introducing a simple adaptation of BBS, and discuss extensions to COA both for our construction and for ReCrypt [EPRS17]. In keeping with the motivations discussed in Section III: 5, we focus on constructions that fully re-randomise a ciphertext so as to avoid similar problems to the key encapsulation approach.

Chapter structure. In Section IV: 2 we formally discuss existing work. We define *token robustness* in Section IV: 3. In Section IV: 4 we give a formal definition for the existing intuition of unidirectionality, and present *maximal irreversibility* – the first security definition that considers reversal attacks. In Section IV: 5 we use maximal irreversibility together with token robustness to create a stronger definition for *best-achievable unidirectionality*, and present a scheme which meets this definition. Finally, in Section IV: 6 we define Ciphertext Origin Authentication as a means of providing authenticated PRE and discuss how to achieve this, presenting explicit extensions to our scheme and to ReCrypt.

IV: 2 Related work

In this section, we discuss existing definitions of directionality in re-encryption primitives (Section IV: 2.1), give a detailed discussion on ciphertext-dependence (Section IV: 2.2), and discuss some related work that implies all ciphertext components change during re-encryption (Section IV: 2.3). Finally, we discuss the existing concept of DOA and how it can be achieved (Section IV: 2.4). Note that much existing work is written in the context of symmetric PRE schemes, but uses underlying mathematics associated with public-key schemes. We therefore note that whilst some of the work we cite was written for a slightly different model, they still create expectations in what a definition for public-key PRE should achieve, and provide guidelines in how to create an appropriate definition.

IV: 2.1 Directionality

Directionality is usually given as an informal definition. The manner in which unidirectionality is described is not consistent in the literature. For example, in [EPRS17] a scheme is described as being unidirectional if there is no algorithm $\mathsf{Invert}()$ which, given $\Delta_{i,j}$, can output $\Delta_{j,i}$, whereas unidirectionality in [LT18] is described as there being no algorithm that can find sk_j given sk_i and $\Delta_{i,j}$. These definitions are both given for symmetric re-encryption schemes such as updatable encryption (see Appendix A). It is not immediately clear whether these two definitions are equivalent, or whether they are well-suited to public-key PRE schemes.

One attempt to formalise the definition of directionality in public-key PRE is given by Ivan and Dodis in [ID03], but the authors do not view directionality as a security definition, rather a classification of PRE schemes. They therefore give one definition for unidirectional PRE schemes and another for bidirectional PRE schemes, as opposed to defining directionality separately from PRE. Furthermore, the definition of unidirectionality in [ID03] assumes that a unidirectional scheme is single-hop, which is not necessarily the case. A more recent work which informally describes unidirectionality is based on the idea that no PPT algorithm can output a token that can re-encrypt to the old key [EPRS17], whereas other works such as [PRSV17] are based on the idea that no PPT algorithm can output an equivalent encryption of the old ciphertext, which is to say, any encryption of the message under the old key.

Recall that before lattice-based PRE schemes, creating unidirectional PRE schemes was difficult, and single-hop schemes emerged as a means of achieving unidirectionality [LV08b, SC09, CWYD10]. These schemes achieve unidirectionality by having two distinct levels of ciphertext which have different formats — level 0 for ciphertexts which can be re-encrypted, and level 1 for ciphertexts which cannot be re-encrypted. It is this format change which prevents a ciphertext from being re-encrypted more than once. However, this approach does not convey how easy it is for a malicious server to reverse the re-encryption process. It also does not allow for multiple re-encryptions, meaning it is unsuitable for access revocation and key rotation. As we discussed in Section III: 4, in many multi-hop schemes the size of the ciphertext grows linearly with each re-encryption, meaning the schemes do not have transparency and thus we have similar problems in defining what is meant by unidirectionality. The related problem of multi-hop unidirectional proxy re-signatures is addressed in [LV08a], however their construction works by providing the message along with the signature and thus this technique cannot be easily adapted to re-encryption.

Existing PRE schemes which are both unidirectional and multi-hop such as [PRSV17] do not address ciphertext-dependence. Current methods for achieving this cannot easily be applied to their scheme. Key scraping attacks (see Section III: 4.2) in the context of subscriber revocation are discussed in [MS17], where they consider an adversary who receives portions of a ciphertext as long as the total size is sufficiently less than the total size of the ciphertext. Whilst their definition of revocation does not explicitly model an adversary seeking to reverse a re-encryption, their rationale of limiting the amount of information the adversary can access with respect to old states is in keeping with ours – that it may be more realistic to assume that the old keys can be retained than old ciphertexts due to storage constraints.

There does not appear to be existing work on either symmetric or public-key PRE that explicitly models a malicious server seeking to perform unauthorised re-encryptions.

IV: 2.2 Ciphertext-Dependence

To our knowledge, the only existing work that addresses limiting re-encryptions that the proxy can perform is [EPRS17]. The particular contribution of interest is *ciphertext-*

dependence, where update tokens can only be used to re-encrypt specific ciphertexts:

Definition IV.1. *Let* $\mathsf{ReKeyGen}$ *in a PRE scheme be redefined to take additional information* $\tilde{c}$ *about ciphertext* c *as input:* $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j, \tilde{c}) \to \Delta^{c}_{i,j}$*. The PRE scheme is* ciphertext-dependent *if for all* $c_1 \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m_1)$ *and* $c_2 \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m_2)$ *such that* $c_1 \neq c_2$*, and all re-encryption tokens* $\Delta^{c_1}_{i,j,} \xleftarrow{\$} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j, \tilde{c}_1)$*, then:*

$$\Pr\Big[\mathsf{Dec}(\mathsf{sk}_j, \mathsf{ReEnc}(\Delta^{c_1}_{i,j}, c_2)) = m_2\Big] \leq \mathsf{negl}(\lambda) \tag{IV.1}$$

for some negligible function $\mathsf{negl}(\lambda)$[1]*.*
If $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$ *does not take* $\tilde{c}$ *as input and update tokens can re-encrypt a ciphertext under the source key, then the scheme is* ciphertext-independent

In existing work [EPRS17] and in our scheme, $\tilde{c}$ denotes the header of the ciphertext. For simplicity we will assume this is the case for ciphertext-dependent schemes in the remainder of this chapter. We note that not all applications require ciphertext-dependence. For example in key expiry, all ciphertexts under the old key need to be re-encrypted, and this is used as motivation in creating a ciphertext-independent updatable encryption scheme in [LT18]. To reflect this, most of the security games in this chapter have variants for both ciphertext-dependent and ciphertext-independent PRE schemes.

To give an idea of how ciphertext-dependent schemes can be constructed, we describe $\mathsf{ReCrypt}$, the construction given in [EPRS17]. Before we do this, we must first define Pseudo-Random Functions (PRFs) and Key-Homomorphic Pseudo-Random Functions (KH-PRFs).

Definition IV.2. *A* Pseudo-Random Function (PRF) *is a function* $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ *that is indistinguishable from a function that outputs genuine random values in* $\mathcal{Y}$*.*

Definition IV.3. *A* Key-Homomorphic Pseudo-Random Function (KH-PRF) *is a PRF* $\mathcal{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *such that*

$$\mathcal{F}(k_1, x) + \mathcal{F}(k_2, x) = \mathcal{F}(k_1 + k_2, x)$$

where $(\mathcal{K}, +)$*,* $(\mathcal{Y}, +)$ *are groups.*

[1]The part including the probability of being able to re-encrypt was not included formally in [EPRS17], but we add it here to formalise the understanding of ciphertext-dependence as presented in [EPRS17].

```
$\mathsf{Setup}(1^\lambda) \rightarrow \text{params}$
a symmetric encryption scheme $\mathcal{SE}$
a KH-PRF $\mathcal{F} : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$
a hash function $h$
$\text{params} = (\mathcal{SE}, \mathcal{F}, h)$
**return** params
```

```
$\mathsf{KeyGen}(1^\lambda) \xrightarrow{\$} k$
$k \xleftarrow{\$} \mathcal{SE}.\mathsf{KeyGen}(1^\lambda)$
```

```
$\mathsf{Enc}(k, m) \xrightarrow{\$} c$
$x, y \xleftarrow{\$} \mathcal{K}$
$\chi = x + y$
$\tau = h(m) + \mathcal{F}(x, 0)$
$\tilde{c} = \mathcal{SE}.\mathsf{Enc}(k, (\chi, \tau))$
$m_1 || m_2 || \ldots || m_n \leftarrow m$
$\bar{c}_0 = y$
    $\bar{c}_i = m_i + \mathcal{F}(x, i)$
$\bar{c} \leftarrow \bar{c}_0 || \bar{c}_1 || \ldots || \bar{c}_n$
**return** $c = (\tilde{c}, \bar{c})$
```

```
$\mathsf{Dec}(k, c) \rightarrow m$
$(\tilde{c}, \bar{c}) \leftarrow c$
$(\chi, \tau) \leftarrow \mathcal{SE}.\mathsf{Dec}(k, \tilde{c})$
**if** $(\chi, \tau) = \perp$ : **return** $\perp$
$y = \bar{c}_0$
**for** $1 \leq i \leq n$ :
    $m_i = \bar{c}_i - \mathcal{F}(\chi - y, i)$
**if** $h(m) + \mathcal{F}(\chi - y, 0) \neq \tau$ :
    **return** $\perp$
**else return** $m$
```

```
$\mathsf{ReKeyGen}(k_i, k_j, \tilde{c}) \xrightarrow{\$} \Delta^c_{i,j}$
$(\chi, \tau) \leftarrow \mathcal{SE}.\mathsf{Dec}(k, \tilde{c}_0)$
**if** $(\chi, \tau) = \perp$ : **return** $\perp$
$x', y' \xleftarrow{\$} \mathcal{K}$
$\chi' = \chi + x' + y'$
$\tau' = \tau + \mathcal{F}(x', 0)$
$\tilde{c}' = \mathcal{SE}.\mathsf{Enc}(k_j, (\chi', \tau'))$
$\Delta^c_{i,j} = (\tilde{c}', x', y')$
**return** $\Delta^c_{i,j}$
```

```
$\mathsf{ReEnc}(\Delta^c_{i,j}, c) \rightarrow c'$
$(\tilde{c}', x', y') \leftarrow \Delta^c_{i,j}$
$y = \bar{c}_0$
$\bar{c}'_0 = y' + y$
**for** $1 \leq i \leq n$ :
    $\bar{c}'_i = \bar{c}_i + \mathcal{F}(x', i)$
$c = (\tilde{c}', \bar{c}')$
**return** $c$
```

Figure IV.1: ReCrypt [EPRS17], a ciphertext-dependent key rotation scheme built using a symmetric encryption scheme $(\mathcal{SE}.\mathsf{KeyGen}, \mathcal{SE}.\mathsf{Enc}, \mathcal{SE}.\mathsf{Dec})$ and KH-PRF $\mathcal{F}$.

The authenticated key rotation scheme ReCrypt of [EPRS17] is given in Figure IV.1. Because ReCrypt is a ciphertext-dependent scheme, ReCrypt.ReKeyGen also takes a ciphertext header $\tilde{c}$ as input. ReCrypt utilises a symmetric encryption scheme $\mathcal{SE} = (\mathcal{SE}.\mathsf{KeyGen}, \mathcal{SE}.\mathsf{Enc}, \mathcal{SE}.\mathsf{Dec})$ to encrypt the ciphertext header, and a KH-PRF $\mathcal{F}$ to encrypt the body of the ciphertext.

ReCrypt uses an approach similar to the key encapsulation approach, but where both the header and the body of the ciphertext are re-randomised during re-encryption. The update token given by ReCrypt.ReKeyGen includes the new ciphertext header, together with the values needed to update the body of the ciphertext accordingly. Essentially, re-encryption adds new randomness x', y' to the existing randomness x, y used to encrypt the message, and uses the key-homomorphic properties of $\mathcal{F}$ to perform this update without removing or giving away the original randomness.

Ciphertexts are of the form

$$\big(\tilde{c} = \mathcal{SE}.\mathsf{Enc}(k, (\chi, \tau)),\ \bar{c} = (y, \{m_i + \mathcal{F}(x, i)\}_{i=1}^{n})\big),$$

where $\chi = x + y$ and $\tau = h(m) + \mathcal{F}(x, 0)$. Re-encrypted ciphertexts will be of the form

$$\big(\tilde{c}' = \mathcal{SE}.\mathsf{Enc}(k, (\chi', \tau')),\ \bar{c}' = (y' + y, \{m_i + \mathcal{F}(x + x', i)\}_{i=1}^{n})\big),$$

where $\chi' = x + x' + y + y'$ and $\tau' = h(m) + \mathcal{F}(x + x', 0)$. Since fresh ciphertexts and re-encrypted ciphertexts have the same form (in other words, ReCrypt has transparency), it is easy to see that the correctness of re-encryption follows from the correctness of encryption. ReCrypt is an authenticated scheme in the sense that the tag τ can be used to verify that the received message is the same one that was sent.

Recall that in order for a re-encryption scheme to be practically useful, it must offer some efficiency gain over downloading and re-encrypting locally. The need to have the re-encryption initiator download (or retain) the ciphertext header is justified by arguing that the construction of ReCrypt allows for much larger messages to be re-encrypted. The overall efficiency justification in [EPRS17] is through having a smaller update token than the ciphertext, putting the bulk of the necessary processing on the server during the re-encryption operation. However, we note that current implementations of ReCrypt do not offer significant efficiency gains over alternative updatable encryption schemes, as existing PRFs rely on comparatively expensive operations such as modular exponentiation. See [LT18, Table 2] for further details.

Observations. ReCrypt is unidirectional in the sense that, in general, outputs of the token generation function ReCrypt.ReKeyGen$(k_i, k_j, \tilde{c})$ cannot be used to re-encrypt arbitrary re-encrypted ciphertexts under k_j ($c' \leftarrow$ ReCrypt.ReEnc$(\Delta_{i,j}^{c}, c)$) back to encryptions of the same message under k_i. This is because new update tokens will not contain the same randomness as the token $\Delta_{i,j}^{c}$ used in the update, and since re-encryption consists of reversible arithmetic operations, other tokens that do not contain the same randomness will not be able to reverse the transformation. However, since no new randomness is added during ReCrypt.ReEnc, if the adversary obtains the update token $\Delta_{i,j}^{c}$ that was used to re-encrypt c to c', they can trivially revert $\bar{c}'$ back to $\bar{c}$. The adversary cannot recover $\tilde{c}$ from the update token, as the update token contains a replacement header as opposed to a value to update the header with. The security games given in [EPRS17] capture the adversary learning update tokens resulting from calls to $\mathsf{ReKeyGen}(k_i, k_j, \tilde{c})$, but not the specific update token used in creating the challenge re-encryption. Therefore the idea of unidirectionality conveyed in [EPRS17] is conditioned on the adversary not learning the specific update token used. These observations fuel our motivation for defining unidirectionality and reversal attacks more formally.

IV: 2.3 Full re-randomisation

It is often not considered a requirement that re-encryption updates all components of the ciphertext. However, full re-randomisation must be considered a necessary security property for applications such as access revocation and key expiry. Recall from the key encapsulation approach described in Chapter II that if parts of the ciphertext remain static, the PRE scheme may not be appropriate for key rotation or access revocation.

One existing definition that implies all components of a ciphertext are updated upon re-encryption is UP-REENC security in [EPRS17], defined for key rotation schemes. Informally, UP-REENC security implies that given a re-encrypted ciphertext, an adversary should be unable to tell which of two potential ciphertexts has been re-encrypted. It models an adversary that can learn update tokens, re-encryptions and corrupted keys, subject to the trivial win condition. We use UP-REENC as a starting point for implying full re-randomisation in this chapter, but note that we discuss this area in much greater detail in Section V: 2.1 when we explore Post-Compromise Security (PCS). For easier relation to the public-key setting, in Definition IV.4 we describe a public-key variant of UP-REENC.

Definition IV.4. *Consider the following security game:*

```
$\mathsf{ReEncIND\text{-}CPA}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$
---
$\mathcal{K}_{\mathsf{corrupt}}, \mathcal{T}_{\mathsf{honest}} = \emptyset$
$\kappa := 0$
$b \xleftarrow{\$} \{0,1\}$
$state \xleftarrow{(\$)} \mathcal{A}_0^{\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Corrupt}}}(1^\lambda)$
$b' \xleftarrow{(\$)} \mathcal{A}_1^{\mathsf{O}_{\mathsf{ReKeyGen}}, \mathsf{O}_{\mathsf{ReEnc}}, \mathsf{O}_{\mathsf{Challenge}}}(1^\lambda, state)$
return $(b' = b)$
```

```
$\mathsf{O}_{\mathsf{KeyGen}}(1^\lambda) \xrightarrow{\$} \mathsf{pk}_\kappa$
---
$\kappa = \kappa + 1$
$(\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
return $\mathsf{pk}_\kappa$
```

```
$\mathsf{O}_{\mathsf{Corrupt}}(i) \rightarrow \mathsf{sk}_i$
---
$\mathcal{K}_{\mathsf{corrupt}}.\mathsf{add}\{\mathsf{sk}_i\}$
return $\mathsf{sk}_i$
```

```
$\mathsf{O}_{\mathsf{ReKeyGen}}(i,j) \xrightarrow{(\$)} \Delta_{i,j}$
----------------------------------------
if $(\mathsf{sk}_i \notin \mathcal{K}_{\mathsf{corrupt}}) \wedge (\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}})$ :
  return $\perp$
$\boxed{\text{if } (\mathsf{sk}_j \notin \mathcal{K}_{\mathsf{corrupt}}) \wedge (\mathsf{sk}_i \in \mathcal{K}_{\mathsf{corrupt}}) :}$
  $\boxed{\text{return } \perp}$
$\Delta_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$\mathcal{T}_{\mathsf{honest}}.\mathsf{add}\{i, j, \Delta_{i,j}\}$
return $\Delta_{i,j}$
```

```
$\mathsf{O}_{\mathsf{ReEnc}}(i,j,[\Delta_{i,j}],c) \xrightarrow{(\$)} c'$
----------------------------------------
if $(\mathsf{sk}_i \notin \mathcal{K}_{\mathsf{corrupt}}) \wedge (\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}})$ :
  return $\perp$
$\boxed{\text{if } (\mathsf{sk}_j \notin \mathcal{K}_{\mathsf{corrupt}}) \wedge (\mathsf{sk}_i \in \mathcal{K}_{\mathsf{corrupt}}) :}$
  $\boxed{\text{return } \perp}$
if $((i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}})$ :
  $\Delta_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$c' \overset{(\$)}{\leftarrow} \mathsf{ReEnc}(\Delta_{i,j}, c)$
return $c'$
```

```
$\mathsf{Chal}_{\mathsf{ReEnc}}(i,j,c_0,c_1) \xrightarrow{\$} c_b$
----------------------------------------
if $(|c_0| \neq |c_1|)$ :
  return $\perp$
if $(\mathsf{sk}_i \notin \mathcal{K}_{\mathsf{corrupt}}) \wedge (\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}})$
  return $\perp$
$\Delta_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$c_0'^* \overset{(\$)}{\leftarrow} \mathsf{ReEnc}(\Delta_{i,j}, c_0)$
$c_1'^* \overset{(\$)}{\leftarrow} \mathsf{ReEnc}(\Delta_{i,j}, c_1)$
$\mathsf{called} \leftarrow \mathsf{true}$
return $c_b'^*$
```

where $\boxed{\textit{boxed conditions}}$ *apply only to bidirectional PRE schemes. A PRE scheme* $\mathcal{PRE}$ *is* ϵ-Re-Encryption Indistinguishable against Chosen Plaintext Attacks (ϵ-ReEnc-IND-CPA-secure) *if for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*:*

$$\Pr\left[\mathsf{ReEncIND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(\lambda) = 1\right] \leq \frac{1}{2} + \epsilon. \tag{IV.2}$$

If ϵ *is negligible as determined by the security parameter* λ *then we say* $\mathcal{PRE}$ *is* Re-Encryption Indistinguishable against Chosen Plaintext Attacks (ReEnc-IND-CPA-secure).

The key difference between PRE-IND-CPA-security and ReEnc-IND-CPA-security is that in the former the adversary is comparing potential messages with a ciphertext, whereas the latter compares potential ciphertexts with a re-encrypted ciphertext. This definition can easily be extended to symmetric PRE by providing the adversary with encryption oracles for both keys, see [EPRS17].

In ReCrypt (Figure IV.1), re-randomisation is achieved by having randomness added in the creation of the update token, and then using this token to update every block of the ciphertext. PRE schemes that fully re-randomise the ciphertext include [HRsV07], which uses obfuscation to re-encrypt ciphertexts. The obfuscated function first decrypts then encrypts without revealing either the message or any secret keys during this process. New randomness is added during the ReEnc operation as opposed to during $\mathsf{ReKeyGen}$. In contrast, [ABH09] adds new randomness to the ciphertext both in $\mathsf{ReKeyGen}$ and in ReEnc. Because existing work that re-randomises can do so in either $\mathsf{ReKeyGen}$ or ReEnc, we consider the possibility that either $\mathsf{ReKeyGen}$ or ReEnc (or both) are randomised, but do not make assumptions that either algorithm is randomised.

IV: 2.4 Data Origin Authentication

Finally, we describe a simplified approach to DOA, which we build on when we define COA in Section IV: 6. DOA enables parties to verify the source of a message – this includes both verifying the identity of the sender and the integrity of the message. One approach to this in the symmetric setting is using a Message Authentication Code (MAC), a keyed function that creates a tag on a message that can be used to verify that the message has not been changed, and that it must have originated from a party in possession of the appropriate secret key. One approach to authenticated symmetric encryption is to compute a MAC on the message to create a tag τ, then encrypt (m, τ). The decryptor can then use τ to verify the message[2]. This is essentially the approach used in $\mathsf{ReCrypt}$ [EPRS17] (Figure IV.1) to provide authenticated re-encryption.

Alternatively, digital signatures (Definition II.3) may provide DOA in settings where the same party who created the message also encrypted it, as is the case in a number of scenarios. After creating the message, the encryptor signs the hash of the message using their unique signature key, $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{ssk}, h(m))$ before encrypting (m, σ). After retrieving (m', σ'), the verifier uses the verification key svk to verify whether σ' is a signature on m' using the encryptor's key. As only the encryptor should know their signing key ssk, only they would have been able to compute σ', meaning that they must have been the one to encrypt the ciphertext. This only works in where we can assume the same party who

[2]This approach only works as long as the verifier is trusted. However, we consider this to be a reasonable assumption in this case, as the act of re-encrypting to the receiver's key already implies they are considered to be trusted.

created the message also encrypted it.

IV: 3 Token Robustness

In this section we define what it means for a malicious proxy to be unable to re-encrypt a ciphertext to a key it has never been encrypted under before. We discuss the other possibility of unauthorised re-encryptions, namely re-encrypting ciphertexts back to keys they have previously been encrypted under in Section IV: 4.

Recall that in a ciphertext-dependent PRE scheme, $\mathsf{ReKeyGen}$ takes an additional input $\tilde{c}$ from the ciphertext to be re-encrypted, with the intention that the resulting update token can only re-encrypt that ciphertext. Ciphertext-dependence does not eliminate the possibility that the adversary can create an update token that allows it to correctly re-encrypt a ciphertext to a key it has never been under before. Ciphertext-dependence is mainly concerned with what can be re-encrypted using the update token, but does not consider whether the token reveals information that might make other re-encryptions possible. For example, if the token contains the ciphertext header, and then during re-encryption the proxy verifies that this matches the ciphertext, then other valid update tokens can easily be created by swapping out the header in the update token. This scheme would be ciphertext-dependent, but it would be easy for a malicious proxy to forge an update token that results in unauthorised re-encryptions. We therefore define *token robustness* – a stronger notion than ciphertext-dependence.

Informally, token robustness states that even with access to a token generation oracle, an adversary cannot create a new valid token which re-encrypts a ciphertext to a key it was never previously encrypted under. This extends ciphertext-dependence by explicitly modelling an adversary who sees a number of legitimate update tokens and attempts to perform an unauthorised re-encryption. This is designed to compliment unidirectionality, which examines re-encrypting to a key which the ciphertext has previously been under.

Definition IV.5. *We say that a PRE scheme $\mathcal{PRE}$ has* (κ, ϵ)-token robustness *if for all* PPT *adversaries $\mathcal{A}$:*

$$\Pr\left[\mathsf{TokRob}^{\mathcal{PRE}}_{\mathcal{A},\kappa}(\lambda) = 1\right] \leq \epsilon, \tag{IV.3}$$

where TokRob *is given in Figure IV.2. If κ is polynomial and ϵ is negligible with respect to*

```
$\mathsf{TokRob}^{\mathcal{PRE}}_{\mathcal{A},\kappa}(\lambda)$
$\text{params} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}_1, \mathsf{sk}_1), \dots, (\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$\mathcal{C}_{\mathsf{honest}}, \mathcal{T}_{\mathsf{honest}} := \emptyset$
for all $m \in \mathcal{M} : \mathsf{chain}[m] := \emptyset$
$(c, i, j, \Delta^{\mathcal{A}}) \overset{(\$)}{\leftarrow} \mathcal{A}^{\mathsf{O_{Enc}}, \mathsf{O_{ReKeyGen}}}(1^\lambda, \mathsf{pk}_1, \dots, \mathsf{pk}_\kappa)$
if $(i, c) \notin \mathcal{C}_{\mathsf{honest}}$ : return $\perp$
$c' \leftarrow \mathsf{ReEnc}(\Delta^{\mathcal{A}}, c)$
if $(\mathsf{Dec}(\mathsf{sk}_i, c) = \mathsf{Dec}(\mathsf{sk}_j, c')) \wedge (j \notin \mathsf{chain}[m])$ :
   return 1
else : return 0
```

```
$\mathsf{O_{Enc}}(i, m) \rightarrow c$
$c \overset{\$}{\leftarrow} \mathsf{Enc}(\mathsf{pk}_i, m)$
$\mathsf{chain}[m].\mathsf{add}\{i\}$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{i, c\}$
return $c$
```

```
$\mathsf{O_{ReKeyGen}}(i, j \boxed{, c}) \rightarrow \Delta^c_{i,j}$
$\Delta^{\boxed{c}}_{i,j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j \boxed{, \tilde{c}})$
$\mathsf{chain}[\mathsf{Dec}(\mathsf{sk}_i, c)].\mathsf{add}\{j\}$
$\mathcal{T}_{\mathsf{honest}}.\mathsf{add}\{i, j \boxed{, c}, \Delta^{\boxed{c}}_{i,j}\}$
return $\Delta^{\boxed{c}}_{i,j}$
```

```
$\mathsf{O_{ReEnc}}(i, j, \Delta^c_{i,j}, c) \rightarrow c'$
if $\Delta^{\boxed{c}}_{i,j} = \perp$ :
   $\Delta^{\boxed{c}}_{i,j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j \boxed{, \tilde{c}})$
else if $(i, j, \Delta^{\boxed{c}}_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}$ :
   return $\perp$
$c' \leftarrow \mathsf{ReEnc}(\Delta^{\boxed{c}}_{i,j}, c)$
if $(i, c) \in \mathcal{C}_{\mathsf{honest}}$
   $\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{j, c'\}$
$\mathsf{chain}[\mathsf{Dec}(\mathsf{sk}_i, c)].\mathsf{add}\{j\}$
return $c'$
```

Figure IV.2: The token robustness game TokRob for assessing whether a ciphertext-dependent PRE scheme allows an adversary to create unauthorised update tokens.

the security parameter λ then we say that $\mathcal{PRE}$ has token robustness.

In the TokRob game, the adversary attempts to output a token $\Delta^{\mathcal{A}}$ which re-encrypts a target ciphertext c from being under pk_i to being under pk_j, where the underlying message has never been encrypted under pk_j. It has access to an encryption oracle $\mathsf{O_{Enc}}$, a token generation oracle $\mathsf{O_{ReKeyGen}}$ and a re-encryption oracle $\mathsf{O_{ReEnc}}$. The list $\mathsf{chain}[m]$ records keys m has been encrypted (or can be re-encrypted) under by appending the appropriate keys whenever $\mathsf{O_{Enc}}$ or $\mathsf{O_{ReKeyGen}}$ are called. The adversary cannot win by submitting a token that re-encrypts the key to another key in the chain. Note that we cannot define token robustness using an indistinguishability game as it is not modelling a passive scenario where the adversary's goal is to infer information, but an active one where the adversary is trying to create a value it should not be able to, similarly to creating a forged signature.

This game aims to model a malicious server trying to re-encrypt ciphertexts provided by the client, and therefore another condition is that the adversary's target ciphertext C must be an output of the $\mathsf{O}_{\mathsf{Enc}}$ oracle (or a re-encryption of such a ciphertext), and the challenger maintains a list $\mathcal{C}_{\mathsf{honest}}$ to keep track of these ciphertexts. This ensures the adversary gains no additional advantage from storing information created when encrypting the ciphertext. For example, in an ElGamal-based scheme, the encryption algorithm samples a random y and sets $\tilde{c} = g^y$. If the adversary encrypts the message for themselves then they learn y, which the server would not know in the cloud storage application. The list $\mathcal{C}_{\mathsf{honest}}$ is used to keep track of oracle-generated ciphertexts.

Theorem IV.1. *No ciphertext-independent PRE scheme has token robustness.*

Proof. If a PRE scheme is not ciphertext-dependent, then the same update token can be used to re-encrypt more than one ciphertext. In the TokRob game, let $c_1 \leftarrow \mathsf{O}_{\mathsf{Enc}}(i, m_1), c_2 \leftarrow \mathsf{O}_{\mathsf{Enc}}(i, m_2)$. Then the adversary can submit (i, j, c_1) to $\mathsf{O}_{\mathsf{ReKeyGen}}$ to obtain $\Delta_{i,j}^{c_1}$ and then submit $(c_2, i, j, \Delta_{i,j})$ (in other words, set $c^{\mathcal{A}} = c_2, \Delta^{\mathcal{A}} = \Delta_{i,j}$). Since $j \notin \mathsf{chain}[m_2]$, the adversary wins the game with probability 1. $\square$

Ciphertext-dependence alone does not imply token robustness. Recall our earlier example of a ciphertext-dependent PRE scheme which had the intended ciphertext header as part of the token, and during re-encryption verified that this matches the input ciphertext. Then an adversary can trivially craft a valid token for a different ciphertext by replacing the new ciphertext header in the update token with the header of a different ciphertext under the same key. Since we have by Theorem IV.1 that a token robust scheme must be ciphertext-dependent, we see that token robustness is a stronger notion than ciphertext-dependence. Token robustness requires more than a check that the given token is meant for a specific ciphertext by comparing values – it requires some mathematical computation where a correct re-encryption is only possible using the appropriate update token.

IV: 4 Directionality Revisited

Whilst token robustness covers the risk of ciphertexts being re-encrypted to keys they have never previously been under, the problem of re-encrypting to a key the ciphertext was previously encrypted under relates to unidirectionality. Recall from Definition III.4 that the existing idea of unidirectionality states that an update token $\Delta_{i,j}$ cannot be used to re-encrypt a ciphertext under pk_j to pk_i. We argue that this notion is not sufficient for access control as it is lacks formality and it is unclear whether it covers the possibility that re-encryptions from pk_i to pk_j can be reversed. This informal notion also does not couple well with ciphertext-dependence or token robustness. In this section we will elaborate on this claim before offering a more fine-grained security definition, which we call $\overline{\lambda}$-*irreversible*.

IV: 4.1 Problems with traditional directionality

Unidirectionality is required in a number of applications for security reasons. However, it has not yet been defined as a security property. This is problematic, as it is not immediately obvious whether different authors' informal descriptions of unidirectionality amount to the same thing. Formal security allows us a precise understanding of what kinds of re-encryptions are prevented. A particular issue is that, if a single-hop PRE scheme prevents $\Delta_{j,i}$ being learned from $\Delta_{i,j}$ (which is one possible interpretation of unidirectionality), this does not convey whether it is possible for $\Delta_{i,j}$ to reverse a re-encryption, which may be possible if ReEnc is deterministic.

We therefore formulate a security game Uni in Figure IV.4, which we believe formalises the informal understanding of unidirectionality in the literature. In Uni, the adversary receives two public keys $\mathsf{pk}_i, \mathsf{pk}_j$ and aims to take a challenge ciphertext c_j which is an encryption of an unknown message, and produce a ciphertext c_i such that $\mathsf{Dec}(\mathsf{sk}_i, c_i) = \mathsf{Dec}(\mathsf{sk}_j, c_j)$. The adversary receives access to a token oracle $\mathsf{O}_{\mathsf{ReKeyGen}}$ that returns tokens from pk_i to pk_j[3]. The adversary can choose the number of times N that a challenge ciphertext can have been re-encrypted (its level), in which case the challenger generates the intermediate keys, and returns the public keys to the adversary together with the ciphertext. This accommodates both single and multi-hop PRE schemes, where $N \in \{0, 1\}$ for single-hop schemes.

Definition IV.6. *A PRE scheme $\mathcal{PRE}$ is* unidirectional *if for all* PPT *algorithms $\mathcal{A}$ there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\Pr\left[\mathsf{Uni}_{\mathcal{A}}^{\mathcal{PRE}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,. \tag{IV.4}$$

A PRE scheme $\mathcal{PRE}$ is bidirectional *if there exists a* PPT *algorithm $\mathcal{A}$ that wins the*

[3]In a ciphertext-dependent scheme, this oracle takes a ciphertext c as input, otherwise it takes no input. Tokens generated by the challenge oracle also take the appropriate ciphertexts as input.

```
$\mathsf{Uni}^{\mathcal{PRE}}_{\mathcal{A},\overline{\lambda}}(\lambda)$
    $\text{params} \leftarrow \mathsf{Setup}(\lambda)$
    $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$
    $\mathcal{C}_{\mathsf{chal}} = \emptyset$
    $(c_0, c_1) \overset{(\$)}{\leftarrow} \mathcal{A}^{\mathsf{O}_{\mathsf{ReKeyGen}}, \mathsf{Chal}_{\mathsf{Enc}}}(1^\lambda, \mathsf{pk}_0, \mathsf{pk}_0)$
    **if** $(c_1) \notin \mathcal{C}_{\mathsf{chal}}$ : **return** $\perp$
    **if** $(\mathsf{Dec}(\mathsf{sk}_i, c_i) = \mathsf{Dec}(\mathsf{sk}_j, c_j))$ :
        **return** $1$
    **else** : **return** $0$

$\mathsf{O}_{\mathsf{ReKeyGen}}(\boxed{c}) \overset{(\$)}{\to} \Delta^{c}_{0,1}$
    $\Delta^{\boxed{c}}_{0,1} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_0, \mathsf{pk}_1 \boxed{, \tilde{c}})$
    **return** $\Delta^{\boxed{c}}_{0,1}$

$\mathsf{Chal}_{\mathsf{Enc}}(N) \overset{(\$)}{\to} c$
    $m \overset{\$}{\leftarrow} \mathcal{M}$
    **if** $(N = 0)$ :
        $c_N \leftarrow \mathsf{Enc}(\mathsf{pk}_1, m)$
    **else** :
        $(\overline{\mathsf{pk}}_0, \overline{\mathsf{sk}}_0), \dots, (\overline{\mathsf{pk}}_{N-1}, \overline{\mathsf{sk}}_{N-1}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$
        $\overline{\mathsf{pk}}_N = \mathsf{pk}_1$
        $c_0 \leftarrow \mathsf{Enc}(\overline{\mathsf{pk}}_0, m)$
        **for** $\ell \in \{1, \dots, N\}$ :
            $\Delta^{\boxed{c_{\ell-1}}}_{\ell-1,\ell} \leftarrow \mathsf{ReKeyGen}(\overline{\mathsf{sk}}_{\ell-1}, \overline{\mathsf{pk}}_\ell \boxed{, \tilde{c}_{\ell-1}})$
            $c_\ell \leftarrow \mathsf{ReEnc}(\Delta^{\boxed{c_{\ell-1}}}_{\ell-1,\ell}, c_{\ell-1})$
    $\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\{c_N\}$
    **return** $(c_N, \overline{\mathsf{pk}}_0, \dots, \overline{\mathsf{pk}}_{N-1})$
```

Figure IV.3: The unidirectionality game Uni – a formal security game capturing the traditional notion of unidirectionality. $\boxed{\text{Boxed}}$ values show the variant for a ciphertext-dependent scheme.

unidirectionality game Uni *with non-negligible probability.*

This can easily be extended to symmetric PRE schemes by making the obvious adjustments. We believe that this definition formally captures the intuition behind unidirectionality as it is currently understood in the literature. We hope that for applications where unidirectionality is a security requirement, this definition can be used to assess the suitability of potential PRE schemes.

IV: 4.2 Directionality reconsidered

Whilst reversal attacks are not of concern for email forwarding, the original motivation for PRE, it is an important consideration for access control and key expiry as reversing a re-encryption can mean regranting access to a revoked user. Uni does not cover reversal attacks, as challenge ciphertexts have never been encrypted under pk_i. A server or revoked user aiming to perform a reversal attack would at some point know the old ciphertext and potentially the information used to perform the re-encryption. As a malicious server aiming to perform an unauthorised re-encryption could attempt such an attack, reversal attacks warrant further consideration.

Before we proceed, we must address the obvious point. An assumption often made, particularly in the access control literature, is that once a message (or ciphertext) is known, it is known forever. However, this assumption does not always make sense in cloud storage applications when considering revoked users. One of the most common reasons for outsourcing storage is limited local space. Therefore, a user's ability to retrieve a file at one point does not necessarily mean they have the capacity to retain all files they have ever had access to. Revoked users may try to decrypt a file that was previously encrypted under a key known to them. In many existing schemes, if a revoked user is able to retain limited information about the old ciphertext, they have an advantage in creating a re-encryption under the old key. For example, in the key encapsulation method, as the data encryption key is static, any entity who knows this key can create a re-encryption of the message under any public key. Since the data encryption key is typically much shorter than the message, it is unnecessary for such adversaries to retain full messages if they can instead retain keys, This could make a significant difference in how realistic such an attack is for constrained devices. Given increasing trends in having external storage as the default storage option (see Section I: 1), we believe definitions addressing revocation that consider bounded storage are worthy of consideration.

We also note that existing notions of unidirectionality do not fit well with ciphertext-dependence. For ciphertext-dependent schemes, reversal attacks become more significant than traditional unidirectionality since tokens are specific to each ciphertext and thus the act of maliciously retaining the token implies a willingness to store something for each ciphertext. If it is assumed that the server retains the update token, we should also consider other information which they might retain.

Now that we have argued that current notions of unidirectionality are not suitable for access control, we present a security definition for reversal attacks. We explain the key principles behind this definition.

- **Principle 1:** *Malicious storage cannot be prevented.* It is impossible for one party to prevent another from storing extra information without additional assumptions. In particular we cannot prevent the server from retaining the old ciphertext.

- **Principle 2:** *The amount of storage required to reverse a process has a lower bound.* Whilst we cannot prevent a malicious server from retaining an old ciphertext, we can ensure there is no 'easier' way for them to obtain the old ciphertext. By 'easier', we mean that the server needs significantly less storage than it would if retaining the components of the original ciphertext that change during re-encryption. This is similar to the motivation behind an *economically rational server* considered in [vJO+12] – the server would not double its storage in order to circumvent the scheme.

There are schemes such as [ABH09] where storage of the update token alone cannot reverse a particular re-encryption, but retaining elements of the update token and elements of the original ciphertext can reverse a re-encryption. For example, if the update token contains a replacement ciphertext header, then reversal can be trivial by retaining the original ciphertext header. If storing some component of the original ciphertext makes a reversal attack successful, then we should assume that the adversary will do so, especially if the amount of storage needed is at most the size of the update token. For practical reasons, header values and update tokens are often designed to be small (as in [EPRS17]), and thus can easily be retained. Current models of unidirectionality permit an adversary to retain the update token in order to attempt to re-encrypt. We argue that there is no reason to restrict the information an adversary may store to just update tokens, especially if the adversary can retain other information and have a greater chance of success.

We now define a reversal attack game which takes into account the amount of information an adversarial server may have retained during the re-encryption process using a storage parameter $\overline{\lambda}$. The following game has an adversary in three stages $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$. All three adversaries receive the security parameter λ, storage parameter $\overline{\lambda}$, public keys and system parameters as input.

Stage 1: $\mathcal{A}_0$ receives a randomly chosen message m and decides which keys $\mathsf{pk}_i, \mathsf{pk}_j$ should be used for encryption and re-encryption.

Stage 2: $\mathcal{A}_1$ receives the ciphertext c and update token $\Delta_{i,j}$ and determines what should be retained in the state $st_{\mathcal{A}}$, which is bounded by a storage parameter $\overline{\lambda}$. Note this adversary never receives the message. Since this adversary knows the storage

```
$\underline{\mathsf{Rev\text{-}ReEnc}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda, 1^{\overline{\lambda}}, \kappa)}$
$\text{params} \leftarrow \mathsf{Setup}(\lambda)$
$(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$m \xleftarrow{\$} \mathcal{M}$
$(i, j) \xleftarrow{(\$)} \mathcal{A}_0(1^\lambda, 1^{\overline{\lambda}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_\kappa, m)$
$c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m)$
$\Delta^{\boxed{c}}_{i,j} \xleftarrow{\$} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j \boxed{, \tilde{c}})$
$st_{\mathcal{A}} \xleftarrow{(\$)} \mathcal{A}_1(1^\lambda, 1^{\overline{\lambda}}, \mathsf{pk}_1, \ldots \mathsf{pk}_\kappa, \Delta^{\boxed{c}}_{i,j}, C)$
$\textbf{if } |st_{\mathcal{A}}| > \overline{\lambda} : \textbf{return } \perp$
$c' \leftarrow \mathsf{ReEnc}(\Delta^{\boxed{c}}_{i,j}, c)$
$c_{\mathcal{A}} \xleftarrow{(\$)} \mathcal{A}_2(1^\lambda, 1^{\overline{\lambda}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_\kappa, st_{\mathcal{A}}, c')$
$\textbf{if } \mathsf{Dec}(\mathsf{sk}_i, c_{\mathcal{A}}) = \mathsf{Dec}(\mathsf{sk}_j, c') :$
  $\textbf{return } 1$
$\textbf{else } : \textbf{return } 0$
```

Figure IV.4: The reversal game Rev-ReEnc. The $\mathcal{A}_0$ maintains a state bounded by $\overline{\lambda}$ which is passed onto $\mathcal{A}_1$ who attempts to reverse the re-encryption. $\boxed{\text{Boxed}}$ values show the variant for a ciphertext-dependent scheme.

bound, it can compute many potential states before selecting which one will be passed on to $\mathcal{A}_2$.

Stage 3: $\mathcal{A}_2$ receives the state $st_{\mathcal{A}}$ forwarded by $\mathcal{A}_1$ and the re-encrypted ciphertext c', and uses this to try to output a ciphertext $c_{\mathcal{A}}$ which is an encryption of the same message under the original key. This adversary never receives the message, original ciphertext or the update token — they only receive the information retained by $\mathcal{A}_1$.

Note that the ciphertext $c_{\mathcal{A}}$ output by $\mathcal{A}_2$ does not need to be the original ciphertext — it can be any encryption of m under pk_i. This emulates the scenario where the server must decide how much information to retain about the old ciphertext and update token before later attempting to reverse the re-encryption (or revert to an equivalent ciphertext – another encryption of the same message under the same key).

Definition IV.7. *Given a PRE scheme $\mathcal{PRE}$, let $|\Delta|$ denote the size of tokens given by $\mathcal{PRE}$ with security parameter λ, let $|\tilde{c}|$ denote the size of the ciphertext headers, let $|\bar{c}|$ denote the size of the ciphertext body, and let s be the total size of ciphertext components updated by* ReEnc. *Then for $\overline{\lambda} \in \{0, |\Delta|, |\tilde{c}|, |\bar{c}|, s\}$, we define the* advantage *of an adversary*

$\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ *of winning the* Rev-ReEnc *game given in Figure IV.4 as:*

$$\mathbf{Adv}^{\mathcal{A},\overline{\lambda},\kappa}_{\mathsf{Rev\text{-}ReEnc}}(1^{\lambda})(1^{\lambda}) = \left| \Pr\left[\mathsf{Rev\text{-}ReEnc}^{\mathcal{PRE}}_{\mathcal{A}}(1^{\lambda}, 1^{\overline{\lambda}}, \kappa) = 1\right] - \frac{1}{2^{s-\overline{\lambda}}} \right|, \tag{IV.5}$$

where κ is polynomial in terms of λ and $\frac{1}{2^{s-\overline{\lambda}}}$ is the probability that an adversary who has retained $\overline{\lambda}$ bits of c can correctly guess the remaining bits.

We say that a proxy re-encryption scheme $\mathcal{PRE}$ is $\overline{\lambda}$-irreversible *if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, the advantage of winning the game is negligible:*

$$\mathbf{Adv}^{\mathcal{A},\overline{\lambda},\kappa}_{\mathsf{Rev\text{-}ReEnc}}(1^{\lambda})(1^{\lambda}) \leq \mathsf{negl}(\lambda)\,. \tag{IV.6}$$

Conversely, a PRE scheme is $\overline{\lambda}$-reversible *if there exists a* PPT *algorithm $\mathcal{A}$ that can win the* $\mathsf{Rev\text{-}ReEnc}^{\mathcal{PRE}}_{\mathcal{A},\overline{\lambda}}(\lambda)$ *game with state $st_{\mathcal{A}}$ of size at most $\overline{\lambda}$, with non-negligible probability.*

Remark. If a PRE scheme is $\overline{\lambda}$-irreversible, then it is not $\overline{\lambda}'$-irreversible for $\overline{\lambda}' < \overline{\lambda}$. Similarly, if a scheme is $\overline{\lambda}$-reversible, then it is also $\overline{\lambda}'$-reversible for $\overline{\lambda}' > \overline{\lambda}$.

We restrict ourselves to logical choices of $\overline{\lambda}$. We note that typically $|\Delta| = |\tilde{c}|$, and that in PRE schemes which do not rely on a hybrid key encapsulation mechanism, components of the ciphertext header and body are also usually the same size.

We briefly note that constructing this notion as an indistinguishability game is difficult since the state which the adversary outputs is not fixed. For example, if the adversary stores a truncated hash of the original ciphertext then the game cannot compute another ciphertext, which makes indistinguishability difficult. Since this definition aims to convey that an adversary should not be able to re-encrypt back to the old key, we do not consider the lack of indistinguishability as a significant drawback.

We now formulate a definition for *maximum irreversibility.* Informally, the amount of storage needed to reverse a re-encryption is at least the size of the old ciphertext components that were updated.

Definition IV.8. *A PRE scheme is* maximally irreversible *if it is s-reversible, where s is the total size of the ciphertext components updated by* ReEnc.

Clearly, *maximum irreversibility* is stronger than traditional notions of unidirectionality as the adversary sees an original ciphertext under pk_i.

IV: 4.2.1 Observations

1. The storage bound $\overline{\lambda}$ can be considered similarly to the security parameter λ in that the larger $\overline{\lambda}$ is, the more secure the scheme is. However, even small values for $\overline{\lambda}$ are still meaningful since they convey how easy it is to reverse a re-encryption and can therefore be used to compare different schemes.
2. The most useful values which $\overline{\lambda}$ can take are $\overline{\lambda} = |\Delta|$, as this is comparable to traditional bidirectionality, or $\overline{\lambda} = s$, as this makes a scheme maximally irreversible. In general, useful values are in the range $|\Delta| \leq \overline{\lambda} \leq |s|$.
3. If a scheme is both ReEnc-IND-CPA-secure and maximally irreversible then it is $|C|$-irreversible.
4. All traditionally bidirectional[4] schemes can be shown to be $|\Delta|$-reversible, but there also exist $|\Delta|$-reversible schemes which are *not* traditionally bidirectional, as we shall see in Section IV: 4.3. Since more attacks are covered, saying that a scheme is $|\Delta|$-reversible is stronger than saying it is bidirectional in the traditional sense.
5. Our definition also applies to schemes where the ciphertext is not fully re-randomised upon re-encryption (not ReEnc-IND-CPA-secure), since best-achievable unidirectionality only requires that the state is the size of the number of components which are updated as opposed to the entire ciphertext.
6. We assume that the ciphertext is as compact as it could be, so for all ciphertexts c there is no compression function which allows c to be stored as c^* with $|c^*| < |c|$. For example, IND\$-CPA-security implies that ciphertexts are indistinguishable from random strings, so for schemes with this property only storing a subset of the bits means the adversary can do no better than guessing to obtain the remainder of the ciphertext. If the ciphertexts can be compressed, we can apply the definition to a compressed version of the PRE scheme by having encryption compress the ciphertext, decryption decompress it and have the re-encryption function perform decompression and compression.

Token robust schemes can be either unidirectional or bidirectional. In other words, token robustness says nothing about directionality. However, note that if a scheme is token robust and we can prove that the only way of reversing a re-encryption is by storing a

[4]The informal notion that an update token $\Delta_{i,j}$ can also re-encrypt from pk_j to pk_i.

state the size of the original ciphertext, then this is the best notion of unidirectionality that can be achieved without assuming that the adversary honestly deletes old ciphertexts. In Section IV: 5 we use this definition to define *best-achievable unidirectionality.* By considering unidirectionality in this way, the problem of creating a unidirectional multi-hop PRE scheme may be solved more easily using token robustness and could therefore lead to more unidirectional multi-hop schemes that are practically implementable.

IV: 4.3 Existing schemes under the new definition

Some traditionally unidirectional schemes are $|\Delta|-$reversible. For ciphertext-dependent schemes, this means they are bidirectional. In both [EPRS17] and [BLMR13], the update token consists of the new header and another value used to change the body of the ciphertext using an arithmetic operation. We can generalise this by saying $\Delta = (\Delta_0, \Delta_1)$, where $\Delta_0 = \tilde{c}'$ and $(\Delta_1)^{-1}$ is easily computable. To reverse the re-encryption, the adversary $\mathcal{A}_1$ retains the old header $\tilde{c}$, and computes the inverse of Δ_1, setting $st_{\mathcal{A}} = (\tilde{c}, \Delta_1^{-1})$. Then $\mathcal{A}_2$ can recover $c \leftarrow \mathsf{ReEnc}(st_{\mathcal{A}}, c')$ to win the game. Note that $\mathcal{A}$ does not need to retain Δ_0 as this is contained in the new ciphertext. The state $st_{\mathcal{A}}$ is clearly the same size as $\Delta = (\tilde{c}', \Delta_1)$, and we can therefore consider such schemes to be $|\Delta|$-reversible. Since any adversary willing to store information of size up to $|\Delta|$ will not restrict themselves to retaining Δ alone, our definition better reflects the intuition behind bidirectionality. In particular, because [EPRS17] is ciphertext-dependent, it should be considered bidirectional under these realistic assumptions.

Some existing bidirectional schemes are maximally irreversible. In the multi-hop PRE scheme of [CH07], $\mathsf{ReKeyGen}$ takes two secret keys as input and the ciphertext includes a number of components including $B = (g^a)^r$, where $\mathsf{pk} = g^a, \mathsf{sk} = a$ and $r \xleftarrow{\$} \mathbb{Z}_q$. The re-encryption token takes as input two secret keys a, b and outputs $\Delta_{a,b} = b/a$, which is then used to update B and no other part of the ciphertext. Since both the ciphertext element B and the re-keying token $\Delta_{a,b}$ are integers modulo q, an adversary hoping to reverse the re-encryption by storing $\Delta_{a,b}$ could have simply retained B. Particularly for applications where there is one message per key pair, the server would need to store one token per re-encrypted ciphertext, in which case they could have retained every original ciphertext [5]. Similarly, the original symmetric PRE scheme [BBS98] may also be considered maximally irreversible.

IV: 5 Proxy Re-Encryption in the Malicious model

We now combine our contributions in Sections IV: 3 and IV: 4 to define a model where the server cannot perform encryptions unless initiated by a client. We thus explore malicious behaviours other than breaking confidentiality. We discuss some conditions which apply to re-encryption generally, before explaining the stronger conditions specific to our setting. Clearly, *correctness* is a necessary property of all PRE schemes. We consider ReEnc-IND-CPA-secure as another necessary condition, despite the fact that this is not the case in much existing work [BBS98, AFGH05, CH07, LV08a, CWYD10], as we aim for PRE to be used for revocation and key expiry.

In the malicious model, we must ensure that giving the server the ability to perform some re-encryptions does not mean they can perform unauthorised re-encryptions. In particular, we want our setting to consider revoked users who are honest-but-curious in that they may try to decrypt re-encrypted ciphertexts, but do not collude with the server directly. We thus require a means of ensuring that only authorised re-encryptions are possible. The inability to perform unauthorised re-encryptions breaks down to two necessary properties:

1. **Token Robustness.** *No matter how many re-encryption tokens the server sees, the server cannot use these to form a token which encrypts a ciphertext to a new key.* This means the adversary is unable to share messages with users who have not had access to them before. Ciphertext-dependence is reasonably trivial to achieve for ElGamal-based schemes such as [BBS98] by having the randomness used to encrypt the message input to ReKeyGen. We build on this existing technique [EPRS17] to create a token robust scheme.

2. **Maximal irreversibility.** *The token used to perform a re-encryption cannot be used to reverse that re-encryption.* This is particularly necessary when considering revocation and key expiry. If re-encryption has been performed to revoke access, then

[5] As the value B is unique for each ciphertext, retaining B for one ciphertext does not allow a different ciphertext to be re-encrypted whereas the update token can re-encrypt any ciphertext in either direction. This further demonstrates why token robustness is a necessary requirement.

reversing that re-encryption regrants access to the revoked user. Our definition of maximal irreversability conveys this under realistic assumptions.

We combine these definitions to form the following definition which is a requirement for a PRE scheme used to enforce changes to access control policy on a malicious server.

Definition IV.9. *A PRE scheme is* best-achievable unidirectional *if it is both token robust and maximally irreversible.*

Token robustness implies that a token $\Delta_{i,j}^{c_i}$ cannot be used to re-encrypt a ciphertext $c_j \xleftarrow{\$} \mathsf{Enc}(pk_j, m)$ where $c_j \neq c_i$ (except with negligible probability), which covers the traditional notion of unidirectionality. Coupled with maximal irreversibility, this means that given a re-encrypted ciphertext c_j under pk_j, the only way that the adversary can produce a ciphertext c_i such that $\mathsf{Dec}(\mathsf{sk}_i, c_i) = \mathsf{Dec}(\mathsf{sk}_j, c_j)$, where pk_i is the original key, is by retaining a state the size of the original ciphertext during the re-encryption process.

We show in Section IV: 5.1 that it is possible to have a scheme which is best-achievable unidirectional using a simple adaptation of ElGamal-based PRE, and prove its security under the DDH assumption.

IV: 5.1 A Secure PRE scheme in the malicious model

Recall that for PRE suitable for access control, we require a multi-hop scheme. For PRE in the malicious model, we require a scheme which is unidirectional, ciphertext-dependent and token robust. We present our scheme in Figure IV.5, based on ElGamal encryption.

Lemma IV.1. *The scheme given in Figure IV.5 is correct.*

Proof. Encryptions have the form $c = (g^y, m \cdot g^{xy})$ for message m and public key g^x. Decrypting using secret key x results in $m' = \tilde{c}^{-x} \cdot \bar{c} = (g^y)^{-x} \cdot m \cdot g^{xy} = m$, as required.

The update token resulting from $\mathsf{ReEnc}(\mathsf{sk}_i = x_i, \mathsf{pk}_j = g^{x_j}, \tilde{c} = g^y)$ has the form $\Delta_{i,j}^{c} = (g^{y'}, g^{x_j y' - x_i y})$, meaning that re-encrypted ciphertexts have the form $c_j = (g^{y'}, m \cdot g^{x_i y} \cdot g^{x_j y' - x_i y}) = (g^{y'}, m \cdot g^{x_j y'})$, which is the same as a fresh ciphertext under g^{x_j}, so correctness follows. □

```
$\mathsf{Setup}(1^\lambda) \to \text{params}$
    $p$ large prime
    $g$ a generator of $\mathbb{Z}_p$
    $\mathcal{M} = \mathbb{Z}_p$
    $\mathcal{T} = \mathcal{K} = \mathbb{Z}_p^*$
    return params $= (p, g, \mathcal{M}, \mathcal{K}, \mathcal{T})$

$\mathsf{KeyGen}(1^\lambda) \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$
    $x \xleftarrow{\$} \mathbb{Z}_p^*$
    $\mathsf{pk} = g^x, \mathsf{sk} = x$
    return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Enc}(\mathsf{pk}, m) \xrightarrow{\$} c$
    $y \xleftarrow{\$} \mathbb{Z}_p^*$
    $\tilde{c} = g^y$
    $\bar{c} = m \cdot \mathsf{pk}^y = m \cdot g^{xy}$
    return $c = (\tilde{c}, \bar{c})$

$\mathsf{Dec}(\mathsf{sk}, c) \to m$
    $m' = \tilde{c}^{-\mathsf{sk}} \cdot \bar{c} = (g^y)^{-x} \cdot m \cdot g^{xy}$
    return $m'$

$\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j, \tilde{c}) \xrightarrow{\$} \Delta_{i,j}^c$
    $y' \xleftarrow{\$} \mathbb{Z}_p^*$
    $a = g^{y'}$
    $b = (\mathsf{pk}_j)^{y'} \cdot \tilde{c}^{-\mathsf{sk}_i} = g^{x_j y'} \cdot g^{-x_i y}$
    return $\Delta_{i,j}^c = (a, b)$

$\mathsf{ReEnc}(\Delta_{i,j,}^c, c) \to c'$
    $(\tilde{c}, \bar{c}) \leftarrow c$
    $(a, b) \leftarrow \Delta_{i,j}^c$
    $\tilde{c}' = a = g^{y'}$
    $\bar{c}' = \bar{c} \cdot b = m \cdot g^{x_i y} \cdot g^{x_j y' - x_i y}$
    return $c' = (\tilde{c}', \bar{c}')$
```

Figure IV.5: An ElGamal-based scheme similar to the one given by Blaze et al [BBS98] which is best-achievable unidirectional and token robust.

IV: 5.2 Security analysis

First we show our construction is ReEnc-IND-CPA, then best-achievable unidirectional, and finally token robust.

Theorem IV.2. *The scheme described in Figure IV.5 is ReEnc-IND-CPA under the DDH assumption.*

Proof. Recall that in the ReEnc-IND-CPA game, the adversary does not learn the precise update token used to perform the re-encryption (similarly to [EPRS17]). Therefore it suffices to show that a re-encrypted ciphertext has the same form as a fresh encryption of the message under the new key. In other words, re-encrypted ciphertexts under x_j are identically distributed to ciphertexts encrypted for the first time under x_j. We have already demonstrated this in the proof of Lemma IV.1.

Proving ReEnc-IND-CPA therefore reduces to ElGamal being IND-CPA. □

Theorem IV.3. *The scheme in Figure IV.5 is best-achievable unidirectional.*

We prove this through two lemmas, first showing that the scheme is maximally irreversible then that it has token robustness.

Lemma IV.2. *The scheme described in Figure IV.5 is maximally irreversible under the DDH assumption.*

In this proof we need to show that the adversary must retain a state of size $\overline{\lambda} = s = 2|x|$, where $x \in \mathbb{Z}_p$, as the token and ciphertext components are all the same size ($|\Delta| = |\tilde{c}| = |\bar{c}|$) and all elements of $\mathbb{Z}_p$, and both ciphertext components are updated during re-encryption. We do not consider the advantage of the adversary storing $g^{x_i y - x_j y'}$ and only part of g^y (or vice versa) here, but in Section IV: 6.3 we discuss unidirectionality when $\overline{\lambda} \in \{0, 1, \ldots, s-1, s\}$.

Proof. As the old header is replaced independently of the update token, the update token alone cannot be used to reverse a ciphertext. If an adversary has both the re-encryption token used as well as the first component of the old ciphertext, then reversing the re-encryption is trivial. We demonstrate that retaining both values would require the same amount of storage as retaining the original ciphertext, and therefore is maximally irreversible.

First we observe that $\bar{c}$ and $\Delta_{i,j}^{c(1)}$ can be considered interchangeably as components to be retained in $st_{\mathcal{A}}$. For simplicity, we only consider $\Delta_{i,j}^{c(1)}$ as a potential input to $st_{\mathcal{A}}$. There are only two values which can revert re-encryption of the ciphertext – the old ciphertext header $\tilde{c}$ and $\Delta_{i,j}^{c(1)}$. For the scheme to be best-achievable unidirectional, we need to show that both values must be retained for a successful reversal attack.

We begin by showing that an adversary who only retains $\Delta_{i,j}^{c(1)} = g^{x_i y - x_j y'}$ cannot derive $\tilde{c} = g^y$. We first recall the DDH assumption which states, given (g^a, g^b, g^e) where either $e = ab$ or $e \xleftarrow{\$} \mathbb{Z}_p^*$, no PPT adversary can distinguish which was given.

In addition to $g^{x_i y - x_j y'}$, we also assume that the server knows the public keys g^{x_i}, g^{x_j} and the header of the new file, $g^{y'}$. For simpler notation, let $a = x_i, b = x_j, d = y, e = y'$. The adversary's task is therefore: given (a, g^b, g^e, g^{be-ad}), find g^d. We show that if there exists an adversary $\mathcal{B}$ such that $\mathcal{B}(a, g^b, g^e, g^{be-ad}) \to g^d$ for $a, b, d, e \xleftarrow{\$} \mathbb{Z}_p^*$, then this would break DDH — given a DDH challenge (g^a, g^b, g^f), we could sample $a, d \xleftarrow{\$} \mathbb{Z}_p^*$ and call $\mathcal{B}(a, g^b, g^e, g^{f-ad}) \to g^{d'}$. If $g^{d'} = g^d$ then $f = be$, breaking DDH. We therefore conclude that g^y cannot be recovered from $(x_i, g^{x_j}, g^{y'}, g^{x_j y' - x_i y})$.

For the other direction, we note that the adversary being able to calculate g^{bd-ac} from (a, g^b, g^d, g^c) would clearly also break DDH. We therefore conclude that $g^{x_j y' - x_i y}$ cannot be recovered from $(x_i, g^{x_j}, g^{y'}, g^y)$.

It remains to show that the adversary cannot output an equivalent ciphertext $(g^{\hat{y}}, m \cdot g^{x_i \hat{y}})$ for some $\hat{y} \in \mathbb{Z}_p^*$ given $(g^{y'}, m \cdot g^{x_j y'}), x_i, g^{x_j}$. To do this the adversary would need to be able to derive m by calculating $g^{x_j y'}$, which would again imply the DDH assumption can be broken. $\square$

This proof shows that the attacker needs to store at least as much as if they were storing the original ciphertext. This means any successful method for obtaining a copy of the message under the old key will be less efficient than simply retaining the original ciphertext.

Lemma IV.3. *The scheme in Figure IV.5 has token robustness under the CDH assumption.*

Proof. To win the token robustness game, the adversary must output a token which re-encrypts an honestly-generated ciphertext so that it is under a key that it has not been encrypted under before.

Recall that the encryption oracle $\mathsf{Enc}(i, m)$ outputs a ciphertext of the form $(g^y, m \cdot g^{x_i y})$ and the token generation oracle $\mathsf{ReKeyGen}(i, j, c)$ outputs a re-encryption token of the form $(g^{y'}, g^{x_j y' - x_i y})$. Let the challenge input ciphertext be denoted $c_i^{\mathcal{A}} = (g^y, m \cdot g^{x_i y})$. Then the adversary must output a token of the form $\Delta_{\mathcal{A}} = (g^{y'}, g^{x_j y' - x_i y})$, where $j \notin \mathsf{chain}(m)$. It may be possible that querying two completely different keys and a different ciphertext $c_l = (g^{y_l}, m \cdot g^{x_l y_l})$ results in a token $\Delta_{l,k}^{c_l} = (g^{y_k}, g^{x_k y_k - x_l y_l})$, where $g^{x_k y_k - x_l y_l} = g^{x_i y - x_j y'}$, but this only occurs with negligible probability.

The token generation oracle means that the adversary receives tokens of the form $\Delta_{i,k}^{c_i^{\mathcal{A}}} = (g^{y''}, g^{x_k y'' - x_i y})$. The adversary's goal is to output an update token $\Delta^{\mathcal{A}} = (g^{y'}, g^{x_j y' - x_i y})$, where $x_i = sk_i$ and $c = (g^y, m \cdot g^{xy})$ is the ciphertext output by $\mathcal{A}$ to be re-encrypted to a new key. We note that it is trivial for the adversary to calculate $g^{x_j y'}$ for some $y' \xleftarrow{\$} \mathbb{Z}_p^*$ from pk_j by setting $\mathsf{pk}_j^{y'} = g^{x_j y'}$. To create a winning $\Delta_{i,j}^{\mathcal{A}}$, it remains for $\mathcal{A}$ to calculate $g^{x_i y}$. Since the ciphertext is honestly generated ($c_i^{\mathcal{A}} \in \mathcal{C}_{\mathsf{honest}}$), the adversary does not know y.

Since factoring is a difficult problem modulo p (otherwise, ElGamal would not be

IND-CPA secure), g^{xy} cannot easily be extracted from tokens of the form $\Delta_{i,k,}^{c_i^{\mathcal{A}}} = g^{x_k y'' - x_i y}$ for some $x_k \leftarrow \mathsf{KeyGen}(1^\lambda), y'' \in \mathbb{Z}_p^*$. We conclude that the adversary can only compute ciphertexts of the correct form for keys pk_i where $i \in \mathsf{chain}(\mathsf{Dec}(\mathsf{sk}_i, c_i^{\mathcal{A}}))$. Since the adversary must output a token for a new key to win the game, we conclude the scheme is token robust. □

We have shown that our scheme is suitable for the malicious model according to the goals we outlined at the beginning of this chapter. This means a malicious server will be unable to perform unauthorised re-encryptions on stored files, as much as is possible to guarantee given realistic storage assumptions.

IV: 6 Authenticating Re-Encryption

We now present an alternative approach to ensuring that re-encryptions have been initiated by honest parties by revisiting the traditional notion of DOA. Recall from Section IV: 2.4 that traditionally, data origin authentication is intended for settings where the party who created the message also encrypts it. This means that it is sufficient for the appropriate tag to tie the identity of the sender to the message prior to encryption. However, for PRE this is not always the case. In situations where there is one key per party, then the act of having created the re-encryption key would imply knowledge of sk_i, meaning that the receiver can be sure that re-encryption was initiated by the party whose public key is pk_i. Therefore a correct re-encryption implies that the re-encryption was initiated by a party with access to the message.

However, in applications where more than one party shares an encryption key, such as some access control systems, proof of having used this key is not sufficient to authenticate the encryptor / re-encryption initiator. For the DOA mechanism discussed in Section IV: 2.4, the underlying signature will not change when the ciphertext is re-encrypted, so the original encryptor will be the one whose identity is tied to the ciphertext. Whilst this may be desirable in some applications, our focus is on authenticating who initiated the most recent re-encryption of the ciphertext. Since existing methods for achieving DOA should be simple extensions to any PRE scheme, we restrict our focus to COA. Both signatures and PRE could be combined to create an authenticated PRE scheme. This is useful in auditing changes to access control policy, or enabling users to verify which user has revoked their access.

For example, consider the scenario where keys represent classifications, so groups with the appropriate classification have the secret key. Now suppose that Alice shares a file with Mallory. She re-encrypts the ciphertext so it can now be decrypted by anyone in Mallory's group. Mallory now wants to re-encrypt this file to share it with Bob without Alice's permission. In a traditional PRE scheme, when this second re-encryption happens, it is unknown which of the people in Mallory's group generated the update token, so there is no proof that Mallory is the one who shared the file without authorisation. Digital rights management can be considered a real world application of this scenario – if Mallory is pirating music and trying to pass off pirated copies as legitimate, then anyone can verify that Mallory is the distributor and not the owner of the rights. We acknowledge that Mallory could simply share the file unencrypted, but this is an unavoidable risk when sharing files.

We aim to define a concept whereby the identity of the re-encryption initiator (or encryptor in the case of a fresh encryption) can be verified from the ciphertext in a way that carries the same unforgeability guarantees as DOA. In other words, Alice will be able to verify that Mallory was the party who performed the re-encryption. Now that we have outlined the distinction between DOA and what we are trying to achieve, we formulate the notion of *Ciphertext Origin Authentication (COA).*

IV: 6.1 Correctness upon verification

Most Authenticated Encryption (AE) and Signcryption schemes [Zhe97] use a mechanism where during the decryption process an authentication check is made, and the receiver is supposed to terminate the decryption process if the check fails. Such a check is also made in [EPRS17]. However, if corrupted users are being considered then honest termination of a process is not guaranteed – a malicious user could ignore the outcome of the authentication check and derive the message regardless. We therefore aim for compulsory COA that provides stronger security against users who want to change the access controls without being caught, as the message can *only* be derived if identity is verified correctly. We call this *correctness upon verification* and consider it a secondary goal.

The most intuitive way of proving which entity encrypted a message is to show proof of knowledge of the secret information used to form the new ciphertext. In our scheme outlined in Figure IV.5 this is the value y. However, the COA check must not leak y as this will enable decryption using the public key. Therefore, we need the initiator to prove that they know y without revealing it.

Recall the basic ElGamal signature scheme Figure II.3. The hash of the message m is signed so as to enable the signing of arbitrary length messages. We can see correctness as

$$\begin{aligned}\mathsf{svk}^{\sigma_0} \cdot \sigma_0^{\sigma_1} &= (g^{\mathbf{x}})^r \cdot r^s \\ &= g^{\mathbf{x}r} \cdot (g^k)^{(h(m)-\mathbf{x}r)k^{-1}} \\ &= g^{\mathbf{x}r} \cdot g^{h(m)-\mathbf{x}r} \\ &= g^{h(m)}.\end{aligned}$$

We can obtain a mandatory COA check by replacing g^y in our original scheme (Figure IV.5) with an ElGamal signature on y signed using the initiator's signing key. Note that we do not need to sign the hash of y, as y is already a random value in the correct range. We also adapt the Verify algorithm so that instead of it returning 0 or 1 to indicate a pass or fail, it derives a specific value a. We call the resulting algorithm VerRetrieve and describe it in Figure IV.6.

```
$\mathsf{sigKeyGen}(1^\lambda) \overset{(\$)}{\rightarrow} (\mathsf{ssk}, \mathsf{svk})$
    $\mathbf{x} \overset{\$}{\leftarrow} \mathbb{Z}_p^*$
    $\mathsf{ssk} = \mathbf{x}, \mathsf{svk} = (g^{\mathbf{x}})$
    return $(\mathsf{ssk}, \mathsf{svk})$

$\mathsf{Sign}(\mathsf{ssk}, y) \overset{(\$)}{\rightarrow} \sigma$
    $\mathbf{x} \leftarrow \mathsf{ssk}$
    $k \overset{\$}{\leftarrow} \mathbb{Z}_p, r = g^k$
    $s = (y - \mathbf{x}r)k^{-1} \mod p-1$
    $\sigma = (r, s)$
    return $\sigma$

$\mathsf{VerRetrieve}(\mathsf{svk}, \sigma) \rightarrow a$
    $(r, s) = \sigma$
    $a = \mathsf{svk}^r \cdot r^s = (g^{\mathbf{x}})^r \cdot r^s$
    return $a$
```

Figure IV.6: A variation on ElGamal signatures where verification returns a value a as opposed to a bit.

Ciphertexts now have the form $c = (\sigma, m \cdot g^{xy})$. By the correctness of ElGamal signatures, $\mathsf{VerRetrieve}(\mathsf{svk} = g^{\mathbf{x}}, \sigma)$ should return $a = g^y$ if σ was signed using $\mathsf{ssk} = \mathbf{x}$, since $(g^{\mathbf{x}})^r \cdot r^s = g^{\mathbf{x}r} \cdot g^{k(y-\mathbf{x}r)k^{-1}} = g^y$. Since y is not revealed during this process, this adaptation should still meet the security properties proved in Section IV: 5.2. Since g^y is needed for decryption, if follows that correctly deriving g^y is necessary to derive the correct message. Correctly deriving g^y relies on correct verification during decryption, meaning that verification is not optional, and so the decryptor cannot deny knowledge of who encrypted the message.

However, if the scheme is only adapted with the change described in this section, then there is no confirmation that the obtained message m is the correct one, so by extension the receiver cannot verify that g^y is correct. Suppose that Alice encrypts a message m to obtain a ciphertext (σ_A, c_A), and that she now shares this with Mallory (so Mallory knows x). Mallory can forge Alice's signature on a message m_M of her choice encrypted under keys for which Mallory knows the corresponding secret $\mathsf{sk}' = x'$ by computing the following:

```
$\mathsf{Forge}(m_M, g^{\mathsf{x}}_A, x', \sigma_A) \to (\sigma_A, c_M)$
$\sigma_A = (r_A, s_A)$
$g^y = (g^{\mathsf{x}}_A)^r \cdot r_A^{s_A}$
$c_M = (g^y, m_M \cdot (g^y)^{x'})$
**return** $(\sigma_A, c_M)$
```

Then receivers of (σ_A, c_M) will conclude that Alice encrypted the message m_M, not Mallory. We note that the encryption used here can either be done using the same secret key $(x' = x)$ or a different one $(x' \neq x)$. This is a realistic scenario in access control schemes where decryption keys are shared, and the attack means that anyone who knows the secret key can forge the encryptor's signature. Therefore in order to have COA, we also need a message integrity check.

We propose adapting the encryption mechanism to replace $\bar{c} = m \cdot g^{xy}$ with $\bar{c} = (g^{x\hat{y}\cdot h(m)}, m \cdot g^{xy})$, where $\hat{y} \xleftarrow{\$} \mathbb{Z}_p^*$, and adding the matching signature to the header. So $\tilde{c}$ now contains two signatures – one used to derive g^y and the other used to verify the message m.

We redefine $\mathsf{ReKeyGen}$ as $\mathsf{authReKeyGen}$, which additionally takes the correct signature verification key svk_A and re-encryption initator's signing key ssk_B as input. To achieve best-achievable unidirectionality, $\mathsf{authReKeyGen}$ selects a new $\hat{y}$ uniformly at random so that new entropy is added throughout the ciphertext. Then to verify the derived message m', the receiver derives $\hat{a}$ from σ and confirms that $a^{xh(m')} = \bar{c}_0$. If they match then we

```
$\mathsf{Setup}(1^\lambda) \rightarrow \text{params}$
    $p$ large prime
    $g$ a generator of $\mathbb{Z}_p$
    return $(p, g)$

$\mathsf{EncKeyGen}(1^\lambda) \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$
    $x \xleftarrow{\$} \mathbb{Z}_p^*$
    $\mathsf{pk} = g^x, \mathsf{sk} = x$
    return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{sigKeyGen}(1^\lambda) \xrightarrow{\$} (\mathsf{svk}, \mathsf{ssk})$
    $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_p^*$
    $\mathsf{ssk} = \mathbf{x}, \mathsf{svk} = g^{\mathbf{x}}$
    return $(\mathsf{svk}, \mathsf{ssk})$

$\mathsf{Sign}(\mathsf{ssk}, y) \xrightarrow{\$} \sigma$
    $k \xleftarrow{\$} \mathbb{Z}_p, r = g^k$
    $s = (y - \mathsf{ssk} \cdot r)k^{-1} \mod p - 1$
    $\sigma = (r, s)$
    return $\sigma$

$\mathsf{VerRetrieve}(\mathsf{svk}, \sigma) \rightarrow a$
    $(r, s) \leftarrow \sigma$
    $a = \mathsf{svk}^r \cdot r^s$
    return $a$

$\mathsf{authEnc}(\mathsf{ssk}, \mathsf{pk}, m) \xrightarrow{\$} c$
    $y \xleftarrow{\$} \mathbb{Z}_p^*, \sigma \leftarrow \mathsf{Sign}(\mathsf{ssk}, y)$
    $\hat{y} \xleftarrow{\$} \mathbb{Z}_p^*, \hat{\sigma} \leftarrow \mathsf{Sign}(\mathsf{ssk}, \hat{y})$
    $\tilde{c} = (\sigma, \hat{\sigma})$
    $\bar{c} = (\mathsf{pk}^{\hat{y} \cdot h(m)}, m \cdot \mathsf{pk}^y)$
    return $c = (\tilde{c}, \bar{c})$

$\mathsf{authDec}(\mathsf{svk}, \mathsf{sk}, c) \rightarrow m$
    $(\tilde{c}, \bar{c}) \leftarrow c$
    $(\sigma, \hat{\sigma}) \leftarrow \tilde{c}$
    $a \leftarrow \mathsf{VerRetrieve}(\mathsf{svk}, \sigma)$
    $\hat{a} \leftarrow \mathsf{VerRetrieve}(\mathsf{svk}, \hat{\sigma})$
    $m' = a^{-\mathsf{sk}} \cdot \bar{c}_1$
    if $\hat{a}^{\mathsf{sk} \cdot h(m')} \neq \bar{c}_0$ :
        return $\perp$
    else :
        return $m'$

$\mathsf{authReKeyGen}(\mathsf{svk}_A, \mathsf{ssk}_B, \mathsf{sk}_i, \mathsf{pk}_j, \tilde{c}) \xrightarrow{\$} \Delta_{i,j}^c$
    $(\sigma, \hat{\sigma}) \leftarrow \tilde{c}$
    $a \leftarrow \mathsf{VerRetrieve}(\mathsf{svk}_A, \sigma)$
    $\hat{a} \leftarrow \mathsf{VerRetrieve}(\mathsf{svk}_A, \hat{\sigma})$
    $y' \xleftarrow{\$} \mathbb{Z}_p^*, \sigma' \leftarrow \mathsf{Sign}(\mathsf{ssk}_B, y')$
    $\hat{y}' \xleftarrow{\$} \mathbb{Z}_p^*, \hat{\sigma}' \leftarrow \mathsf{Sign}(\mathsf{ssk}_B, \hat{y}')$
    $\tilde{c}' = (\sigma', \hat{\sigma}')$
    $\Delta_{i,j}^c = (\tilde{c}', (\mathsf{pk}_j)^{\hat{y}'} \cdot \hat{a}^{-\mathsf{sk}_i}, (\mathsf{pk}_j)^{y'} \cdot a^{-\mathsf{sk}_i})$
    return $\Delta_{i,j}^c$

$\mathsf{ReEnc}(\Delta_{i,j}^c, c) \rightarrow c'$
    $(\tilde{c}', \Delta_1, \Delta_2) = \Delta_{i,j}^c$
    $\bar{c}' = (\bar{c}_0 \cdot \Delta_1, \bar{c}_1 \cdot \Delta_2)$
    return $(\tilde{c}', \bar{c}')$
```

Figure IV.7: The scheme given in Figure IV.5 adapted such that it also provides ciphertext origin authentication. Receivers must verify the encryptor (re-encryption initiator) of the ciphertext to decrypt the message.

have both message integrity and COA. Therefore this mechanism provides a means of pairing g^y with m and associating this with the identity of the encryptor (or re-encryption initiator). The full adapted scheme is given in Figure IV.7.

IV: 6.2 COA in other schemes

COA can be added to other schemes. It is sufficient to create a signature using the encryptor's signing key and the randomness used to form the ciphertext and changing re-encryption and decryption accordingly. We now demonstrate how to extend existing schemes to offer COA. For other ElGamal-based schemes including [BBS98, LV08b, LT18], a similar adaptation to the one made in Section IV: 6.1 can be made.

We now propose an extension to ReCrypt [EPRS17]. Whilst ReCrypt uses symmetric PRE for re-encryption, we believe that COA is still valid for symmetric encryption. Recall that a COA check can either be optional or compulsory, depending on whether corrupted users are considered. In ReCrypt, the message integrity check is optional and it is therefore arguable whether an optional COA check suffices in their model. We first describe a simple extension which creates an optional check here.

For a basic extension, the ciphertext should replace χ with $(\chi, \sigma = \mathsf{Sign}(\mathbf{x}, \chi))$ and decryption should include the step $\mathsf{Verify}((g^{\mathbf{x}}), \sigma, \chi)$ and only return m if this outputs 1. We note that this may be preferable to our scheme in Figure IV.5 depending on the application, as it could theoretically permit the re-encryption of longer messages (dependent on the existence of an efficient KH-PRF, $\mathcal{F}$), with the caveat that it is not best-achievable unidirectional.

Since ReCrypt is not an ElGamal-based scheme, the adaptation for a mandatory COA check is more complicated. Essentially, we replace $x, y \xleftarrow{\$} \mathcal{K}$ and $\chi = x + y$ in the ciphertext header by having $x, y \xleftarrow{\$} \{i, j, : i + j = g^r\}$ for $r \xleftarrow{\$} \mathbb{Z}_p^*$ and setting $\chi = g^r$. Then by signing r, $\mathsf{VerRetrieve}$ will return $g^r = \chi$ and decryption is adjusted accordingly. Note that we move y from the ciphertext body into the ciphertext header, as it is now needed to generate update tokens. The full extension is described in Figure IV.8.

IV: 6.3 Other choices for $\overline{\lambda}$

Recall that in our definition of unidirectionality (Definition IV.7), the security parameter $\overline{\lambda}$ is set according to the size of ciphertext components ($\overline{\lambda} \in \{0, |\Delta|, |\tilde{c}|, |\bar{c}|, s\}$, where s is the total size of ciphertext components updated during re-encryption). A more fine-grained definition would be to allow $\overline{\lambda}$ to be any number of bits $\overline{\lambda} \in \{0, 1, \ldots, s\}$. We decided against this definition since, although it is more fine-grained in terms of security, it makes proofs much harder for what we consider not to be a significant distinction in practice. The main difference in terms of proofs is that the more fine-grained options for $\overline{\lambda}$ require additional proofs that an adversary who only retains part of a component has no real advantage in calculating the rest. Ultimately, whilst retaining certain values such as the old header or the update token make sense, it is harder to envisage or justify an adversary

```
$\mathsf{Setup}(1^\lambda) \to \text{params}$
    a symmetric encryption scheme $\mathcal{SE}$
    a KH-PRF $\mathcal{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$
    a hash function $h$
    params $= (\mathcal{SE}, \mathcal{F}, h)$
    return params

$\mathsf{encKeyGen}(1^\lambda) \overset{(\$)}{\to} k$
    $k \overset{\$}{\leftarrow} \mathcal{SE}.\mathsf{KeyGen}(1^\lambda)$

$\mathsf{sigKeyGen}(1^\lambda) \overset{(\$)}{\to} (\mathsf{ssk}, \mathsf{svk})$
    $\mathbf{x} \overset{\$}{\leftarrow} \mathbb{Z}_p^*$
    $\mathsf{ssk} = \mathbf{x}, \mathsf{svk} = g^{\mathbf{x}}$
    return $(\mathsf{ssk}, \mathsf{svk})$

$\mathsf{authEnc}(k, \mathsf{ssk}, m) \overset{(\$)}{\to} c$
    $r \overset{\$}{\leftarrow} \mathbb{Z}_p^*$
    $x, y \overset{\$}{\leftarrow} \{i, j, : i + j = g^r\}$
    $\sigma \leftarrow \mathsf{Sign}(\mathsf{ssk}, r)$
    $\tau = h(m) + F(x, 0)$
    $\tilde{c} = (\mathcal{E}_k(\sigma, \tau), y)$
    $m_1 || m_2 || \ldots || m_n \leftarrow m$
    for $1 \leq i \leq n$ :
        $\bar{c}_i = m_i + F(x, i + 1)$
    $\bar{c} \leftarrow \bar{c}_0 || \bar{c}_1 || \ldots || \bar{c}_n$
    return $c = (\tilde{c}, \bar{c})$

$\mathsf{authDec}(k, \mathsf{svk}, (\tilde{c}, \bar{c})) \to m$
    $(\tilde{c}, \bar{c}) \leftarrow c$
    $(\sigma, \tau) \leftarrow \mathcal{D}_k(\tilde{c}_0)$
    if $(\sigma, \tau) = \perp$ : return $\perp$
    $\chi \leftarrow \mathsf{VerRetrieve}(\mathsf{svk}, \sigma)$
    for $1 \leq i \leq n$ :
        $m_i = \bar{c}_i - F(\chi - y, i + 1)$
    if $h(m) + F(\chi - y, 0) \neq \tau$ :
        return $\perp$
    else : return $m$

$\mathsf{authReKeyGen}(k_i, \mathsf{svk}_A, k_j, \mathbf{x}_B, \tilde{c}) \overset{(\$)}{\to} \Delta_{i,j}^c$
    $(\sigma, \tau) \leftarrow \mathcal{D}_{k_i}(\tilde{c}_0)$
    if $(\sigma, \tau) = \perp$ : return $\perp$
    $\chi \leftarrow \mathsf{VerRetrieve}(\mathsf{svk}_A, \sigma)$
    $y = \tilde{c}_1, x = \chi - y$
    $r' \overset{\$}{\leftarrow} \mathbb{Z}_p^*$
    $x', y' \overset{\$}{\leftarrow} \{i, j, : i + j = g^{r'}\}$
    $\sigma' \leftarrow \mathsf{Sign}(\mathsf{ssk}_B, r')$
    $a = x' - x$
    $\tau' = \tau + F(a, 0)$
    $\tilde{c}' = (\mathcal{E}_{k_j}(\sigma', \tau'), y')$
    return $\Delta_{i,j}^c = (\tilde{c}', a)$

$\mathsf{ReEnc}(\Delta_{i,j}^c, c) \to c'$
    $(\tilde{c}', a) \leftarrow \Delta_{i,j}^c$
    for $1 \leq i \leq n$ :
        $\bar{c}_i' = \bar{c}_i + F(a, i + 1)$
    return $(\tilde{c}', \bar{c}')$
```

Figure IV.8: ReCrypt [EPRS17] with compulsory Ciphertext Origin Authentication.

who is additionally retaining some number of bits of other values.

To demonstrate how proofs become more complex for more fine-grained choices of $\overline{\lambda}$, we now show that our scheme in Figure IV.5 is also best-achievable unidirectional in this stricter sense, on the condition that the prime p chosen by the Setup algorithm is a Mersenne prime, which is to say that $p = 2^n - 1$ for some n. To prove that our scheme is best-achievable unidirectional, we need the following lemma, after which best-achievable unidirectionality follows from a trivial adaptation of Lemma IV.2.

Lemma IV.4. *Let $p = 2^n - 1$ be a (Mersenne) prime and let $a, b \xleftarrow{\$} \mathbb{Z}_p^*$. Then, for all* PPT *algorithms $\mathcal{B}$ there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that:*

$$\Pr\left[\mathcal{B}(g^a, g^b, [g^{ab}]_{\overline{\lambda}}) \to g^{ab}\right] \leq \frac{1}{2^{n-\overline{\lambda}}} + \mathsf{negl}(\lambda)\,, \qquad \text{(IV.7)}$$

where $[g^{ab}]_{\overline{\lambda}}$ denotes the $\overline{\lambda}$ known bits of g^{ab}.

Proof. Since p is a Mersenne prime it has the form $2^n - 1$, and so is n bits long and there is a bijection between integers in $\mathbb{Z}_p$ and bitstrings of length n. In other words every integer in $\mathbb{Z}_p$ can be represented as a bitstring. Therefore random elements of $\mathbb{Z}_p$ can be modelled as random n-bit strings. More specifically, elements g^a, where $a \xleftarrow{\$} \mathbb{Z}_p^*$, can be considered as random n-bit strings.

We proceed by induction.

Base case: $\overline{\lambda} = 0$. By the CDH assumption, the adversary cannot compute g^{ab} with probability significantly greater than $\frac{1}{2^n}$.

Assume for $\overline{\lambda} = i$. An adversary who knows the first i bits of g^{ab} cannot calculate the remaining $n - i$ bits with probability higher than $\frac{1}{2^{n-i}}$.

We now consider a variant of the DDH game which factors in the adversary's knowledge of i bits of g^{ab}. We denote by $[g^{ab}]_{\overline{\lambda}}$ the $\overline{\lambda}$ known bits of g^{ab}. Let a, b be selected uniformly at random and let c be selected at random with the condition that $[g^c]_i = [g^{ab}]_i$: $a, b \xleftarrow{\$} \mathbb{Z}_p^*, c \xleftarrow{\$} \{x \in \mathbb{Z}_p^* : [g^x]_i = [g^{ab}]_i\}$. A consequence of assuming that our hypothesis is correct for i is that an adversary cannot distinguish between (g^a, g^b, g^{ab}) and (g^a, g^b, g^c) with non-negligible probability. If our assumption was false, the adversary would have an advantage in this game with probability $> \frac{1}{2^{n-i}}$.

Show for $\overline{\lambda} = i + 1$. For $a, b \xleftarrow{\$} \mathbb{Z}_p^*, c \xleftarrow{\$} \{x \in \mathbb{Z}_p^* : [g^x]_i = [g^{ab}]_i\}$, we have that $[g^c]_{i+1} \neq [g^{ab}]_{i+1}$ with probability $\frac{1}{2}$. In other words, the bit which the adversary is trying to predict is going to be different from g^{ab} in g^c with probability $\frac{1}{2}$. Suppose for a contradiction that the adversary has a non-negligible advantage in distinguishing (g^a, g^b, g^{ab}) from (g^a, g^b, g^c). This would mean an adversary has an advantage of winning the variant of DDH in case i half the time, whenever $[g^c]_{i+1} \neq [g^{ab}]_{i+1}$. This makes the assumption in case i false. Equivalently, the adversary would have an advantage of winning DDH with probability $\frac{1}{2^i}$. Therefore, by contradiction, an adversary has no significant advantage calculating another bit of g^{ab} over guessing. Overall, the adversary cannot compute the remaining bits g^{ab} with probability significantly greater than $\frac{1}{2^{n-i-1}}$.

We conclude that knowledge of the first $\overline{\lambda}$ bits of g^{ab} in addition to (g^a, g^b) does not help the adversary compute the remaining bits of g^{ab} with probability greater than $\frac{1}{2^{n-\overline{\lambda}}}$ when p is a Mersenne prime. $\square$

IV: 7 Summary

We revisited the notion of unidirectionality in PRE schemes and provided a formal security definition that covers reversal attacks. We have shown how, under this new definition, existing PRE schemes which are considered traditionally bidirectional may be considered unidirectional, and vice versa. We also outlined properties that a PRE scheme needs in order to be considered secure in the malicious model, in particular defining token robustness — the inability of the server to forge update tokens. Finally, we introduced a new notion of COA for authenticated PRE, discussed how to implement this and gave schemes meeting these definitions.

Chapter V

Post-Compromise Secure PRE

Contents

In this Chapter

In Section III: 5 we discussed the application of using PRE for access revocation, or as a key rotation mechanism to enforce key life cycles. We used the key encapsulation mechanism to demonstrate that if re-encryption does not fully re-randomise a ciphertext, particularly the part of the ciphertext that contains the message (as is the case for a number of schemes in the literature), then it may be possible for parties who have the original ciphertext and the old secret key to derive the message from a re-encrypted ciphertext. In Section IV: 2.3 we gave one definition that implies the ciphertext is re-randomised upon re-encryption, but did not discuss existing approaches to this, or implications of re-randomisation in detail. In this chapter, we seek to create a strong definition which implies that only the new secret key must remain uncompromised for the re-encrypted ciphertext to hide the message. Our overall aim is to define a security property such that schemes with this property can be used for access revocation and key expiry, and provide a fitting construction.

*The contributions of this chapter are joint work with Alex Davidson, Amit Deo and Keith M. Martin. My personal contribution includes the definition of Post-Compromise Security for PRE and its analysis, and the construction (*BV-PRE*) that meets the definition. Amit Deo provided the variant of* BV-PRE *that is source-hiding and its analysis. Elements of this chapter were published at ACISP 2019 [DDLM19a], including our definition of PCS, its analysis and (*BV-PRE*. A full version of the published work which includes the proofs is published on the Cryptology ePrint Archive [DDLM19b].*

Motivation for exploring Post-Compromise Secure PRE was provided by Katriel Cohn-Gordon.

V: 1 Introduction

For PRE to be useful in enforcing key rotation and access revocation, we would need corruption of past keys to not impact the security of re-encrypted ciphertexts. We are inspired by the work of Cohn-Gordon et al. [CCG16] who define Post-Compromise Security (PCS) for messaging protocols. The high-level definition of PCS is as follows:

Definition V.1 ([CCG16])**.** *'A protocol between Alice and Bob provides* Post-Compromise Security (PCS) *if Alice has a security guarantee about communication with Bob, even if Bob's secrets have already been compromised.'*

Note that PCS differs from *forward security*, which conveys that the compromise of future states does not affect the security of past ones. PCS conveys whether a cryptographic scheme can regain security after a session or party has been compromised. The difference can be seen visually in Figures V.1 and V.2. A definition of PCS for PRE schemes is clearly required in order to indicate that such schemes are appropriate for access revocation and key rotation.

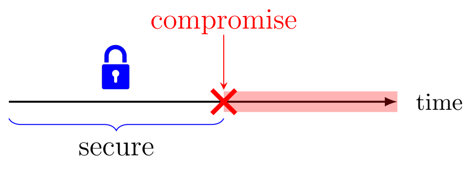

Figure V.1: Forward security implies states *before* compromise remain secure.

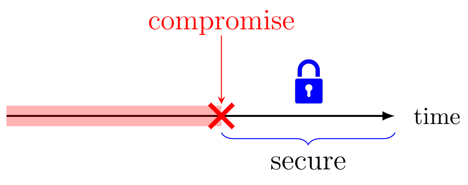

Figure V.2: Post-compromise security implies states *after* compromise are secure.

As Definition V.1 only provides the bare intuition, the 'security guarantee' is left open so that it can be tailored to the application, which is what we will do for PCS in PRE in this chapter.

Motivation for PCS PRE. One particular application of post-compromise PRE of interest is to complement PCS of messages in transit, by giving PCS to backed-up messages stored in the cloud. The Signal Protocol for encrypting messages in transit between two devices provides strong, modern, provable security guarantees including PCS [CCG16]. However, most users expect to keep their messages when they lose their device or buy a new one; thus, popular Signal implementations such as WhatsApp back up client messages to public cloud services. Unfortunately, this backup is encrypted using a static encryption key. This means that while messages in transit have PCS, these properties are lost once messages are backed up. If an adversary compromises a device and obtains the static cloud backup key, they can retain this to compromise future messages once they are backed up.

Assuming that all message history is stored locally, the updated message history could be encrypted under a new key and re-uploaded at regular time intervals, but this will have a huge cost both in terms of computation and bandwidth, particularly as much messaging is done via smartphones. A PRE scheme with PCS could be used instead, so that the PCS of messages in transit is extended to message backups.

Another use of PCS PRE is to revoke access from subscribers. For example, data stored by third parties may be subject to contractual agreements, as is the case with streaming services such as Netflix, who buy the rights to share content owned by production companies. The streaming services make the content available to subscribers only for the period of time indicated in a license, and must therefore ensure that this content cannot be accessed either by non-subscribers, or subscribers residing in a geographical region not specified by the license. They also must ensure that, as much as possible, content cannot be permanently downloaded and thereby stolen by a malicious subscriber.

V: 1.1 Contributions

In this chapter, we set out the first formalisation of PCS for PRE schemes. In our model, the adversary cannot distinguish which of two ciphertexts has been re-encrypted given the old key, old ciphertexts and the token used to perform the re-encryption. In other words, we view a compromise as the loss of all previous public and secret states associated with a given ciphertext, and limit the information that must remain secret to the current secret key alone. To date there is no related security definition that gives the adversary the update token used in the challenge re-encryption. Our definition implies unidirectionality as opposed to treating unidirectional and bidirectional schemes differently, and that additional randomness beyond that given in the update token is added upon re-encryption. Since we do not make as many assumptions concerning which algorithms are deterministic or on the flow of re-encryption operations, our security model can be applied to more general PRE schemes and applications than similar definitions in the literature (see Section V: 2).

We analyse our model, proving several results that associate PCS with existing security models for PRE and related primitives such as updatable encryption [LT18], and provide separating examples that distinguish PCS as a separate security characteristic in its own right. We demonstrate in Theorems V.1 and V.2 conditions under which schemes that have PCS are also IND-HRA-secure. As these additional properties are common in practical PRE schemes, this means that proving PCS is sufficient for proving that such schemes provide message confidentiality. One of our major contributions in Theorem V.3 is to show that a PRE scheme that is both *source-hiding* and secure against chosen plaintext attacks also has PCS, meaning one of the six PRE schemes presented by Fuchsbauer et al. [FKKP18] has PCS.

However, source-hiding PRE schemes were designed with different goals in mind (namely security with adaptive key corruptions, which we discuss in Chapter VI) and do not necessarily yield the most practical lattice-based PREs with PCS. We therefore give a new PRE scheme, pcBV-PRE, adapted from BV-PRE– the practical scheme of Polyakov et al. [PRSV17] whose security depends on the RLWE assumption. We prove that our adaptation achieves PCS and is IND-HRA-secure via a tighter reduction than using the combination of properties previously mentioned, meaning that source-hiding is sufficient, but not necessary for achieving PCS.

The advantage of this is that we can prove security for general re-encryption graphs, as opposed to only acyclic graphs as in [PRSV17], or only trees and chains as in [FKKP19]. Moreover, the other existing lattice-based source-hiding PRE scheme [FKKP19, Construction 2.b] ([FKKP18, Construction 7.b]) is forced into sub-optimal parameter choices that render their assumptions much stronger (with respect to the approximation factors of solving worst-case lattice problems), and much less efficient, and dramatically limits the number of times a ciphertext can be re-encrypted. Our new scheme is much more efficient than [FKKP19, Construction 2.b] since we leverage the speed of BV-PRE with minimal changes.

Chapter structure. We begin by reviewing related work in Section V: 2. In Section V: 3 we define Post-Compromise Security (PCS) and show how our definition relates to other definitions in the literature, giving separating examples and security reductions. In Section V: 4, we give an explicit, efficient construction, pcBV-PRE, by modifying the lattice-based PRE scheme BV-PRE [PRSV17] to show that our notion of PCS can be satisfied by natural extensions of current practical PRE schemes. We give an analysis of pcBV-PRE in Section V: 5 and discuss alternative techniques for proving PCS that do not rely on source-hiding.

V: 2 Related work

In this section, we explore three main areas – re-encryption security definitions that model re-randomisation (Section V: 2.1), results that demonstrate how to prove IND-HRA security (Section V: 2.2), and finally BV-PRE which we use as a basis for our construction (Section V: 2.3).

V: 2.1 Ciphertext re-randomisation

As explained in Section III: 4.2, to effectively model security after key revocation, we need a definition which implies that re-encryption updates all components of the ciphertext. In particular, we aim for a scheme which adds new randomness during re-encryption as a means of preventing successful reversal attacks, so that the re-encrypted ciphertext is as secure as a fresh encryption under the new key. We refer to this as *re-randomisation.*

There is little work on requiring re-encryption to also re-randomise. In Section IV: 4.2 we discussed defining the inability to regain the original ciphertext by considering an adversary who retains some information about the ciphertext prior to re-encryption. However, this approach does not account for the possibility that multiple revoked users store parts of the original ciphertexts and collude, as is a concern in [CCV12, Coh19, PRSV17]. Modelling these possibilities leads to more complicated proofs than alternative approaches, as we saw in Section IV: 6.3.

One such alternative is to take the same approach as for ReEnc-IND-CPA-security (Definition IV.4), where the adversary must guess which of two ciphertexts has been re-encrypted. For example, in [CH07], Canetti and Hohenberger discuss a notion for re-randomising ciphertexts (which they call *unlinkability*) in the context of creating a CCA definition. More recent work discussing re-randomisation is in the related areas of key rotation and updatable encryption (see Appendix A). Recall that these differ from PRE in that they are symmetric schemes, updates happen sequentially from k_i to k_{i+1}, and they name full re-randomisation as a necessary security goal.

In their paper on key rotation, Everspaugh et al. defined UP-REENC [EPRS17], another notion that implies full re-randomisation upon re-encryption. Interestingly, similarly to the ReEnc-IND-CPA experiment $\mathsf{ReEncIND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(\lambda)$ (Section IV: 2.3), their challenge oracle takes key indices as input but no update token, meaning that unless $\mathsf{ReKeyGen}$ is deterministic, the adversary never learns the update token used to perform the encryption. This is crucial to their security proof as their construction, $\mathsf{ReCrypt}$ (Figure IV.1), allows an adversary who has learned the update token to reverse the re-encryption of the challenge ciphertext and therefore does not cover full compromise of the user who generated the update token. In Chapter IV, we argued that unidirectionality and compromise of update tokens is an important factor to consider in PRE. We therefore aim for a strong notion of PCS which implies that only the new secret key must remain uncompromised for the re-encrypted ciphertext to be secure.

PCS for Updatable Encryption. A particular work of note is [LT18], where Lehmann and Tackmann define IND-UPD – a form of PCS for *updatable encryption.* This is similar in nature to $\mathsf{IND\text{-}HRA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ (Definition III.8), except that instead of the challenge being an encryption of either m_0 or m_1, it is a re-encryption of either c_0 or c_1 as in $\mathsf{ReEncIND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(\lambda)$.

A main difference in security for updatable encryption is that instead of re-encrypting to arbitrary keys, IND-UPD considers *epochs* where keys are updated sequentially, meaning the resulting re-encryption graph will be a chain (or group of chains). We describe a selective variant of the IND-UPD experiment in Figure V.3. Note that we adapt the notation and syntax used in [LT18] to that introduced in Chapter III, but that our formulation is equivalent to the one given in [LT18]. IND-UPD is modelled using an oracle $\mathsf{O}_{\mathsf{Next}}$ which updates the global state by incrementing the epoch number e, generating a key k_e for the new epoch and the update token $\Delta_{e-1,e}$ from the previous epoch to the current one. $\mathsf{O}_{\mathsf{Enc}}$ only encrypts under the current epoch key k_e, and $\mathsf{O}_{\mathsf{ReEnc}}$ only re-encrypts from the previous epoch to the current one. Unlike the definitions we have given so far, update tokens and re-encryptions of the challenge are generated by $\mathsf{O}_{\mathsf{Next}}$ and only learned by the adversary when it calls $\mathsf{O}_{\mathsf{LearnTok}}$ or $\mathsf{O}_{\mathsf{LearnChal}}$ respectively.

Definition V.2. *An updatable encryption scheme $\mathcal{UE}$ is* ϵ-sIND-UPD-secure (selectively ϵ-IND-UPD-secure) *if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:*

$$\Pr\left[\mathsf{sIND\text{-}UPD}_{\mathcal{A}}^{\mathcal{UE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon,$$

where $\mathsf{sIND\text{-}UPD}_{\mathcal{A}}^{\mathcal{UE}}(1^\lambda)$ *is defined in Figure V.3.*

If ϵ is negligible as parameterised by λ, then we say the scheme is (selectively) IND-UPD-secure.

Whilst IND-UPD is defined for updatable encryption schemes, it can easily be applied to symmetric PRE schemes when restricted to applications where the resulting encryption graph will be a chain. We give $\mathsf{pksIND\text{-}UPD}$, a simple adaptation of $\mathsf{sIND\text{-}UPD}$ for public-key schemes, in Appendix D.

We now make several observations on some of the conditions imposed in $\mathsf{sIND\text{-}UPD}_{\mathcal{A}}^{\mathcal{UE}}(1^\lambda)$:

- *IND-UPD-security does not imply that ciphertexts are re-randomised upon re-encryption.* This is because of the condition in $\mathsf{O}_{\mathsf{ReEnc}}$ that if ReEnc is deterministic, the adversary cannot have re-encrypted either of the potential challenge ciphertexts $\bar{c}_0, \bar{c}_1$ chosen by $\mathcal{A}_0$.
- *The adversary cannot learn the update token Δ_{e^*-1,e^*} leading to the epoch e^* where the challenge is created.* This is enforced as a condition in $\mathsf{O}_{\mathsf{LearnTok}}$.
- *An updatable encryption scheme can be bidirectional and still be IND-UPD.* This is enforced by the challenge graph as bidirectionality adds an undirected edge to the graph in $\mathsf{O}_{\mathsf{LearnTok}}$, meaning the adversary cannot have learned any update tokens

```
$\mathsf{sIND\text{-}UPD}^{\mathcal{UE}}_{\mathcal{A}}(1^\lambda)$
    $b \xleftarrow{\$} \{0,1\}$
    $e := 0, k_0 \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda), e^* := \bot, \mathcal{DRG} := \{v_0\}$
    $\mathcal{K}_{\mathsf{corrupt}}, \mathcal{K}_{\mathsf{chal}}, \mathcal{C}_{\mathsf{honest}}, \mathcal{C}_{\mathsf{chal}} = \emptyset$
    $(c_0, c_1, state) \xleftarrow{(\$)} \mathcal{A}_0^{\mathsf{O_{Enc}}, \mathsf{O_{Next}}, \mathsf{O_{ReEnc}}, \mathsf{O_{Corrupt}}, \mathsf{O_{LearnTok}}}(1^\lambda)$
    if $\big((e, c_0) \notin \mathcal{C}_{\mathsf{honest}}\big) \vee \big((e, c_1) \notin \mathcal{C}_{\mathsf{honest}}\big) \vee (|c_0| \neq |c_1|)$ :
        return $\bot$
    $e^* := e, \bar{c}_0 := c_0, \bar{c}_1 := c_1$
    $c_e^* \xleftarrow{(\$)} \mathsf{ReEnc}(\Delta_{e-1,e}, c_b)$
    $\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\{(e, c_e^*)\}, \mathcal{K}_{\mathsf{chal}}.\mathsf{add}\{k_e\}$
    $b' \xleftarrow{(\$)} \mathcal{A}_1^{\mathsf{O_{Enc}}, \mathsf{O_{Next}}, \mathsf{O_{ReEnc}}, \mathsf{O_{LearnTok}}, \mathsf{O_{ReEnc}}}(1^\lambda, state)$
    $\mathcal{K}_{\mathsf{chal}} \leftarrow \mathsf{UpdateChallengeKeys}(\mathcal{K}_{\mathsf{chal}}, \mathcal{DRG})$
    if $\mathcal{K}_{\mathsf{chal}} \cap \mathcal{K}_{\mathsf{corrupt}} \neq \emptyset$ : return $\bot$
    return $b' = b$

$\mathsf{O_{Enc}}(m) \xrightarrow{\$} c_e$
    $c_e \xleftarrow{\$} \mathsf{Enc}(k_e, m)$
    $\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(e, c_e)\}$
    return $c_e$

$\mathsf{O_{Next}}()$
    $e \leftarrow e + 1$
    $\mathcal{DRG}.\mathsf{add}\{v_e\}$
    $k_e \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
    $\Delta_{e-1,e} \leftarrow \mathsf{ReKeyGen}(k_{e-1}, k_e)$
    if $e^* \neq \bot$ :
        $c_e^* \leftarrow \mathsf{ReEnc}(\Delta_{e-1,e}, c_{e-1}^*)$
        $\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\{(e, c_e^*)\}$

$\mathsf{O_{ReEnc}}(c_{e-1}) \rightarrow c_e$
    if $(e-1, c_{e-1}) \notin \mathcal{C}_{\mathsf{honest}}$ :
        return $\bot$
    if (ReEnc is deterministic) $\wedge$ $(e = e^*)$ :
        if $(c_{e-1} = \bar{c}_0) \vee (c_{e-1} = \bar{c}_1)$ :
            return $\bot$
    $c_e \leftarrow \mathsf{ReEnc}(\Delta_{e-1,e}, c_{e-1})$
    $\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(e, c_e)\}$
    return $c_e$

$\mathsf{O_{Corrupt}}(i) \rightarrow k_i$
    $\mathcal{K}_{\mathsf{corrupt}}.\mathsf{add}\{k_i\}$
    return $k_i$

$\mathsf{O_{LearnTok}}(i) \rightarrow \Delta_{i-1,i}$
    if $(i > e) \vee (i = e^*)$ : return $\bot$
    if $\mathcal{PRE}$ is unidirectional :
        $\mathcal{DRG}.\mathsf{add}\{\overrightarrow{e}_{i-1,i}\}$ // directed edge
    else ($\mathcal{PRE}$ is bidirectional) :
        $\mathcal{DRG}.\mathsf{add}\{e_{i-1,i}\}$ // undirected edge
    return $\Delta_{i-1,i}$

$\mathsf{O_{LearnChal}}() \xrightarrow{(\$)} c_e^*$
    $\mathcal{K}_{\mathsf{chal}}.\mathsf{add}\{k_e\}$
    return $c_e^*$
```

Figure V.3: A selective variant of IND-UPD [LT18] (with adapted notation). The adversary needs to determine which of two ciphertexts was re-encrypted. e denotes the current epoch, e^* denotes the epoch where the challenge is called, and c_i^* denotes the challenge ciphertext under k_i.

used to create a challenge ciphertext.

We will address these points when we build our own definition in Section V: 3. We provide separating examples in Section V: 3.3 to demonstrate that our definition of PCS is stronger.

V: 2.2 Proving HRA security from CPA

We note that, as Cohen points out in his motivation of IND-HRA security (Definition III.8), re-encryption schemes where re-encryption appends an encryption of the old secret key using the new secret key to the ciphertext ($\Delta_{i,j} \overset{\$}{\leftarrow} \mathsf{Enc}(\mathsf{pk}_j, \mathsf{sk}_i)$, $c' \leftarrow (c, \Delta_{i,j})$) are considered secure in the CPA setting. This is a result of the strict separation of keys meaning that no re-encryptions from challenge keys to corrupted ones can be made, as we depicted in Figure III.5. Such constructions do not meet our intuition of PCS as we need to consider the compromise of keys that were once used for the challenge ciphertext. We also note that honest re-encryptions are considered in IND-UPD, although this terminology is not used in [LT18]. For these reasons, we choose to create a definition of PCS with honest re-encryptions.

As we choose to allow for honest re-encryptions in our definition of PCS, existing work that proves HRA security from CPA security with additional properties is relevant to our work. There are a few properties defined in the literature that a PRE scheme can have that lift the scheme from being CPA secure to being HRA secure. One such property that was defined by Cohen is *re-encryption simulatability.*

Definition V.3. *[Coh19, Definition 7]. A PRE scheme $\mathcal{PRE}$ is* re-encryption simulatable *if there exists a* PPT *algorithm* ReEncSim *such that with high probability over* aux*, for all* $m \in \mathcal{M}$*:*

$$(\mathsf{ReEncSim}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{sk}_j, c, m), \mathsf{aux}) \approx_s (\mathsf{ReEnc}(\Delta_{i,j}, c), \mathsf{aux}),$$

where $\approx_s$ denotes statistical indistinguishability, and values are sampled according to

$$(\mathsf{pk}_i, \mathsf{sk}_i) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$$
$$(\mathsf{pk}_j, \mathsf{sk}_j) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$$
$$\Delta_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$$
$$c \overset{\$}{\leftarrow} \mathsf{Enc}(\mathsf{pk}_i, m)$$
$$\mathsf{aux} = (\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{sk}_j, c, m).$$

Cohen outlines a theorem stating that PRE-IND-CPA-secure PRE schemes that have this property are IND-HRA-secure [Coh19, Theorem 5]. The proof shows how an adversary $\mathcal{A}_{\mathsf{CPA}}$ in $\mathsf{PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ can simulate $\mathsf{IND\text{-}HRA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ by replacing the re-encryptions that cannot be made in $\mathsf{PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ with outputs of $\mathsf{ReEncSim}$. Re-encryption simulatability implies that this simulated game and the real game are indistinguishable, and so $\mathcal{A}_{\mathsf{CPA}}$ can use a successful adversary $\mathcal{A}_{\mathsf{HRA}}$ in $\mathsf{IND\text{-}HRA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ to win $\mathsf{PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$.

Fuchsbauer et al. [FKKP19] expand on Cohen's work in defining *source-hiding* as part of their work on proving security with adaptive key corruptions. We note that the definition of a PRE scheme given in [FKKP19] has explicit levelling, where the encryption function takes a level ℓ for the ciphertext to be encrypted at. Informally, in a source-hiding scheme, a re-encrypted ciphertext is indistinguishable from a fresh encryption of the same underlying message using the same key, even given both key pairs, the old ciphertext and the update token and the message. This means re-encrypted ciphertexts reveal no history as to the keys they were previously encrypted under, or similarities between components of previous ciphertexts. We give a formal description of the game defining the source-hiding property in Figure V.4.

```
$\mathsf{SH}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$
$b \xleftarrow{\$} \{0,1\}$
$\mathsf{called} := \mathsf{false}$
$(\mathsf{pk}_0, \mathsf{sk}_0) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$(\mathsf{pk}_1, \mathsf{sk}_1) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$\Delta_{0,1} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_0, \mathsf{pk}_1)$
$b' \xleftarrow{(\$)} \mathcal{A}^{\mathsf{Chal}^{\mathsf{SH}}_{\mathsf{ReEnc/Enc}}}(1^\lambda, (\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1), \Delta_{0,1})$
return $b'$
```

```
$\mathsf{Chal}^{\mathsf{SH}}_{\mathsf{ReEnc/Enc}}(m^*, \ell^*) \xrightarrow{(\$)} (c^*, c'^{(b)})$
if $\mathsf{called} \vee (\ell^* + 1 > L)$ :
    return $\perp$
$c^* \leftarrow \mathsf{Enc}(\mathsf{pk}_0, m^*, \ell^*)$
$c'^{(0)} \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{0,1}, c^*)$
$c'^{(1)} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_1, m^*, \ell^* + 1)$
$\mathsf{called} \leftarrow \mathsf{true}$
return $(c^*, c'^{(b)})$
```

Figure V.4: Experiments for the source-hiding property. Here, ℓ denotes a *level* for the ciphertext to be encrypted at. L is the correctness bound (the number of times a ciphertext can be re-encrypted without breaking the correctness conditions).

Definition V.4. *[FKKP19, Definition 9]. A PRE scheme $\mathcal{PRE}$ is said to be* ϵ-source-hiding *if for all* PPT *adversaries $\mathcal{A}$:*

$$\Pr\left[\mathsf{SH}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon$$

where $\mathsf{SH}^{\mathcal{PRE}}_{\mathcal{A}}(1^{\lambda})$ *is defined in Figure V.4.*

If ϵ *is negligible as parameterised by* λ*, then we say the scheme is* source-hiding (SH).

We describe a simple variant of source-hiding for symmetric PRE schemes in Appendix C for easier use with symmetric re-encryption primitives.

Fuchsbauer et al. also define *weak key privacy* and show PRE schemes with weak key privacy that are source-hiding and IND-CPA-secure in the traditional sense (Definition II.7) are also secure against adaptive HRAs, with sub-exponential loss for some re-encryption graphs. More specifically, they show quasi-polynomial loss in security when lifting to adaptive corruption for trees and chains, and exponential loss otherwise. The loss is due to the fact that the definition of source-hiding is single-challenge and has only two key pairs. This means the simulator needs to guess in advance which keys the adversary will ask for an honest re-encryption of, as well as which ciphertext will be re-encrypted so that the simulator can integrate the source-hiding challenge into the right place. This only works if they guess correctly, which leads to the loss in security. We use and discuss this technique in Section V: 3.4.

V: 2.3 BV-PRE

Finally, we discuss BV-PRE [PRSV17], the PRE scheme that forms the basis for our own construction. Recall that lattice-based cryptography provides a means of creating a unidirectional, multi-hop PRE scheme with transparency. Unlike unidirectional ElGamal-based schemes, unidirectional lattice-based schemes to not need to rely on bilinear pairings, which typically result in single-hop schemes, as discussed in Section III: 4.In 2017, Polyakov, Rohloff, Sahu and Vaikuntanthan presented BV-PRE [PRSV17], a unidirectional, multi-hop PRE scheme with transparency based on BV encryption (see Section II: 4.2). They also present an efficient software implementation for BV-PRE.

However, Polyakov et al. assume a different workflow in their definition of a PRE scheme, introducing additional parties. This is partly because they define PRE specifically in the context of publish/subscribe systems, where PRE is used to re-encrypt ciphertexts under the publisher's public key to ciphertexts under the subscriber's public key. The parties and workflow assumed in [PRSV17] are as follows:

- A *policy authority* generates a key pair for the publisher, Alice, and sends her the public key.

- When Alice has created her subscribe-able content, she encrypts it then stores it with a *PRE server*.
- When Bob subscribes, he sends his public key to the policy authority, who uses it to create an update token $\Delta_{A,B}$ from Alice's key to Bob's, which is sent straight to the PRE server.
- The PRE server then re-encrypts Alice's ciphertext before forwarding the re-encrypted ciphertext to Bob, who can then access the content.

In this workflow, it is not necessarily the case that the public keys used for generating update tokens are also used for encryption. We describe $\mathsf{BV\text{-}PRE}$ in Figure V.5.

This construction uses the key switching technique described in [BV11] to reduce error growth. This technique re-linearises a ciphertext by splitting it into parts before processing, as determined by a relinearisation window r:

$$c = \sum_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} c_\iota \cdot (2^r)^\iota. \tag{V.1}$$

It is important to minimise the size of the error added upon re-encryption, as if the error gets too large then the message will not decrypt correctly. Therefore, error growth directly affects the correctness bound, as we shall demonstrate for our scheme, $\mathsf{pcBV\text{-}PRE}$, in Section V: 5.1.

$\mathsf{BV\text{-}PRE}$ uses the algorithm $\mathsf{Preprocess}$ to output the 'public'[1] key 'pk_j' used for re-encryption. We note that, given $\Delta_{i,j} = \{(\beta_\iota, \gamma_\iota)\}_{\iota=0}^{\lfloor \log_2(q)/r \rfloor}$ and 'pk_j', it is possible to calculate sk_i by computing

$$s_\iota \cdot (2^r)^\iota = \theta_\iota - \gamma_\iota$$

for $\iota \in \{0, 1, \ldots, \lfloor \log_2(q)/r \rfloor\}$. Therefore, the 'public' key used to create update tokens cannot actually be made public. It is this fact that Cohen uses to prove that $\mathsf{BV\text{-}PRE}$ is not IND-HRA-secure [Coh19, Theorem 4].

However, Polyakov et al. created a fast, optimised implementation of $\mathsf{BV\text{-}PRE}$, which has a lower time and space complexity than existing lattice-based PRE schemes. They

[1]This key is also referred to as 'public' in [PRSV17], acknowledging that it cannot actually be made public.

```
$\mathsf{Setup}(1^\lambda) \to$ params
    Choose positive integers $q, n$
    Choose plaintext modulus $p$
    Choose key switching window $r$
    return $pp = (q, n, p.r)$

$\mathsf{KeyGen}(1^\lambda) \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$
    $a \xleftarrow{\$} \mathcal{U}_q$
    $s, e \xleftarrow{\$} \chi_e$
    $b = a \cdot s + pe$
    $\mathsf{sk} = s, \mathsf{pk} = (a, b)$
    return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Enc}(\mathsf{pk}, m) \xrightarrow{\$} c$
    $(a, b) \leftarrow \mathsf{pk}$
    $v, e_0, e_1 \xleftarrow{\$} \chi_e$
    $c_0 = b \cdot v + pe_0 + m$
    $c_1 = a \cdot v + pe_1$
    return $c = (c_0, c_1)$

$\mathsf{Dec}(\mathsf{sk}, c) \to m$
    $s \leftarrow \mathsf{sk}$,
    $(c_0, c_1) \leftarrow c$
    $m' = c_0 - s \cdot c_1 \mod p$
    return $m'$

$\mathsf{Preprocess}(1^\lambda, \mathsf{sk}_j) \xrightarrow{\$}$ '$\mathsf{pk}_j$'
    $s_j \leftarrow \mathsf{sk}_j$
    for $\iota \in \{0, 1, \ldots, \lfloor \log_2(q)/r \rfloor\}$ :
        $\beta_\iota \xleftarrow{\$} \mathcal{U}_q$
        $e_\iota \xleftarrow{\$} \chi_e$
        $\theta_\iota = \beta_\iota \cdot s_j + pe_\iota$
    return '$\mathsf{pk}_j$' $= \{(\beta_\iota, \theta_\iota)\}_{\iota=0}^{\lfloor \log_2(q)/r \rfloor}$

$\mathsf{ReKeyGen}(\mathsf{sk}_i,$ '$\mathsf{pk}_j$'$) \to \Delta_{i,j}$
    $s_i \leftarrow \mathsf{sk}_i, \{(\beta_\iota, \theta_\iota)\}_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} \leftarrow$ '$\mathsf{pk}_j$'
    for $\iota \in \{0, 1, \ldots, \lfloor \log_2(q)/r \rfloor\}$ :
        $\gamma_\iota = \theta_\iota - s_\iota \cdot (2^r)^\iota$
    $\Delta_{i,j} = \{(\beta_\iota, \gamma_\iota)\}_{\iota=0}^{\lfloor \log_2(q)/r \rfloor}$
    return $\Delta_{i,j}$

$\mathsf{ReEnc}(\Delta_{i,j}, c) \to c'$
    $\{(\beta_\iota, \gamma_\iota)\}_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} \leftarrow \Delta_{i,j}$
    $(c_0, c_1) \leftarrow c$
    $c'_0 = c_0 + \sum_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} (c_1^{(\iota)} \cdot \gamma_\iota)$
    $c'_1 = \sum_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} (c_1^{(\iota)} \cdot \beta_\iota)$
    return $c' = (c'_0, c'_1)$
```

Figure V.5: BV-PRE [PRSV17], a PRE scheme based on the BV encryption scheme (Figure II.2). $\mathcal{U}_q$ is the discrete uniform distribution over $\mathcal{R}_q$ and χ_e is a B_e-bounded discrete Gaussian error distribution.

also make optimised parameter choices and discuss trade-offs, concluding that BV-PRE is suitable for use on low-resource embedded systems. We refer the interested reader to [PRSV17, Sections 7–9] for details. We note therefore that if BV-PRE can be made fully public-key and fit the traditional PRE workflow, then this will result in a desirable PRE scheme from a practical perspective.

V: 3 Strong Post-Compromise Security for PRE

In this section, we create a definition for strong PCS for public-key PRE schemes. We begin by justifying our motivation, explaining why we believe existing definitions are insufficient for PCS.

We have two main motivations for creating a new definition for PCS in the context of PRE. The first is that there is currently no definition that implies unidirectionality and ciphertext re-randomisation. We believe that the distinction of whether a PRE scheme is unidirectional or bidirectional is vital in the post-compromise scenario to model the corruption of used update tokens, and hence we define PCS to explicitly mean that schemes meeting the definition must be unidirectional. We note that there may be some applications where bidirectionality is preferred (such as using PRE as to share files between trusted parties), we aim for unidirectionality as this can be used for a larger number of applications, and allows us to model the compromise of the update tokens used to create a challenge re-encryption. The second motivation is a matter of existing definitions inherently assuming which of the algorithms in the PRE scheme are probabilistic, which introduces problems when considering a post-compromise scenario. We now discuss these in further detail.

Explicit unidirectionality. We use IND-UPD [LT18], the most relevant definition in the existing literature to make our case. IND-UPD places restrictions based on inferable information, as defined by the [LT18] notions of directionality:

- When sk_j cannot be derived from sk_i and $\Delta_{i,j}$ (LT-unidirectional)[2]
- When sk_j can be derived from sk_i and $\Delta_{i,j}$ (LT-bidirectional).

In the LT-unidirectional case, the adversary can acquire re-encryption tokens from a corrupted key to a challenge key, but not the other way around. In the LT-bidirectional case, the adversary is additionally prevented, by definition, from learning tokens from challenge keys to corrupted keys or vice versa. These preventions are necessary to enforce the trivial win condition. This means that for bidirectional schemes, the adversary queries tokens in such a way that the resulting re-encryption graphs form disjoint sub-graphs – one containing corrupted keys and the other containing challenge keys. We consider this too restrictive for the intuition of PCS.

Since the derivability of the new secret key is implicit information to the game, meeting the definition itself says nothing about directionality and therefore the extent to which

[2]The general understanding of unidirectionality is not so strong - the new key does not necessarily have to be derivable, but the token and old key should lead to the message being learned. We suspect that for symmetric PRE schemes, LT-unidirectionality is equivalent to the general understanding but we do not claim this to be true as we do not present any formal comparison in this thesis.

update token compromise is of concern. We believe that it is better for directionality to be explicit as it allows practitioners to understand the security of a scheme more clearly.

Assuming probabilistic algorithms. We use two existing security definitions which explicitly consider re-randomisation [EPRS17, LT18] as an example of the problems this creates. The [EPRS17] definition of a *key rotation scheme* assumes that $\mathsf{ReKeyGen}$ is randomised but ReEnc is deterministic. This leads to a necessary condition that the update token used to create the challenge re-encryption cannot be learned by the adversary, otherwise the adversary could use it to re-encrypt the input ciphertexts locally and compare this to the challenge to win the game. The adversary is allowed to learn other tokens going from corrupted to challenge keys using an oracle $\mathsf{O}_{\mathsf{ReKeyGen}}$, but not the exact token used to perform the re-encryption. This means, UP-REENC [EPRS17] does not model compromise of the update token used. To model compromise of the token used, as we aim to do, it is therefore important that new randomness is introduced in ReEnc to prevent trivial downgrading of the challenge ciphertext if the adversary compromises the update token used.

In the [LT18] definition of an updatable encryption scheme, the opposite assumption is made – that ReEnc is randomised and $\mathsf{ReKeyGen}$ is deterministic. This means that for keys $\mathsf{sk}_i, \mathsf{pk}_j$, there is only one update token $\Delta_{i,j}$. This is reflected in the $\mathsf{sIND\text{-}UPD}^{\mathcal{UE}}_{\mathcal{A}}(1^\lambda)$ security game (and $\mathsf{pksIND\text{-}UPD}^{\mathcal{UE}}_{\mathcal{A}}(1^\lambda)$) by having one update token $\Delta_{i-1,i}$ created when k_i is generated and later having oracles reveal tokens to the adversary. This is less fitting for schemes where $\mathsf{ReKeyGen}$ is randomised and there are multiple tokens per key pair[3]. More importantly, such an assumption implicitly rules out some methods for masking the secret key in the update token, which is important for PCS. $\mathsf{BV\text{-}PRE}$ [PRSV17] is an example of this, where knowledge of the key 'pk_j' together with $\Delta_{i,j}$ can be used to derive sk_i. Another example is ElGamal-based symmetric PRE schemes (e.g. [BBS98, LT18]) where update tokens have the form $\Delta_{i,j} = \mathsf{sk}_j/\mathsf{sk}_i$. Clearly, given the update token, compromise of the old key leads to compromise of the new key. Introducing some randomness gives a means of masking the new key, which is necessary for unidirectionality. It also means that the client does not need to fully rely on the proxy to be assured of new randomness, which is

[3]Interestingly, other works such as [FKKP19] take a similar approach to the adversary learning update tokens, despite their assuming randomised tokens.

more appropriate for some trust scenarios. As we aim for security against malicious proxies, we would prefer for the client to provide trustworthy new randomness for the re-encrypted ciphertext.

For constructions where randomness is added in both $\mathsf{ReKeyGen}$ and ReEnc, neither definition is suitable. It is therefore of interest to create a security notion for PCS which factors in the possibility that both the $\mathsf{ReKeyGen}$ and ReEnc algorithms are probabilistic.

V: 3.1 Post-Compromise Security

We model PCS using an adversary $\mathcal{A}$ who chooses two ciphertexts (whose decryption key can be known) and a re-encryption token, and receives a challenge ciphertext which is a re-encryption of one of the original ciphertexts created using the specified token. $\mathcal{A}$ attempts to distinguish which ciphertext was re-encrypted. This models the compromise of all key-related material prior to the challenge re-encryption.

Unlike some previous definitions, we allow $\mathcal{A}$ to corrupt the secret key with which the update token was generated. As in IND-HRA security, we also allow $\mathcal{A}$ to re-encrypt honestly (oracle)-generated non-challenge ciphertexts to corrupted keys. Challenges are obtained via the challenge oracle $\mathsf{Chal}^{\mathsf{PC}}_{\mathsf{ReEnc}}$ which will only accept as input honestly-generated ciphertexts of the same length, and honestly-generated update tokens. The challenger maintains the lists $\mathcal{C}_{\mathsf{honest}}$ and $\mathcal{T}_{\mathsf{honest}}$ to enforce this.

Here we present a formal definition of PCS with selective key corruptions for general PRE. The first stage adversary $\mathcal{A}_0$ can access a key generation oracle $\mathsf{O}_{\mathsf{KeyGen}}$ and key corruption oracle $\mathsf{O}_{\mathsf{Corrupt}}$ and the second stage adversary $\mathcal{A}_1$ can access the challenge oracle $\mathsf{Chal}^{\mathsf{PC}}_{\mathsf{ReEnc}}$ and re-encryption oracle $\mathsf{O}_{\mathsf{ReKeyGen}}$, and use these to adaptively determine the re-encryption graph. As in previous definitions, $\mathcal{K}_{\mathsf{corrupt}}$ and $\mathcal{C}_{\mathsf{chal}}$ are also maintained, which record corrupted keys and challenge ciphertexts, respectively, and are used to enforce the trivial win condition.

Definition V.5. *A PRE scheme $\mathcal{PRE}$ is said to have* (selective) ϵ-Post-Compromise Security (ϵ-PCS) *if for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*:*

$$\Pr\left[\mathsf{PostComp}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda) = 1\right] \le \frac{1}{2} + \epsilon,$$

```
$\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$
$b \xleftarrow{\$} \{0,1\}$
$\mathcal{K}_{\mathsf{corrupt}}, \mathcal{T}_{\mathsf{honest}}, \mathcal{C}_{\mathsf{honest}}, \mathcal{C}_{\mathsf{chal}} = \emptyset$
$\kappa := 0, \mathsf{called} \leftarrow \mathsf{false}$
$state \xleftarrow{(\$)} \mathcal{A}_0^{\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Corrupt}}}(1^\lambda)$
$b' \xleftarrow{(\$)} \mathcal{A}_1^{\mathsf{O}_{\mathsf{Enc}}, \mathsf{O}_{\mathsf{ReKeyGen}}, \mathsf{O}_{\mathsf{ReEnc}}^{\mathsf{PC}}, \mathsf{Chal}_{\mathsf{ReEnc}}^{\mathsf{PC}}}(1^\lambda, state)$
return $(b' = b)$
```

```
$\mathsf{O}_{\mathsf{KeyGen}}(1^\lambda) \xrightarrow{\$} \mathsf{pk}$
$\kappa = \kappa + 1$
$(\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
return $\mathsf{pk}_\kappa$
```

```
$\mathsf{O}_{\mathsf{Corrupt}}(i) \rightarrow \mathsf{sk}_i$
$\mathcal{K}_{\mathsf{corrupt}}.\mathsf{add}\{\mathsf{sk}_i\}$
return $\mathsf{sk}_i$
```

```
$\mathsf{O}_{\mathsf{Enc}}(i, m) \xrightarrow{\$} c$
$c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m)$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(i, c)\}$
return $c$
```

```
$\mathsf{O}_{\mathsf{ReKeyGen}}(i, j) \xrightarrow{(\$)} \Delta_{i,j}$
if $(\mathsf{sk}_i \notin \mathcal{K}_{\mathsf{corrupt}}) \wedge (\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}})$ :
  return $\perp$
$\Delta_{i,j} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$\mathcal{T}_{\mathsf{honest}}.\mathsf{add}\{i, j, \Delta_{i,j}\}$
return $\Delta_{i,j}$
```

```
$\mathsf{O}_{\mathsf{ReEnc}}^{\mathsf{PC}}(i, j, [\Delta_{i,j}], c)$
if $\big((i,c) \notin \mathcal{C}_{\mathsf{honest}}\big) \vee \big(((i,c) \in \mathcal{C}_{\mathsf{chal}}) \wedge (\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}})\big)$ :
  return $\perp$
if $((i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}})$ :
  $\Delta_{i,j} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$c' \xleftarrow{(\$)} \mathsf{ReEnc}(\Delta_{i,j}, c)$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(j, c')\}$
if $(i, c) \in \mathcal{C}_{\mathsf{chal}}$ :
  $\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\{(j, c')\}$
return $c'$
```

```
$\mathsf{Chal}_{\mathsf{ReEnc}}^{\mathsf{PC}}(i, j, [\Delta_{i,j},]\ c_0, c_1)$
if $\mathsf{called} \vee (\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}}) \vee (|c_0| \neq |c_1|)$ :
  return $\perp$
if $\big((i, c_0) \notin \mathcal{C}_{\mathsf{honest}}\big) \vee \big((i, c_1) \notin \mathcal{C}_{\mathsf{honest}}\big)$
  return $\perp$
if $((i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}})$ :
  $\Delta_{i,j} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$c'_0 \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, c_0)$
$c'_1 \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, c_1)$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(j, c'_b)\}, \mathcal{C}_{\mathsf{chal}}.\mathsf{add}\{(j, c'_b)\}$
$\mathsf{called} \leftarrow \mathsf{true}$
return $c'_b$
```

Figure V.6: The PCS game. This reflects full compromise of the old secret key and update token used to perform the re-encryption. Unlike $\mathsf{IND\text{-}HRA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$, there are no conditions that only apply to bidirectional schemes.

where $\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ *is defined in Figure V.6.*

If ϵ is negligible as parameterised by the security parameter, then we say the scheme has (selective) Post-Compromise Security (PCS)*.*

We give a definition of PCS for symmetric PRE schemes in Appendix C.

V: 3.2 Basic observations

We make some observations on what our definition of PCS implies. We note that, as we are modelling ciphertext independent schemes, it is sufficient to use the informal definition of unidirectionality.

Lemma V.1. *No PRE scheme where* ReEnc *is deterministic has PCS.*

Proof. If ReEnc is deterministic then $\mathcal{A}$ can submit $(c_0, c_1, i, j, \Delta_{i,j})$ to $\mathsf{Chal}^{\mathsf{PC}}_{\mathsf{ReEnc}}$ to learn a challenge c'. Then $\mathcal{A}$ can locally compute $c'_0 \leftarrow \mathsf{ReEnc}(\Delta_{i,j}, c_0)$ and compare this with c'. If they match then output $b' = 0$, otherwise output $b' = 1$. This approach wins with probability 1. □

Lemma V.2. *PCS implies unidirectionality.*

Proof. We show that if a scheme is bidirectional then it cannot have PCS. Bidirectionality implies that an update token $\Delta_{i,j}$, can be used to re-encrypt from pk_j to pk_i. The adversary $\mathcal{A}$ proceeds as follows: $\mathcal{A}$ first corrupts a key sk_i, then creates two ciphertexts $c_0 \xleftarrow{\$} \mathsf{O}_{\mathsf{Enc}}(i, m_0), c_1 \xleftarrow{\$} \mathsf{O}_{\mathsf{Enc}}(i, m_1)$ and update token $\Delta_{i,j} \xleftarrow{(\$)} \mathsf{O}_{\mathsf{ReKeyGen}}(i, j)$, before submitting $(c_0, c_1, i, j, \Delta_{i,j})$ to the challenge oracle, receiving a challenge c'^{*}_b. $\mathcal{A}$ then re-encrypts c'^{*}_b back to pk_i, and decrypts the result using sk_i to win the game. □

This means an easy way to disprove PCS for existing schemes is to either point out that ReEnc is deterministic or to show bidirectionality.

Lemma V.3. *PCS implies maximal irreversibility (Definition IV.8).*

Proof. Clearly, PCS implies the re-encryption cannot be reversed given the old ciphertext and update token, which involves retaining a state larger than the original ciphertext. □

V: 3.3 Separating examples

We now demonstrate the relationship between PCS and existing security notions and constructions by means of a number of separating examples. In this way, we demonstrate that PCS is stronger than existing similar notions. Here we use $\mathsf{pksIND\text{-}UPD}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$, the public-key variant of $\mathsf{sIND\text{-}UPD}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$ (see Appendix D), for more direct comparison, but note that similar claims can be made between $\mathsf{sIND\text{-}UPD}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$ and the symmetric variant of $\mathsf{PostComp}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$ described in Appendix C.

Lemma V.4. *pksIND-UPD-security does not imply PCS.*

Proof. Let $\mathcal{PRE}$ be a pksIND-UPD-secure PRE scheme where ReEnc is deterministic. By Lemma V.1, this scheme does not have PCS. □

Lemma V.5. *Let $\mathcal{PRE}$ be a PRE scheme where* $\mathsf{ReKeyGen}$ *is deterministic. If $\mathcal{PRE}$ has PCS, then it is pksIND-UPD-secure.*

Proof sketch. Assume κ key pairs are generated, and that t keys are corrupted by the end of the game. The $\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ adversary $\mathcal{A}$ can simulate $\mathsf{pksIND\text{-}UPD}_{\mathcal{B}}^{\mathcal{PRE}}(1^\lambda)$. As in $\mathsf{pksIND\text{-}UPD}_{\mathcal{B}}^{\mathcal{PRE}}(1^\lambda)$, $\mathcal{A}$ maintains an epoch counter e. Before the challenge is issued, $\mathcal{A}$ can simulate $\mathsf{O}_{\mathsf{Next}}$ by incrementing e, corrupting sk_{e-1} and obtaining $\mathsf{O}_{\mathsf{ReKeyGen}}(e-1,e)$ between the old key and the new. This will not affect the trivial win condition in $\mathsf{PostComp}$, as the corruption of keys and tokens leading to the challenge is permitted. When $\mathcal{B}_0$ ends, $\mathcal{A}_0$ sets $e^* := e$ before guessing which remaining keys with index $i > e^*$ that $\mathcal{B}_1$ will corrupt, which leads to a polynomial loss in security. $\mathcal{A}_1$ creates the challenge ciphertext by calling $c^*_{\tilde{e}} \overset{(\$)}{\leftarrow} \mathsf{Chal}^{\mathsf{PC}}_{\mathsf{ReEnc}}(c_0, c_1, \tilde{e}-1, \tilde{e})$. $\mathcal{A}_1$ can update both challenge and honest ciphertexts using $\mathsf{O}_{\mathsf{ReEnc}}$, and corrupting tokens can be simulated with calls to $\mathsf{O}_{\mathsf{ReKeyGen}}$.

We note that this is not a perfect simulation, as in $\mathsf{pksIND\text{-}UPD}_{\mathcal{B}}^{\mathcal{PRE}}(1^\lambda)$ the challenge ciphertext is encrypted through every key. However, in schemes which have been proven to be IND-UPD, re-encrypted ciphertexts are independent of the old secret key, and we conjecture that this is a requirement of IND-UPD, in which case the simulation is indistinguishable. We note that it is difficult to prove exact relations as the definitions are ultimately intended for different re-encryption primitives with different applications in mind. However, we present this comparison as IND-UPD is the most relevant definition with which to compare PCS PRE.

Lemma V.6. *IND-HRA-security $\not\Longrightarrow$ PCS.*

Proof. Let $\overline{\mathcal{PRE}} = (\overline{\mathsf{KeyGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}}, \overline{\mathsf{ReKeyGen}}, \overline{\mathsf{ReEnc}})$ be a IND-HRA-secure PRE scheme, and let $(\mathcal{SE}.\mathsf{KeyGen}, \mathcal{SE}.\mathsf{Enc}, \mathcal{SE}.\mathsf{Dec})$ be a IND-CPA-secure symmetric encryption scheme for the traditional notion of IND-CPA security. We now define a PRE scheme $\mathcal{PRE} = \{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc}\}$ using the key encapsulation method as follows:

- $\mathsf{KeyGen}(1^\lambda) \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk}) : (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \overline{\mathsf{KeyGen}}(1^\lambda)$

- $\mathsf{Enc}(\mathsf{pk}, m) \xrightarrow{\$} c : k \xleftarrow{\$} \mathcal{SE}.\mathsf{KeyGen}(1^\lambda), c_0 \xleftarrow{\$} \overline{\mathsf{Enc}}(\mathsf{pk}, k), c_1 \xleftarrow{\$} \mathcal{SE}.\mathsf{Enc}(k, m),\ c = (c_0, c_1)$
- $\mathsf{Dec}(\mathsf{sk}, c) \rightarrow m' : k' \leftarrow \mathsf{Dec}(\mathsf{sk}, c_0), m' \leftarrow \mathcal{SE}.\mathsf{Dec}(k', c_1)$
- $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) \xrightarrow{(\$)} \Delta_{i,j} : \Delta_{i,j} \xleftarrow{(\$)} \overline{\mathsf{ReKeyGen}}(\mathsf{sk}_i, \mathsf{pk}_j)$
- $\mathsf{ReEnc}(\Delta_{i,j}, c) \xrightarrow{(\$)} c' : c'_0 \xleftarrow{(\$)} \overline{\mathsf{ReEnc}}(\Delta_{i,j}, c_0),\ c' = (c'_0, c_1)$

This scheme is also IND-HRA-secure, but does not have PCS. An adversary $\mathcal{A}$ can submit two ciphertexts to the challenge oracle, and compare the second part of the challenge re-encryption with the submitted ciphertexts to win the game. □

Lemma V.7. *PCS $\not\Rightarrow$ PRE-IND-CPA security (or by extension IND-HRA security).*

Proof. Let $\overline{\mathcal{PRE}} = (\overline{\mathsf{KeyGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}}, \overline{\mathsf{ReKeyGen}}, \overline{\mathsf{ReEnc}})$ be a PRE scheme that is PRE-IND-CPA-secure and has PCS. We use it to construct the following PRE scheme:

- $\mathsf{KeyGen}(1^\lambda) \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk}) : (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \overline{\mathsf{KeyGen}}(1^\lambda)$
- $\mathsf{Enc}(\mathsf{pk}, m) \xrightarrow{\$} c : c_1 \xleftarrow{\$} \overline{\mathsf{Enc}}(\mathsf{pk}, m), c := (m, c_1)$
- $\mathsf{Dec}(\mathsf{sk}, c) \rightarrow m' : m' \leftarrow \overline{\mathsf{Dec}}(\mathsf{sk}, c_1)$
- $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) \xrightarrow{(\$)} \Delta_{i,j} : \Delta_{i,j} \xleftarrow{(\$)} \overline{\mathsf{ReKeyGen}}(\mathsf{sk}_i, \mathsf{pk}_j)$
- $\mathsf{ReEnc}(\Delta_{i,j}, c) \xrightarrow{(\$)} c' : c'_0 \xleftarrow{\$} \overline{\mathsf{Enc}}(\mathsf{pk}_j, 0), c'_1 \xleftarrow{(\$)} \overline{\mathsf{ReEnc}}(\Delta_{i,j}, c_1),\ c' = (c'_0, c'_1)$

Clearly this scheme is not PRE-IND-CPA-secure (or by extension IND-HRA-secure), as fresh ciphertexts contain the plaintext. However, the scheme has PCS, as re-encryptions c'_1 will be unrelated to c_1, since $\overline{\mathcal{PRE}}$ has PCS, and the creation of c'_0 is independent of both c_0 and $\Delta_{i,j}$. □

This counterexample relies on the underlying encryption scheme not being $\mathsf{IND\text{-}CPA}$, which is unlikely to be a concern in practice. We now present two theorems which suggest that many practical PRE schemes with PCS are also IND-HRA-secure.

Theorem V.1. *Let $\mathcal{PRE}$ be a PRE scheme with transparency. Then if $\mathcal{PRE}$ is ϵ-PCS, it is also ϵ-IND-HRA-secure.*

Proof. For a PRE scheme $\mathcal{PRE}$ with ϵ-PCS, no PPT adversary can guess the underlying message m_b of the challenge re-encryption $c_b'^*$ with advantage greater than ϵ, as this would lead to a win with advantage greater than ϵ. As the adversary knows the underlying messages of the challenge input ciphertexts c_0 and c_1 and has transparency, it follows that fresh encryptions of the same message under the same key are indistinguishable regardless of level, particularly level-0 ciphertexts (otherwise known as fresh ciphertexts). Therefore, no PPT adversary can distinguish a fresh encryption of m_b with advantage greater than ϵ, even given the ability to re-encrypt as described in $\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ and $\mathsf{IND\text{-}HRA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$. We conclude that $\mathcal{PRE}$ is also ϵ-IND-HRA-secure. $\square$

We note that the converse to Theorem V.1 does not hold – a PRE scheme with transparency that is IND-HRA-secure is not necessarily PCS. This is demonstrated by a similar example to the one in Lemma V.7, where $\overline{\mathcal{PRE}}$ also has transparency.

Theorem V.2. *Let $\mathcal{PRE}$ be a PRE scheme with PCS where the underlying encryption scheme is IND-CPA-secure (in the traditional PKE setting). Then $\mathcal{PRE}$ is also IND-HRA-secure.*

Proof. As in proof of Theorem V.1, we note that PCS implies that re-encryption does not reveal the underlying message, either in the re-encryption process or in th re-encrypted ciphertext. As fresh ciphertexts also do not reveal the underlying message, the result follows. $\square$

This means that, to prove that a PRE scheme with PCS is IND-HRA-secure, it is sufficient to demonstrate that either the underlying encryption scheme is IND-CPA-secure, or that the scheme has transparency. As transparency is desirable from an efficiency perspective and that there has been a recent trend in creating new schemes with transparency, we conjecture that this result will apply to a significant number of PRE schemes that provably have PCS.

V: 3.4 PCS via source-hiding

In this section we show that a PRE scheme that is both source-hiding and PRE-IND-CPA-secure also has PCS.

Theorem V.3 (main)**.** *Let $\mathcal{PRE}$ be a PRE scheme with transparency which is both PRE-IND-CPA-secure and source-hiding. Then $\mathcal{PRE}$ also has PCS.*

More specifically, if $\mathcal{PRE}$ is ϵ_1-PRE-IND-CPA-secure and ϵ_2-source-hiding, then it also has ϵ-PCS where

$$\epsilon \leq \epsilon_1 + \kappa(\kappa-1)(Q_E + Q_{HRE} + 1)(Q_{HRE} + 1) \cdot \epsilon_2 < \mathsf{negl}(\lambda)\,,$$

for some negligible function $\mathsf{negl}(\lambda)$, where Q_E is the number of queries $\mathcal{A}$ makes to $\mathsf{O}_{\mathsf{Enc}}$, and Q_{HRE} is the number of re-encryption queries for non-challenge ciphertexts that $\mathcal{A}$ makes to $\mathsf{O}_{\mathsf{ReEnc}}$.

The intuition behind this is that source-hiding implies that non-challenge ciphertexts can be replaced by fresh encryptions at the appropriate level. Therefore, an adversary in the CPA experiment can simulate the HRA experiment and gain the advantage of any successful PCS adversary. We give a reduction-based proof based on the proof of [FKKP18, Lemma 7] that a PRE scheme that is both source-hiding and PRE-IND-CPA-secure has PCS.

Proof. We prove this theorem using a sequence of game hops, breaking the proof down into a number of lemmas. Let $\mathsf{PostCompSH}$ be a variant of the $\mathsf{PostComp}$ game where all re-encryptions of non-challenge ciphertexts and the re-encryption made by $\mathsf{Chal}^{\mathsf{PC}}_{\mathsf{ReEnc}}$ are replaced with fresh encryptions at the appropriate level. We first demonstrate that $\mathsf{PostComp}$ and $\mathsf{PostCompSH}$ are computationally indistinguishable, before showing how $\mathcal{PRE}$ being PRE-IND-CPA-secure implies there is no PPT adversary that can win $\mathsf{PostCompSH}$ with non-negligible advantage.

Lemma V.8. *The advantage ϵ_2' of any PPT algorithm in distinguishing between $\mathsf{PostCompSH}$ and $\mathsf{PostComp}$ is bounded by*

$$\epsilon_2' \leq \kappa(\kappa-1)(Q_E + Q_{HRE} + 1)(Q_{HRE} + 1) \cdot \epsilon_2.$$

Proof. We use a hybrid argument where re-encryptions are replaced one at a time. Let $\mathsf{H}^0 := \mathsf{PostComp}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$, and let H^k be identical to H^{k-1} except that the k^{th} re-encryption is replaced by a fresh encryption, limited to re-encryptions of non-challenge ciphertexts or the re-encryption used in $\mathsf{Chal}^{\mathsf{PC}}_{\mathsf{ReEnc}}$. We see that $\mathsf{H}^{Q_{HRE}+1} = \mathsf{PostCompSH}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$.

We describe how an adversary $\mathcal{A}$ trying to win the source-hiding game SH with challenge bit $d \xleftarrow{\$} \{0,1\}$ can perfectly simulate either H^{k-1} or H^{k}, depending on d. Therefore, if there exists a PPT distinguisher $\mathcal{D}$ that, with advantage δ, outputs $d'_{\mathcal{D}} = 0$ to indicate that it is playing H^{k-1} and $d'_{\mathcal{D}} = 1$ to indicate that it is playing H^{k}, then $\mathcal{A}$ can run $\mathcal{D}$ as a subroutine and set $d' = d'_{\mathcal{D}}$ to win SH. Because $Q_{HRE} + 1$ replacements need to be made in total (all the honest re-encryptions plus the re-encryption performed during the challenge), the advantage of distinguishing between $\mathsf{PostComp}$ and $\mathsf{PostCompSH}$ is bounded by $\epsilon'_2 \leq \delta \cdot (Q_{HRE} + 1)$.

$\mathcal{A}$ first sets $b \xleftarrow{\$} \{0,1\}$ as the challenge bit in $\mathsf{PostComp}^{\mathcal{PRE}}(1^\lambda)$. Note that SH only has two keypairs $(\mathsf{pk}_0^{\mathsf{SH}}, \mathsf{sk}_0^{\mathsf{SH}})$, $(\mathsf{pk}_1^{\mathsf{SH}}, \mathsf{sk}_1^{\mathsf{SH}})$, and that the challenge $\mathsf{Chal}^{\mathsf{SH}}_{\mathsf{ReEnc/Enc}}(m, \ell)$ returns both the fresh encryption c^* and the challenge $c^{(b)}$. Therefore, $\mathcal{A}$ must first guess which ciphertext c and key indexes i^*, j^* will be input for the k^{th} replacement. $\mathcal{A}$ responds to the k^{th} re-encryption query as follows:

- It takes the two keypairs from the source-hiding game $(\mathsf{pk}_0^{\mathsf{SH}}, \mathsf{sk}_0^{\mathsf{SH}}), (\mathsf{pk}_1^{\mathsf{SH}}, \mathsf{sk}_1^{\mathsf{SH}})$, sets these as $(\mathsf{pk}_{i^*}, \mathsf{sk}_{i^*}), (\mathsf{pk}_{j^*}, \mathsf{sk}_{j^*})$ respectively, and then self-generates the remaining $\kappa - 2$ key pairs for $\mathsf{PostComp}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$.
- When $\mathcal{D}$ makes the query that leads to the creation of c under pk_i (which could be either through $\mathsf{O}_{\mathsf{Enc}}$ or $\mathsf{O}_{\mathsf{ReEnc}}$)), if $i \neq i^*$ then the reduction aborts. Otherwise the reduction takes the underlying message and uses it to query $\mathsf{Chal}^{\mathsf{SH}}_{\mathsf{ReEnc/Enc}}(m, \ell) \xrightarrow{\$} (c^*, c'^{(b)})$ and returns c^* to the adversary. Note that c^* is created via oracles, either by $\mathsf{O}_{\mathsf{Enc}}(i, m)$ or by calling $\mathsf{O}_{\mathsf{ReEnc}}$ on an oracle-created ciphertext. This means that for a pair (i, c) where c is an oracle-created ciphertext under pk_i, $\mathcal{A}$ will know the underlying message m and the number of times ℓ that the ciphertext has been (or appears to have been) re-encrypted.
- Let $\mathsf{O}_{\mathsf{ReEnc}}(i^*, j, (\Delta_{i^*,j}), c)$ be the k^{th} re-encryption that $\mathcal{D}$ queries, where c is an honest ciphertext. If $c \neq c^*$ or $j \neq j^*$, the reduction aborts. Otherwise, $\mathcal{A}$ returns the challenge $c'^{(b)}$ it received from the source-hiding game.

Note that for the challenge ciphertext, because $\mathcal{A}$ has chosen b they can easily simulate the challenge.

To distinguish between H^{i} and H^{i+1}, the adversary must have guessed i, j correctly, resulting in a loss of $\kappa(\kappa - 1)$, and must also have guessed which encryption will be the input for the k^{th}re-encryption, resulting in a loss of at most $(Q_E + Q_{HRE} + 1)$. Because $\mathcal{PRE}$ is ϵ_2-source-hiding by assumption, we conclude that $\delta \leq \kappa(\kappa-1)(Q_E+Q_{HRE}+1)\cdot\epsilon_2$. As there are $Q_{HRE}+1$ hybrids, by the triangle inequality we have that $\epsilon_2' \leq \kappa(\kappa-1)(Q_E + Q_{HRE}+1)(Q_{HRE}+1)\cdot\epsilon_2$, as required. □

Lemma V.9. *If $\mathcal{PRE}$ is ϵ_1-PRE-IND-CPA-secure, then the advantage ϵ_1' of winning $\mathsf{PostCompSH}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ is bounded by*

$$\epsilon_1' \leq \epsilon_1.$$

Proof. We demonstrate how an adversary $\mathcal{A}$ in $\mathsf{PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ can simulate $\mathsf{PostCompSH}_{\mathcal{B}}^{\mathcal{PRE}}(1^\lambda)$. Thus, if there exists a PPT adversary $\mathcal{B}$ that wins $\mathsf{PostCompSH}_{\mathcal{B}}^{\mathcal{PRE}}(1^\lambda)$ (returning $b'_{\mathcal{B}}$) with advantage ϵ_1', then $\mathcal{A}$ can win $\mathsf{PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ with the same advantage by returning $b' = b'_{\mathcal{B}}$. $\mathcal{A}$ simulates $\mathsf{PostComp}_{\mathcal{B}}^{\mathcal{PRE}}(1^\lambda)$ by responding to oracle queries made by $\mathcal{B}$ as follows:

- $\mathcal{A}$ simulates $\mathsf{O}_{\mathsf{Enc}}$ as described in $\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$, also setting lookup tables $\mathcal{C}_{\mathsf{msg}}[i,c] := m$ for the underlying message m and $\mathcal{C}_{\mathsf{lev}}[i,c] := 0$ for the ciphertext level.
- $\mathcal{A}$ forwards calls to $\mathsf{O}_{\mathsf{ReKeyGen}}$ to the same oracle in $\mathsf{PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$.
- $\mathcal{A}$ simulates $\mathsf{O}_{\mathsf{ReEnc}}(i,j,c)$ for non-challenge ciphertexts by retrieving $m \leftarrow \mathcal{C}_{\mathsf{msg}}[i,c]$ and $\ell \leftarrow \mathcal{C}_{\mathsf{lev}}[i,c]$ and returning $c' \overset{\$}{\leftarrow} \mathsf{Enc}(\mathsf{pk}_j, m, \ell+1)$. $\mathcal{A}$ sets $\mathcal{C}_{\mathsf{msg}}[j,c'] := m$, $\mathcal{C}_{\mathsf{lev}}[j,c'] := \ell+1$.

  For challenge ciphertexts, if $\mathcal{B}$ has set $\Delta_{i,j} = \perp$ then $\mathcal{A}$ generates $\Delta_{i,j} \overset{(\$)}{\leftarrow} \mathsf{O}_{\mathsf{ReKeyGen}}(i,j)$ and returns $c' \overset{\$}{\leftarrow} \mathsf{ReEnc}(\Delta_{i,j}, c)$.
- To simulate $\mathsf{Chal}_{\mathsf{ReEnc}}^{\mathsf{PC}}(i^*, j^*, [\Delta_{i^*,j^*}], c_0^*, c_1^*)$, $\mathcal{A}$ first retrieves $m_0 \leftarrow \mathcal{C}_{\mathsf{msg}}[i^*, c_0^*]$, $\ell_0 \leftarrow \mathcal{C}_{\mathsf{lev}}[(i^*, c_0^*)]$, $m_1 \leftarrow \mathcal{C}_{\mathsf{msg}}[i^*, c_1^*]$, $\ell_1 \leftarrow \mathcal{C}_{\mathsf{lev}}[i^*, c_1^*]$. Then $\mathcal{A}$ queries $\mathsf{Chal}_{\mathsf{Enc}}^{\mathsf{CPA}}(j, m_0, m_1) \overset{\$}{\rightarrow} c'^{(b)}$. Because $\mathcal{PRE}$ has transparency, $c'^{(b)}$ will be indistinguishable from any re-encrypted ciphertext $c' \overset{(\$)}{\rightarrow} \mathsf{ReEnc}(\Delta_{i,j}, c_b^*)$ at level $\ell_b + 1$.

The resulting simulation is identical to $\mathsf{PostCompSH}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$. Therefore, if there exists an adversary $\mathcal{B}$ that wins $\mathsf{PRE\text{-}IND\text{-}CPA}_{\mathcal{B}}^{\mathcal{PRE}}(1^\lambda)$ with advantage ϵ_1', then $\mathcal{A}$ can win $\mathsf{PostCompSH}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ with the same advantage. We conclude that $\epsilon_1' \leq \epsilon_1$, as required. □

By Lemmas V.8 and V.9 and the triangle inequality, we see that

$$\epsilon \leq \kappa(\kappa - 1)(Q_E + Q_{HRE} + 1)(Q_{HRE} + 1) \cdot \epsilon_2 + \epsilon_1.$$

This concludes the proof of Theorem V.3. □

Remarks. We note that a multi-key, multi-challenge variant of source-hiding will result in a simpler reduction, as the simulator will no longer need to guess in advance which keys and ciphertexts will be used to replace the re-encrypted ciphertext with a fresh encryption. We conjecture that proving that a PRE scheme meets a multi-key, multi-challenge variant of source-hiding should not incur a significant loss, as proving source-hiding requires a statistical argument that ciphertexts are identically distributed. This reasoning should apply to multiple keys and ciphertexts as long as they are honestly generated. Indeed, it is possible that the overall loss from using a multi-key, multi-challenge notion of source-hiding will mean the overall loss of security is less severe than it is in the current approach.

Secondly, we conjecture that there exists a tighter reduction based on the IND-CPA security of the underlying encryption scheme, as in [FKKP18, Theorem 6]. Doing this requires swapping the re-encryption challenge in the post-compromise game with the encryption challenge from the PKE IND-CPA game, which is only defined for one key and therefore results in a loss of security of at least $1/\kappa$, as a simulator must guess which key the challenge will be called under. Notions of key privacy may help with this. For example, source key privacy for update tokens would allow simulator to simulate update tokens from unknown source keys. Key privacy of ciphertexts may mean it does not matter which key the challenge is called under, as in [LT18].

Finally, we conjecture that there is the potential for a tighter relation between IND-HRA-security and PCS, as at a high level the only re-encryption that needs replacing here is the one performed by $\mathsf{Chal}^{\mathsf{PC}}_{\mathsf{ReEnc}}$. Because the single-challenge definition of source-hiding has limited application in that it only means that re-encryptions $c' \overset{(\$)}{\leftarrow} \mathsf{ReEnc}(\Delta_{i,j}, c)$ of fresh ciphertexts $c \overset{\$}{\leftarrow} \mathsf{Enc}(\mathsf{pk}_i, m, [\ell])$ can be replaced by fresh encryptions, whereas we need this property to hold even when c is itself a re-encrypted ciphertext. It should be possible that such a replacement can be made without needing to replace all previous re-encryptions. Informally, this follows trivially from an iterative argument, or from the statistical argument outlined above.

We will revisit these remarks in Chapter VI.

V: 3.5 Existing PRE schemes that satisfy PCS

In [FKKP19, Construction 2.b], Fuchsbauer et al. present a PRE scheme which is both PRE-IND-CPA-secure and source-hiding. This construction is based on the hardness of the decision learning with errors problem with sub-exponential noise-to-modulus ratio. Using the result of Theorem V.3, we see that their scheme has PCS. We note that, subject to the remarks given in Section V: 3.4, it is possible that [FKKP18, Construction 2, Construction 4] also have PCS, as, whilst these constructions are not source-hiding, they do re-randomise ciphertexts upon re-encryption.

We describe a similar scheme to [FKKP19, Construction 2.b] in Section V: 5.2, where we explain how the parameters required to make the scheme source-hiding carry inherent inefficiencies which we aim to avoid, motivating our new construction.

V: 4 An Efficient Construction from Lattices

We introduce a natural construction with PCS, based on BV-PRE – the RLWE construction described in Figure V.5. Whilst Theorem V.3 shows that source-hiding can lead to PCS, the existing constructions that are provably source-hiding depend on making sub-optimal parameter choices that significantly impact the schemes' practicality, as we shall explore in Section V: 5.2. Our construction has PCS but is not source-hiding, implying that source-hiding is not necessary for PCS. This means that our construction can utilise much better parameter choices in terms of efficiency. Furthermore, our construction does not rely on strong assumptions or heavy techniques such as obfuscation as in [HRsV07, CCV12]. We also achieve transparency, meaning the cost of decryption does not grow for repeatedly re-encrypted ciphertexts. Recall from the original motivations of PRE that if extra computation is needed to decrypt a re-encrypted ciphertext then this may outweigh the benefits of outsourcing re-encryption.

Our construction makes some adaptations to BV-PRE to fit the workflow of PRE; making use of the key resampling technique of [BV11] to re-randomise the ciphertext. We conjecture that any scheme that permits similar re-randomisation can be proven secure using similar methods.

V: 4.1 Adapting BV-PRE for PCS

The underlying scheme, BV-PRE is based on the BV-encryption scheme [BV11] (Figure II.2), which is based on RLWE. This scheme is parameterised by ciphertext modulus q, plaintext modulus $p \geq 2$, ring dimension n, polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[n]/\langle x^n + 1\rangle$ and relinearisation window r.

Recall that BV-PRE is not fully public-key, relying on an additional 'public' key 'pk'$_j$ for the target key sk_j to generate update tokens. However, this key together with the token can be used to derive the old secret key. We get around this problem using the key resampling technique ReSample which takes a public key pk_j and outputs a fresh public key pk'_j with the same underlying secret. The key resampling technique [BV11], can be found in Figure V.7.

```
ReSample(pk)
$(a, b) \leftarrow \mathsf{pk}$
$\bar{v}, \bar{e_0}, \bar{e_1} \xleftarrow{\$} \chi_e$
$\bar{a} = a\bar{v} + p\bar{e_0}$
$\bar{b} = b\bar{v} + p\bar{e_1}$
return $\mathsf{pk} = (\bar{a}, \bar{b})$
```

Figure V.7: Similified key resampling technique [BV11].

For the resampled key, observe that $\bar{b} = \bar{a}s + p(e\bar{v} + \bar{e_1} - \bar{e_0}s)$ and is therefore also suitable for encryption. Furthermore, $(\bar{a}, \bar{b})$ is computationally indistinguishable from a freshly generated public key for s. Our idea is to replace the 'public' key used to create update tokens in BV-PRE with a resampled public key. We use the same idea to introduce new randomness in ReEnc.

Resampling leads us to define a new security game for resampled keys in Figure V.9, where given a public key (a, b), the challenger either returns a resampled key or a uniformly sampled pair. This is similar to RLWE challenges for fresh keys. Note that we use the same relinearisation technique as [PRSV17] to reduce error growth, where ciphertexts are broken down as described in Equation (V.1).

We give our construction, pcBV-PRE, in Figure V.8. It builds on BV-PRE by having the proxy add further randomness in the ReEnc operation. Recall that this is necessary for

```
KeyGen(1^λ)
    $a \xleftarrow{\$} \mathcal{U}_q$,
    $s, e \xleftarrow{\$} \chi_e$
    $b = a \cdot s + pe$
    $\mathsf{sk} = s$
    $\mathsf{pk} = (a, b)$
    return $(\mathsf{pk}, \mathsf{sk})$

Enc(pk, m) $\xrightarrow{\$} c$
    $(a, b) \leftarrow \mathsf{pk}$
    $v, e_0, e_1 \xleftarrow{\$} \chi_e$
    $c_0 = b \cdot v + pe_0 + m$
    $c_1 = a \cdot v + pe_1$
    return $c = (c_0, c_1)$

Dec(sk, c) $\to m'$
    $s \leftarrow \mathsf{sk}$
    $m' = c_0 - s \cdot c_1 \mod p$
    return $m'$

ReKeyGen(sk_i, pk_j) $\xrightarrow{\$} \Delta_{i,j}$
    for $\iota \in \{0, 1, \ldots, \lfloor \log_2(q)/r \rfloor\}$ :
        $(\bar{a_\iota}, \bar{b_\iota}) \xleftarrow{\$} \mathsf{ReSample}(\mathsf{pk}_j)$
        $\gamma_\iota = \bar{b_\iota} - \mathsf{sk}_i \cdot (2^r)^\iota$
    $\Delta = \{(\bar{a_\iota}, \gamma_\iota)\}_{\iota=0}^{\lfloor \log_2(q)/r \rfloor}$
    $\Delta_{i,j} = (\Delta, \mathsf{pk}_j)$
    return $\Delta_{i,j}$

ReEnc(Δ_{i,j}, c) $\xrightarrow{\$} c'$
    $\left(\{(\bar{a_\iota}, \gamma_\iota)\}_{\iota=0}^{\lfloor \log_2(q)/r \rfloor}, \mathsf{pk}_j\right) \leftarrow \Delta_{i,j}$
    $(c_0, c_1) \leftarrow c$
    $(\hat{a}, \hat{b}) \xleftarrow{\$} \mathsf{ReSample}(\mathsf{pk}_j)$
    $c'_0 = c_0 + \left(\sum_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} (c_1^{(\iota)} \cdot \gamma_\iota)\right) + \hat{b}$
    $c'_1 = \left(\sum_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} (c_1^{(\iota)} \cdot \bar{a_\iota})\right) + \hat{a}$
    return $c' = (c'_0, c'_1)$
```

Figure V.8: pcBV-PRE. This adapts BV-PRE [PRSV17] (Figure V.5) to be fully public-key, whilst taking steps to minimise computation and bandwidth. The key resampling algorithm ReSample is described in Figure V.7.

a scheme to have PCS, as otherwise an adversary could re-encrypt locally to obtain the same re-encryption.

V: 5 Analysis of pcBV-PRE

We now analyse pcBV-PRE, giving a correctness bound (Equation (V.5)), and demonstrate that pcBV-PRE has PCS (Theorem V.4) and is IND-HRA-secure (Theorem V.7). We do not leverage Theorem V.3 to prove PCS, as pcBV-PRE is not source-hiding. In Section V: 5.2 we use a variant of pcBV-PRE that is source-hiding to explicitly demonstrate how schemes that are provably source-hiding (at least, using known techniques) rely on parameter choices that make the scheme far less practical. We instead prove security of pcBV-PRE using observations of re-sampling RLWE samples as described in Figure V.7. We therefore show that source-hiding is not a necessary requirement for a PRE scheme to have PCS.

V: 5.1 Correctness of pcBV-PRE

Much of the analysis used to argue correctness from [PRSV17] holds for our modified scheme in Figure V.8, as pcBV-PRE is very similar to BV-PRE. Therefore, we keep the discussion fairly brief and refer to [PRSV17] for full details.

Recall that the decryption algorithm computes $c_0 - c_1 s$ and then performs a reduction modulo p. In the case that (c_0, c_1) is a ciphertext produced by the Enc algorithm, we have that

$$c_0 - c_1 s = p(ev + e_0 + e_1 s) + m, \tag{V.2}$$

so decryption is successful when $p(ev + e_0 + e_1 s) + m$ (i.e. the error plus the message) does not wrap around modulo q. However, the correctness condition of decryption is different when considering ciphertexts output by the ReEnc algorithm. To see how, let $(c_0', c_1') \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, (c_0, c_1))$. We also note that for $\mathsf{ReSample}(i,j) \xrightarrow{\$} (\bar{a}, \bar{b}) = (a\bar{v} + p\bar{e_0}, b\bar{v} + p\bar{e_1})$ where $b = as + pe$, we get

$$\bar{b} - s\bar{a} = p(e\bar{v} + \bar{e_1} - \bar{e_0} s). \tag{V.3}$$

Using notation consistent with Figure V.7, we have that

$$\begin{aligned} c_0' - c_1' s_j &= c_0 + \left(\sum_{\iota=0}^{\lfloor \log(q)/r \rfloor} c_1^{(\iota)} \left(\bar{b_\iota} - s_i (2^r)^\iota - s_j \bar{a_\iota} \right) \right) + (\hat{b} - \hat{a} s_j) \\ &= c_0 - c_1 s_i + \left(\sum_{\iota=0}^{\lfloor \log(q)/r \rfloor} c_1^{(\iota)} \left(\bar{b_\iota} - s_j \bar{a_\iota} \right) \right) + (\hat{b} - \hat{a} s_j) \\ &\overset{(V.3)}{=} c_0 - c_1 s_i + \left(\sum_{\iota=0}^{\lfloor \log(q)/r \rfloor} c_1^{(\iota)} \cdot p(e\bar{v_\iota} + \bar{e}_0^{(\iota)} s_j + \bar{e}_1^{(\iota)}) \right) + p(e\hat{v} + \hat{e_0} s_j + \hat{e_1}), \end{aligned}$$

where all arithmetic is performed over the integers modulo q. This shows that the error grows by an additive term of pE with each re-encryption where

$$E = \left(\sum_{\iota=0}^{\lfloor \log(q)/r \rfloor} c_1^{(\iota)} \cdot (e\bar{v_\iota} + \bar{e}_0^{(\iota)} s_j + \bar{e}_1^{(\iota)}) \right) + (e\hat{v} + \hat{e_0} s_j + \hat{e_1}).$$

Assume that χ_e is (B, δ)-bounded for some small δ (we quantify these values later). Then we have that the noise grows by at most

$$p||E||_\infty \leq pn(2^r - 1)(\lfloor \log(q)/r \rfloor + 1)(2B^2n + B) + p(2B^2n + B) =: G(n, q, r, B) \quad \text{(V.4)}$$

on each re-encryption taking into consideration the multiplication of degree $n-1$ polynomials. Therefore, after ℓ re-encryptions, the noise has grown by an additive term of size $\ell \cdot G(n, q, r, B)$. Therefore, the condition for successful decryption after ℓ re-encryptions is

$$p(2B^n + B) + \ell G(n, q, r, B) + p/2 \leq q/2, \quad \text{(V.5)}$$

where G is defined in Equation (V.4). Note that if our error distribution has $\sigma_e > \omega(\log n)$, then we can set $B = \sigma_e\sqrt{n}$ and $\delta = 2^{-n+1}$ [MR04,LTV12]. In order to choose the parameters of the system, one needs to ensure that Equation (V.5) is satisfied. Note that this analysis is a worst-case one. It is shown in [PRSV17] that the central limit theorem can be used to essentially alter the form of B and change all occurrences of n into occurrences of $\sqrt{n}$ in Equation (V.5) by allowing for a tunable failure probability. We omit this more practical method of correctness analysis for brevity.

V: 5.2 pcBV-PRE and source-hiding

We first demonstrate that pcBV-PRE is not source-hiding, before describing a variant of pcBV-PRE that is source-hiding, at the cost of requiring less practical parameter choices.

Lemma V.10. pcBV-PRE *is not source-hiding.*

Proof. Fresh ciphertexts in pcBV-PRE are distinguishable from re-encrypted ciphertexts, as re-encrypted ciphertexts have a bigger error than fresh encryptions. As we have not defined levelling explicitly in pcBV-PRE, all fresh ciphertexts a smaller error bound than re-encrypted ciphertexts. Since an adversary with the secret key can derive the error, they will be able to infer whether a ciphertext is fresh or a re-encryption from the size of the error. □

The reason pcBV-PRE is not source-hiding is because fresh encryptions have different noise magnitudes compared with re-encryptions. One method of overcoming this is using a noise 'blurring' approach [CCL+14]. This involves 'blurring' or 'drowning' the noise in fresh encryptions so that it is distributed in the same way as re-encryptions. To do this, we modify the ciphertext pairs in the Enc and ReEnc algorithms to add fresh noise to them. This fresh noise is taken from an error distribution such that it 'blurs' out all the old noise that was introduced from the encryption and re-encryption operations. This is the approach used in [FKKP19, Construction 2.b].

We now present a formal description of this amendment to our scheme. We define pcBV-PRE+ in the same way as pcBV-PRE except that we introduce levelling through blurring widths $\mathbf{E}_\ell$ associated to level ℓ encryption. Specifically, we define $\mathsf{Enc}(\mathsf{pk}, m; \ell) = (c_0, c_1) = (bv + pe_0 + m + pf_\ell, av + pe_1)$ where $f_\ell \xleftarrow{\$} [-\mathbf{E}_\ell, \mathbf{E}_\ell]^n$. We also have ReEnc take a level ℓ as input, where the input ciphertext has already been re-encrypted $\ell - 1$ times, and have ReEnc add f_ℓ to c_0'.

Lemma V.11. pcBV-PRE+ *is computationally source hiding provided that* $\mathbf{E}_\ell \gg \mathbf{E}_{\ell-1}$ *and* $\mathbf{E}_0 \gg (\sigma n)^2$*, where* $x \gg y$ *denotes that* x *is asymptotically larger than* y*.*

Proof. For correctly sampled $\mathsf{aux} = (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_0, \mathsf{sk}_1, \Delta_{0,1})$, we will show that the distributions of

$$\Big(\mathsf{aux}, c := \mathsf{Enc}(\mathsf{pk}_0, m; \ell), c'^{(1)} := \mathsf{Enc}(\mathsf{pk}_1, m; \ell + 1)\Big) \tag{V.6}$$

and

$$\Big(\mathsf{aux}, c := \mathsf{Enc}(\mathsf{pk}_0, m; \ell), c'^{(0)} := \mathsf{ReEnc}(\Delta_{0,1}, \mathsf{pk}_1, c; \ell + 1)\Big) \tag{V.7}$$

are computationally indistinguishable for any valid choice of m and ℓ.

We begin by assessing the error size of fresh encryptions (Equation (V.6)). Let $\mathsf{pk}_1 = (a_1, b_1 := a_1 \cdot s_1 + pe_1)$ where s_1, e_1 are the secret errors associated with pk_1, and let $c = (b_1 \cdot \tilde{v} + p(\tilde{e_0} + f_{\ell+1}) + m, a_1 \cdot \tilde{v} + p\tilde{e_1})$ be a fresh ciphertext where $\tilde{v}, \tilde{e_0}, \tilde{e_1} \xleftarrow{\$} \chi_\sigma$ are the error terms sampled during Enc as described in Figure V.8, and $f_{\ell+1} \xleftarrow{\$} [-\mathbf{E}_{\ell+1}, \mathbf{E}_{\ell+1}]^n$. Using the notation of the source-hiding experiment, equivalently we can write

$$c'^{(1)} = ((a_1 \cdot \tilde{v}) \cdot s_1 + p(\tilde{e_0} + e_1\tilde{v} + f_{\ell+1}) + m, a_1 \cdot \tilde{v} + p\tilde{e_1}) \tag{V.8}$$

as the form of a valid level $\ell + 1$ encryption of m under pk_1.

We note that as s_1 is included in aux, we can calculate $c_0 - c_1 s_1 = m + p(\tilde{e_0} s_1 - \tilde{e_1} + e_1\tilde{v} + f_\ell)$ and use this to derive the error term $\tilde{e_1} s_1 - \tilde{e_0} - e_1\tilde{v} + f_{\ell+1}$. We therefore need to demonstrate that this error is indistinguishable from the error of a ciphertext that has been re-encrypted $\ell+1$ times.

Let $f'_{\ell+1} \xleftarrow{\$} [-\mathbf{E}_{\ell+1}, \mathbf{E}_{\ell+1}]$. The distribution of $\tilde{e_1}s_1 - \tilde{e_0} - e_1\tilde{v} + f'_{\ell+1}$ is statistically indistinguishable from the distribution of $f_{\ell+1}$ as long as $\mathbf{E}_{\ell+1} \gg (\sigma n)^2$, as this is greater than the largest coefficient of $\tilde{e_1}s_1 - \tilde{e_0} - e_1\tilde{v}$ with all but negligible probability. Therefore, the distribution of Equation (V.8) is statistically indistinguishable from the distribution of

$$\begin{aligned}&\left((a_1 \cdot \tilde{v}) \cdot s_1 + p(\tilde{e_0} + e_1\tilde{v} + (f'_{\ell+1} + \tilde{e_1}s_1 - \tilde{e_0} - e_1\tilde{v})) + m, a_1 \cdot \tilde{v} + p\tilde{e_1}\right)\\ =&\left((a_1 \cdot \tilde{v} + p\tilde{e_1}) \cdot s_1 + pf'_{\ell+1} + m, a_1 \cdot \tilde{v} + p\tilde{e_1}\right). \qquad \text{(V.9)}\end{aligned}$$

In turn, the RLWE assumption implies Equation (V.9) is computationally indistinguishable from

$$\left(u \cdot s_1 + pf'_{\ell+1} + m, u\right),$$

where $u \xleftarrow{\$} \mathcal{U}_q$. As the arguments hold for any correctly distributed aux, it follows that the distribution of Equation (V.6) is computationally indistinguishable from

$$\left(\mathsf{aux}, c := \mathsf{Enc}(\mathsf{pk}_0, m; \ell), (u \cdot s_1 + pf'_{\ell+1} + m, u)\right). \qquad \text{(V.10)}$$

We move on to assessing the error size of re-encrypted ciphertexts (Equation (V.7)). To complete the proof, we will show that the distribution of Equation (V.7) is also computationally indistinguishable from the distribution of Equation (V.10) via a similar argument. Let $\mathsf{pk}_0 = (a_0, b_0 := a_0 \cdot s_0 + pe_0)$ where s_1, e_1 are the secret errors associated with pk_0. As before, we fix a valid ciphertext $(c_0, c_1) = ((a_0 \cdot s_0 + pe_0)\tilde{v} + p(\tilde{e_0} + f_\ell) + m, a_0 \cdot \tilde{v} + p\tilde{e_0})$ where $\tilde{v}, \tilde{e_0}, \tilde{e_1} \xleftarrow{\$} \chi_\sigma$, meaning that $c_0 - c_1 s_0 = m + p(\tilde{e_0}s_0 - \tilde{e_1} + e_0\tilde{v} + f_\ell)$. Writing $\Delta_{0,1} = \{(\bar{a_\iota}, \bar{b_\iota} - s_0 \cdot (2^r)^\iota)\}_{\iota=0}^{\lfloor \log_2(q)/r \rfloor}$ where $\{(\bar{a_\iota}, \bar{b_\iota})\}_{\iota=0}^{\lfloor \log_2(q)/r \rfloor}$ are the outputs of the calls to $\mathsf{ReSample}(\mathsf{pk}_1)$ used in $\mathsf{ReKeyGen}$ as described in Figure V.8, we have that $\bar{b_\iota} = \bar{a_\iota} \cdot s_1 + p(e_1\bar{v_\iota} + \bar{e_1}^{(\iota)} - \bar{e_0}^{(\iota)}s_1)$ where $\bar{v_\iota}, \bar{e_0}^{(\iota)}, \bar{e_1}^{(\iota)} \xleftarrow{\$} \chi_\sigma$ are the error terms sampled in $\mathsf{ReSample}$. In addition, we have the re-randomisation term $(\hat{a}, \hat{b})$ generated by calling $\mathsf{ReSample}(\mathsf{pk}_1)$ in ReEnc as described in Figure V.8, which satisfies $\hat{b} = \hat{a} \cdot s_1 + p(e_1\hat{v} + \hat{e_1} - \hat{e_0}s_1)$ where $\hat{v}, \hat{e_0}, \hat{e_1} \xleftarrow{\$} \chi_\sigma$. Let $c'_1 = \left(\sum_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} c_1^{(\iota)} \cdot \bar{a_\iota}\right) + \hat{a}$, $e_c = \tilde{e_0}s_0 - \tilde{e_1} + e_0\tilde{v}$, $e_t = \left(\sum_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} c_1^{(\iota)}(e_1\bar{v_\iota} + \bar{e_1}^{(\iota)} - \bar{e_0}^{(\iota)}s_1)\right)$ and $e_p = e_1\hat{v} + \hat{e_1} - p\hat{e_0}s_1$. Then for a re-encrypted ciphertext $c'^{(0)} = (c'_0, c'_1)$:

$$\begin{aligned}
c'_0 &= c_0 + \left(\sum_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} c_1^{(\iota)} \cdot (\bar{b}_\iota - s_i \cdot (2^r)^\iota) \right) + \hat{b} + pf_{\ell+1} \\
&= c_0 - c_1 s_0 + \left(\sum_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} c_1^{(\iota)} \cdot \bar{b}_\iota \right) + \hat{b} + pf_{\ell+1} \\
&= m + p(e_c + f_\ell) \\
&\quad + \left(\sum_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} c_1^{(\iota)} \cdot \left(\bar{a}_\iota \cdot s_1 + p(e_1 \bar{v}_\iota + \bar{e_1}^{(\iota)} - \bar{e_0}^{(\iota)} s_1) \right) \right) + \hat{a} \cdot s_1 + pe_p + pf_{\ell+1} \\
&= c'_1 \cdot s_1 + p(e_c + e_t + e_p + f_\ell + f_{\ell+1}) + m,
\end{aligned}$$

where $f_{\ell+1} \xleftarrow{\$} [-\mathbf{E}_{\ell+1}, \mathbf{E}_{\ell+1}]$. Then we have

$$c'^{(0)} = \left(c'_1 \cdot s_1 + p(e_c + e_t + e_p + f_\ell + f_{\ell+1}) + m, \quad c'_1\right). \tag{V.11}$$

Let $f'_{\ell+1} \xleftarrow{\$} [-\mathbf{E}_{\ell+1}, \mathbf{E}_{\ell+1}]$. Then the distribution of $f'_{\ell+1} - e_c - e_t - e_p - f_\ell$ is statistically indistinguishable from the distribution of $f_{\ell+1}$ so long as $\mathbf{E}_{\ell+1} \gg \max\{\mathbf{E}_\ell, (\sigma n)^2 \cdot (\lfloor \log_2(q)/r \rfloor + 2)\}$. Therefore Equation (V.11) is distributed statistically close to the distribution of

$$\begin{aligned}
&\left(c'_1 \cdot s_1 + p(e_c + e_t + e_p + f_\ell + (f'_{\ell+1} - e_c - e_t - e_p - f_\ell)) + m, \quad c'_1\right) \\
&= \left(c'_1 \cdot s_1 + pf'_{\ell+1} + m, \quad c'_1\right).
\end{aligned} \tag{V.12}$$

Finally, using a RLWE assumption on $\hat{a}$, the term c'_1 is statistically indistinguishable from uniform, meaning that the distribution of Equation (V.12) is computationally indistinguishable from $(u \cdot s_1 + pf'_{\ell+1}, u)$, where $u \xleftarrow{\$} \mathcal{U}_q$. Since these arguments work for any valid aux and ciphertext (c_0, c_1), it follows that the distribution of Equation (V.7) is computationally indistinguishable from the distribution of

$$\left(\mathsf{aux}, c := \mathsf{Enc}(\mathsf{pk}_0, m; \ell), (u \cdot s_1 + pf'_{\ell+1}, u)\right),$$

as required. □

The advantage of this approach is that in the resulting scheme, the secret key does not reveal whether a ciphertext is fresh or a re-encryption. Another advantage is that the keys the ciphertext was previously encrypted under remain hidden. This is vital since in the source-hiding security property, the adversary receives all of the available decryption keys and thus can decrypt any ciphertext they want.

With this in mind, there are a number of reasons why requiring a scheme to be provably source-hiding is a hindrance. This 'blurring' approach is the only one to date that has been used to prove source-hiding, but it requires at least a sub-exponential noise-to-modulus ratio which considerably harms performance and security. This is because the RLWE assumption (and respectively LWE assumption for [FKKP19, Construction 2.b]) that is used becomes much stronger, since the approximation factors of the related ideal lattice problems become sub-exponential rather than polynomial in n. In particular, there are quantum polynomial time algorithms solving ideal lattice problems with approximation factor $\exp(\tilde{O}(\sqrt{n}))$ [BS16, CDPR16, CDW17], providing evidence that we cannot simply use an arbitrary noise-to-modulus ratio while retaining the same security guarantees. Moreover, the increase in modulus that is required makes standard operations much slower. As a by-product, both schemes allow only a constant number of re-encryptions. This can be seen in how the error size for each level must increase dramatically so as to 'blur out' the errors of previous levels. We conclude that incorporating source-hiding (at least, using known methods) results in a scheme that is impractical, based on much stronger assumptions and also heavily restricted in the number of re-encryptions that can occur.

The intention with our construction was to give a practical PRE scheme with PCS with minimal restrictions and from weaker assumptions. Since pcBV-PRE is very close to BV-PRE, which is comparatively fast (see [PRSV17] for exact figures), and our construction only adds extra sampling, loss of efficiency is minimal.

V: 5.3 Implications of resampling

Before we prove that pcBV-PRE has PCS, we demonstrate some consequences of using the resampling technique described in Figure V.7. We will use these in our proof.

Lemma V.12. *Let* $\mathsf{ReSample\text{-}ncRLWE}_{\mathcal{A}}(1^\lambda)$ *be defined as in Figure V.9. Then, for all*

```
$\mathsf{ReSample\text{-}ncRLWE}_{\mathcal{A}}(1^\lambda)$
    $d \xleftarrow{\$} \{0,1\}$
    $\kappa = 0$
    $d' \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{KeyGen}},\mathsf{Chal}_{\mathsf{RRLWE}}}(1^\lambda)$
    return $(d' = d)$
```

```
$\mathsf{O}_{\mathsf{KeyGen}}()$
    $\kappa \leftarrow \kappa + 1$
    $a \xleftarrow{\$} \mathcal{U}_q$
    $s, e \xleftarrow{\$} \chi_e$
    $b = a \cdot s + pe$
    $\mathsf{pk}_\kappa = (a, b), \mathsf{sk}_\kappa = s$
    return $\mathsf{pk}_\kappa$
```

```
$\mathsf{Chal}_{\mathsf{RRLWE}}(i)$
    $(a, b) \leftarrow \mathsf{pk}_i$
    $v, e_0, e_1 \xleftarrow{\$} \chi_e$
    $(\bar{a}_0, \bar{b}_0) = (av + pe_0, bv + pe_1)$
    $(\bar{a}_1, \bar{b}_1) \xleftarrow{\$} \mathcal{U}_q \times \mathcal{U}_q$
    return $(\bar{a}_d, \bar{b}_d)$
```

Figure V.9: A variant of RLWE for resampling. Note that the challenge oracle can be queried more than once.

PPT *adversaries $\mathcal{A}$, there exists a RLWE distinguisher $\mathcal{D}$ such that:*

$$\Pr\left[\mathsf{ReSample\text{-}ncRLWE}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \frac{1}{2} + (\kappa + 2Q) \cdot \mathbf{Adv}_{(\mathcal{D})}^{RLWE_{\phi,q,\chi_e}}(1^\lambda),$$

where κ is the number of keys generated in $\mathsf{ReSample\text{-}ncRLWE}_{\mathcal{A}}(1^\lambda)$ *and Q is the number of calls that $\mathcal{A}$ makes to* $\mathsf{Chal}_{\mathsf{RRLWE}}$. *Because $\mathcal{A}$ is* PPT, *κ and Q must also be polynomial, meaning the overall advantage of winning* $\mathsf{ReSample\text{-}ncRLWE}_{\mathcal{A}}(1^\lambda)$ *is negligible, by the RLWE assumption.*

Proof. Let $\mathcal{D}$ receive an RLWE challenge, where either the sample is genuine ($d = 0$) or it is random ($d = 1$). We demonstrate via a hybrid argument that $\mathcal{D}$ can simulate $\mathsf{ReSample\text{-}ncRLWE}_{\mathcal{A}}(1^\lambda)$.

Let $\mathsf{H}_0 := \mathsf{ReSample\text{-}ncRLWE}_{\mathcal{A}}(1^\lambda)$ with challenge bit $d = 0$, and let H_i be defined in the same way as H_{i-1} except that the first i public keys have been replaced with uniform random values. Suppose there exists an adversary $\mathcal{B}$ that can distinguish between H_{i-1} and H_i with advantage ϵ. We demonstrate how an RLWE distinguisher $\mathcal{D}$ can break RLWE with the same advantage by simulating H_i and running $\mathcal{B}$ as a subroutine. $\mathcal{D}$ can simulate H_{i-d} by responding to calls $\mathcal{B}$ makes to $\mathsf{O}_{\mathsf{KeyGen}}$ as follows:

- if $k < i$ then $\mathcal{D}$ returns $(u_0, u_1) \xleftarrow{\$} \mathcal{U}_q^2$,
- if $k = i$ $\mathcal{D}$ returns an RLWE challenge (a, b),
- if $k > i$ then $\mathcal{D}$ returns a genuine RLWE sample.

Then $\mathcal{D}$ returns $d' = d'_{\mathcal{A}}$. The maximal advantage in distinguishing between H_i and H_{i-1} is therefore $\mathbf{Adv}_{(\mathcal{D})}^{RLWE_{\phi,q,\chi_e}}(1^\lambda)$.

Now consider a further hybrid argument $\bar{\mathsf{H}}_0 \ldots \bar{\mathsf{H}}_Q$ where $\bar{\mathsf{H}}_0$ is identical to H_κ and $\bar{\mathsf{H}}_i$

is identical to $\bar{\mathsf{H}}_{i-1}$ except that the ith output of $\mathsf{O}_{\mathsf{Challenge}}$ is replaced with $2 \cdot (\bar{a}, \bar{b}) \xleftarrow{\$} \mathcal{U}_q^2$.

Because all public keys are now random elements of $\mathcal{U}_q$, this means that all inputs to ReSample are uniform random elements, meaning that $\bar{a}_0 = av + pe_0$ and $\bar{b}_0 = bv + pe_1$ are also RLWE samples. Therefore, using the RLWE assumption twice more for each call to $\mathsf{Chal}_{\mathsf{RRLWE}}$, we see that the maximal advantage in distinguishing between $\bar{\mathsf{H}}_i$ and $\bar{\mathsf{H}}_{i-1}$ is $2 \cdot \mathbf{Adv}_{(\mathcal{D})}^{RLWE_{\phi,q,\chi_e}}(1^\lambda)$. Alternatively, this can be seen in the same way as the IND-CPA security of the underlying encryption scheme [PRSV17, proof of Theorem 5.1], as ReSample is the same as encrypting 0.

Overall, we get that the number of times we rely on RLWE is $\kappa + 2Q$. This completes the proof. $\square$

To get to selective key corruption models, we need to consider a variant of ReSample-ncRLWE where $\mathcal{A}$ is also given access to a key corruption oracle. We describe this variant, which we call ReSample-RLWE, in Figure V.10.

```
$\mathsf{ReSample\text{-}RLWE}_{\mathcal{A}}(1^\lambda)$
$d \xleftarrow{\$} \{0,1\}$
$\kappa = 0$
$\mathcal{K}_{\mathsf{corrupt}}, \mathcal{K}_{\mathsf{chal}} = \emptyset$
$d' \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Corrupt}}, \mathsf{Chal}_{\mathsf{RRLWE}}(i)}(1^\lambda)$
return $(d' = d)$
```

```
$\mathsf{O}_{\mathsf{KeyGen}}()$
$\kappa \leftarrow \kappa + 1$
$a \xleftarrow{\$} \mathcal{U}_q$
$s, e \xleftarrow{\$} \chi_e$
$b = a \cdot s + pe$
$\mathsf{pk}_\kappa = (a, b)$
$\mathsf{sk}_\kappa = s$
return $\mathsf{pk}_\kappa$
```

```
$\mathsf{O}_{\mathsf{Corrupt}}(i)$
if $\mathsf{sk}_i \in \mathcal{K}_{\mathsf{chal}}$ :
    return $\perp$
$\mathcal{K}_{\mathsf{corrupt}}.\mathsf{add}\{\mathsf{sk}_i\}$
return $\mathsf{sk}_i$
```

```
$\mathsf{Chal}_{\mathsf{RRLWE}}(i)$
if $\mathsf{sk}_i \in \mathcal{K}_{\mathsf{corrupt}}$ :
    return $\perp$
$(a, b) \leftarrow \mathsf{pk}_i$
$v, e_0 \xleftarrow{\$} \chi_e, e_1 \xleftarrow{\$} \chi_e$
$(\bar{a}_0, \bar{b}_0) = (av + pe_0, bv + pe_1)$
$(\bar{a}_1, \bar{b}_1) \xleftarrow{\$} \mathcal{U}_q \times \mathcal{U}_q$
$\mathcal{K}_{\mathsf{chal}}.\mathsf{add}\{\mathsf{sk}_i\}$
return $(\bar{a}_d, \bar{b}_d)$
```

Figure V.10: A variant of RLWE for resampling with key corruptions, where $d \in \{0,1\}$. As in ReSample-ncRLWE, the challenge oracle can be queried more than once.

To prove that no PPT adversary can win $\mathsf{ReSample\text{-}RLWE}_{\mathcal{A}}(1^\lambda)$ with non-negligible advantage, we rely on the following assumption, which we call the *challenge independence assumption*. Let $\{q\}$ denote the set of queries made in a security game Game that are independent of any challenge queries. Then $\{q\}$ can give no advantage to an adversary hoping to win Game. This is relevant for key corruptions as, if the adversary corrupts keys which are never linked to any challenge keys, the challenge independence assumption means these keys give the adversary no additional advantage. This principle is similar to the intuition of how re-encryption simulatability and source-hiding imply a PRE scheme is also

IND-HRA-secure, as replacing re-encryptions with ciphertexts independent of the challenge secret key cannot yield an advantage in learning challenge keys. A similar assumption is made in [FKKP19], where Fuchsbauer et al. mention that it suffices to consider the challenge graph – the subgraph of the $\mathcal{DRG}$ that is reachable from the challenge node.

Lemma V.13. *Let* ReSample-RLWE$_{\mathcal{A}}(1^\lambda)$ *be defined as in Figure V.10. Then an adversary who wins* ReSample-ncRLWE$_{\mathcal{A}}(1^\lambda)$ *with advantage* ϵ *can win* ReSample-RLWE$_{\mathcal{A}}(1^\lambda)$*with advantage* ϵ *by the challenge independence assumption.*

Proof. We note that all keys are generated independently and that in ReSample-RLWE$_{\mathcal{A}}(1^\lambda)$ keys either become challenge keys or corrupted keys. Furthermore, ReSample-RLWE$_{\mathcal{A}}(1^\lambda)$ gives no information linking individual samples. This means the challenge independence assumption, meaning that providing the adversary with the corruption oracle, $\mathsf{O}_{\mathsf{Corrupt}}$, gives no additional advantage in winning. □

Combining Lemma V.12 and Lemma V.13 gives the following result.

Lemma V.14. *For all* PPT *adversaries* $\mathcal{A}$*, there exists an RLWE distinguisher* $\mathcal{D}$ *such that:*

$$\Pr\Big[\mathsf{ReSample\text{-}RLWE}_{\mathcal{A}}(1^\lambda) = 1\Big] \leq \frac{1}{2} + (\kappa + 2Q) \cdot \mathbf{Adv}_{(\mathcal{D})}^{RLWE_{\phi,q,\chi_e}}(1^\lambda), \tag{V.13}$$

where κ *is the number of keys generated in* ReSample-ncRLWE$_{\mathcal{A}}(1^\lambda)$ *and* Q *is the number of calls that* $\mathcal{A}$ *makes to* $\mathsf{Chal}_{\mathsf{RRLWE}}$*. As in Lemma V.12, this means the advantage of winning* ReSample-ncRLWE$_{\mathcal{A}}(1^\lambda)$ *is negligible by the RLWE assumption.*

V: 5.4 Post-Compromise Security of pcBV-PRE

Recall from Section V: 3.4 that pcBV-PRE is not source-hiding, and we therefore do not leverage Theorem V.3. Instead we use an alternative method to show that re-encryptions of non-challenge ciphertexts can be replaced without detection.

Theorem V.4. *For all* PPT *adversaries* $\mathcal{A}$*, there exists an RLWE distinguisher* $\mathcal{D}$ *such* *that:*

$$\Pr\left[\mathsf{PostComp}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} +$$
$$(2\,(\kappa + (Q_{RKu\to u} + Q_{RE\perp u\to u})(\lfloor \log_2(q)/r \rfloor + 1) + Q_{Cu} + 1) + Q_{Hu\to c}) \cdot \mathbf{Adv}_{\mathcal{D}}^{RLWE_{\phi,q,\chi_e}}(1^\lambda),$$

where κ *is the number of keys generated during* $\mathsf{PostComp}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda)$, $Q_{RKu\to u}$ *is the number of queries* $\mathcal{A}$ *makes to* $\mathsf{O}_{\mathsf{ReKeyGen}}(i,j)$ *where* sk_i *and* sk_j *are uncorrupted,* $Q_{RE\perp u\to u}$ *is the number of queries* $\mathcal{A}$ *makes to* $\mathsf{O}_{\mathsf{ReKeyGen}}(c,i,j,\Delta_{i,j})$ *where* sk_i *and* sk_j *are uncorrupted and* $\Delta_{i,j} = \perp$, *and* Q_{Cu} *is the number of non-challenge ciphertexts under corrupted keys (counting both fresh encryptions and re-encryptions).*
Because $\mathcal{A}$ *is* PPT, *we note that* $\kappa, Q_{RKu\to u}, Q_{RE\perp u\to u}$ *and* Q_{Cu} *are all polynomial, meaning that the overall advantage is negligible by the RLWE assumption and thus* pcBV-PRE *has PCS.*

Proof. We give a brief informal argument before giving a full proof. We first demonstrate that the scheme is secure for a variant of the game where no re-encryptions from uncorrupted to corrupted keys are permitted. We call this variant $\mathsf{CPA\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$.

This proof follows a hybrid argument using a sequence of game hops beginning with $\mathsf{CPA\text{-}PostComp}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda)$. We replace:

1. outputs of the ReSample used within pcBV-PRE.ReKeyGen where sk_i and sk_j are uncorrupted with uniform random values, and
2. outputs of pcBV-PRE.ReKeyGen where sk_i and sk_j are uncorrupted with uniform random values.

In the final game hop, the challenge ciphertext given to the adversary is a uniformly sampled value and thus the adversary has no advantage in this game. This implies that $\mathsf{CPA\text{-}PostComp}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda)$ when $b = 0$ is indistinguishable from when $b = 1$. This first part of the proof of Theorem V.4 is formulated in Theorem V.5.

To prove security for the full $\mathsf{PostComp}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda)$ game including honest re-encryptions, we use a similar idea to re-encryption simulatability to demonstrate how re-encryptions from uncorrupted to corrupted keys can be simulated without the update token or knowledge of the old secret key. We demonstrate how this simulated game is indistinguishable from $\mathsf{PostComp}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda)$, meaning that pcBV-PRE PCS. This second part of the proof of Theorem V.4 is formulated in Theorem V.6.

We now proceed to the first stage of the full proof.

Theorem V.5. *Let* $\mathsf{CPA\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ *be a variant of* $\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ *where no re-encryptions from uncorrupted to corrupted keys are permitted whatsoever.*

Then for all PPT *adversaries* $\mathcal{A}$*, there exists an RLWE distinguisher* $\mathcal{D}$ *such that:*

$$\Pr\left[\mathsf{CPA\text{-}PostComp}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \\ (\kappa + 2(Q_{RKu\to u} + Q_{RE\bot u\to u})(\lfloor \log_2(q)/r \rfloor + 1) + 2) \cdot \mathbf{Adv}_{\mathcal{D}}^{RLWE_{\phi,q,\chi_e}}(1^\lambda),$$

where $\kappa, Q_{RKu\to u}$ *and* $Q_{RE\bot u\to u}$ *are as described in Theorem V.4.*

Proof. We prove this theorem using a sequence of game hops as described in Lemmas V.15 to V.17.

Lemma V.15. *Let* Game_0 *identical to* $\mathsf{CPA\text{-}PostComp}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda)$*. Let* Game_1 *the same as* Game_0*, except that all update tokens* $\left(\{(\beta_\iota, \gamma_\iota = \theta_\iota - \mathsf{sk}_i \cdot (2^r)^\iota)\}_{\iota=0}^{\lfloor \log_2(q)/r \rfloor}, \mathsf{pk}_j\right)$ *between uncorrupted keys are replaced with* $\left(\{(\beta_\iota, \theta_\iota)\}_{\iota=0}^{\lfloor \log_2(q)/r \rfloor}, \mathsf{pk}_j\right)$*. Then for all* PPT *adversaries* $\mathcal{A}$*:*

$$|\Pr[1 \leftarrow \mathsf{Game}_0(\mathcal{A})] - \Pr[1 \leftarrow \mathsf{Game}_1(\mathcal{A})]| \\ \leq (\kappa + 2(Q_{RKu\to u} + Q_{RE\bot u\to u})(\lfloor \log_2(q)/r \rfloor + 1)) \cdot \mathbf{Adv}_{\mathcal{D}}^{RLWE_{\phi,q,\chi_e}}(1^\lambda).$$

In other words, the advantage in distinguishing between Game_0 *and* Game_1 *is the at most the advantage of distinguishing one of* $\kappa + 2(Q_{RKu\to u} + Q_{RE\bot u\to u})(\lfloor \log_2(q)/r \rfloor + 1)$ *RLWE samples from uniform.*

Proof. Note that if $x = r + u \in \mathcal{R}_q$, where $r \xleftarrow{\$} \mathcal{U}_q$, then x is indistinguishable from $x' \xleftarrow{\$} \mathcal{U}_q$, and vice versa. This applies as long as r is unknown, even if x is known. This means we can replace $\Delta = \{(\bar{a}_\iota, \bar{b}_\iota - \mathsf{sk}_i)_{\iota=0}^{\lfloor \log_2(q)/r \rfloor}\} \xleftarrow{\$} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$, where $\{(\bar{a}_\iota, \bar{b}_\iota) \xleftarrow{\$} \mathsf{ReSample}(\mathsf{pk}_j)\}_{\iota=0}^{\lfloor \log_2(q)/r \rfloor}$, with $\Delta = \{(\bar{a}_\iota, \bar{b}_\iota)_{\iota=0}^{\lfloor \log_2(q)/r \rfloor}\} \xleftarrow{\$} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$ if and only if we can replace calls to $\mathsf{ReSample}(\mathsf{pk}_j)$ with uniform random values.

We demonstrate how an adversary $\mathcal{B}$ for $\mathsf{ReSample\text{-}RLWE}_{\mathcal{B}}(1^\lambda)$ with challenge bit $d \xleftarrow{\$} \{0,1\}$ can simulate Game_d:

- $\mathcal{B}$ samples $b \xleftarrow{\$} \{0,1\}$ to be the bit determining the challenge re-encryption.
- When $\mathcal{A}$ calls $\mathsf{O}^{\mathsf{Game}_d}_{\mathsf{KeyGen}}$ and $\mathsf{O}^{\mathsf{Game}_d}_{\mathsf{Corrupt}}$, $\mathcal{B}$ forwards these queries to the equivalent oracles in $\mathsf{ReSample\text{-}RLWE}$.
- When $\mathcal{A}$ calls $\mathsf{O}^{\mathsf{Game}_d}_{\mathsf{ReKeyGen}}(i,j)$:
  - If sk_i is not corrupted and sk_j is corrupted, $\mathcal{B}$ returns $\perp$.
  - If sk_i is corrupted, $\mathcal{B}$ returns $\Delta_{i,j} \xleftarrow{\$} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$.
  - If sk_i and sk_j are not corrupted, $\mathcal{B}$ uses the $\mathsf{ReSample\text{-}RLWE}$ challenger to obtain $\{(\bar{a}_\iota, \bar{b}_\iota) \xleftarrow{\$} \mathsf{Chal}_{\mathsf{RRLWE}}(j)\}_{\iota=0}^{\lfloor \log_2(q)/r \rfloor}$ and sets $\Delta \leftarrow \{(\bar{a}_\iota, \bar{b}_\iota)\}_{\iota=0}^{\lfloor \log_2(q)/r \rfloor}$.
- When $\mathcal{A}$ calls $\mathsf{O}^{\mathsf{Game}_d}_{\mathsf{ReEnc}}(i,j,\Delta_{i,j},c)$, $\mathcal{B}$ responds as described in $\mathsf{PostComp}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$, using the same method as for $\mathsf{O}^{\mathsf{Game}_d}_{\mathsf{ReKeyGen}}$ if $\Delta_{i,j} = \perp$.
- When $\mathcal{A}$ calls $\mathsf{Chal}^{\mathsf{Game}_d}_{\mathsf{ReEnc}}(i,j,\Delta_{i,j},c_0,c_1)$, $\mathcal{B}$ responds as it would for $\mathsf{O}^{\mathsf{Game}_d}_{\mathsf{ReEnc}}(i,j,\Delta_{i,j},c_b)$.

If $d = 0$ then resamples are genuine so this perfectly simulates Game_0, and if $d = 0$ then this perfectly simulates Game_1. Therefore, for an adversary $\mathcal{A}$ that outputs $d'_{\mathcal{A}} = d$ with advantage δ, $\mathcal{B}$ can set $d' = d'_{\mathcal{A}}$ and win $\mathsf{ReSample\text{-}RLWE}_{\mathcal{B}}(1^\lambda)$ with the same advantage.

Overall, $\mathcal{B}$ calls $\mathsf{Chal}_{\mathsf{RRLWE}}$ $\lfloor \log_2(q)/r \rfloor + 1$ times for each call $\mathcal{A}$ makes to $\mathsf{O}_{\mathsf{ReKeyGen}}(i,j)$ where sk_i is uncorrupted, and a further $\lfloor \log_2(q)/r \rfloor + 1$ times whenever $\mathcal{A}$ calls $\mathsf{O}_{\mathsf{ReEnc}}(c,i,j,\Delta_{i,j} = \perp)$ where sk_i is uncorrupted. In total, $\mathsf{Chal}_{\mathsf{RRLWE}}$ was called at most $(Q_{RKu\to u} + Q_{RE\perp u \to u})(\lfloor \log_2(q)/r \rfloor + 1)$ times. Lemma V.14 gives us the result. □

Lemma V.16. *Let* Game_2 *be identical to* Game_1*, except that when the challenge oracle* $\mathsf{Chal}^{\mathsf{Game}_2}_{\mathsf{ReEnc}}(i,j,[\Delta_{i,j},]\ c_0,c_1)$ *is queried, calls to* $\mathsf{ReSample}(\mathsf{pk}_j)$ *are replaced with uniform random values. Then for all* PPT *adversaries* $\mathcal{A}$*, there exists an RLWE distinguisher* $\mathcal{D}$ *such that:*

$$\left| \Pr\left[1 \xleftarrow{(\$)} \mathsf{Game}_1(\mathcal{A})\right] - \Pr\left[1 \xleftarrow{(\$)} \mathsf{Game}_2(\mathcal{A})\right] \right| \leq 2 \cdot \mathbf{Adv}^{RLWE_{\phi,q,\chi_e}}_{\mathcal{D}}(1^\lambda).$$

In other words, the advantage in distinguishing between Game_1 *and* Game_2 *is the same as distinguishing two RLWE samples from uniform.*

Proof. Without loss of generality, we assume that when $\mathcal{A}$ queries the challenge oracle $\mathsf{Chal}_{\mathsf{ReEnc}}(c_0, c_1, i, j, \Delta_{i,j})$, it uses an oracle-generated updated token $\Delta_{i,j} \in \mathcal{T}_{\mathsf{honest}}$. Let $d \xleftarrow{\$} \{0,1\}$ be the challenge bit in $\mathsf{ReSample\text{-}RLWE}_{\mathcal{B}}(1^\lambda)$. We demonstrate that if there exists an adversary $\mathcal{A}$ that outputs $d'_{\mathcal{A}} = d$ for Game_{d+1} with advantage δ, then there exists an adversary $\mathcal{B}$ who can win $\mathsf{ReSample\text{-}RLWE}_{\mathcal{B}}(1^\lambda)$.

As before, we show how $\mathcal{B}$ can use $\mathsf{ReSample\text{-}RLWE}_{\mathcal{B}}(1^\lambda)$ to simulate Game_{d+1}. $\mathcal{B}$ first samples $b \xleftarrow{\$} \{0,1\}$ to be the challenge bit in Game_{d+1}, and responds to all oracles except the challenge as described in Lemma V.15. When $\mathcal{A}$ calls $\mathsf{Chal}^{\mathsf{Game}_d}_{\mathsf{ReEnc}}(i, j, \Delta_{i,j}, c_0, c_1)$, $\mathcal{B}$ performs the appropriate checks, then follows the procedure for $\mathsf{pcBV\text{-}PRE.ReEnc}(\Delta_{i,j}, c_b)$ except that it replaces $\mathsf{ReSample}(\mathsf{pk}_j)$ with a call to $\mathsf{Chal}_{\mathsf{RRLWE}}(j)$.

When $d = 0$, this simulates Game_1 perfectly and when $d = 1$, this simulates Game_2 perfectly. Therefore, $\mathcal{B}$ can run $\mathcal{A}$ on the simulation and return $d' = d'_{\mathcal{A}}$ to win with the same advantage. Since $\mathsf{Chal}_{\mathsf{RRLWE}}(j)$ has only been called once, we rely on the RLWE assumption two extra times. This gives the result. □

Lemma V.17. *Let* Game_3 *be identical to* Game_2*, except that* $\mathsf{Chal}^{\mathsf{Game}_3}_{\mathsf{ReEnc}}(i, j, \Delta_{i,j}, c_0, c_1)$ *returns uniform random values. For all adversaries* $\mathcal{A}$*:*

$$\Pr\left[1 \overset{(\$)}{\leftarrow} \mathsf{Game}_2(\mathcal{A})\right] = \Pr\left[1 \overset{(\$)}{\leftarrow} \mathsf{Game}_3(\mathcal{A})\right].$$

Proof. In both Game_2 and Game_3, the adversary submits two ciphertexts $c_0^* = (c_{0,0}^*, c_{0,1}^*)$ and $c_1^* = (c_{1,0}^*, c_{1,1}^*)$ when it queries the challenge oracle. In Game_2, the challenger responds by setting $c_0 = c_{0(b)}^*$ and $c_1 = c_{1(b)}^*$ for uniformly chosen $b \xleftarrow{\$} \{0,1\}$ and returning the challenge ciphertext given by

$$c_0' = c_0 + b^* + \sum_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} (c_1^{(\iota)} \cdot \gamma_i), \quad c_1' = a^* + \sum_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} (c_1^{(\iota)} \cdot a_i),$$

using the relinearisation breakdown described in Equation (V.1). In particular, a^* and b^* are the result of calling $\mathsf{ReSample}(\mathsf{pk}_j)$ in Game_2 on an honest public key and are therefore uniform random values that are used once and never revealed to $\mathcal{A}$. Therefore, c_0' and c_1' are also independent uniform random values in Game_2, which is the exact case in Game_3.

This argument holds both when the challenge bit $b = 0$ and when $b = 1$. This concludes the proof of this lemma. $\square$

Using Lemmas V.15 to V.17 together with the triangle inequality, we get that

$$\Pr\left[\mathsf{CPA\text{-}PostComp}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + (\kappa + 2(Q_{RKu\to u} + Q_{RE\perp u\to u})(\lfloor \log_2(q)/r \rfloor + 1) + 2) \cdot \mathbf{Adv}_{\mathcal{D}}^{RLWE_{\phi,q,\chi_e}}(1^\lambda).$$

This concludes the proof of Theorem V.5. $\square$

Theorem V.6. *If for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*:*

$$\Pr\left[\mathsf{PostComp}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon + (\kappa + 2Q_{Cu} + Q_{Hu\to c}) \cdot \mathbf{Adv}_{\mathcal{D}}^{RLWE_{\phi,q,\chi_e}}(1^\lambda),$$

where ϵ *is the maximal advantage of winning* $\mathsf{CPA\text{-}PostComp}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda)$*, and* κ, Q_{Cu} *and* $Q_{Hu\to c}$ *are as described in Theorem V.4.*

Proof. This is similar to the proof outline of [Coh17, Theorem 5]. We observe that an adversary $\mathcal{B}$ for $\mathsf{CPA\text{-}PostComp}_{\mathcal{B}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda)$ can simulate $\mathsf{PostComp}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda)$ by replacing the outputs from calls to $\mathsf{O}_{\mathsf{ReEnc}}(i, j, [\Delta_{i,j},]c)$, where $\mathsf{sk}_i \notin \mathcal{K}_{\mathsf{corrupt}}$, $\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}}$ and $(i, c) \in \mathcal{C}_{\mathsf{honest}}$ but $(i, c) \notin \mathcal{C}_{\mathsf{chal}}$, with a fake re-encryption. If this replacement is not detectable, then $\mathcal{B}$ can call $\mathcal{A}$ as a subroutine, output the same guess and win with the same advantage.

Without loss of generality, for simplicity we give the proof for the variant of our construction described in Figure V.11, where $\lfloor \log_2(q)/r \rfloor = 0$ $(r \geq \log_2(q))$.

Note that for $b_i = a_i s_i + p e_i$,

$$b_i - a_i s_i = p e_i. \tag{V.14}$$

Let $(\mathsf{pk}_i, \mathsf{sk}_i) = ((a_i, b_i = a_i s_i + p e_i), s_i)$, $(\mathsf{pk}_j, \mathsf{sk}_j) = ((a_j, b_j = a_j s_j + p e_j), s_j)$, let $\bar{a} = a_j \bar{v} + p\bar{e_1}$, $\bar{b} = b_j \bar{v} + p\bar{e_0}$, $\hat{a} = a_j \hat{v} + p\hat{e_1}$, $\hat{b} = b_j \hat{v} + p\hat{e_0}$ and let $c, \Delta_{i,j}, c'$ be as described in Figure V.11. We first note that by Equation (V.2), a genuine re-encryption will result in

```
$c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m)$
    $(a_i, b_i) \leftarrow \mathsf{pk}_i$
    $\tilde{v}, \tilde{e_0}, \tilde{e_1} \xleftarrow{\$} \chi_e$
    $c_0 = b_i\tilde{v} + p\tilde{e_0} + m$
    $c_1 = a_i\tilde{v} + p\tilde{e_1}$

$m' \leftarrow \mathsf{Dec}(\mathsf{sk}_i = s_i, c)$
    $(c_0, c_1) \leftarrow c$
    $m' = c_0 - c_1 s_i \mod p$

$\Delta_{i,j} \xleftarrow{\$} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
    $(\bar{a}, \bar{b}) \leftarrow \mathsf{ReSample}(\mathsf{pk}_j)$
    $\gamma = \bar{b} - s_i$
    $\Delta_{i,j} = (\bar{a}, \gamma)$

$c' \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, c)$
    $(\bar{a}, \gamma) \leftarrow \Delta_{i,j}$
    $(\hat{a}, \hat{b}) \leftarrow \mathsf{ReSample}(\mathsf{pk}_j)$
    $c'_0 = c_0 + c_1\gamma + \hat{b}$
    $c'_1 = c_1\bar{a} + \hat{a}$
```

Figure V.11: A simplified version of pcBV-PRE (Figure V.8).

a ciphertext $c' = (c'_0, c'_1)$, where

$$\begin{aligned}
c'_0 &= c_0 + c_1\gamma + \hat{b} \\
&= c_0 + c_1(\bar{b} - s_i) + \hat{b} \\
&= c_0 - s_i c_1 + c_1\bar{b} + \hat{b} \\
&\overset{(V.2)}{=} p(\tilde{e_0}s_i - \tilde{e_1} + e_i\tilde{v}) + m + c_1(b_j\bar{v} + p\bar{e_0}) + b_j\hat{v} + p\hat{e_0} \\
&= b_j(c_1\bar{v} + \hat{v}) + p(c_1\bar{e_0} + \hat{e_0} + \tilde{e_0}s_i - \tilde{e_1} + e_i\tilde{v}) + m, \\
c'_1 &= c_1\bar{a} + \hat{a} \\
&= c_1(a_j\bar{v} + p\bar{e_1}) + a_j\hat{v} + p\hat{e_1} \\
&= a_j(c_1\bar{v} + \hat{v}) + p(c_1\bar{e_1} + \hat{e_1}).
\end{aligned}$$

If we let

$$\mathbf{v} = c_1\bar{v} + \hat{v}, \qquad \mathbf{e_0} = c_1\bar{e_0} + \hat{e_0} + \tilde{e_0}s_i - \tilde{e_1} + e_i\tilde{v}, \qquad \mathbf{e_1} = c_1\bar{e_1} + \hat{e_1}, \tag{V.15}$$

we get

$$\begin{aligned}
c'_0 &= b_j\mathbf{v} + p\mathbf{e_0} + m, \\
c'_1 &= a_j\mathbf{v} + p\mathbf{e_1}.
\end{aligned} \tag{V.16}$$

Therefore, if the secret key s_i (and therefore e_i), message m and random values $\tilde{v}, \tilde{e_0}, \tilde{e_1}$ were known, an alternative way to compute a genuine re-encryption would be to sample

$\bar{v}, \bar{e_0}, \bar{e_1}, \hat{v}, \hat{e_0}, \hat{e_1} \xleftarrow{\$} \chi_e$ and compute c' as described in Equations (V.15) and (V.16).

We note that, as $\bar{v}, \bar{e_0}, \bar{e_1}, \hat{v}, \hat{e_0}, \hat{e_1}$ are freshly sampled for one-time use, $\mathcal{B}$ can easily sample its own values from χ_e and use these to calculate $\mathbf{v}, \mathbf{e_1}$. However, since the simulator does not know s_i, e_i, they cannot compute $\mathbf{e_0}$. Instead $\mathcal{B}$ can sample its own $s_i{}', e_i{}' \xleftarrow{\$} \chi_e$ and use these to create

$$\mathbf{e_0'} = c_1\bar{e_0} + \hat{e_0} + \tilde{e_0}s_i{}' - \tilde{e_1} + e_i{}'\tilde{v},$$

and create a fake re-encryption:

$$c'^* = (b_j\mathbf{v} + p\mathbf{e_0'} + m, a_j\mathbf{v} + p\mathbf{e_1}). \tag{V.17}$$

We need to demonstrate that this replacement will not be detected.

We note that, since pk_i is uncorrupted, the distinguisher does not know $\mathsf{sk}_i = s_i$, and therefore cannot use the input ciphertext c to learn the ciphertext error $E_c = \tilde{e_0}s_i - \tilde{e_1} + e_i\tilde{v}$, as otherwise they could derive m from c without knowing the secret key, thereby breaking RLWE. However, as they do know $\mathsf{sk}_j = s_j$, they can compute the error of a re-encrypted ciphertext. Since

$$\begin{aligned}
c_0' - c_1's_j &= (c_0 + c_1\gamma + \hat{b}) - (c_i\bar{a} + \hat{a})s_j \\
&= c_1(\gamma - \bar{a}s_j) + c_0 + \hat{b} - \hat{a}s_j \\
&= c_1(\bar{b} - s_i - \bar{a}s_j) + c_0 + \hat{b} - \hat{a}s_j \\
&= c_0 + c_1(\bar{b} - \bar{a}s_j - s_i) + (b_j\hat{v} + p\hat{e_0}) - (a_j\hat{v} + p\hat{e_1})s_j \\
&= c_0 - c_1s_i + c_1(\bar{b} - \bar{a}s_j) + (b_j - a_js_j)\hat{v} + p(\hat{e_0} - \hat{e_1}s_j) \\
&\overset{(V.14)}{=} m + pE_c + c_1p(e_j\bar{v} + \bar{e_0} - \bar{e_1}s_j) + pe_j\hat{v} + p(\hat{e_0} - \hat{e_1}s_j) \\
&= m + p(E_c + \hat{e_0} - \hat{e_1}s_j + e_j\hat{v} + c_1(e_j\bar{v} + \bar{e_0} - \bar{e_1}s_j)),
\end{aligned}$$

the error on a genuine re-encryption c' is

$$E_{c'} = \tilde{e_0}s_i - \tilde{e_1} + e_i\tilde{v} + \hat{e_0} - \hat{e_1}\underline{s_j} + \underline{e_j}\hat{v} + \underline{c_1}(\underline{e_j}\bar{v} + \bar{e_0} - \bar{e_1}\underline{s_j}),$$

and for a fake re-encryption c'^* is

$$E_{c'^*} = \tilde{e_0}{s_i}' - \tilde{e_1} + {e_i}'\tilde{v} + \hat{e_0} - \hat{e_1}\underline{s_j} + \underline{e_j}\hat{v} + \underline{c_1}(\underline{e_j}\bar{v} + \bar{e_0} - \bar{e_1}\underline{s_j}),$$

where underlined values are known to the adversary.

As $c_1 = a_i\tilde{v} + p\tilde{e_1}$, and a_i is public, (a_i, c_1) is an RLWE sample with secret $\tilde{v}$ and error $p\tilde{e_1}$. Because $\tilde{v}$ and $\tilde{e_1}$ are chosen independently of s_i and e_i, it follows that even if s_i and e_i were later corrupted, $\tilde{v}$ and $\tilde{e_1}$ should remain unknown. As E_c is the only other value derived using $\tilde{v}, \tilde{e_1}$ and it is also unknown to the adversary, we can therefore argue that c_1 is indistinguishable from a value chosen uniformly at random, by the RLWE assumption.

Rearranging, we need to demonstrate that

$$E_{c'} = \underline{c_1}\bar{e_0} + \hat{e_0} - \hat{e_1}\underline{s_j} + \underline{e_j}\hat{v} + \underline{c_1}(\underline{e_j}\bar{v} - \bar{e_1}\underline{s_j}) + E_c \tag{V.18}$$

and

$$E_{c'^*} = \underline{c_1}\bar{e_0} + \hat{e_0} - \hat{e_1}\underline{s_j} + \underline{e_j}\hat{v} + \underline{c_1}(\underline{e_j}\bar{v} - \bar{e_1}\underline{s_j}) + E_c^*, \tag{V.19}$$

where $E_c^* = \tilde{e_0}{s_i}' - \tilde{e_1} + {e_i}'\tilde{v}$, appear to have the same distribution to any entity that only knows m, c, s_j and $c'^{(*)}$.

We now observe that because c is an encryption under an uncorrupted key, we can apply the RLWE assumption to suggest that both c_0 and c_1 are indistinguishable from uniform random elements of $\mathcal{R}_q$. We can demonstrate this explicitly via $\mathsf{ReSample\text{-}RLWE}_{\mathcal{A}}(1^\lambda)$, but omit this full explanation for brevity. In doing this, we invoke the RLWE assumption at least $\kappa + 2Q_{Cu}$ times (Lemma V.14).

Because c_1 appears to be a uniform element of $\mathcal{R}_q$, $c_1\bar{e_0} + \hat{e_0}$ is also distributed like an RLWE sample with secret $\bar{e_0}$ and error $\hat{e_0}$. This invokes the RLWE assumption once more per re-encryption replacement (an additional $Q_{Hu\to c}$ times). The distributions of Equations (V.18) and (V.19) are therefore indistinguishable, under the RLWE assumption, from the distributions of a modified version of Equations (V.18) and (V.19) where $c_1\bar{e_0} + \hat{e_0}$ is replaced with $u \xleftarrow{\$} \mathcal{U}_q$[4].

[4]This is a simplification, as we need u to be such that the resulting $E_{c'}E_{c'^*}$ are the appropriate size for

Finally, we note that if w is an element of $\mathcal{R}_q$ and $u \xleftarrow{\$} \mathcal{U}_q$, then $u+w$ is also distributed as a uniform element of $\mathcal{R}_q$. Therefore, because $c_1\bar{e_0} + \hat{e_0}$ can be replaced with random elements of the appropriate distribution, we can replace the entirety of both E'_c and E'^{*}_c with random elements from the appropriate distribution for the number of times the ciphertext has been re-encrypted. We conclude that $\mathbf{e_0}$ and $\mathbf{e'_0}$ are indistinguishable.

We get an analogous result for the full scheme by setting

$$\mathbf{v} = \left(\sum_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} c_1^{(\iota)} \cdot \bar{v}_\iota \right) + \hat{v}, \quad \mathbf{e'_0} = \left(\sum_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} c_1^{(\iota)} \cdot \bar{e_0}^{(\iota)} \right) + \hat{e_1} + \tilde{e_0} {s_i}' - \tilde{e_1} + {e_i}'\tilde{v},$$

$$\mathbf{e_1} = \left(\sum_{\iota=0}^{\lfloor \log_2(q)/r \rfloor} c_1^{(\iota)} \cdot \bar{e_1}^{(\iota)} \right) + \hat{e_1}.$$

Thus, $\mathcal{B}$ can use fake re-encryptions to simulate $\mathsf{PostComp}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda)$ and run $\mathcal{A}$ to win $\mathsf{CPA\text{-}PostComp}_{\mathcal{B}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda)$ with overall advantage of at most $\left(\epsilon + (\kappa + 2Q_{Cu} + Q_{Hu\to c}) \cdot \mathbf{Adv}_{\mathcal{D}}^{RLWE_{\phi,q,\chi_e}}(1^\lambda)\right)$, by Lemma V.14 and the triangle inequality. This completes the proof.

□

The proof of Theorem V.4 follows from Theorem V.5 and Theorem V.6, together with the triangle inequality. □

Remarks. We conjecture that it is possible to generalise the replacement technique used for pcBV-PRE in place of source-hiding in order to demonstrate HRA-security for a larger number of schemes. As also note that, as we replace all uncorrupted keys at once as opposed to needing to replace one key at a time our proof is not restricted to acyclic re-encryption graphs. In contrast, other proofs such as those given in [PRSV17, FKKP18] replace one key at a time, in an order that depends on the re-encryption graph. For example, [PRSV17] replaces keys and update tokens in a topological order from sinks to sources defined by the acyclic re-encryption graph. As uncorrupted secret keys are only used for update tokens and re-encryptions leading away from a vertex, this ordering is important in showing indistinguishability between game hops. We revisit these ideas in Chapter VI.

the number of re-encryptions. As long as they are indistinguishably drawn from the appropriate range, the result still holds.

V: 5.5 IND-HRA security of pcBV-PRE

Recall from Lemma V.7 that PCS does not imply IND-HRA security. We therefore now demonstrate that pcBV-PRE is IND-HRA-secure.

Theorem V.7. pcBV-PRE *is IND-HRA-secure. More specifically, for all* PPT *adversaries* $\mathcal{A}$*, there exists an RLWE distinguisher* $\mathcal{D}$ *such that:*

$$\Pr\left[\mathsf{IND\text{-}HRA}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \\ (2\left(\kappa + (Q_{RKu\to u} + Q_{RE\bot u\to u})(\lfloor \log_2(q)/r \rfloor + 1) + Q_{Cu} + 1\right) + Q_{Hu\to c}) \cdot \mathbf{Adv}_{\mathcal{D}}^{RLWE_{\phi,q,\chi e}}(1^\lambda),$$

where $\kappa, Q_{RKu\to u}, Q_{RE\bot u\to u}, Q_{Cu}$ *and* $Q_{Hu\to c}$ *are as described in Theorem V.4.*
Because $\mathcal{A}$ *is* PPT*, we note that the above terms are also all polynomial, meaning that the overall advantage is negligible by the RLWE assumption.*

This can be demonstrated using a similar argument to that of Theorem V.4, except that the challenge that is replaced with a uniform value is a fresh encryption as opposed to a re-encrypted ciphertext.

V: 6 Summary

In this chapter, we have presented the strongest notion of Post-Compromise Security for PRE to date. By strongest, we mean that security holds even when the old secret key sk_i and update token $\Delta_{i,j}$ used to perform the challenge re-encryption are known to the adversary. We have also shown that existing Post-Compromise Security notions are implied by our notion of security, and we have presented separating examples showing that the opposite implication does not hold. We have also shown that PCS can be achieved via a number of existing PRE security notions which immediately shows that at least one existing PRE scheme satisfies PCS [FKKP19], and that PCS and transparency imply IND-HRA-security. We presented pcBV-PRE, an efficient construction of a PRE scheme with PCS based on lattices that also has transparency.

Chapter VI

Adaptive Adversaries Explored

Contents

In this Chapter

In this chapter, we study security with respect to adaptive key corruptions. Unlike the selective security definitions we have studied so far, in the adaptive model the adversary can corrupt keys at any point, as long as they do not corrupt any key that can be used to directly decrypt the challenge ciphertext (or any re-encryption of the challenge ciphertext). A simple way of relating selective and adaptive security is by noting that a selective adversary can simulate the adaptive game successfully if it correctly guesses in the first stage which keys the adaptive adversary will corrupt, an approach known as *complexity leveraging*. However, this leads to a security loss that is exponential in the number of keys. Therefore, useful work on proving adaptive security should demonstrate how to prove adaptive security at a smaller loss. We relate selective and adaptive security without using complexity leveraging, by extending observations already made in the literature and identifying notions of key privacy for update tokens to prove adaptive security for both CPA and Honest Re-encryption Attacks (HRA)-style definitions.

The contributions of this chapter are joint work with Benjamin Dowling. My personal contribution includes the the extension of the challenge graph observation (Section VI: 4), the definitions of re-encryption replaceability (Definitions VI.11 and VI.14) and all reduction-based proofs. Benjamin Dowling assisted in defining the appropriate definitions of key privacy (Definitions VI.6 to VI.8) and provided the statistical proof linking fixed and selective IND-CPA security (Lemma VI.5).

The contents of this chapter are currently unpublished.

VI: 1 Introduction

Recall that one of the motivations in modelling security for PRE is to consider an untrusted proxy. One natural extension to existing work is to model more powerful adversaries. Most security in PRE, including the definitions we have seen so far, is defined in the selective key corruption model, where a two-stage adversary $\mathcal{A}^{\mathsf{sel}} = (\mathcal{A}_0, \mathcal{A}_1)$ first gets to corrupt keys, and then learn challenges, update tokens, and re-encryptions. A stronger model formalised in [FKKP19] is that of adaptive security, where a single-stage adversary $\mathcal{A}^{\mathsf{ad}}$ can corrupt keys at any stage in the game, as long as they do no corrupt any keys that allow them to trivially decrypt the challenge ciphertext. This means that the adversary can corrupt keys in response to information gained from challenge ciphertexts, subject to the trivial win condition. A trivial approach to proving adaptive security given selective security is to have an adversary $\mathcal{A}^{\mathsf{sel}}$ in the selective game attempt to simulate the adaptive game by guessing which keys the adaptive adversary $\mathcal{A}^{\mathsf{ad}}$ will corrupt. If $\mathcal{A}^{\mathsf{sel}}$ guesses correctly, then it can use $\mathcal{A}^{\mathsf{ad}}$ as a subroutine to win the selective game with the same advantage. However, as this simulation only works if $\mathcal{A}^{\mathsf{sel}}$ correctly guesses the corrupted keys, this technique has a loss exponential in the number of keys generated, κ.

Recent work [FKKP19] has demonstrated how adaptive security can be proven at a quasi-polynomial loss, namely at least $\kappa^{O(\log(\kappa))}$ for certain types of re-encryption graph (namely trees and chains, see Figures III.3 and III.4), assuming that the re-encryption graph is known in advance. Recall that the re-encryption graph can reflect applications, so this result is useful for some common applications including key rotation. However, for general re-encryption graphs (and therefore general applications of PRE), the results of [FKKP19] still lead to an exponential loss.

Similarly, [LT18, KLR19a] prove adaptive security for updatable encryption schemes using a key-insulation technique, which limits the necessary amount of guessing by indicating groups of keys that will not be corrupted. This method works because in updatable encryption the resulting re-encryption graph is a chain, meaning any need to guess specific regions of keys in the re-encryption graph is limited compared with general re-encryption graphs.

VI: 1.1 Contributions

In this chapter, we seek to further reduce the loss when proving adaptive security, using methods that apply to all re-encryption graphs. More specifically, we take the concepts introduced in [FKKP19, LT18, KLR19a], extend existing observations and use notions of key privacy in PRE to prove security with adaptive key corruptions at a smaller loss.

One of our contributions is to draw attention to a discrepancy in the literature in terms of how selective security is defined – some work defines this in the same way we did in Section III: 3 where $\mathcal{A}_0$ can access a key generation oracle $\mathsf{O}_{\mathsf{KeyGen}}$ and key corruption oracle $\mathsf{O}_{\mathsf{Corrupt}}$ whereas other work instead gives the first-stage adversary an honest key generation oracle $\mathsf{O}_{\mathsf{HonKeyGen}}$ and a corrupted key generation oracle $\mathsf{O}_{\mathsf{CorKeyGen}}$. We explain why this discrepancy might pose a problem, before presenting arguments as to why the two variants can be considered equivalent.

In practice, proofs of security in PRE usually involve describing how update tokens from uncorrupted keys can be simulated without detection. We formalise this property by defining *token source privacy*, which, when present, indicates that update tokens do not reveal their underlying source key. We describe a number of variants of token source privacy, and use these to relate existing security properties of PRE schemes. One of our main contributions is to demonstrate that PRE-IND-CPA-secure (with selective key corruptions) PRE schemes with token source privacy are also adaptively PRE-IND-CPA-secure.

We use an observation made in [FKKP19] to demonstrate how a weaker property than source-hiding is sufficient for our needs, and define *re-encryption replaceability* as an alternative which can demonstrate how adaptively PRE-IND-CPA-secure PRE schemes with this property are also adaptively IND-HRA-secure. We make a number of observations which indicate that re-encryption replaceability is easier to demonstrate than source-hiding, suggesting that a number of existing re-encryption schemes (both public-key and symmetric) have this property.

Our results offer a number of improvements over those presented in [FKKP19]:

- Our methodology can be applied to the traditional approach where re-encryption graphs are not known in advance, but adaptively defined by the adversary via queries to $\mathsf{O}_{\mathsf{ReKeyGen}}$ and $\mathsf{O}_{\mathsf{ReEnc}}$.
- Some of the properties we define are already met by a number of re-encryption schemes, meaning that some of our results apply more widely.
- We avoid the need to rely on complexity leveraging by using multi-key, multi-challenge security definitions that prevent the need to guess where replacements will be made.

A summary of our contributions can be seen in Figure VI.1. Our main result is as follows:

Theorem VI.1 (informal Theorems VI.4, VI.7 and VI.13). *If a PRE scheme has token-source privacy, is re-encryption replaceable, and its underlying encryption scheme is IND-CPA-secure, then it is also Indistinguishable against Honest Re-Encryption Attacks with adaptive key corruptions (adaptively IND-HRA-secure).*

Finally, we demonstrate similar results for adaptive Post-Compromise Security. We use these results to demonstrate that $\mathsf{pcBV\text{-}PRE}$ (Figure V.8) is secure with respect to adaptive key corruptions for PRE-IND-CPA security, IND-HRA security and PCS.

Chapter structure. We begin by discussing related work in Section VI: 2, before presenting variants of the indistinguishability notions introduced in Section III: 3 with respect to different forms of key corruption in Section VI: 3. In Section VI: 4 we discuss the challenge graph observation, an observation made in [FKKP19] which we expand on to demonstrate security at a smaller loss. In Section VI: 5 we define several notions of key privacy in PRE. We then use these to give a general result for proving adaptive security for IND-CPA-secure

$$\begin{tikzcd}
\mathsf{PKE\text{-}IND\text{-}CPA} \arrow[d, Rightarrow, "+\ \mathsf{wTSP}"', "\textbf{Theorem VI.3}"] \arrow[dr, Rightarrow, "+\ \mathsf{wTSP}", "\textbf{Theorem VI.4}"'] \arrow[drr, Rightarrow, "+\ \mathsf{TSP}", "\textbf{Theorem VI.6}"'] & & \\
\mathsf{f\text{-}PRE\text{-}IND\text{-}CPA} \arrow[r, Rightarrow, "\textbf{Section VI: 6.1}"'] \arrow[d, Rightarrow, "+\ \mathsf{ReEncSim}"', "\textbf{[Coh19, Theorem 5]}"] & \mathsf{s\text{-}PRE\text{-}IND\text{-}CPA} \arrow[r, Rightarrow, "+\ \mathsf{TSP}", "\textbf{Theorem VI.7}"'] & \mathsf{a\text{-}PRE\text{-}IND\text{-}CPA} \arrow[d, Rightarrow, "+\ \mathsf{ReEncRep}"', "\textbf{Theorem VI.13}"] \\
\mathsf{f\text{-}IND\text{-}HRA} & \mathsf{s\text{-}IND\text{-}HRA} & \mathsf{a\text{-}IND\text{-}HRA}
\end{tikzcd}$$

Figure VI.1: A summary of the individual relations proven in this chapter. wTSP is weak Token Source Privacy (Definition VI.6), TSP is Token Source Privacy (Definition VI.7). ReEncRep is re-encryption replaceability (Definition VI.14).

definitions in Section VI: 6 and extend this to IND-HRA definitions in Section VI: 7 and PCS in Section VI: 7.4, demonstrating how pcBV-PRE is also secure with respect to adaptive key corruptions.

VI: 2 Related work

Jafargholi et al. [JKK+17] relate selective and adaptive security for a number of cryptographic primitives, including generalized selective decryption, constrained PRFs, Yao garbled circuits and secret sharing over access structures defined via monotone circuits. They demonstrate how it is not always necessary for a selective adversary to guess all queries the adaptive adversary will make, and that instead only some subset of information needs to be known. More specifically, Jafargholi et al. demonstrate how, when using a hybrid argument,[1] proving that two neighbouring hybrids are indistinguishable often requires only some partial information about the queries the adaptive adversary will make. They use directed graphs to demonstrate how queries are connected, and introduce *graph pebbling games* to describe what needs to be simulated in an indistinguishability proof between two neighbouring hybrids – if a vertex has a pebble on it, then all edges from that vertex are simulated, otherwise they are as expected. They demonstrate how this approach can lead to proving adaptive security at a sub-exponential loss.

[1]A hybrid argument demonstrates indistinguishability between two games by defining a sequence of interim games leading from one game to the other by replacing specified values, then showing indistinguishability between the interim games. This approach is also known as using *game-hops*.

Fuchsbauer et al. [FKKP19] apply this framework specifically to PRE for both PRE-IND-CPA and IND-HRA security. They define a number of pebbling games based on the re-encryption graph to prove adaptive security at a smaller loss for certain types of re-encryption graph. Specifically, they demonstrate a quasi-polynomial loss $\kappa^{O(\log \kappa)}$ for trees and chains. However, for general re-encryption graphs, the security loss is exponential in their depth. They also introduce a new definition of key privacy for update tokens, which they use to prove adaptive IND-CPA security, and they introduce *source-hiding* as a means of proving adaptive IND-HRA security, as discussed in Section V: 5.2. However, recall that one of the downsides of this technique is that known schemes that are provably source-hiding use a blurring technique [CCL+14] that severely limits the number of times a ciphertext can be re-encrypted and still decrypt correctly. Furthermore, the way in which source-hiding is defined results in the need for some complexity leveraging. We define an alternative property, re-encryption replaceability, which is weaker than source-hiding but sufficient to demonstrate adaptive IND-HRA security.

The main other related work that proves adaptive security for re-encryption schemes is in the related primitive of *updatable encryption* [LT18, KLR19a]. In particular, in [KLR19a] Klooß et al. use a *key-insulation* approach to prove adaptive security without relying entirely on complexity leveraging. This entails showing how certain parts of the re-encryption graph are independent of areas where keys are corrupted, meaning that these parts can be indistinguishably simulated without knowledge of the secret keys. This is similar to the idea of partial commitments expressed in [JKK+17, FKKP19]. Klooß et al. also introduce the concept of *randomness-preserving* and use this as a means of simulating re-encryptions under uncorrupted keys that are indistinguishable from genuine re-encryptions. They conjecture that their work can be extended to prove adaptive security in unidirectional PRE by using notions of key privacy in PRE to enable key-insulation.

Key privacy (also known as key anonymity) was first defined in [BBDP01] in the context of PKE schemes. Key privacy captures the notion that an adversary cannot distinguish between ciphertexts encrypted under one public key from another public key. This notion was later extended to PRE schemes, first defined in [ABH09] for unidirectional, single-hop PRE schemes. In the [ABH09] definition, the adversary must instead distinguish whether a given update token allows the adversary to re-encrypt ciphertexts between two specified and uncorrupted keys, or is instead a random element of the update token space. They demonstrate that in a PRE scheme that meets this definition, ciphertexts also have key-privacy in the traditional sense. In [FKKP19], a weaker notion of key privacy is defined, which implies that only the source key (pk_i for token $\Delta_{i,j}$) is hidden. Neither definition of key-privacy in PRE considers corrupted keys.

VI: 3 Preliminaries

In this section we define existing models of key corruption in detail. We begin by defining a multi-key variant of traditional IND-CPA security for PKE schemes. We will demonstrate how security in PRE with adaptive key corruptions can be proven from this comparatively simple definition together with some additional properties.

Definition VI.1. *Consider the following security game:*

```
$\mathsf{PKE\text{-}IND\text{-}CPA}^{\mathcal{PKE}}_{\mathcal{A}}(1^\lambda)$
    $\kappa := 0, \mathsf{called} := \mathsf{false}$
    $b \xleftarrow{\$} \{0,1\}$
    $b' \xleftarrow{(\$)} \mathcal{A}^{\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{Chal}_{\mathsf{Enc}}}(1^\lambda, state)$
    return $(b' = b)$

$\mathsf{O}_{\mathsf{KeyGen}}(1^\lambda) \xrightarrow{\$} \mathsf{pk}_\kappa$
    $\kappa = \kappa + 1$
    $(\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
    return $\mathsf{pk}_\kappa$

$\mathsf{Chal}^{\mathsf{CPA}}_{\mathsf{Enc}}(i, m_0, m_1) \xrightarrow{\$} c_b^*$
    if $(\mathsf{called} = \mathsf{true}) \vee (|m_0| \neq |m_1|)$ :
        return $\perp$
    $c_0^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m_0)$
    $c_1^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m_1)$
    $\mathsf{called} \leftarrow \mathsf{true}$
    return $c_b^*$
```

A PKE scheme is said to be (ϵ, κ)-PKE-IND-CPA-secure *if for all* PPT *adversaries* $\mathcal{A}$*:*

$$\Pr\left[\mathsf{PKE\text{-}IND\text{-}CPA}^{\mathcal{PKE}}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon,$$

where κ is the maximum number of keys generated in $\mathsf{PKE\text{-}IND\text{-}CPA}^{\mathcal{PKE}}_{\mathcal{A}}(1^\lambda)$*. If ϵ is negligible and κ is polynomial as determined by the security parameter λ then we say $\mathcal{PRE}$ is* PKE Indistinguishable against Chosen Plaintext Attacks (PKE-IND-CPA-secure).

We believe this definition is met by most, if not all IND-CPA PRE schemes in the literature, as can be demonstrated by an inductive argument on the number of keys generated.

VI: 3.1 Fixed vs selective key corruptions

One of our contributions is to highlight a discrepancy in how selective security is defined. At a high level it is always defined in the same way – the first stage adversary $\mathcal{A}_0$ determines which keys are corrupted, before the second stage adversary $\mathcal{A}_1$ gets access to $\mathsf{Chal}_{\mathsf{Enc}}, \mathsf{O}_{\mathsf{ReKeyGen}}$, or $\mathsf{O}_{\mathsf{ReEnc}}$. However the discrepancy comes in whether keys are *generated* as corrupted keys, or whether keys are generated and then later corrupted. The former is the model we have discussed in this thesis so far. In the latter, instead of $\mathcal{A}_0$ having access to $\mathsf{O}_{\mathsf{KeyGen}}$ and $\mathsf{O}_{\mathsf{Corrupt}}$, they gain access to an honest key generation oracle, and a corrupted key generation oracle, defined as follows:

```
$\underline{\mathsf{O}_{\mathsf{HonKeyGen}}(1^\lambda) \xrightarrow{\$} \mathsf{pk}_\kappa}$
$\kappa = \kappa + 1$
$(\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$\mathcal{DRG}.\mathsf{add}\{v_\kappa\}$
return $\mathsf{pk}_\kappa$

$\underline{\mathsf{O}_{\mathsf{CorKeyGen}}(1^\lambda) \xrightarrow{\$} (\mathsf{pk}_\kappa, \mathsf{sk}_\kappa)}$
$\kappa = \kappa + 1$
$(\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$\mathcal{DRG}.\mathsf{add}\{v_\kappa\}$
$\mathcal{K}_{\mathsf{corrupt}}.\mathsf{add}\{\mathsf{sk}_\kappa\}$
return $(\mathsf{pk}_\kappa, \mathsf{sk}_\kappa)$
```

We refer to this as the *fixed key model.* We refer to the fixed key corruption variants of the CPA experiment (Definition III.7) and HRA experiment (Definition III.8) defined by this alteration as $\mathsf{f\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ $\mathsf{f\text{-}IND\text{-}HRA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$, respectively.

Definition VI.2. *We say that a PRE scheme, $\mathcal{PRE}$, is* (ϵ, κ)-Indistinguishable against Chosen Plaintext Attacks with fixed key corruptions *if for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*:*

$$\Pr\Big[\mathsf{f\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1\Big] \leq \frac{1}{2} + \epsilon,$$

where κ is the maximum number of keys generated in $\mathsf{f\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$*. If ϵ is negligible and κ is polynomial as determined by λ, then we say $\mathcal{PRE}$ is* Indistinguishable against Chosen Plaintext Attacks with fixed key corruptions (fixed-key PRE-IND-CPA-secure)*.*

We say that a PRE scheme, $\mathcal{PRE}$, is (ϵ, κ)-Indistinguishable against Honest Re-Encryption Attacks with fixed key corruptions if for all PPT *adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*:*

$$\Pr\Big[\mathsf{f\text{-}IND\text{-}HRA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1\Big] \leq \frac{1}{2} + \epsilon,$$

where κ is the maximum number of keys generated in $\mathsf{f\text{-}IND\text{-}HRA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$*. If ϵ is negligible* *and κ is polynomial as determined by λ, then we say $\mathcal{PRE}$ is Indistinguishable against Honest Re-Encryption Attacks with fixed key corruptions (fixed-key IND-HRA-secure).*

In the chapter, we also parametrise the selective variants of the definitions described in Section III: 3:

Definition VI.3. *If a PRE scheme, $\mathcal{PRE}$, is ϵ-PRE-IND-CPA-secure subject to a maximum of κ key pairs, then we say $\mathcal{PRE}$ is selectively (ϵ, κ)-PRE-IND-CPA-secure. If ϵ is negligible and κ is polynomial as determined by λ, then we say $\mathcal{PRE}$ is selectively Indistinguishable against Chosen Plaintext Attacks (PRE-IND-CPA-secure).*

Similarly, if $\mathcal{PRE}$ is ϵ-IND-HRA-secure subject to a maximum of κ key pairs, then we say $\mathcal{PRE}$ is selectively (ϵ, κ)-IND-HRA-secure. If ϵ is negligible and κ is polynomial as as determined by λ, then we say $\mathcal{PRE}$ is selectively Indistinguishable against Honest Re-Encryption Attacks (IND-HRA-secure).

If $\mathcal{PRE}$ is ϵ-PCS subject to a maximum of κ key pairs, then we say $\mathcal{PRE}$ has selective (ϵ, κ)-PCS. If ϵ is negligible and κ is polynomial as as determined by λ, then we say $\mathcal{PRE}$ has selective Post-Compromise Security (PCS).

The crucial difference between fixed-key and selective security is that in the selective we model the adversary can choose whether to learn the secret key based on the public key. This difference affects how things can be proven in the different models. For example, in [PRSV17] security is defined in the fixed-key model, and therefore proofs are also in this model. As security depends on RLWE, the proof replaces all public keys generated by $\mathsf{O}_{\mathsf{HonKeyGen}}(1^\lambda)$ with uniform random elements. However, in the selective model this approach does not work, as it is unknown whether keys will be corrupted when they are generated. As the secret key does not exist when the public key is actually a uniform random value, the challenger will be unable to return it if the adversary chooses to corrupt that key. It is this reason that led to us defining the resample-RLWE experiment in Section V: 5.3, and using reductions to this as opposed to standard RLWE challenges. In the fixed-key variant, standard RLWE would have been sufficient.

We remark that, in the symmetric setting, as there is no public key, $\mathsf{O}_{\mathsf{HonKeyGen}}(1^\lambda)$ generates a key but returns nothing to the adversary, meaning that fixed and selective key corruptions are equivalent, as $\mathcal{A}_0$ receives no information when keys are generated with which to decide whether to corrupt those keys.

The existing work [FKKP19] that relates selective and adaptive key corruption in PRE defines selective security using a corruption oracle. This is reasonable in the sense that as the selective variant is closer to the adaptive model, making the two easier to compare. However, the constructions presented in [FKKP18] (which is the full version of [FKKP19]) are variants of constructions presented in existing work, where they are proven secure in the fixed-key model. In [FKKP18], proofs that the constructions are adaptively secure leverage these results, referring to them as selectively secure despite the discrepancy. Therefore, it appears that Fuchsbauer et al. assume that the fixed and selective variants are equivalent. We therefore investigate in Section VI: 6.1 whether fixed key and selective security can indeed be considered equivalent.

VI: 3.2 Definitions for adaptive security

Recall that in selective security (Definitions III.7 and III.8), the trivial win condition was enforced in the oracles, by $\mathsf{Chal}_{\mathsf{Enc}}$ returning $\perp$ if given a corrupted key as input, and by $\mathsf{O}_{\mathsf{ReKeyGen}}$ and $\mathsf{O}_{\mathsf{ReEnc}}$ also returning $\perp$ for forbidden queries. This type of enforcement is possible because these oracles are only given to the second-stage adversary $\mathcal{A}_1$, at a point when the sets of corrupted and uncorrupted keys are known. However in the adaptive key corruption model, it is more complex to use conditions in $\mathsf{O}_{\mathsf{ReKeyGen}}$, $\mathsf{O}_{\mathsf{ReEnc}}$ and $\mathsf{Chal}_{\mathsf{Enc}}$ to prevent trivial wins, as whether or not a key will later be corrupted is unknown at the time these oracles are queried. For example, the adversary could call $\mathsf{Chal}_{\mathsf{Enc}}(i, m_0, m_1)$ whilst sk_i is uncorrupted and later corrupt sk_i, resulting in a trivial win if we were to not change how the trivial win condition is enforced. For this reason, the trivial win condition in adaptive games is instead enforced using the Directed Re-encryption Graph (DRG) (Section III: 3.1). When the adversary queries $\mathsf{O}_{\mathsf{ReKeyGen}}$ or $\mathsf{O}_{\mathsf{ReEnc}}$, the challenger updates a DRG. Then after the adversary outputs its guess, the challenger uses the DRG to update the set of challenge keys $\mathcal{K}_{\mathsf{chal}}$, as described in Figure VI.2. Then, if any challenge keys have been corrupted, the challenger returns $\perp$, and otherwise returns $(b' = b)$ as usual.

We now define PRE-IND-CPA security and IND-HRA security notions in the adaptive key corruption model.

```
$\mathsf{UpdateChallengeKeys}(\mathcal{K}_{\mathsf{chal}}, \mathcal{DRG}) \to \mathcal{K}_{\mathsf{chal}}$

$\textbf{for } \mathsf{sk}_i \in \mathcal{K}_{\mathsf{chal}}, : \quad \textbf{for } j \in \{1, \dots, \kappa\} :$
    $\textbf{if } \exists$ a path between $v_i$ and $v_j$ in $\mathcal{DRG}$ :
      $\mathcal{K}_{\mathsf{chal}}.\mathsf{add}\{sk_j\}$
$\textbf{return } \mathcal{K}_{\mathsf{chal}}$
```

Figure VI.2: Updating the set of challenge keys in an adaptive security game.

Definition VI.4. *Consider the following security game:*

```
$\mathsf{a\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$

$\mathcal{K}_{\mathsf{chal}}, \mathcal{K}_{\mathsf{corrupt}}, \mathcal{T}_{\mathsf{honest}}, \mathcal{DRG} = \emptyset$
$\kappa := 0, \mathsf{called} := \mathsf{false}$
$b \overset{\$}{\leftarrow} \{0,1\}$
$b' \overset{(\$)}{\leftarrow} \mathcal{A}^{\mathsf{O_{KeyGen}}, \mathsf{O_{Corrupt}}, \mathsf{O_{ReKeyGen}}, \mathsf{O^{CPA}_{ReEnc}}, \mathsf{Chal^{CPA}_{Enc}}}(1^\lambda)$
$\mathcal{K}_{\mathsf{chal}} \leftarrow \mathsf{UpdateChallengeKeys}(\mathcal{K}_{\mathsf{chal}}, \mathcal{DRG})$
$\textbf{if } \mathcal{K}_{\mathsf{chal}} \cap \mathcal{K}_{\mathsf{corrupt}} \neq \emptyset :$
  $\textbf{return } \bot$
$\textbf{else } :$
  $\textbf{return } (b' = b)$
```

```
$\mathsf{O_{KeyGen}}(1^\lambda) \overset{\$}{\to} \mathsf{pk}_\kappa$

$\kappa = \kappa + 1$
$(\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$
$\mathcal{DRG}.\mathsf{add}\{v_\kappa\}$
$\textbf{return } \mathsf{pk}_\kappa$
```

```
$\mathsf{O_{Corrupt}}(i) \to \mathsf{sk}_i$

$\mathcal{K}_{\mathsf{corrupt}}.\mathsf{add}\{\mathsf{sk}_i\}$
$\textbf{return } \mathsf{sk}_i$
```

```
$\mathsf{O_{ReKeyGen}}(i,j) \overset{(\$)}{\to} \Delta_{i,j}$

$\Delta_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$\mathcal{T}_{\mathsf{honest}}.\mathsf{add}\{i, j, \Delta_{i,j}\}$
$\mathcal{DRG}.\mathsf{add}\{\vec{e}_{i,j}\} \boxed{\mathcal{DRG}.\mathsf{add}\{e_{i,j}\}}$
$\textbf{return } \Delta_{i,j}$
```

```
$\mathsf{O^{CPA}_{ReEnc}}(i, j, [\Delta_{i,j}], c) \overset{(\$)}{\to} c'$

$\textbf{if } ((i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}) :$
  $\Delta_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$c' \overset{(\$)}{\leftarrow} \mathsf{ReEnc}(\Delta_{i,j}, c)$
$\mathcal{DRG}.\mathsf{add}\{\vec{e}_{i,j}\} \boxed{\mathcal{DRG}.\mathsf{add}\{e_{i,j}\}}$
$\textbf{return } c'$
```

```
$\mathsf{Chal^{CPA}_{Enc}}(i, m_0, m_1) \overset{\$}{\to} c^*_b$

$\textbf{if } (\mathsf{called} = \mathsf{true}) \vee (|m_0| \neq |m_1|) :$
  $\textbf{return } \bot$
$c^*_0 \overset{\$}{\leftarrow} \mathsf{Enc}(\mathsf{pk}_i, m_0)$
$c^*_1 \overset{\$}{\leftarrow} \mathsf{Enc}(\mathsf{pk}_i, m_1)$
$\mathcal{K}_{\mathsf{chal}}.\mathsf{add}\{\mathsf{sk}_i\}$
$\mathsf{called} \leftarrow \mathsf{true}$
$\textbf{return } c^*_b$
```

where $\boxed{\textit{boxed conditions}}$ *apply only to bidirectional PRE schemes.*

A PRE scheme, $\mathcal{PRE}$, is said to be adaptively (ϵ, κ)-Indistinguishable against Chosen Plaintext Attacks with adaptive key corruptions ((ϵ, κ)-adaptively PRE-IND-CPA-secure) *if for all* PPT *adversaries* $\mathcal{A}$*:*

$$\Pr\left[\textsf{a-PRE-IND-CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon,$$

where κ *is the maximum number of keys generated in* $\textsf{a-PRE-IND-CPA}_{\mathcal{A}}^{\mathcal{PKE}}(1^\lambda)$.

If ϵ *is negligible and* κ *is polynomial as determined by the security parameter* λ *then we say* $\mathcal{PRE}$ *is* Indistinguishable against Chosen Plaintext Attacks with adaptive key corruptions (adaptively PRE-IND-CPA-secure).

Definition VI.5. *Consider the following security game (where shaded parts highlight the differences from* $\textsf{a-PRE-IND-CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$*):*

```
$\textsf{a-IND-HRA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$
$\mathcal{K}_{\textsf{chal}}, \mathcal{K}_{\textsf{corrupt}}, \mathcal{T}_{\textsf{honest}}, \mathcal{DRG} = \emptyset$
$\kappa := 0, \textsf{called} := \textsf{false}$
$b \xleftarrow{\$} \{0,1\}$
$b' \xleftarrow{(\$)} \mathcal{A}^{\textsf{O}_{\textsf{KeyGen}}, \textsf{O}_{\textsf{Corrupt}}, \textsf{O}_{\textsf{Enc}}, \textsf{O}_{\textsf{ReKeyGen}}, \textsf{O}^{\textsf{HRA}}_{\textsf{ReEnc}}, \textsf{Chal}^{\textsf{HRA}}_{\textsf{Enc}}}(1^\lambda)$
$\mathcal{K}_{\textsf{chal}} \leftarrow \textsf{UpdateChallengeKeys}(\mathcal{K}_{\textsf{chal}}, \mathcal{DRG})$
if $\mathcal{K}_{\textsf{chal}} \cap \mathcal{K}_{\textsf{corrupt}} \neq \emptyset$ :
  return $\perp$
else :
  return $(b' = b)$
```

```
$\textsf{O}_{\textsf{KeyGen}}(1^\lambda) \xrightarrow{\$} \textsf{pk}_\kappa$
$\kappa = \kappa + 1$
$(\textsf{pk}_\kappa, \textsf{sk}_\kappa) \xleftarrow{\$} \textsf{KeyGen}(1^\lambda)$
$\mathcal{DRG}.\textsf{add}\{v_\kappa\}$
return $\textsf{pk}_\kappa$
```

```
$\textsf{O}_{\textsf{Corrupt}}(i) \rightarrow \textsf{sk}_i$
$\mathcal{K}_{\textsf{corrupt}}.\textsf{add}\{\textsf{sk}_i\}$
return $\textsf{sk}_i$
```

```
$\textsf{O}_{\textsf{Enc}}(i, m) \xrightarrow{\$} c$
$c \xleftarrow{\$} \textsf{Enc}(\textsf{pk}_i, m)$
$\mathcal{C}_{\textsf{honest}}.\textsf{add}\{(i, c)\}$
return $c$
```

```
$\textsf{O}_{\textsf{ReKeyGen}}(i, j) \xrightarrow{(\$)} \Delta_{i,j}$
$\Delta_{i,j} \xleftarrow{(\$)} \textsf{ReKeyGen}(\textsf{sk}_i, \textsf{pk}_j)$
$\mathcal{T}_{\textsf{honest}}.\textsf{add}\{i, j, \Delta_{i,j}\}$
$\mathcal{DRG}.\textsf{add}\{\overrightarrow{e}_{i,j}\}$ $\boxed{\mathcal{DRG}.\textsf{add}\{e_{i,j}\}}$
return $\Delta_{i,j}$
```

```
$\mathsf{O}^{\mathsf{HRA}}_{\mathsf{ReEnc}}(i, j, [\Delta_{i,j}], c) \overset{(\$)}{\to} c'$
---------------------------------------------
if $(i, c) \notin \mathcal{C}_{\mathsf{honest}}$ : return $\bot$
if $((i, c) \in \mathcal{C}_{\mathsf{chal}}) \wedge (\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}})$ : return $\bot$
if $((i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}})$ :
    $\Delta_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$c' \overset{(\$)}{\leftarrow} \mathsf{ReEnc}(\Delta_{i,j}, c)$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{j, c'\}$
if $(i, c) \in \mathcal{C}_{\mathsf{chal}}$ :
    $\mathcal{DRG}.\mathsf{add}\{\vec{e}_{i,j}\}$ $\boxed{\mathcal{DRG}.\mathsf{add}\{e_{i,j}\}}$
    $\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\{(j, c')\}$
return $c'$
```

```
$\mathsf{Chal}^{\mathsf{HRA}}_{\mathsf{Enc}}(i, m_0, m_1) \overset{\$}{\to} c^*_b$
---------------------------------------------
if $(\mathsf{called} = \mathsf{true}) \vee (|m_0| \neq |m_1|)$ :
    return $\bot$
$c^*_0 \overset{\$}{\leftarrow} \mathsf{Enc}(\mathsf{pk}_i, m_0)$
$c^*_1 \overset{\$}{\leftarrow} \mathsf{Enc}(\mathsf{pk}_i, m_1)$
$\mathcal{K}_{\mathsf{chal}}.\mathsf{add}\{\mathsf{sk}_i\}$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(i, c)\}, \mathcal{C}_{\mathsf{chal}}.\mathsf{add}\{(i, c)\}$
$\mathsf{called} \leftarrow \mathsf{true}$
return $c^*_b$
```

where $\boxed{\textit{boxed conditions}}$ *apply only to bidirectional PRE schemes.*

Similarly, $\mathcal{PRE}$ is said to be adaptively (ϵ, κ)-Indistinguishable against Honest Re-Encryption Attacks with adaptive key corruptions ((ϵ, κ)-adaptively IND-HRA-secure) *if for all* PPT *adversaries $\mathcal{A}$:*

$$\Pr\left[\mathsf{a\text{-}IND\text{-}HRA}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon,$$

where κ is the maximum number of keys generated in $\mathsf{a\text{-}IND\text{-}HRA}^{\mathcal{PKE}}_{\mathcal{A}}(1^\lambda)$.

If ϵ is negligible and κ is polynomial as determined by the security parameter λ then we say $\mathcal{PRE}$ is Indistinguishable against Honest Re-Encryption Attacks with adaptive key corruptions (adaptively IND-HRA-secure).

VI: 4 The Challenge Graph Observation

We now discuss a crucial observation made by Fuchsbauer et al. in their work on adaptively secure PRE [FKKP19]. They point out that to prove security, strictly speaking, it is sufficient to only consider the '*challenge graph*' – the subgraph that is reachable from the challenge vertex. Therefore, in their proofs of security, their game-hops only replace outputs of oracle queries in the challenge graph. We believe this is reasonable, as any subgraph that is not reachable from the challenge subgraph represents keys that the challenge ciphertext has never been encrypted under, and that were generated independently of the challenge key. It is therefore reasonable to conclude that any information the adversary learns about these keys cannot give any advantage in winning the game.

Recall that a major difference between [FKKP19] and other work defining security in PRE is that the re-encryption graph is fixed at the start of the game. This change is necessary in [FKKP19] as the security proofs utilise a hybrid argument that replaces the outputs of oracle queries in an order determined by the graph. Because the [FKKP19] setting assumes that the re-encryption graph is known at the start of the game, the challenge graph is also apparent from the beginning (see Figure VI.3), making it simple for any simulator to know which oracle queries need to be simulated. Unfortunately, in the traditional model

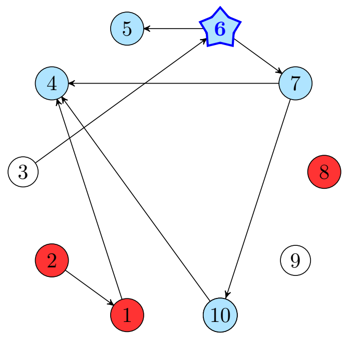

Figure VI.3: The *challenge graph* is the subgraph reachable from the **challenge node**. In this example, this means the challenge graph consists of vertices v_4,v_5, v_6, v_7, v_{10} and edges $\vec{e}_{6,5}$, $\vec{e}_{6,7}$, $\vec{e}_{7,4}$, $\vec{e}_{7,10}$, $\vec{e}_{10,4}$.

where the re-encryption graph is determined during the course of the experiment, the challenge graph is also not known in advance, making it difficult to know which queries need to be simulated and which do not.

However, we can extend their observation to the adaptive graph setting by observing that even when the challenge graph is not known in advance, we know that certain oracle queries will lead to a subgraph that is not reachable from the challenge graph, as a consequence of the trivial win condition. For example, if $\mathcal{A}$ learns a token $\Delta_{i,j}$ and corrupts sk_j, then neither pk_i nor pk_j can be in the challenge graph. If we use the same logic as Fuchsbauer et al., we conclude that it suffices to only consider adversaries that never make $\mathsf{O}_{\mathsf{Corrupt}}$, $\mathsf{O}_{\mathsf{ReKeyGen}}$ or $\mathsf{O}_{\mathsf{ReEnc}}$ queries that result in vertices being unreachable from the challenge graph. For example, we do not need to consider adversaries that call $\mathsf{O}_{\mathsf{Corrupt}}(i)$, $\mathsf{O}_{\mathsf{Corrupt}}(j)$ and $\mathsf{O}_{\mathsf{ReKeyGen}}(i,j)$. We note this is similar to the challenge independence assumption discussed in Section V: 5.3. We further note this also implies that learning update tokens between corrupted keys can give no advantage in winning the game, meaning that it suffices to only consider adversaries that corrupt sk_i, sk_j and learn some $\Delta_{i,j}$ (in any order). We refer to this as the *challenge graph observation*.

We shall explore this in detail in Section VI: 7.1, and in Section VI: 7.2 will demonstrate that this observation implies that a property weaker than source-hiding is sufficient for demonstrating adaptive IND-HRA security.

We remark that similar assumptions can be made for other primitives. Informally, any queries made in a security game Game on keys that are independent of any challenge keys cannot give any advantage to an adversary hoping to win Game. We do not formalise this here as the challenge independence assumption is difficult to generalise correctly, so we restrict ourselves to defining the challenge independence assumption for PRE schemes only.

VI: 5 Key Privacy Notions

In contrast to prior work on key privacy in PRE, we give more specific terms so as to be clear both in terms of whether we are referring to ciphertexts or update tokens, and in the case of update tokens, which keys are hidden. In what follows, we specifically examine *token source privacy*. We will discuss a number of variants of this, but overall the following definitions capture the notion that, given an update token, the adversary cannot tell whether it re-encrypts from the expected source secret key, or an unknown source key. We give different variants as we find that weaker variants are sufficient to relate fixed and selective PRE security from PKE-IND-CPA-security, but stronger variants are required to prove adaptive security. Defining these variants, allows us to be more precise in the properties required for our results.

As in the security games discussed so far, in all our key privacy notions the adversary is not allowed to corrupt any challenge keys. We note that when challenges are update tokens $\Delta^*_{i,j}$, the challenge key is the target key sk_j. This is because if the adversary learns $\mathsf{sk}_j \overset{(\$)}{\leftarrow} \mathsf{Chal}_{\mathsf{ReKeyGen}}(i_0, i_1, j)$ then they will be able to trivially win by computing $c \overset{\$}{\leftarrow} \mathsf{Enc}(\mathsf{pk}_{i_0}, m)$, followed by $c' \overset{(\$)}{\rightarrow} \mathsf{ReEnc}(\Delta^*, c)$ and verifying whether $m \overset{?}{=} \mathsf{Dec}(\mathsf{sk}_j, c')$.

Our first definition of token source privacy in Definition VI.6 is similar to FKKP-weak key privacy [FKKP19, Definition 8]. Whereas Fuchsbauer et al. call their definition 'weak' in the sense that only the source key is hidden in an update token, we use 'weak' to mean the adversary does not get to corrupt any secret keys related to challenges i.e. $\mathsf{Chal}_{\mathsf{ReKeyGen}}(i, j)$ returns $\perp$ if either sk_i or sk_j are corrupted. We also give a stronger variant in Definition VI.7 – called *token source privacy* – where the adversary is allowed to learn the source private keys of challenge tokens. This is clearly stronger than the weak variant, as the adversary has access to more information. Note that in both of these variants, only $\mathcal{A}_0$ has access to $\mathsf{O}_{\mathsf{Corrupt}}$, and only $\mathcal{A}_1$ has access to $\mathsf{Chal}_{\mathsf{ReKeyGen}}$. Finally, we introduce an adaptive version of token source privacy in Definition VI.8, where the adversary can corrupt keys and learn challenges in any order.

Definition VI.6. *Consider the following security game:*

```
$\mathsf{wTSP}^{\mathcal{PRE}}_{\mathcal{A},\kappa}(1^\lambda)$
$\mathcal{K}_{\mathsf{corrupt}} = \emptyset$
$(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$(\overline{\mathsf{pk}}_1, \overline{\mathsf{sk}}_1), \ldots, (\overline{\mathsf{pk}}_\kappa, \overline{\mathsf{sk}}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$b \xleftarrow{\$} \{0, 1\}$
$state \xleftarrow{(\$)} \mathcal{A}_0^{\mathsf{O}_{\mathsf{Corrupt}}}(\mathsf{pk}_1, \ldots, \mathsf{pk}_\kappa)$
$b' \xleftarrow{(\$)} \mathcal{A}_1^{\mathsf{Chal}_{\mathsf{ReKeyGen}}}(1^\lambda, state)$
return $(b' = b)$
```

```
$\mathsf{O}_{\mathsf{Corrupt}}(i) \to \mathsf{sk}_i$
$\mathcal{K}_{\mathsf{corrupt}}.\mathsf{add}\{\mathsf{sk}_i\}$
return $\mathsf{sk}_i$
```

```
$\mathsf{Chal}_{\mathsf{ReKeyGen}}(i, j) \overset{(\$)}{\to} \Delta^{*b}$
if $((\mathsf{sk}_i \in \mathcal{K}_{\mathsf{corrupt}}) \vee (\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}}))$ :
    return $\perp$
$\Delta^{*0} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$\Delta^{*1} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\overline{\mathsf{sk}}_i, \mathsf{pk}_j)$
return $\Delta^{*b}$
```

A PRE scheme, $\mathcal{PRE}$, is said to have (ϵ, κ)-weak Token Source Privacy ((ϵ, κ)-weak TSP) *if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:*

$$\Pr\left[\mathsf{wTSP}^{\mathcal{PRE}}_{\mathcal{A},\kappa}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon.$$

If ϵ is negligible and κ is polynomial as determined by the security parameter λ then we say $\mathcal{PRE}$ has weak Token Source Privacy (weak TSP).

We note that weak TSP is a strictly weaker notion than the definition of key privacy given in [ABH09], as their definition masks both the source and target keys. Therefore, the schemes which are proven to have ABH-key-privacy have weak TSP.

Recall that in Section V: 5.3 we used the challenge independence assumption to demonstrate that if there is a strict separation between challenge keys and corrupted keys, and the experiment gives no information linking the two, then the existence of corrupted keys can provide no advantage in winning the experiment.

Lemma VI.1. *Consider the variant of the weak TSP experiment with no key corruption. This is equivalent to the weak TSP experiment, by the challenge independence assumption.*

This means that to demonstrate that a scheme has weak TSP it is sufficient to demonstrate that the scheme meets the variant of weak TSP with no $\mathsf{O}_{\mathsf{Corrupt}}$ (which is equivalent to a multi-key variant of FKKP-weak key privacy).

We now present a stronger notion of token source privacy, where the adversary is allowed to corrupt the source keys of challenge tokens.

Definition VI.7. *Consider the following security game:*

```
$\mathsf{TSP}^{\mathcal{PRE}}_{\mathcal{A},\kappa}(1^\lambda)$
$\mathcal{K}_{\mathsf{corrupt}}, \mathcal{K}_{\mathsf{chal}} = \emptyset$
$(\mathsf{pk}_1, \mathsf{sk}_1), \dots, (\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$(\overline{\mathsf{pk}}_1, \overline{\mathsf{sk}}_1), \dots, (\overline{\mathsf{pk}}_\kappa, \overline{\mathsf{sk}}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$b \xleftarrow{\$} \{0,1\}$
$state \xleftarrow{(\$)} \mathcal{A}_0^{\mathsf{O}_{\mathsf{Corrupt}}}(\mathsf{pk}_1, \dots, \mathsf{pk}_\kappa)$
$b' \xleftarrow{(\$)} \mathcal{A}_1^{\mathsf{Chal}_{\mathsf{ReKeyGen}}}(1^\lambda)$
**return** $(b' = b)$

$\mathsf{O}_{\mathsf{Corrupt}}(i) \to \mathsf{sk}_i$
$\mathcal{K}_{\mathsf{corrupt}}.\mathsf{add}\{\mathsf{sk}_i\}$
**return** $\mathsf{sk}_i$

$\mathsf{Chal}_{\mathsf{ReKeyGen}}(i,j) \xrightarrow{(\$)} \Delta^{*b}$
**if** $(\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}})$ :
    **return** $\perp$
$\Delta^{*0} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$\Delta^{*1} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\overline{\mathsf{sk}}_i, \mathsf{pk}_j)$
**return** $\Delta^{*b}$
```

A PRE scheme, $\mathcal{PRE}$, is said to have (ϵ, κ)-Token Source Privacy ((ϵ, κ)-TSP) *if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:*

$$\Pr\left[\mathsf{TSP}^{\mathcal{PRE}}_{\mathcal{A},\kappa}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon.$$

If ϵ is negligible and κ is polynomial as determined by the security parameter λ then we say $\mathcal{PRE}$ has Token Source Privacy (TSP).

Recall that typically, proofs that a PRE scheme is PRE-IND-CPA-secure or IND-HRA-secure use a hybrid argument where keys together with tokens and re-encryptions from those keys are replaced. We avoid the need to restrict ourselves to acyclic graphs by instead replacing all update tokens with uncorrupted source keys in one step, utilising our multi-key, multi-challenge notions of token source privacy. This is in contrast to existing work, where often proofs replace values in a particular order which often depends on the re-encryption graph. For example, in [FKKP19], the pebbling games used in security proofs can only be defined assuming a single source vertex, meaning that graphs must be acyclic.

Similarly, in the proofs of [PRSV17, Theorem 4.3, Theorem 5.1], uncorrupted public keys are replaced with uniform random values in an order which depends on the unique ordering presented by a directed acyclic graph.

Remark. For a public-key PRE scheme to be secure, it is necessary that, given $\Delta_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$, it should not be possible to derive sk_i from pk_j. Recall that BV-PRE [PRSV17] uses an alternative 'public' key to create update tokens, and the exposure of this key leads to the source key being learned from the update token, and hence this key cannot be public in practice. One means of ensuring the source secret key remains secret is by using some random input to 'mask' the keys being used, as in pcBV-PRE. We therefore conjecture that for public-key PRE schemes with a probabilistic ReKeyGen, token-source privacy is likely to be a side-effect of the existing security of the system, meaning that our main results which utilise token source privacy can be applied.

Definition VI.8. *Consider the following security game:*

```
$\mathsf{a\text{-}TSP}^{\mathcal{PRE}}_{\mathcal{A},\kappa}(1^\lambda)$
$\mathcal{K}_{\mathsf{corrupt}} = \emptyset$
$(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$
$(\overline{\mathsf{pk}}_1, \overline{\mathsf{sk}}_1), \ldots, (\overline{\mathsf{pk}}_\kappa, \overline{\mathsf{sk}}_\kappa) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$
$b \overset{\$}{\leftarrow} \{0,1\}$
$b' \overset{(\$)}{\leftarrow} \mathcal{A}^{\mathsf{O}_{\mathsf{Corrupt}}, \mathsf{Chal}_{\mathsf{ReKeyGen}}}(\mathsf{pk}_1, \ldots, \mathsf{pk}_\kappa)$
return $(b' = b)$
```

```
$\mathsf{O}_{\mathsf{Corrupt}}(i) \to \mathsf{sk}_i$
if $\mathsf{sk}_i \in \mathcal{K}_{\mathsf{chal}}$ : return $\perp$
$\mathcal{K}_{\mathsf{corrupt}}.\mathsf{add}\{\mathsf{sk}_i\}$
return $\mathsf{sk}_i$
```

```
$\mathsf{Chal}_{\mathsf{ReKeyGen}}(i, j) \overset{(\$)}{\to} \Delta^{*b}$
if $(\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}})$ :
  return $\perp$
$\Delta^{*0} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$\Delta^{*1} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\overline{\mathsf{sk}}_i, \mathsf{pk}_j)$
$\mathcal{K}_{\mathsf{chal}}.\mathsf{add}\{\mathsf{sk}_j\}$
return $\Delta^{*b}$
```

A PRE scheme, $\mathcal{PRE}$, is said to have (ϵ, κ)-adaptive Token Source Privacy ((ϵ, κ)-adaptive TSP) *if for all* PPT *adversaries $\mathcal{A}$:*

$$\Pr\left[\mathsf{a\text{-}TSP}^{\mathcal{PRE}}_{\mathcal{A},\kappa}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon.$$

If ϵ is negligible and κ is polynomial as determined by the security parameter λ then we say $\mathcal{PRE}$ has adaptive Token Source Privacy (adaptive TSP).

In $\mathsf{a\text{-}TSP}^{\mathcal{PRE}}_{\mathcal{A},\kappa}(1^\lambda)$, the trivial win condition is enforced by $\mathsf{Chal}_{\mathsf{ReKeyGen}}(i, j)$ returning $\perp$ if sk_j is corrupted, and by $\mathsf{O}_{\mathsf{Corrupt}}(i)$ returning $\perp$ if sk_i is a challenge key. The main difference between TSP and adaptive TSP is that the adversary can corrupt keys in response to challenge tokens, as long as they do not corrupt any keys that a challenge token re-encrypts to.

VI: 5.1 Key privacy observations

We now make some basic observations about these notions of source key privacy.

Lemma VI.2. *If a PRE scheme $\mathcal{PRE}$ has (ϵ, κ)-adaptive TSP, then it also has (ϵ, κ)-TSP, and if $\mathcal{PRE}$ has (ϵ, κ)-TSP, then it also has (ϵ, κ)-weak TSP.*
In other words, (ϵ, κ)-adaptive TSP $\implies$ (ϵ, κ)-TSP $\implies$ (ϵ, κ)-weak TSP.

Proof. Each security game gives the adversary strictly more information or ability. This means that adversaries in $\mathsf{a\text{-}TSP}^{\mathcal{PRE}}_{\mathcal{A},\kappa}(1^\lambda)$ can perfectly simulate $\mathsf{TSP}^{\mathcal{PRE}}_{\mathcal{A},\kappa}(1^\lambda)$, and that adversaries in $\mathsf{TSP}^{\mathcal{PRE}}_{\mathcal{A},\kappa}(1^\lambda)$ can perfectly simulate $\mathsf{wTSP}^{\mathcal{PRE}}_{\mathcal{A},\kappa}(1^\lambda)$, and therefore gain the same advantage as any successful adversary for the weaker games. $\square$

Lemma VI.3. *If a PRE scheme, $\mathcal{PRE}$, has TSP, then $\mathcal{PRE}.\mathsf{ReKeyGen}$ must be probabilistic.*

Proof. In $\mathsf{TSP}^{\mathcal{PRE}}_{\mathcal{A},\kappa}(1^\lambda)$, $\mathcal{A}$ can call $\mathsf{Chal}_{\mathsf{ReKeyGen}}(i, j) \overset{(\$)}{\rightarrow} \Delta^*$ and then $\mathsf{O}_{\mathsf{Corrupt}}(i) \rightarrow \mathsf{sk}_i$. This means $\mathcal{A}$ can compute their own token $\Delta_{i,j} \overset{\$}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$.

If $\mathsf{ReKeyGen}$ is deterministic then $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$ only has one output. Therefore, $\mathcal{A}$ can output $b = 0$ if $\Delta^* = \Delta_{i_0,j}$ and $b' = 1$ otherwise to always win the game. We conclude that PRE schemes with token source privacy must be probabilistic. $\square$

We note that a similar result to Lemma VI.3 is shown in [ABH09, Lemma 1] for (what we refer to as) weak TSP, but we chose to provide an explicit proof for easier comparison in our multi-key approach.

Lemma VI.4. *If a PRE scheme has TSP, it must be unidirectional.*

Proof. We show that if $\mathcal{PRE}$ is bidirectional, then it cannot have TSP. Bidirectionality implies that, given a token $\Delta_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$ it is possible to derive a token $\Delta_{j,i}$ that successfully re-encrypts from pk_j to pk_j. Therefore, if $\mathcal{A}_0$ corrupts sk_i and $\mathcal{A}_1$ calls $\mathsf{Chal}_{\mathsf{ReKeyGen}}(i, j) \overset{(\$)}{\rightarrow} \Delta^*$, they can compute some $\Delta_{j,i}$. If $\mathcal{A}_1$ sets some $c \overset{\$}{\leftarrow} \mathsf{Enc}(\mathsf{pk}_j, m)$, then they can compute $m' \leftarrow \mathsf{Dec}(\Delta_{j,i}, \mathsf{ReEnc}(\Delta^*, c))$. Therefore, $\mathcal{A}_1$ can output $b = 0$ if $m' = m$ and $b' = 1$ otherwise to always win the game. We conclude that PRE schemes with token source privacy must be unidirectional. $\square$

We note that Lemma VI.2 does not apply to weak TSP. Furthermore, by Lemma VI.2, schemes with adaptive TSP also must be unidirectional and have a probabilistic $\mathsf{ReKeyGen}$ (Lemmas VI.3 and VI.4).

A crucial and initially surprising result which we use going forward, is to demonstrate that adaptive token source privacy is already implied by token source privacy.

Theorem VI.2. *If a PRE scheme $\mathcal{PRE}$ is (ϵ, κ)-TSP, it is (ϵ, κ)-adaptive TSP.*

Proof. We first demonstrate that the adaptive adversary cannot corrupt any keys that could not have been corrupted by a selective adversary. In both $\mathsf{TSP}^{\mathcal{PRE}}_{\mathcal{A},\kappa}(1^\lambda)$ and $\mathsf{a\text{-}TSP}^{\mathcal{PRE}}_{\mathcal{A},\kappa}(1^\lambda)$, the set of challenge keys is created by adding $\{\mathsf{sk}_j\}$ whenever $\mathsf{Chal}_{\mathsf{ReKeyGen}}(i, j)$ is queried. We note that because of the trivial win condition, the only keys that the adaptive adversary can corrupt are those which are a source in the re-encryption graph, in other words, keys that are not reachable from any other vertex. This is true for all re-encryption graphs. Without loss of generality, we can assume that by the end of the game, both the selective and adaptive adversaries have ensured that every key is either a challenge key or a corrupted key. It follows that for every re-encryption graph, the adaptive adversary does not corrupt any keys that are not corrupted by the selective adversary.

It may be the case that the challenge graph that gives the highest advantage varies depending on the specific set of keys, and so the adaptive adversary has an advantage in determining this graph. The crucial difference between the selective and adaptive adversaries is that after receiving a challenge token $\Delta^*_{i,j}$ the adaptive adversary can decide whether sk_i should become a corrupted key or a challenge key, whereas the selective adversary can only choose for sk_i to be a challenge key. More specifically, assuming a challenge $\Delta^*_{i,j}$ has been learned, the adaptive adversary can choose whether to extend the re-encryption graph by querying $\mathsf{Chal}_{\mathsf{ReKeyGen}}(h, i)$ for some key index h (adding sk_i to the set of challenge keys), or whether to corrupt sk_i. We note that the adaptive adversary cannot decide to corrupt sk_j, as this breaks the trivial win condition. It remains to show that this ability cannot give the adaptive adversary any advantage in determining b that the selective adversary does not have.

We note that a challenge token in the TSP experiment only depends on the two input keys. This differs from confidentiality notions of PRE schemes, as in these the challenge ciphertext can be re-encrypted through a number of keys and therefore be a product of all prior keys. This means that for any individual challenge token, any possible advantage in distinguishing the source key must come from values created using the same source keys, specifically other challenge tokens from the same source key.

Suppose that sk_i is uncorrupted, and that the adversary has learned a set of challenge tokens with the same source key $(\{\Delta^*_{i,j} \overset{\$}{\leftarrow} \mathsf{Chal}_{\mathsf{ReKeyGen}}(i,j) : j \in \{1,\ldots,\kappa\}\})$. Now suppose this reveals some information about the source key which helps the adversary to determine whether sk_i should become a corrupted key or a challenge key, and that leads to an additional advantage in the adaptive game but not the selective game. If this information could have been inferred from the public keys without any challenge tokens, then the selective adversary would also be able to determine whether sk_i should be corrupted. For the adaptive adversary to have a greater advantage, it would have to learn something that depended on the set of challenge tokens. However, if $\Delta^*_{i,j}$ reveals whether the specific underlying secret key sk_i of these tokens $\{\Delta^*_{i,j}\}$ should be corrupted or challenged, then this implies it is possible to infer the source key, meaning the selective adversary could use the information to determine b from $\Delta^*_{i,j}$, and gain the same advantage. We conclude that TSP implies that being able to adaptively corrupt source keys after receiving challenge tokens gives no advantage to the adversary in the adaptive TSP experiment. $\square$

VI: 6 Adaptive CPA security

In this section, we demonstrate how a PRE scheme with TSP may also be adaptively PRE-IND-CPA-secure. We begin by examining key corruptions in PRE-IND-CPA security. We first discuss the relationship between fixed and selective security and whether these can be considered equivalent, before demonstrating how to prove adaptive PRE-IND-CPA security at a sub-exponential loss when re-encryption graphs are determined adaptively.

VI: 6.1 Fixed and Selective PRE-IND-CPA security

Here, we argue that it is reasonable to consider fixed-key PRE-IND-CPA security (Definition VI.2) and selective PRE-IND-CPA security (Definition III.7) to be equivalent. We demonstrate this through two different approaches. In the first approach, we demonstrate that if a PRE scheme is PKE-IND-CPA-secure and has weak token source privacy, then it is both fixed-key PRE-IND-CPA-secure and selectively PRE-IND-CPA-secure. In the second approach, we use a statistical argument to show that if the number of corrupted keys in $\mathsf{f\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ and $\mathsf{PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ is the same, then the difference in advantage between the two experiments is negligible.

Approach 1

We begin by proving that a PRE scheme which has PKE-IND-CPA security and weak token source privacy is both fixed-key PRE-IND-CPA-secure and selectively PRE-IND-CPA-secure.

Theorem VI.3. *Let $\mathcal{PRE}$ be a PRE scheme that is (ϵ_1, κ)-PKE-IND-CPA-secure and has (ϵ_2, κ)-weak TSP. Then $\mathcal{PRE}$ is $(\epsilon_1 + \epsilon_2, \kappa)$-fixed-key PRE-IND-CPA-secure.*

Proof. Suppose there exists an adversary $\mathcal{B}$ that wins $\mathsf{f\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{B}}^{\mathcal{PRE}}(1^\lambda)$ with advantage ϵ. If an adversary $\mathcal{A}$ in $\mathsf{PKE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ can simulate $\mathsf{f\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$, then they can call $\mathcal{B}$ as a subroutine and output the same guess to gain the same advantage.

We demonstrate how $\mathcal{A}$ can simulate $\mathsf{f\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$. Let $\mathsf{Game}_0 = \mathsf{f\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$, and let Game_1 be a variant of $\mathsf{f\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ where update tokens between uncorrupted keys, $\Delta_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$, are replaced by tokens from independent source keys, $\overline{\Delta_{i,j}} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\overline{\mathsf{sk}_i}, \mathsf{pk}_j)$, where $(\overline{\mathsf{sk}_i}, \overline{\mathsf{pk}_i}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$ is independent of $(\mathsf{pk}_i, \mathsf{sk}_i)$. $\mathcal{A}$ can simulate Game_1 by generating its own key pairs $(\overline{\mathsf{pk}_i}, \overline{\mathsf{sk}_i})$ for each uncorrupted key pair, and use these as the source keys for token generation and re-encryption queries between uncorrupted keys.

We now show that no PPT distinguisher $\mathcal{D}$ can distinguish between Game_0 and Game_1 with advantage greater than ϵ_2. Let $\mathcal{B}$ be an adversary in the wTSP experiment with challenge bit $d \in \{0,1\}$. $\mathcal{B}$ can perfectly simulate Game_d as follows:

- $\mathcal{B}$ first samples $b \overset{\$}{\leftarrow} \{0,1\}$ to be the challenge bit in Game_d.
- When $\mathcal{D}$ calls $\mathsf{O}_{\mathsf{HonKeyGen}}$, $\mathcal{B}$ returns one of the public keys it received in the wTSP experiment that it has not returned in a call to $\mathsf{O}_{\mathsf{CorKeyGen}}$.
- When $\mathcal{D}$ calls $\mathsf{O}_{\mathsf{CorKeyGen}}$, $\mathcal{B}$ takes one of the public keys it received in the wTSP experiment that it has not returned in a call to $\mathsf{O}_{\mathsf{HonKeyGen}}$, corrupts the corresponding secret key then returns $(\mathsf{pk}, \mathsf{sk})$ to $\mathcal{D}$.

- When $\mathcal{D}$ calls $\mathsf{O}_{\mathsf{ReKeyGen}}(i, j)$:
  - If sk_i is a corrupted key, then $\mathcal{B}$ generates $\Delta_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$ and returns $\Delta_{i,j}$ to $\mathcal{D}$.
  - If sk_i is uncorrupted and sk_j is corrupted, $\mathcal{B}$ returns $\perp$, as specified in both Game_0 and Game_1.
  - If both sk_i and sk_j are uncorrupted, then $\mathcal{B}$ calls the wTSP challenge oracle $\mathsf{Chal}_{\mathsf{ReKeyGen}}(i, j) \overset{(\$)}{\rightarrow} \Delta^*_{i,j}$ and returns $\Delta^*_{i,j}$ to $\mathcal{D}$.
- When $\mathcal{D}$ calls $\mathsf{O}_{\mathsf{ReEnc}}(i, j, \Delta_{i,j}, c)$, $\mathcal{B}$ first performs the checks described in $\mathsf{f\text{-}PRE\text{-}IND\text{-}CPA}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$. If $(i, j, \Delta_{i,j}) \in \mathcal{T}_{\mathsf{honest}}$, then $\mathcal{B}$ calculates $\Delta_{i,j}$ as described for calls to $\mathsf{O}_{\mathsf{ReKeyGen}}$. $\mathcal{B}$ then computes $c' \overset{(\$)}{\leftarrow} \mathsf{ReEnc}(\Delta_{i,j}, c)$ and returns c' to $\mathcal{D}$.
- When $\mathcal{D}$ calls $\mathsf{Chal}_{\mathsf{Enc}}(i, m_0, m_1)$, $\mathcal{B}$ performs the appropriate checks, and if the checks pass computes $c_0 \overset{\$}{\leftarrow} \mathsf{Enc}(\mathsf{pk}_i, m_0), c_1 \overset{\$}{\leftarrow} \mathsf{Enc}(\mathsf{pk}_i, m_1)$ and returns c_b to $\mathcal{D}$.

We see that if $d = 0$, tokens have the correct source keys and thus this simulates Game_0 perfectly, and if $d = 1$, tokens have the independent source keys and thus simulates Game_1 perfectly. As $\mathcal{PRE}$ is (ϵ_2, κ)-weak TSP, we conclude that advantage that $\mathcal{D}$ has in distinguishing between the two is bounded by ϵ_2. We conclude that the advantage ϵ of any PPT adversary winning $\mathsf{f\text{-}PRE\text{-}IND\text{-}CPA}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$ is bounded by $\epsilon \leq \epsilon_1 + \epsilon_2$, as required. □

Theorem VI.4. *If a PRE scheme $\mathcal{PRE}$ is (ϵ_1, κ)-PKE-IND-CPA-secure and (ϵ_2, κ)-weak TSP, then it is also selectively $(\epsilon_1 + \epsilon_2, \kappa)$-PRE-IND-CPA-secure.*

Proof. Similarly to proof of Theorem VI.3, let $\mathsf{Game}_0 = \mathsf{PRE\text{-}IND\text{-}CPA}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$, and let Game_1 be a variant of $\mathsf{PRE\text{-}IND\text{-}CPA}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$ where update tokens between uncorrupted keys $\Delta_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$ are replaced by tokens from independent source keys, $\overline{\Delta_{i,j}} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\overline{\mathsf{sk}}_i, \mathsf{pk}_j)$, where $(\overline{\mathsf{sk}}_i, \overline{\mathsf{pk}}_i) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$ is independent of $(\mathsf{pk}_i, \mathsf{sk}_i)$. An adversary in $\mathsf{PKE\text{-}IND\text{-}CPA}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$ can simulate Game_1 in the same way as described in proof of Theorem VI.3.

The rest of the proof is identical to proof of Theorem VI.3, except that when $\mathcal{D}$ queries $\mathsf{O}_{\mathsf{KeyGen}}$, $\mathcal{B}$ returns the next public key given in the wTSP experiment, and when $\mathcal{D}$ queries $\mathsf{O}_{\mathsf{Corrupt}}$, $\mathcal{B}$ forwards this to the challenger in the wTSP experiment and returns the corresponding secret key. □

The results of Theorems VI.3 and VI.4 are intuitive, as if the message m_b underlying the challenge ciphertext is revealed during re-encryption, then it follows that re-encryption must involve the correct source key, as otherwise it would be possible to distinguish the message without the secret key and so the underlying encryption scheme would not be PKE-IND-CPA-secure.

Remark. As we have discussed, most security in PRE is (or can be) proven by simulating re-encryption tokens from uncorrupted source keys, and showing that this simulation is indistinguishable. This property is precisely what we have formalised in token source privacy. We therefore remark that, if this is the case for all PRE schemes (and we believe it is likely for the majority of schemes), then Theorems VI.3 and VI.4 can be applied. This indicates the possibility that fixed-key and selective PRE-IND-CPA-security are equivalent.

Approach 2

We now demonstrate further reasoning to consider fixed and selective key corruptions equivalent in our second approach, which uses a statistical argument.

Lemma VI.5. *Consider the two security games* $\mathsf{f\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ *and* $\mathsf{PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$*. If the number of non-trivial* $\mathsf{O}_{\mathsf{Corrupt}}$ *queries (queries that do not return* $\perp$*) made by* $\mathcal{A}_0^{\mathsf{sel}}$ *(the first-stage adversary in the selective PRE-IND-CPA experiment), and the number of* $\mathsf{O}_{\mathsf{CorKeyGen}}$ *queries made by* $\mathcal{A}_0^{\mathsf{fix}}$ *(the first-stage adversary in the fixed-key PRE-IND-CPA experiment) is the same, then*

$$\Pr\left[\mathsf{f\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1\right] \approxeq \Pr\left[\mathsf{PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1\right].$$

Proof. If we fix the number of corrupted keys learned by $\mathcal{A}_0^{\mathsf{fix}}$ and $\mathcal{A}_0^{\mathsf{sel}}$ to N, then by the end of the first stage the distribution of information received by both adversaries is identical. In the first stage the adversary can learn the security parameter, a set of public keys, and a subset of secret keys associated with the set of public keys. Since all of these keys are independent in the first stage, if N is fixed, then the distribution of information is identical. Since the oracles and restrictions on oracle queries in both experiments in the second stage is identical (i.e. we can simulate a second-stage selective challenger with access to a second-stage fixed challenger), then it is clear that the distributions of both second-stage adversaries is identical as well. This gives us our result. □

In the first approach, we argued that in practice, proofs of selective PRE-IND-CPA security rely on weak TSP, which we have demonstrated can prove both the fixed and selective variants. Together with the statistical argument given in the second approach, we conclude that it is reasonable to believe the fixed and selective approaches to defining selective PRE-IND-CPA security are equivalent.

VI: 6.2 Adaptive PRE-IND-CPA security

We now move on to demonstrating how TSP can imply adaptive security for PRE-IND-CPA.

Theorem VI.5. *If a PRE scheme is (ϵ_1, κ)-PKE-IND-CPA-secure and has (ϵ_2, κ)-adaptive TSP, then it is also $(\epsilon_1 + \epsilon_2, \kappa)$-adaptively PRE-IND-CPA-secure by the challenge independence assumption.*

Proof. We start by noting that, by the challenge graph observation, it is sufficient to only consider adversaries who do not query $\mathsf{O}_{\mathsf{Corrupt}}(j)$ if they have queried either $\mathsf{O}_{\mathsf{ReKeyGen}}(i, j)$ or $\mathsf{O}_{\mathsf{ReEnc}}(i, j, \Delta_{i,j}, c)$, as this will result in a subgraph that is not in the challenge graph.

Suppose there exists an adversary $\mathcal{B}$ that wins $\mathsf{a\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ with advantage ϵ. If an adversary $\mathcal{A}$ in $\mathsf{PKE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ can simulate $\mathsf{f\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$, then they can call $\mathcal{B}$ as a subroutine and output the same guess to gain the same advantage.

Let $\mathsf{Game}_0 = \mathsf{a\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$, and let Game_1 be a variant of $\mathsf{a\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ where all update tokens, $\Delta_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$, are replaced by tokens from independent source keys, $\overline{\Delta}_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\overline{\mathsf{sk}}_i, \mathsf{pk}_j)$, where $(\overline{\mathsf{sk}}_i, \overline{\mathsf{pk}}_i) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$ is independent of $(\mathsf{pk}_i, \mathsf{sk}_i)$. Recall from Section VI: 4 that the challenge independence assumption implies that a variant of $\mathsf{PKE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ where the adversary can corrupt non-challenge keys (at any point) as long as they do not corrupt any challenge keys is equivalent to the original experiment with no corruption oracle. We note that an adversary $\mathcal{A}$ in this variant can simulate the Game_1 by forwarding calls to $\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Corrupt}}$ and $\mathsf{Chal}_{\mathsf{Enc}}$ to the challenger in the PKE-IND-CPA variant. $\mathcal{A}$ also generates its own key pair $(\overline{\mathsf{pk}}_i, \overline{\mathsf{sk}}_i) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$ for every call to $\mathsf{O}_{\mathsf{KeyGen}}$. Then every time a token is generated between keys sk_i and pk_j, $\mathcal{A}$ instead generates $\overline{\Delta}_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\overline{\mathsf{sk}}_i, \mathsf{pk}_j)$.

We now show that no PPT distinguisher $\mathcal{D}$ can distinguish between Game_0 and Game_1

with advantage greater than ϵ_2. Let $\mathcal{B}$ be an adversary in the $\mathsf{a\text{-}TSP}$ experiment with challenge bit $d \in \{0,1\}$. $\mathcal{B}$ can perfectly simulate Game_d as follows:

- $\mathcal{B}$ first samples $b \xleftarrow{\$} \{0,1\}$ to be the challenge bit in Game_d.
- When $\mathcal{D}$ calls $\mathsf{O}_{\mathsf{KeyGen}}$, $\mathcal{B}$ returns one of the public keys it received in the $\mathsf{a\text{-}TSP}$ experiment.
- When $\mathcal{D}$ calls $\mathsf{O}_{\mathsf{Corrupt}}$, $\mathcal{B}$ takes forwards this to the $\mathsf{a\text{-}TSP}$ challenger.
- When $\mathcal{D}$ calls $\mathsf{O}_{\mathsf{ReKeyGen}}(i,j)$, $\mathcal{B}$ calls the $\mathsf{a\text{-}TSP}$ challenge oracle $\mathsf{Chal}_{\mathsf{ReKeyGen}}(i,j) \xrightarrow{(\$)} \Delta_{i,j}$ and returns $\Delta_{i,j}$ to $\mathcal{D}$.
- When $\mathcal{D}$ calls $\mathsf{O}_{\mathsf{ReEnc}}(i,j,\Delta_{i,j},c)$, $\mathcal{A}$ first performs the checks described in $\mathsf{f\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$. If $(i,j,\Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}$, then $\mathcal{B}$ calculates $\Delta_{i,j}$ as described for calls to $\mathsf{O}_{\mathsf{ReKeyGen}}$. If the checks pass then $\mathcal{B}$ then computes $c' \xleftarrow{(\$)} \mathsf{ReEnc}(\Delta_{i,j},c)$ and returns c' to $\mathcal{D}$.
- When $\mathcal{D}$ calls $\mathsf{Chal}_{\mathsf{Enc}}(i,m_0,m_1)$, $\mathcal{B}$ performs the appropriate checks, and if the checks pass returns $c^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i,m_b)$.

Recall that the challenge graph observation implies that we only need to consider cases where $\mathcal{D}$ does not query both $\mathsf{O}_{\mathsf{Corrupt}}(j)$ and either of $\mathsf{O}_{\mathsf{ReKeyGen}}(i,j)$ or $\mathsf{O}_{\mathsf{ReEnc}}(i,j,\Delta_{i,j},c)$ for some $(i,j,\Delta_{i,j},c)$. This means that the trivial win condition in the $\mathsf{a\text{-}TSP}$ experiment does not restrict the calls that $\mathcal{D}$ makes to $\mathsf{O}_{\mathsf{Corrupt}}$. We see that if $d=0$, tokens have the correct source keys and thus this simulates Game_0 perfectly, and if $d=1$, tokens have independent source keys and thus simulates Game_1 perfectly. As $\mathcal{PRE}$ has (ϵ_2,κ)-adaptive TSP, we conclude that the advantage that $\mathcal{D}$ has in distinguishing between the two is bounded by ϵ_2. Overall, we see that the advantage ϵ of any PPT adversary winning $\mathsf{a\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ is bounded by $\epsilon \leq \epsilon_1 + \epsilon_2$, as required. □

Combining Theorem VI.2 and Theorem VI.5 gives the following result:

Theorem VI.6. *If a PRE scheme is (ϵ_1,κ)-PKE-IND-CPA-secure and (ϵ_2,κ)-TSP, then it is also $(\epsilon_1+\epsilon_2,\kappa)$-adaptively PRE-IND-CPA-secure.*

As many PRE schemes are already proven for the selective PRE-IND-CPA setting, the following theorem may be more useful in demonstrating that these schemes are also secure in the adaptive model.

Theorem VI.7. *If a PRE scheme $\mathcal{PRE}$ is (ϵ_1, κ)-PRE-IND-CPA-secure and (ϵ_2, κ)-TSP, then $\mathcal{PRE}$ is also $(\epsilon_1 + \epsilon_2, \kappa)$-adaptively PRE-IND-CPA-secure.*

Proof. Note if $\mathcal{PRE}$ is (ϵ_1, κ)-PRE-IND-CPA-secure then it is also (ϵ_1, κ)-PKE-IND-CPA-secure, as PRE-IND-CPA security is a strictly stronger security property than PKE-IND-CPA security. Leveraging Theorem VI.6 gives the result. □

In summary, we have demonstrated that PKE-IND-CPA security and token source privacy implies adaptive IND-CPA security. As discussed in Section VI: 5, it appears that token source privacy is already a property of existing public-key PRE schemes where ReKeyGen is probabilistic, and therefore these schemes are adaptively PRE-IND-CPA-secure. In particular, in Section VI: 8 we will demonstrate that pcBV-PRE is adaptively PRE-IND-CPA-secure.

VI: 7 Extension to HRA security and PCS

We now move on to exploring under what conditions an adaptively PRE-IND-CPA-secure PRE scheme is also adaptively IND-HRA-secure. Whilst most of the oracle queries in HRA games can be forwarded to the equivalent CPA game, the distinction comes in how re-encryptions of non-challenge ciphertexts are handled. These cannot be forwarded to the re-encryption oracle in the CPA game, as this will result in an edge being added to the DRG in the CPA game which is not added in the HRA game, which in turn may lead to the trivial win condition being triggered by the CPA challenger but not the HRA challenger. To prove security against Honest Re-Encryption Attacks, one approach is to replace the outputs of $\mathsf{O}_{\mathsf{ReEnc}}(i, j, \Delta_{i,j}, c)$ where c is a non-challenge ciphertext with a value that has been computed without the source secret key sk_i. This means that this new ability to re-encrypt from the challenge key to corrupted keys does not give any information about sk_i (which could be a challenge key), meaning these re-encryptions give no advantage in winning the game. Note that this can be considered an existing use of the challenge independence assumption.

Existing work defines properties that imply such replacements will be undetected. As discussed in Section V: 2.2, one such property is *re-encryption simulatability* [Coh19, Definition 7] (see Definition V.3) which informally states that there exists an algorithm

$\mathsf{ReEncSim}$ which, given $(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{sk}_j, c, m)$, can simulate a re-encryption of a ciphertext without knowing the old secret key, sk_i. In [Coh19, Theorem 5] Cohen[2] proposes a theorem stating that PRE schemes that are selectively PRE-IND-CPA-secure are also adaptively PRE-IND-CPA-secure. In [Coh17, Coh19] Cohen provides a proof sketch that indicates how re-encryptions from uncorrupted keys to corrupted keys can be simulated using $\mathsf{ReEncSim}$ – because the ciphertext is honestly generated, the selective adversary can keep a lookup table and retrieve the underlying message, m, for ciphertext, $c \xleftarrow{\$} \mathsf{O}_{\mathsf{Enc}}(i, m)$, and therefore knows the information necessary to simulate the re-encryption using $\mathsf{ReEncSim}$. As sk_j is corrupted, the selective adversary knows all the necessary information to compute the simulated re-encryption $c' \xleftarrow{(\$)} \mathsf{ReEncSim}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{sk}_j, c, m)$.

Re-Encryption simulatability cannot be used in the same way to prove that adaptive CPA security implies adaptive HRA security. This is because the adaptive adversary can request the re-encryption of a non-challenge ciphertext before any keys are corrupted. As re-encryption simulatability assumes sk_j is known to the adversary, $\mathsf{ReEncSim}$ cannot necessarily be used to make this replacement. This problem does not exist for selective security as in this setting re-encryption happens in the second phase, when the set of corrupted keys is known. Furthermore, the adaptive adversary might later query $\mathsf{O}_{\mathsf{ReKeyGen}}(i, j)$, meaning that the simulated ciphertext would need to remain indistinguishable even if the adaptive adversary later learns some $\Delta_{i,j}$, which is not guaranteed by re-encryption simulatability.

This observation was also made by Fuchsbauer et al., who introduced *source-hiding* [FKKP19, Definition 9] (see Definition V.4) as an alternative property which indicates that re-encryption of honest ciphertexts can be simulated in the adaptive setting. Source-hiding was introduced for a variant of PRE where encryption takes an additional input ℓ to indicate the *level* for the ciphertext (the number of times the ciphertext appears to have been re-encrypted). In other words, $\mathsf{Enc}(\mathsf{pk}, \ell, m)$ produces an encryption of m under pk that appears to have been re-encrypted ℓ times. Recall from Section V: 2.2 that informally, source-hiding implies that re-encrypted ciphertexts have the same distribution as fresh ciphertexts at the appropriate level, even when the adversary knows both key pairs and the update token.

In [FKKP18, Section 4.2.1] Fuchsbauer et al. demonstrate how a PRE scheme that

[2]Cohen's definition of selective security uses the fixed key corruption approach.

is adaptively PRE-IND-CPA-secure and source-hiding is also adaptively IND-HRA-secure. In their proof, the use of source-hiding incurs an exponential security loss whose factor is based on the number of keys, the number of encryption queries and the number of re-encryption queries, as we demonstrated in the proof of Theorem V.3. This is because of how source-hiding is defined. Firstly, source-hiding only has two key pairs, meaning that in order to embed a source-hiding challenge into a reduction, the simulator must guess in advance which keys the adaptive adversary will request the re-encryption for. Secondly, as the source-hiding challenge gives both the re-encrypted ciphertext and the input ciphertext, the simulator must guess which ciphertext will become the input for the re-encryption, and return this at the appropriate place.

As we aim for a smaller security loss, we opt for a different approach. Additional reasons why source-hiding is a less ideal solution include:

1. Most existing PRE schemes do not have explicit levelling, and so it is harder to apply source-hiding to these schemes. Hypothetically, one could define variants of existing PRE schemes to create levelled encryption, but this requires adapting the existing scheme and therefore makes comparisons less clear.
2. All known public-key PRE schemes that are source-hiding such as [FKKP19, Construction 2.b] use a '*blurring*' approach [CCL$^+$14]. Here extra randomness is added on each re-encryption to mask traces of the original ciphertext, and the amount of randomness needed increases dramatically for each level. This means the constructions rely on heavier parameter choices compared to other lattice-based PRE schemes, meaning they are less efficient compared to other constructions. The blurring approach also limits the number of times a ciphertext can be re-encrypted and still decrypt correctly.

As we noted in Section V: 3.4, in practice, security proofs are usually based on the keys as opposed to the update tokens, we believe that a multi-key, multi-challenge notion would lead to a smaller overall loss in proving adaptive HRA security.

VI: 7.1 The Challenge Graph Observation for HRA

In addition to the outlined problems with relying on source-hiding, we conjecture that source-hiding is actually stronger than what is required to make the appropriate replacement, by the challenge graph observation. In source-hiding, the distinguisher receives the maximum amount of information – the message m, old key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$, new key pair $(\mathsf{pk}_j, \mathsf{sk}_j)$ original ciphertext c and update token $\Delta_{i,j}$. This means that both sk_i and sk_j are corrupted. However, we see by the challenge graph observation that if both keys are corrupted, then their respective nodes cannot be in the challenge subgraph and therefore we do not need to consider adversaries that make this combination of queries. We can therefore define a weaker notion which can hopefully be met by a larger number of PRE schemes, opening the possibility of proving adaptive HRA security for more efficient constructions.

We now investigate what kinds of auxiliary information the simulated re-encryption must remain indistinguishable for, according to the challenge graph observation. Suppose that an adversary $\mathcal{A}$ first queries $\mathsf{O}_{\mathsf{Enc}}(i, m) \overset{(\$)}{\to} c$ followed by $\mathsf{O}_{\mathsf{ReEnc}}(i, j, c)$ (see Figure VI.4). To demonstrate possible relations to the challenge, now suppose that $\mathcal{A}$ learns a challenge under a different key pk_{i^*}, followed by an update token from pk_{i^*} to some independent pk_{j^*} (see Figure VI.5). Note that the challenge graph currently contains v_{i^*} and v_{j^*}, as v_{j^*} is reachable from v_{i^*}.

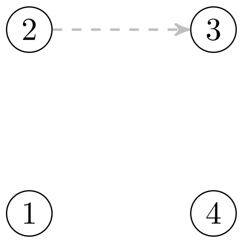

Figure VI.4: $\mathsf{O}_{\mathsf{ReEnc}}(2, 3, c)$ where $c \leftarrow \mathsf{O}_{\mathsf{Enc}}(2, m)$ for some m.

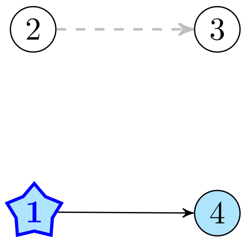

Figure VI.5: Figure VI.4 followed by $\mathsf{Chal}_{\mathsf{Enc}}(1, m_0, m_1)$ and $\mathsf{O}_{\mathsf{ReKeyGen}}(1, 4)$.

Figure VI.6: In these re-encryption graphs, dashed, grey lines represent re-encryptions of non-challenge ciphertexts, whilst solid edges are added for calls to $\mathsf{O}_{\mathsf{ReKeyGen}}$ and calls to $\mathsf{O}_{\mathsf{ReEnc}}$ for challenge ciphertexts.

We now observe that $\mathcal{A}$ has three main options for corruption regarding $\mathsf{sk}_i, \mathsf{sk}_j$ and $\Delta_{i,j}$.

1. **$\mathcal{A}$ later corrupts both sk_i and sk_j.**

   If this happens, then no challenges can be learned under either of these keys, by the trivial win condition. Therefore, v_i and v_j will be unreachable from the challenge vertex and therefore not in the challenge subgraph, as shown in Figure VI.7. We therefore do not need to consider adversaries who query $\mathsf{O}_{\mathsf{ReEnc}}(i, j, \perp, c)$ for a non-challenge ciphertext c, followed by $\mathsf{O}_{\mathsf{Corrupt}}(i)$, and $\mathsf{O}_{\mathsf{Corrupt}}(j)$ (in any order).

2. **$\mathcal{A}$ later corrupts sk_j and learns some $\Delta_{i,j}$.**

   Because there is now a path from v_i to v_j, again we have that no challenge can be learned under either of these keys, so again, v_i and v_j are not in the challenge subgraph, as shown in Figure VI.8. We therefore do not need to consider adversaries who query $\mathsf{O}_{\mathsf{ReEnc}}(i, j, \perp, c)$ for a non-challenge ciphertext c, followed by $\mathsf{O}_{\mathsf{Corrupt}}(j)$, and $\mathsf{O}_{\mathsf{ReKeyGen}}(i, j)$ (in any order).

3. **$\mathcal{A}$ later corrupts sk_i and learns some $\Delta_{i,j}$.**

   Similarly to the first point, we do not need to consider $\mathcal{A}$ also corrupting sk_j. This means a challenge could still be learned under pk_j, as shown in Figure VI.9.

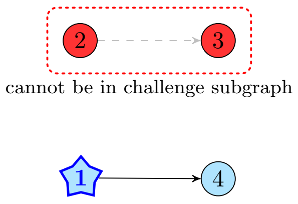

Figure VI.7: Figure VI.5 followed by $\mathsf{O}_{\mathsf{Corrupt}}(3)$ and $\mathsf{O}_{\mathsf{Corrupt}}(2)$ (in any order).

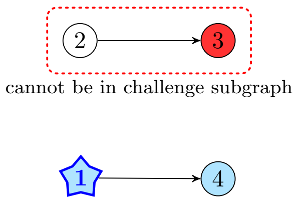

Figure VI.8: Figure VI.5 followed by $\mathsf{O}_{\mathsf{Corrupt}}(3)$ and $\mathsf{O}_{\mathsf{ReKeyGen}}(2, 3)$ (in any order).

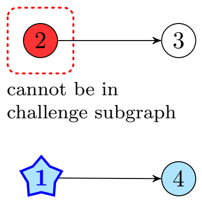

Figure VI.9: Figure VI.5 followed by $\mathsf{O}_{\mathsf{Corrupt}}(2)$ and $\mathsf{O}_{\mathsf{ReKeyGen}}(2, 3)$ (in any order). v_3 could still become part of the challenge graph.

Overall, the challenge graph observation implies that to replace outputs of $\mathsf{O}_{\mathsf{ReEnc}}(i, j, c)$ for a non-challenge c, it is sufficient to consider the case where the replacement is indistinguishable given *either* sk_j *or* $(\mathsf{sk}_i, \Delta_{i,j})$.

VI: 7.2 Our approach: Re-encryption Replaceability

As we have seen, the current method for proving HRA security is to demonstrate that a selective adversary can simulate the HRA game by replacing the re-encryptions of non-challenge ciphertexts with an alternative calculated using information available to the selective adversary. In particular, simulations must be possible when sk_i has not been corrupted, meaning the simulated re-encryption is independent of sk_i. Motivated by existing definitions and the challenge graph observation, in this section we present our new definition, *re-encryption replaceability*.

Similar to re-encryption simulatability, instead of needing a specific function, we just need to demonstrate that there *exists* a function that can make the appropriate replacement. We need the inputs to the function to be those available to the adaptive CPA adversary $\mathcal{A}^{\mathsf{CPA}}$ when no corruptions have been made and no update tokens have been learned, but have the replacement be indistinguishable from a genuine re-encryption even given values the HRA adversary $\mathcal{A}^{\mathsf{HRA}}$ may learn later. Recall that when simulating the HRA game $\mathcal{A}^{\mathsf{CPA}}$ simulates the encryption oracle $\mathsf{O}_{\mathsf{Enc}}(i, m)$ by computing $c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m)$ and updating the appropriate lists. We note that, because $\mathcal{A}^{\mathsf{CPA}}$ performs this encryption locally, it will know any random values used to create the ciphertext, and therefore can use these when simulating re-encryptions. This becomes useful in our proofs as this information may enable the challenger to replace re-encryptions without them needing to know the corresponding secret key, unlike re-encryption simulatability. This idea is similar to those of *randomness-recovery* and *perfect re-encryption* in [KLR19b]. To make discussion simpler, we define the following.

Definition VI.9. *For an algorithm* $\mathsf{F} : \mathcal{X} \to \mathcal{Y}$*, let* $\overline{\mathsf{F}} : \mathcal{X} \to \mathcal{Y} \times \mathcal{R}$ *be the variant of* F *which also returns any values randomly sampled to produce the output. If* F *is deterministic, then we say* $\overline{\mathsf{F}}(x) \to (y, \emptyset)$*.*

For example, for ElGamal encryption (Figure II.1) where $\mathsf{Enc}(\mathsf{pk} = g^x, m)$ returns $c = (g^y, g^{yx} \cdot m)$, $\overline{\mathsf{Enc}}(\mathsf{pk}, m)$ returns (c, y).

Definition VI.10. *For an algorithm* $\mathsf{F} : \mathcal{X} \to \mathcal{Y}$*, let* $\mathsf{det\text{-}F} : \mathcal{X} \times \mathcal{R} \to \mathcal{Y}$ *be the deterministic variant of* F *where the random values are explicitly given as input. In other words,* $\mathsf{det\text{-}F}$ *is such that for* $\overline{\mathsf{F}}(x) \xrightarrow{\$} (y, r)$*,* $\mathsf{det\text{-}F}(x; r) = y$*.*

Finally, we observe that we should not need to know whether the adaptive adversary will later learn sk_j or $(\mathsf{sk}_i, \Delta_{i,j})$ in advance of making the replacement, as long as the same function can be used to replace the re-encryption in both cases.

This inspires our new definition, *re-encryption replaceability.* Informally, if a PRE scheme is re-encryption replaceable, then re-encrypted ciphertexts appear to be independent of any secret keys the ciphertext was previously encrypted under. We define two types – one where indistinguishability still holds if the new secret key is learned, and the other where indistinguishability still holds if the old secret key and an update token are learned. Both will be used in our relation between CPA and HRA security. We note that re-encryption replaceability is optimal in the sense that inputs to $\mathsf{ReEncRep}$ are minimal in terms of what may be known to a selective adversary hoping to simulate the adaptive game, but maximal as it also considers the randomness the selective adversary uses to encrypt ciphertexts, thereby modelling all the information known to the selective adversary. We present two variants, a statistical definition in Definition VI.11 and a game-based definition in Definition VI.14.

We acknowledge that statistical security notions are generally considered to be weaker than computational (game-based) ones. However, in practice, proofs are often statistical. In particular, if either $\mathsf{ReKeyGen}$ or ReEnc are probabilistic, then there is more than one possible re-encrypted ciphertext $c' \overset{(\$)}{\leftarrow} \mathsf{ReEnc}(\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j), c)$ for each input ciphertext, c. Therefore, proofs do not demonstrate that the output is a specific value, but rather that it appears to come from the correct distribution. This appears to be why source-hiding is defined as a statistical notion in [FKKP19]. In other words, we find that in practice, a statistical notion is not necessarily more difficult to prove than a computational notion, and this is why we define the statistical notion of re-encryption replaceability as well as the computational variant.

Definition VI.11. *Consider a PRE scheme, $\mathcal{PRE}$. Let* $(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$*. We say that $\mathcal{PRE}$ is* type-N statistically κ-re-encryption replaceable *for $N \in \{1, 2\}$ if there exists an algorithm* $\mathsf{ReEncRep}$ *such that with overwhelming probability over* aux_N*, for all $m \in \mathcal{M}$, all* $(\mathsf{pk}_i, \mathsf{sk}_i), (\mathsf{pk}_j, \mathsf{sk}_j) \in \{(\mathsf{pk}_\iota, \mathsf{sk}_\iota)\}_{\iota=1}^{\kappa}$ *and all ciphertexts c where* $(c, r_c) \overset{\$}{\leftarrow} \overline{\mathsf{Enc}}(\mathsf{pk}_i, m)$*:*

$$(\mathsf{ReEncRep}(\mathsf{pk}_i, \mathsf{pk}_j, m, c, r_c), \mathsf{aux}_N) \approx_s (\mathsf{ReEnc}(\Delta_{i,j}, c), \mathsf{aux}_N)$$

where $\approx_s$ denotes statistical indistinguishability and inputs are sampled according to

$$\begin{aligned}
&\Delta_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)\\
&\Delta'_{i,j} \overset{(\$)}{\leftarrow} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)\\
&\mathsf{aux}_1 = (\mathsf{pk}_i, \mathsf{pk}_j, c, m, \mathsf{sk}_j)\\
&\mathsf{aux}_2 = (\mathsf{pk}_i, \mathsf{pk}_j, c, m, \mathsf{sk}_i, \Delta'_{i,j}).
\end{aligned}$$

If $\mathcal{PRE}$ is type-N statistically κ-re-encryption replaceable for all $N \in \{1, 2\}$, and the same function $\mathsf{ReEncRep}$ *is used for both types, then we say it is* statistically κ-re-encryption replaceable. *If κ is polynomial as determined by the security parameter, then we say $\mathcal{PRE}$ is* statistically re-encryption replaceable.

In the special case that

$$\mathsf{ReEncRep}(\mathsf{pk}_i, \mathsf{pk}_j, m, c, r_c) = \mathsf{ReEnc}(\Delta_{i,j}, c),$$

we say that $\mathcal{PRE}$ is perfectly re-encryption replaceable.

We note that if $\mathsf{ReKeyGen}$ is deterministic then $\Delta_{i,j} = \Delta'_{i,j}$, in which case the adversary learns the token used to perform the re-encryption.

VI: 7.2.1 Observations on statistical re-encryption replaceability

We begin with some simple observations on statistical re-encryption replaceability, including how it relates to re-encryption simulatability and source-hiding.

Remark. For a PRE scheme to be type-1 statistically re-encryption replaceable, $\mathsf{ReEncRep}$ needs to output ciphertexts that successfully decrypt to m using sk_j. This means that type-1 re-encryption replaceability cannot easily be proven using reductions such as DDH or RLWE, as the output of $\mathsf{ReEncRep}$ cannot be replaced with a uniform random value.

Lemma VI.6. *If a PRE scheme $\mathcal{PRE}$ is source-hiding, then it is statistically $(\epsilon, 2)$-re-encryption replaceable where ϵ is negligible.*

Proof. Source-hiding implies that replacing re-encrypted ciphertexts with fresh encryptions at the appropriate level is indistinguishable even given $(\mathsf{sk}_i, \mathsf{sk}_j, \Delta_{i,j})$, defined over two key pairs. Clearly, indistinguishability holds given only a subset of these values, as is the case for type-1 and type-2 statistical $(\epsilon, 2)$-re-encryption replaceability where ϵ is negligible. □

Lemma VI.7. *Type-1 re-encryption replaceability does not imply re-encryption simulatability, and re-encryption simulatability does not imply type-1 re-encryption replaceability.*

Proof. This is because the inputs to $\mathsf{ReEncRep}$ and $\mathsf{ReEncSim}$ are different. $\mathsf{ReEncRep}$ assumes that r_c is known whereas $\mathsf{ReEncSim}$ does not, and similarly $\mathsf{ReEncSim}$ assumes that

sk_j is known whereas $\mathsf{ReEncRep}$ does not. If the only provable function for re-encryption simulatability depends on sk_j being known, then this function cannot be used to demonstrate re-encryption replaceability, and similarly for r_c and re-encryption replaceability. $\square$

Despite Lemma VI.7, we note that, like re-encryption simulatability, re-encryption replaceability can also be used to demonstrate that a PRE-IND-CPA-secure scheme with this property is also IND-HRA-secure. We do not prove this explicitly here as it will follow trivially from Theorem VI.13.

We now make some observations on what kind of schemes are trivially re-encryption replaceable, focusing on transparency. Transparency is a well-known concept but not formally defined in the literature. Recall from Definition III.6 that, informally, a PRE scheme has transparency if fresh ciphertexts have the same form as re-encrypted ciphertexts, and can be decrypted in the same way, and is called such because the existence of the proxy is essentially hidden. Definition VI.12 is our formalisation of how transparency is currently considered.

Definition VI.12. *Let $\mathcal{PRE}$ be a PRE scheme with correctness bound L. Let $c^{(0)} \xleftarrow{\$} \mathcal{PRE}.\mathsf{Enc}(\mathsf{pk}, m)$, and let $c^{(\ell)}$ be a ciphertext under* pk *with underlying message m that has been re-encrypted ℓ times. Then $\mathcal{PRE}$ has* (weak) transparency *if for all messages m, all key pairs $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathcal{PRE}.\mathsf{KeyGen}(1^\lambda)$, all fresh ciphertexts $c^{(0)} \xleftarrow{\$} \mathcal{PRE}.\mathsf{Enc}(\mathsf{pk}, m)$ and all re-encrypted ciphertexts $c^{(\ell)}$ under* pk *with underlying message m that have been re-encrypted ℓ times:*

$$(c^{(0)}, m, \mathsf{pk}) \approx_s (c^{(1)}, m, \mathsf{pk}) \approx_s \cdots \approx_s (c^{(L)}, m, \mathsf{pk}),$$

and for all $i \in \{0, \ldots, L\}$

$$m = \mathcal{PRE}.\mathsf{Dec}(\mathsf{sk}, c^{(i)}).$$

Definition VI.12 meets the intuition of transparency in the sense that decryption is the same for ciphertexts of all levels and that all ciphertexts c where $\mathsf{Dec}(\mathsf{sk}, c) \to m$ look the same to an adversary who does not know sk. This is in contrast to PRE schemes where re-encryption adds some additional components to the ciphertext, which then need to be processed upon decryption, or when decryption is defined differently for ciphertexts at different levels.

We note that an implication of weak transparency is that for all c such that $\mathsf{Dec}(\mathsf{sk}_i, c) \to m$, there exists an x such that $c = \mathsf{det\text{-}Enc}(\mathsf{pk}_i, m; x)$. We note that x is a function of the inputs used to create the ciphertext i.e. if c' is a re-encrypted ciphertext $c' \leftarrow \mathsf{det\text{-}ReEnc}(\Delta_{i,j}, c; r_{re})$ where $c \leftarrow \mathsf{det\text{-}Enc}(\mathsf{pk}_i, m; r_c)$, and $\Delta_{i,j} \leftarrow \mathsf{det\text{-}ReKeyGen}(\mathsf{sk}_i, pk_j; r_t)$, then x is derived from $(m, \mathsf{pk}_i, r_c, \mathsf{sk}_i, \mathsf{pk}_j, r_t, r_{re})$, or some subset of these values.

However, weak transparency does not imply that a party who knows the secret key cannot infer how many times the ciphertext has been re-encrypted. This is the case in lattice-based PRE schemes, where re-encryption adds further error to the ciphertext, and a party with the secret key can learn the size of the error and may therefore infer how many times the ciphertext was re-encrypted. Recall that in Section V: 5.2, we gave this as the reason why $\mathsf{pcBV\text{-}PRE}$ is not source-hiding.

We therefore also define strong transparency, where fresh ciphertexts (at level $\ell = 0$) under pk_j are identically distributed to re-encrypted ciphertexts under pk_j, even given sk_j and $\Delta_{i,j}$.

Definition VI.13. *Let $\mathcal{PRE}$ be a PRE scheme with correctness bound L. Let $c^{(0)} \xleftarrow{\$} \mathcal{PRE}.\mathsf{Enc}(\mathsf{pk}, m)$, and let $c^{(\ell)}$ be a ciphertext under pk with underlying message m that has been re-encrypted ℓ times. Then $\mathcal{PRE}$ has* strong transparency *if for all messages m, all key pairs $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathcal{PRE}.\mathsf{KeyGen}(1^\lambda)$, all fresh ciphertexts $c^{(0)} \xleftarrow{\$} \mathcal{PRE}.\mathsf{Enc}(\mathsf{pk}, m)$ and all re-encrypted ciphertexts $c^{(\ell)}$ under pk with underlying message m that have been re-encrypted ℓ times:*

$$(c^{(0)}, m, \mathsf{pk}, \mathsf{sk}) \approx_s (c^{(1)}, m, \mathsf{pk}, \mathsf{sk}) \approx_s \cdots \approx_s (c^{(L)}, m, \mathsf{pk}, \mathsf{sk}),$$

and for all $i \in \{0, \ldots, L\}$

$$m = \mathcal{PRE}.\mathsf{Dec}(\mathsf{sk}, c^{(i)}).$$

We now make a similar observation as we did for for weak transparency. Let $\mathcal{R}_c$ be the distribution from which Enc samples randomness, $\mathcal{R}_t$ be the distribution from which $\mathsf{ReKeyGen}$ samples randomness, and $\mathcal{R}_{re}$ be the distribution from which ReEnc samples randomness. Then if the PRE scheme has strong transparency, this implies that for all $(c, r_c) \overset{\$}{\leftarrow} \overline{\mathsf{Enc}}(\mathsf{pk}_i, m)$, $(\Delta_{i,j}, r_t) \overset{(\$)}{\leftarrow} \overline{\mathsf{ReKeyGen}}(\mathsf{pk}_i, \mathsf{sk}_j)$, and $(c', r_{re}) \overset{(\$)}{\leftarrow} \overline{\mathsf{ReEnc}}(\Delta_{i,j}, c)$, there exists a deterministic function $f : \mathcal{R}_c \times \mathcal{R}_t \times \mathcal{R}_{re} \to \mathcal{R}_c$ such that

$$c' = \mathsf{det\text{-}Enc}(\mathsf{pk}_j, m; f(r_c, r_t, r_{re})).$$

We can assume that $\mathcal{R}_c$, $\mathcal{R}_t$ and $\mathcal{R}_{\mathrm{re}}$ are publicly known, as they are included in the descriptions of the algorithms. We can further assume that f is publicly known, as it can be determined from re-encrypting using known key pairs.

For example, the original PRE scheme, BBS [BBS98] (Figure III.8), has strong transparency. Fresh ciphertexts have the form $(m \cdot g^{r_c}, g^{ar_c})$ for $(\mathsf{pk}_i, \mathsf{sk}_i) = (g^a, a)$ where r_c is the randomness sampled by Enc. The update tokens have the form $\Delta = b/a$, and re-encrypted ciphertexts are computed as $c' = (c_0, c_1^{\Delta}) = (m \cdot g^{r_c}, g^{br_c})$. We see that $c' = \mathsf{det\text{-}Enc}(\mathsf{pk}_j, m; r_c)$, meaning that $f(r_c, r_t, r_{re}) = f(r_c, \emptyset, \emptyset) = r_c$.

Similarly, the updatable encryption scheme, RISE [LT18] (see Appendix C: 1), also has strong transparency. Fresh ciphertexts have the form $(g^{xr_c}, g^{r_c} \cdot m)$ for $(\mathsf{pk}_i, \mathsf{sk}_i) = (g^x, x)$ where r_c is the randomness sampled by Enc. The update tokens have the form $\Delta = (x'/x, g^{x'})$, and re-encrypted ciphertexts are computed as $c' = (c_0^{\Delta_0} \cdot \Delta_1^{r_{re}}, c_1 \cdot g^{r_{re}}) = (g^{x'(r_c + r_{re})}, g^{r_c + r_{re}} \cdot m)$. We see that $c' = \mathsf{det\text{-}Enc}(g^{x'}, m; r_c + r_{re})$, meaning that $f(r_c, r_t, r_{re}) = f(r_c, \emptyset, r_{re}) := r_c + r_{re}$. Since addition is carried out modulo q, $f(r_c, r_t, r_{re})$ from the same distribution as r_c.

Clearly, strong transparency implies weak transparency. The main difference between weak and strong transparency is that for schemes with weak transparency, re-encrypted ciphertexts could still contain the secret key, whereas for schemes with strong transparency, re-encrypted ciphertexts are independent of any previous secret keys. Another difference is that with strong transparency, the randomness $f(r_c, r_t, r_{re})$ in a re-encrypted ciphertext comes from the same distribution $\mathcal{R}_c$ as a fresh ciphertext (at level $\ell = 0$). This is not the case for PRE schemes that have weak transparency but not strong transparency, particularly lattice-based constructions such as $\mathsf{BV\text{-}PRE}$ [PRSV17] and $\mathsf{pcBV\text{-}PRE}$ (Figure V.8) due to error growth.

Remark. At present it seems that error growth and therefore weak transparency is an inevitability of lattice-based PRE schemes. The problem of reducing error growth also exists for Fully-Homomorphic Encryption (FHE). As recent public-key unidirectional, multi-hop PRE schemes are based on lattices, we make the distinction between strong and weak transparency so as to clarify which is needed when proving certain properties. This contrasts some existing work such as [ABH09], which appears to assume strong transparency.

Lemma VI.8. *Let $\mathcal{PRE}$ be a PRE scheme with strong transparency where both* $\mathsf{ReKeyGen}$ *and* ReEnc *are deterministic. Then $\mathcal{PRE}$ is perfectly re-encryption replaceable.*

Proof. Let $\mathsf{ReEncRep}$ be as defined as:

```
$\underline{\mathsf{ReEncRep}(\mathsf{pk}_i, \mathsf{pk}_j, m, c, r_c) \xrightarrow{\$} c'^*}$
$r'_t \xleftarrow{\$} \mathcal{R}_t$
$r'_{re} \xleftarrow{\$} \mathcal{R}_{re}$
$c'^* \leftarrow \mathsf{det\text{-}Enc}(\mathsf{pk}_j, m; f(r_c, r'_t, r'_{re}))$
**return** $c'^*$
```

Because $\mathsf{ReKeyGen}$ and ReEnc are deterministic, we have that $r_t = r_{re} = r'_t = r'_{re} = \emptyset$. Therefore,

$$\begin{aligned}\mathsf{ReEncRep}(\mathsf{pk}_i, \mathsf{pk}_j, m, c, r_c) &= \mathsf{det\text{-}Enc}(\mathsf{pk}_j, m; f(r_c, r'_t, r'_{re}))\\ &= \mathsf{det\text{-}Enc}(\mathsf{pk}_j, m; f(r_c, r_t, r_{re}))\\ &= \mathsf{det\text{-}Enc}(\mathsf{pk}_j, m; f(r_c, \emptyset, \emptyset))\\ &= \mathsf{ReEnc}(\Delta_{i,j}, c),\end{aligned}$$

as required. □

The following theorem is a generalisation of the proof technique used in proof of Theorem V.6.

Theorem VI.8. *PRE schemes with strong transparency where the underlying encryption scheme is PKE-IND-CPA secure are statistically re-encryption replaceable.*

Proof. Let r_t denote the random values sampled from distribution $\mathcal{R}_t$ by $\mathsf{ReKeyGen}$ and let r_{re} denote the random values sampled from distribution $\mathcal{R}_{re}$ by ReEnc. Recall that strong transparency implies that there exists a publicly known function f such that re-encrypted ciphertexts can be computed using $\mathsf{det\text{-}Enc}$ and f and computing

$$c' = \mathsf{det\text{-}Enc}(\mathsf{pk}_j, m; f(r_c, r_t, r_{re})).$$

Then $\mathsf{ReEncRep}$ can be defined as in the proof of Lemma VI.8.

We now prove statistical re-encryption replaceability for $\mathsf{ReEncRep}$. For type-1 statistical indistinguishability, we need to show that for all $m \in \mathcal{M}$, and all $(c, r_c) \xleftarrow{\$} \overline{\mathsf{Enc}}(\mathsf{pk}_i, m)$:

$$\begin{aligned}&(\mathsf{det\text{-}Enc}(\mathsf{pk}_j, m; f(r_c, r'_t, r'_{re})), \mathsf{pk}_i, \mathsf{pk}_j, c, m, \mathsf{sk}_j)\\ \approx_s\ &(\mathsf{det\text{-}Enc}(\mathsf{pk}_j, m; f(r_c, r_t, r_{re})), \mathsf{pk}_i, \mathsf{pk}_j, c, m, \mathsf{sk}_j)\end{aligned}$$

Firstly, we note that both r_t and r_{re} are generated afresh when $\mathsf{ReKeyGen}$ and ReEnc are called and therefore only used in the specific creation of a genuine re-encryption. We further note that we can consider $\mathsf{ReEnc}(i, j, c)$ as a black box that generates $\Delta_{i,j}$ and re-encrypts c, returning c' without revealing any interim values. In other words, neither $\Delta_{i,j}$, r_t nor r_{re} are directly given to the distinguisher. The same applies for $\mathsf{ReEncRep}$, meaning that r'_t and r'_{re} are also unknown.

Because the underlying encryption scheme is PKE-IND-CPA secure, encryption does not reveal r_c – otherwise, an adversary could use r_c to compute $c = \mathsf{det\text{-}Enc}(\mathsf{pk}, m_0; r_c)$ and compare this with the challenge ciphertext to break PKE-IND-CPA security. We therefore conclude that no distinguisher can learn r_c from c. However, as sk_j is known, there exists some function g for which the distinguisher learns $g(f(r_c, r_t, r_{re}))$ from $c' \leftarrow \mathsf{det\text{-}Enc}(\mathsf{pk}_j, m; f(r_c, r_t, r_{re}))$ $\big($or $g(f(r_c, r'_t, r'_{re}))$ from $c' \leftarrow \mathsf{det\text{-}Enc}(\mathsf{pk}_j, m; f(r_c, r'_t, r'_{re}))\big)$. In other words, the secret key could allow the adversary to learn something about the randomness used for encryption. For example, in both BBS and RISE, the adversary learns g^{r_c} for $c = \mathsf{det\text{-}Enc}(\mathsf{pk}, m, r_c)$ but not r_c itself.

Clearly, as r_t and r'_t are from the same distribution, r_{re} and r'_{re} are also from the same distribution, and as $r_t, r'_t, r_{re}, r'_{re}$ are unknown then $f(r_c, r_t, r_{re})$ and $f(r_c, r'_t, r'_{re})$, and therefore $g(f(r_c, r_t, r_{re}))$ and $g(f(r_c, r'_t, r'_{re}))$, are indistinguishable. We therefore conclude that replaced re-encryptions will be indistinguishable from genuine ones, even given the current secret key.

Similarly, to show type-2 statistical re-encryption replaceability we need to show that

$$\begin{aligned}&(\mathsf{det\text{-}Enc}(\mathsf{pk}_j, m; f(r_c, r'_t, r'_{re})), \mathsf{pk}_i, \mathsf{pk}_j, c, m, \mathsf{sk}_i, \Delta'_{i,j})\\ \approx_s\ &(\mathsf{det\text{-}Enc}(\mathsf{pk}_j, m; f(r_c, r_t, r_{re})), \mathsf{pk}_i, \mathsf{pk}_j, c, m, \mathsf{sk}_i, \Delta'_{i,j})\end{aligned}$$

We note that because sk_j, is unknown, then $f(r_c, r_t, r_{re})$ $\big($or $f(r_c, r'_t, r'_{re})\big)$ is also unknown. The result follows. $\square$

VI: 7.2.2 Computational Re-Encryption Replaceability

We now define a computational notion of re-encryption replaceability, as this can be used more intuitively for our main results. The intuition is the same, but we shift to a game-based, multi-key, multi-challenge model. In $\mathsf{Replace}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ (Figure VI.10), the adversary must determine whether it is receiving genuine re-encryptions, or outputs of $\mathsf{ReEncRep}$. For each challenge query replacing a re-encryption from pk_i to pk_j, we allow the adversary to either learn some sk_i and/or $\Delta_{i,j}$, or only learn sk_j. This is enforced by recording the queries to $\mathsf{Chal}_{\mathsf{ReEncRep}}$, $\mathsf{O}_{\mathsf{ReKeyGen}}$ and $\mathsf{O}_{\mathsf{Corrupt}}$ in $\mathcal{R}_{\mathsf{Challenge}}$, $\mathcal{T}_{\mathsf{learned}}$ and $\mathcal{K}_{\mathsf{corrupt}}$ respectively, and using abort conditions in these oracles. We also impose the condition in $\mathsf{Chal}_{\mathsf{ReEncRep}}(i, j, c)$ that the adversary cannot have learned some $\Delta_{i,j}$ $((i,j) \notin \mathcal{T}_{\mathsf{learned}})$, as in this case a CPA adversary $\mathcal{A}^{\mathsf{CPA}}$ simulating the in the HRA experiment can compute a genuine re-encryption and does not need to make any replacements. The challenger also maintains a lookup table of underlying messages in *Message*[], and a lookup table of the random inputs used to create a re-encrypted ciphertext in *RandomInputs*[], which it uses to retrieve the appropriate inputs for $\mathsf{ReEncRep}$ when $b = 1$. In $\mathsf{Replace}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$, the $\boxed{\text{boxed constraint}}$ in $\mathsf{Chal}_{\mathsf{ReEncRep}}$ enforces the challenge graph observation depicted in Figure VI.7, the $\boxed{\text{boxed constraint}}$ in $\mathsf{O}_{\mathsf{ReKeyGen}}$ enforce the challenge graph observation depicted in Figure VI.8, and the $\boxed{\text{boxed constraint}}$ in $\mathsf{O}_{\mathsf{Corrupt}}$ enforces the challenge graph observation depicted in Figure VI.9.

Definition VI.14. *A PRE scheme $\mathcal{PRE}$ is* (ϵ, κ)-re-encryption replaceable *if for all* PPT *adversaries $\mathcal{A}$:*

$$\Pr\left[\mathsf{Replace}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon$$

where $\mathsf{Replace}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ *is given in Figure VI.10, and κ is the maximum number of keys*

```
$\mathsf{Replace}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$
$\mathcal{C}_{\mathsf{honest}}, \mathcal{K}_{\mathsf{corrupt}}, \mathcal{T}_{\mathsf{learned}}, \mathcal{R}_{\mathsf{chal}}, = \emptyset$
$b \xleftarrow{\$} \{0,1\}$
$b' \xleftarrow{(\$)} \mathcal{A}^{\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Corrupt}}, \mathsf{O}_{\mathsf{Enc}}, \mathsf{O}_{\mathsf{ReKeyGen}}, \mathsf{Chal}_{\mathsf{ReEncRep}}}(1^\lambda)$
    return $(b' = b)$
```

```
$\mathsf{O}_{\mathsf{KeyGen}}(1^\lambda) \xrightarrow{\$} \mathsf{pk}_\kappa$
$\kappa = \kappa + 1$
$(\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
return $\mathsf{pk}_\kappa$
```

```
$\mathsf{O}_{\mathsf{Corrupt}}(i) \rightarrow \mathsf{sk}_i$
$\boxed{\textbf{if } \exists j \text{ such that } ((j,i) \in \mathcal{R}_{\mathsf{chal}}) :}$
$\boxed{\quad \textbf{return } \perp}$
$\mathcal{K}_{\mathsf{corrupt}}.\mathsf{add}\{\mathsf{sk}_i\}$
return $\mathsf{sk}_i$
```

```
$\mathsf{O}_{\mathsf{Enc}}(i, m) \xrightarrow{\$} c$
$(c, r_c) \xleftarrow{\$} \overline{\mathsf{Enc}}(\mathsf{pk}_i, m)$
$\mathit{Message}[i, c] := m$
$\mathit{RandomInputs}[i, c] := r_c$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(i, c)\}$
return $c$
```

```
$\mathsf{O}_{\mathsf{ReKeyGen}}(i, j) \xrightarrow{(\$)} \Delta_{i,j}$
$\boxed{\textbf{if } ((i,j) \in \mathcal{R}_{\mathsf{chal}}) \wedge (\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}})}$
$\boxed{\quad \textbf{return } \perp}$
$\Delta_{i,j} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$\mathcal{T}_{\mathsf{learned}}.\mathsf{add}\{(i, j)\}$
return $\Delta_{i,j}$
```

```
$\mathsf{Chal}_{\mathsf{ReEncRep}}(i, j, c) \xrightarrow{(\$)} c'^*_b$
if $((i, c) \notin \mathcal{C}_{\mathsf{honest}}) \vee ((i, j) \in \mathcal{T}_{\mathsf{learned}})$ :
    return $\perp$
$\boxed{\textbf{if } (\mathsf{sk}_i \in \mathcal{K}_{\mathsf{corrupt}}) \vee (\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}})}$ :
$\boxed{\quad \textbf{return } \perp}$
$(\Delta_{i,j}, r_t) \xleftarrow{(\$)} \overline{\mathsf{ReKeyGen}}(\mathsf{sk}_i, \mathsf{pk}_j)$
$(c^*_0, r_{re}) \xleftarrow{(\$)} \overline{\mathsf{ReEnc}}(\Delta_{i,j}, c)$
$r_0 := (r_t, r_{re})$
$m \leftarrow \mathit{Message}[i, c]$
$r_c \leftarrow \mathit{RandomInputs}[i, c]$
$(c^*_1, r_1) \xleftarrow{(\$)} \overline{\mathsf{ReEncRep}}(\mathsf{pk}_i, \mathsf{pk}_j, m, c, r_c)$
$\mathit{Message}[j, c^*_b] := m$
$\mathit{RandomInputs}[j, c^*_b] := \mathit{RandomInputs}[i, c] \cup r_b$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(j, c^*_b)\}$
$\mathcal{R}_{\mathsf{chal}}.\mathsf{add}\{(i, j)\}$
return $c^*_b$
```

Figure VI.10: The Re-Encryption Replaceability game. The $\boxed{\text{boxed conditions}}$ enforce the challenge graph observation, which in this case means that for each challenge $c^*_b \xleftarrow{\$} \mathsf{Chal}_{\mathsf{ReEncRep}}(i, j, c)$ for some c, we only need to consider adversaries that know either sk_j or at most $(\mathsf{sk}_i, \Delta_{i,j})$.

generated in $\mathsf{Replace}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$*. If* ϵ *is negligible and* κ *is polynomial as characterised by* λ*, then we say* $\mathcal{PRE}$ *is* re-encryption replaceable.

VI: 7.2.3 Observations on Re-Encryption Replaceability

Here we prove a number of theorems that demonstrate properties that imply re-encryption replaceability.

Theorem VI.9. *If a PRE scheme* $\mathcal{PRE}$ *is perfectly re-encryption replaceable, then it is re-encryption replaceable (for all* κ*).*

The proof of this is trivial, as perfect re-encryption replaceability implies there exists $\mathsf{ReEncRep}$ such that $\mathsf{ReEncRep}(\mathsf{pk}_i, \mathsf{pk}_j, m, c, r_c) = \mathsf{ReEnc}(\Delta_{i,j}, c)$.

We now demonstrate that in a number of cases, it is sufficient to prove that the PRE scheme is secure as defined by the following variant of re-encryption replaceability.

Definition VI.15. *Consider a variant of* $\mathsf{Replace}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ *where* $\mathsf{Chal}_{\mathsf{ReEncRep}}$ *only takes fresh ciphertexts (at level 0) as input. For a given PRE scheme* $\mathcal{PRE}$*, if there is no* PPT *adversary that wins this variant with advantage greater than* ϵ *when a maximum of* κ *key pairs are generated, then we say that* $\mathcal{PRE}$ *is* level-0 (ϵ, κ)-re-encryption replaceable.

Theorem VI.10. *If a PRE scheme,* $\mathcal{PRE}$*, is statistically* κ*-re-encryption replaceable (Definition VI.11), then* $\mathcal{PRE}$ *is also level-0* κ*-re-encryption replaceable.*

Proof. Firstly, we note that because statistical re-encryption replaceability is defined over all κ keys, there is no advantage in choosing which keys a challenge is called on. Secondly, because statistical re-encryption replaceability is defined for all fresh ciphertexts, we should be able to replace all challenges in $\mathsf{Replace}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$.

We now look at game-based re-encryption replaceability. For all $(i,j) \in \mathcal{R}_{\mathsf{Challenge}}$, by the challenge graph observation, at the end of the game, either

- $\mathsf{sk}_j \in \mathcal{K}_{\mathsf{corrupt}}$, and $\mathsf{sk}_i \notin \mathcal{K}_{\mathsf{corrupt}}$ and $(i,j) \notin \mathcal{T}_{\mathsf{learned}}$; or
- $\mathsf{sk}_i \in \mathcal{K}_{\mathsf{corrupt}}$ and/or $(i,j) \in \mathcal{T}_{\mathsf{learned}}$, and $\mathsf{sk}_j \notin \mathcal{K}_{\mathsf{corrupt}}$; or
- $\mathsf{sk}_i, \mathsf{sk}_j \notin \mathcal{K}_{\mathsf{corrupt}}$, and $(i,j) \notin \mathcal{T}_{\mathsf{learned}}$.

This means that in the first case, the adversary's information is limited to aux_1. Since statistical indistinguishability is stronger than computational indistinguishability, statistical type-1 re-encryption replaceability implies no adversary can distinguish a challenge where sk_j is later corrupted. Similarly, in the second case the adversary's information is limited to at most aux_2, meaning that by statistical type-2 re-encryption replaceability no adversary can distinguish a challenge when it later learns $(\mathsf{sk}_i, \Delta_{i,j})$. We note that technically the adversary could either corrupt only sk_i *or* $\Delta_{i,j}$, but this trivially follows from statistical type-2 re-encryption replaceability as the adversary has less information with which to distinguish the challenge.

The remaining difference of note is that in Definition VI.14, the adversary can adaptively decide which set of auxiliary information it receives after receiving the challenge. However, as statistical re-encryption replaceability implies that challenges are statistically indistinguishable in either case, there is no advantage in the adversary being able to decide which auxiliary information it receives. □

We now demonstrate how PRE schemes with weak transparency that are level-0 re-encryption replaceable are re-encryption replaceable (for all levels).

Theorem VI.11. *Let $\mathcal{PRE}$ be a PRE scheme with weak transparency. Then if $\mathcal{PRE}$ is level-0 (ϵ, κ)-re-encryption replaceable, it is also (ϵ, κ)-re-encryption replaceable.*

Proof. Let ℓ be the maximum level for a ciphertext c for which $\mathsf{O}_{\mathsf{ReEncRep}}$ accepts c as input. We proceed by induction on ℓ. We show via an iterative argument that if there exists a function $\mathsf{ReEncRep}$ that can indistinguishably replace re-encryptions of fresh ciphertexts, then it can also be used to replace re-encryptions of re-encrypted ciphertexts.

$\underline{\ell = 0}$: This is the case where all ciphertext inputs to $\mathsf{Chal}_{\mathsf{ReEncRep}}$ are outputs of $\mathsf{O}_{\mathsf{Enc}}$ and therefore at level 0. This clearly follows from $\mathcal{PRE}$ being level-0 re-encryption replaceable.

For $(c, r_c) \overset{\$}{\leftarrow} \overline{\mathsf{Enc}}(\mathsf{pk}_i, m)$, $(\Delta_{i,j}, r_t) \overset{(\$)}{\leftarrow} \overline{\mathsf{ReKeyGen}}(\mathsf{sk}_i, \mathsf{pk}_j)$, let $(c'_0, r_{re}) \overset{(\$)}{\leftarrow} \overline{\mathsf{ReEnc}}(\Delta_{i,j}, c)$ and $c'_1 \overset{(\$)}{\leftarrow} \mathsf{ReEncRep}(\mathsf{pk}_i, \mathsf{pk}_j, m, c, r_c)$. Because c'_0 and c'_1 are computationally indistinguishable even when sk_j is known, we have that $\mathsf{Dec}(\mathsf{sk}_j, c'_0) = \mathsf{Dec}(\mathsf{sk}_j, c'_1) = m$. Because $\mathcal{PRE}$ has weak transparency, it follows that for $b \in \{0, 1\}$, there exists x_b such that

$$c'_b = \mathsf{det\text{-}Enc}(\mathsf{pk}_j, m; x_b),$$

where x_0 is derived from (some subset of) $(\mathsf{pk}_i, m, r_c, \mathsf{sk}_i, \mathsf{pk}_j, r_t, r_{re})$ and x_1 is derived from (some subset of) $(\mathsf{pk}_i, m, r_c, \mathsf{pk}_j)$.

Let g be the function such that an adversary decrypting c where $c \overset{\$}{\leftarrow} \mathsf{det\text{-}Enc}(\mathsf{pk}, m; x)$ learns $g(x)$. In other words, g describes what a party with the decryption key can learn of the randomness used in the encryption. We have that $g(x_0)$ and $g(x_1)$ must be computationally indistinguishable.

Therefore, the existence of $\mathsf{ReEncRep}$ implies that there exists a function f such that $f(\mathsf{pk}_i, \mathsf{pk}_j, m, c, r_c) \overset{(\$)}{\rightarrow} x_1$ where $g(x_1)$ is computationally indistinguishable from $g(x_0)$. We conclude that an acceptable choice for $\mathsf{ReEncRep}$ is

```
$\mathsf{ReEncRep}(\mathsf{pk}_i, \mathsf{pk}_j, m, c, r_c) \overset{(\$)}{\rightarrow} \Delta_{i,j}$
$x_1 \overset{(\$)}{\leftarrow} f(\mathsf{pk}_i, \mathsf{pk}_j, m, c, r_c)$
$c'_1 \leftarrow \mathsf{det\text{-}Enc}(\mathsf{pk}_j, m; x_1)$
**return** $c'_1$
```

We will use this going forward.

$\underline{\ell = n}$: We assume this is true.

$\underline{\ell = n + 1}$: Because the theorem holds for $\ell \leq n$, we have that for all input ciphertexts c that have been re-encrypted at most n times, there exists r_c such that $c = \mathsf{det\text{-}Enc}(\mathsf{pk}_i, m; r_c)$. We further note that as c has been created by oracles, we can use $r \leftarrow \mathit{RandomInputs}[i, c]$ to derive r_c.

Recall that weak transparency implies that for $(c'_0, r_{re}) \overset{(\$)}{\leftarrow} \overline{\mathsf{ReEnc}}(\Delta_{i,j}, c)$ where $(\Delta_{i,j}, r_t) \overset{(\$)}{\leftarrow} \overline{\mathsf{ReKeyGen}}(\mathsf{sk}_i, \mathsf{pk}_j)$, there exists an x_0 such that $c'_0 = \mathsf{det\text{-}Enc}(\mathsf{pk}_j, m; x_0)$, and that x_0 is derived from (some subset of) $(\mathsf{pk}_i, m, r, \mathsf{sk}_i, \mathsf{pk}_j, r_t, r_{re})$. As for the case where $\ell = 0$, the same function f can be used to derive $f(\mathsf{pk}_i, \mathsf{pk}_j, m, c, r_c) \overset{(\$)}{\rightarrow} x_1$ such that $g(x_1)$ and $g(x_0)$ are computationally indistinguishable. We conclude that $\mathsf{ReEncRep}$ yields the result. □

Combining Theorems VI.10 and VI.11 we get the following:

Theorem VI.12. *If $\mathcal{PRE}$ is a PRE scheme with weak transparency and $\mathcal{PRE}$ is statistically κ-re-encryption replaceable, then $\mathcal{PRE}$ is also κ-re-encryption replaceable.*

In summary, to prove that a PRE scheme $\mathcal{PRE}$ has re-encryption replaceability, it suffices to demonstrate that one of the following is true:

- $\mathcal{PRE}$ has perfect re-encryption replaceability (Theorem VI.9).

- $\mathcal{PRE}$ has both weak transparency and statistical re-encryption replaceability (Theorem VI.12).

Furthermore, as the existing schemes that are source-hiding (see Section V: 5.2) also have weak transparency, Lemma VI.6 gives us reason to believe that these schemes are also re-encryption replaceable.

VI: 7.3 Proving adaptive HRA using Re-Encryption Replaceability

We now demonstrate our main result – that if a PRE scheme, $\mathcal{PRE}$, is adaptively PRE-IND-CPA-secure and re-encryption replaceable, then it is also adaptively IND-HRA-secure.

Theorem VI.13. *If $\mathcal{PRE}$ is (ϵ_1, κ)-adaptively PRE-IND-CPA-secure and (ϵ_2, κ)-re-encryption replaceable, then it is $(\epsilon_1 + \epsilon_2, \kappa)$-adaptively IND-HRA-secure.*

The following proof follows the same idea as Cohen's proof outline, except that instead of using $\mathsf{ReEncSim}$ to replace re-encryptions of non-challenge ciphertexts, we use $\mathsf{ReEncRep}$.

Proof. This proof follows the same structure as the proofs of Theorems VI.3 to VI.5. Suppose there exists an adversary $\mathcal{A}^{\mathsf{HRA}}$ that wins $\mathsf{a\text{-}IND\text{-}HRA}_{\kappa}^{\mathcal{PRE}}(1^\lambda)$ with advantage ϵ. If an adversary $\mathcal{A}$ in $\mathsf{a\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ can simulate $\mathsf{a\text{-}IND\text{-}HRA}_{\kappa}^{\mathcal{PRE}}(1^\lambda)$, then they can call $\mathcal{A}^{\mathsf{HRA}}$ as a subroutine and output the same guess to gain the same advantage.

Let $\mathsf{Game}_0 = \mathsf{a\text{-}IND\text{-}HRA}_{\kappa}^{\mathcal{PRE}}(1^\lambda)$, and let Game_1 be a variant of $\mathsf{a\text{-}IND\text{-}HRA}_{\kappa}^{\mathcal{PRE}}(1^\lambda)$ where re-encryptions of non-challenge ciphertexts when no update token has been learned and the source key is not corrupted are replaced by outputs of $\mathsf{ReEncRep}$. We note that $\mathcal{A}$ can simulate Game_1 by forwarding calls to $\mathsf{O}_{\mathsf{KeyGen}}$, $\mathsf{O}_{\mathsf{Corrupt}}$ or $\mathsf{O}_{\mathsf{ReKeyGen}}(i, j)$ to the CPA challenger, and can simulate calls to $\mathsf{O}_{\mathsf{Enc}}$ by encrypting locally and updating $\mathcal{C}_{\mathsf{honest}}$. Recall that $\mathcal{A}$ cannot forward calls to $\mathsf{O}_{\mathsf{ReEnc}}(i, j, \Delta_{i,j}, c)$ when c is a non-challenge ciphertext. Instead, for each honestly-generated ciphertext and key tuple (i, c), $\mathcal{A}$ maintains a list of the random inputs used in the creation of c and the underlying message, as described in $\mathsf{Replace}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$, and uses these as inputs to $\mathsf{ReEncRep}$ to simulate re-encryptions of non-challenge ciphertexts.

We now show that no PPT distinguisher $\mathcal{D}$ can distinguish between Game_0 and Game_1 with advantage greater than ϵ_2. Let $\mathcal{B}$ be an adversary in $\mathsf{Replace}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ with challenge bit $d \in \{0,1\}$. $\mathcal{B}$ can perfectly simulate Game_d as follows:

- $\mathcal{B}$ first samples $b \xleftarrow{\$} \{0,1\}$ to be the challenge bit in Game_d.
- When $\mathcal{D}$ calls $\mathsf{O}_{\mathsf{KeyGen}}$, $\mathsf{O}_{\mathsf{Corrupt}}$, $\mathsf{O}_{\mathsf{Enc}}$ or $\mathsf{O}_{\mathsf{ReKeyGen}}(i,j)$, $\mathcal{B}$ forwards these to the challenger in $\mathsf{Replace}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$.
- When $\mathcal{D}$ calls $\mathsf{O}_{\mathsf{ReEnc}}(i,j,\Delta_{i,j},c)$, $\mathcal{B}$ first performs the checks described in $\mathsf{IND\text{-}HRA}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ and if they pass, proceeds as follows:
  - If $(i,c) \in \mathcal{C}_{\mathsf{chal}}$ and $(i,j,\Delta_{i,j}) \in \mathcal{T}_{\mathsf{honest}}$, then $\mathcal{B}$ returns $c' \xleftarrow{(\$)} \mathsf{ReEnc}(\Delta_{i,j},c)$ after updating the appropriate lists.
  - If $(i,c) \in \mathcal{C}_{\mathsf{chal}}$ and $(i,j,\Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}$, then $\mathcal{B}$ obtains $\Delta_{i,j} \xleftarrow{(\$)} \mathsf{O}_{\mathsf{ReKeyGen}}(i,j)$ then returns $c' \xleftarrow{(\$)} \mathsf{ReEnc}(\Delta_{i,j},c)$ after updating the appropriate lists.
  - If $(i,c) \notin \mathcal{C}_{\mathsf{chal}}$ and $\mathsf{sk}_i \in \mathcal{K}_{\mathsf{corrupt}}$, $\mathcal{B}$ computes $\Delta_{i,j} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\mathsf{sk}_i,\mathsf{pk}_j)$ and returns $c' \xleftarrow{(\$)} \mathsf{ReEnc}(\Delta_{i,j},c)$ after updating the appropriate lists.
  - If $(i,c) \notin \mathcal{C}_{\mathsf{chal}}$ and $\mathsf{sk}_i \notin \mathcal{K}_{\mathsf{corrupt}}$ then $\mathcal{B}$ calls the challenge oracle in $\mathsf{Chal}_{\mathsf{ReEncRep}}(i,j,c) \xrightarrow{(\$)} c_d^*$, updates the appropriate lists and returns c'.
- When $\mathcal{D}$ calls $\mathsf{Chal}_{\mathsf{Enc}}(m_0,m_1)$, $\mathcal{B}$ performs the appropriate checks, and if the checks pass computes $c_0 \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i,m_0)$ and $c_1 \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i,m_1)$ and returns c_b, updating the lists as appropriate.

We note that the challenge graph observation implies that for each pair of keys $\mathsf{pk}_i,\mathsf{pk}_j$, we only need to consider the case when $\mathcal{D}$ does not query both $\mathsf{O}_{\mathsf{Corrupt}}(j)$ and either $\mathsf{O}_{\mathsf{ReKeyGen}}(i,j)$ or $\mathsf{O}_{\mathsf{ReEnc}}(i,j,\Delta_{i,j},c)$ for some $(i,j,\Delta_{i,j},c)$, meaning that the conditions in $\mathsf{Replace}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ do not restrict the calls that $\mathcal{D}$ makes to $\mathsf{O}_{\mathsf{Corrupt}}$. We see that if $d=0$, non-challenge ciphertexts are genuine re-encryptions and thus this simulates Game_0 perfectly, and if $d=1$, non-challenge re-encryptions are replaced and thus simulates Game_1 perfectly. As $\mathcal{PRE}$ is (ϵ_2,κ)-re-encryption replaceable, the advantage that $\mathcal{D}$ has in distinguishing between the two is bounded by ϵ_2. We conclude that the advantage ϵ of any PPT adversary winning $\mathsf{a\text{-}IND\text{-}HRA}_{\kappa}^{\mathcal{PRE}}(1^\lambda)$ is bounded by $\epsilon \leq \epsilon_1 + \epsilon_2$, as required. □

VI: 7.4 Proving adaptive PCS using Re-Encryption Replaceability

We can prove that a PRE scheme has adaptive Post-Compromise Security (adaptive PCS) using the same techniques as in Sections VI: 6 and VI: 7. We do not define adaptive PCS explicitly here as it is reasonably intuitive, but provide a full description in Appendix E.

Theorem VI.14. *Let $\mathcal{PRE}$ be a PRE scheme where for all* PPT *adversaries $\mathcal{A}$:*

$$\Pr\left[\mathsf{CPA\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon_1,$$

where $\mathsf{CPA\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1$ *is the variant of the selective PCS game that forbids any re-encryptions from uncorrupted to corrupted keys, which generates a maximum of κ keys. Then if $\mathcal{PRE}$ is also (ϵ_2, κ)-TSP and (ϵ_3, κ)-re-encryption simulatability, then $\mathcal{PRE}$ has $(\epsilon_1 + \epsilon_2 + \epsilon_3, \kappa)$-adaptive PCS.*

Proof. Similarly to proof of Theorem VI.13, we Theorem VI.14 by using a CPA-variant of PCS.

Lemma VI.9. *Let $\mathcal{PRE}$ be a PRE scheme where for all* PPT *adversaries $\mathcal{A}$,* $\Pr\left[\mathsf{CPA\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon_1$, *where* $\mathsf{CPA\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1$ *is the variant of the selective PCS game that forbids any re-encryptions from uncorrupted to corrupted keys, which generates a maximum of κ keys. Then if $\mathcal{PRE}$ is also (ϵ_2, κ)-TSP, then*

$$\Pr\left[\mathsf{a\text{-}CPA\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + (\epsilon_1 + \epsilon_2),$$

where $\mathsf{a\text{-}CPA\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ *is the adaptive variant of* $\mathsf{CPA\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ *which also generates a maximum of κ keys.*

Proof of Lemma VI.9 is analogous to proof of Theorem VI.7, leveraging the fact that adaptive TSP is implied by TSP (Theorem VI.2).

We can expand into the full adaptive PCS game using re-encryption replaceability.

Lemma VI.10. *Let $\mathcal{PRE}$ be a PRE scheme where for all* PPT *adversaries $\mathcal{A}$* $\Pr\left[\mathsf{a\text{-}CPA\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon_1$, *where at most κ key pairs are generated in* $\mathsf{a\text{-}CPA\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$. *If $\mathcal{PRE}$ is also (ϵ_2, κ)-re-encryption replaceable, then:*

$$\Pr\left[\mathsf{a\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon_1 + \epsilon_2,$$

if at most κ key pairs are generated in $\mathsf{a\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$. *In other words, $\mathcal{PRE}$ has $(\epsilon_1 + \epsilon_2, \kappa)$-adaptive PCS.*

Proof of Lemma VI.10 is analogous to proof of Theorem VI.13. □

VI: 8 Adaptive security of pcBV-PRE

We now use the results demonstrated in this chapter to prove that pcBV-PRE is both adaptively IND-HRA-secure and adaptively Post-Compromise secure.

VI: 8.1 Adaptive PRE-IND-CPA security

Theorem VI.15. *For all* PPT *adversaries $\mathcal{A}$, there exists an RLWE distinguisher $\mathcal{D}$ such that:*

$$\Pr\left[\mathsf{a\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \\ 2(\kappa + (Q_{RK} + Q_{RE\perp})(\lfloor \log_2(q)/r \rfloor + 1) + 1) \cdot \mathbf{Adv}_{\mathcal{D}}^{RLWE_{\phi,q,\chi_e}}(1^\lambda),$$

where κ is the number of key pairs, Q_{RK} is the number of calls to $\mathsf{O}_{\mathsf{ReKeyGen}}$ and $Q_{RE\perp}$ is the number of calls to $\mathsf{O}_{\mathsf{ReEnc}}$ where $\Delta_{i,j} = \perp$.
Because $\mathcal{A}$ is PPT*, κ, Q_{RK} and $Q_{RE\perp}$ must also be polynomial, meaning the overall advantage of winning* $\mathsf{a\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda)$ *is negligible, by the RLWE assumption. Therefore,* pcBV-PRE *is adaptively PRE-IND-CPA-secure.*

Proof. This follows from Theorem VI.6, as we can demonstrate adaptive PRE-IND-CPA security via PKE-IND-CPA-security and TSP.

Lemma VI.11. pcBV-PRE *is $((\kappa + 2) \cdot \epsilon_{\mathsf{RLWE}}, \kappa)$-PKE-IND-CPA-secure, where ϵ_{RLWE} is the maximal advantage of any distinguisher in $RLWE_{\phi,q,\chi_e}$.*

Proof. We note that we can also express pcBV-PRE.Enc using ReSample:

```
pcBV-PRE.Enc(pk, $m$)
---
$(\bar{a}, \bar{b}) \xleftarrow{\$} \mathsf{ReSample}(\mathsf{pk})$
$c_0 = \bar{b} + m$
$c_1 = \bar{a}$
return $c = (c_0, c_1)$
```

Therefore, we can use $\mathsf{ReSample\text{-}RLWE}_{\mathcal{A}}(1^\lambda)$, calling $\mathsf{Chal}_{\mathsf{RRLWE}}$ once when simulating $\mathsf{Chal}_{\mathsf{Enc}}^{\mathsf{CPA}}$ to demonstrate that the challenge ciphertext is indistinguishable from random, meaning that m_b is hidden. Using Lemma V.14 gives the result. □

Lemma VI.12. pcBV-PRE *has* $((\kappa + 2Q_{RK}(\lfloor \log_2(q)/r \rfloor + 1)) \cdot \epsilon_{\mathsf{RLWE}}, \kappa)$*-TSP, where* Q_{RK} *is the number of queries to* $\mathsf{Chal}^{\mathsf{TSP}}_{\mathsf{ReKeyGen}}$*.*

Proof. Again, this can be shown using $\mathsf{ReSample\text{-}RLWE}_{\mathcal{A}}(1^\lambda)$, by calling $\mathsf{Chal}_{\mathsf{RRLWE}}$ in place of ReSample for each simulation of $\mathsf{Chal}^{\mathsf{TSP}}_{\mathsf{ReKeyGen}}(i, j)$ (where sk_j is uncorrupted). This means we call $\mathsf{Chal}_{\mathsf{RRLWE}}$ at most $(Q_{RK} + Q_{RE\perp})(\lfloor \log_2(q)/r \rfloor + 1)$ times. □

Recall from Theorem VI.6 that we can now simulate $\mathsf{a\text{-}PRE\text{-}IND\text{-}CPA}_{\mathcal{A}}(1^\lambda)$ by simulating update tokens to uncorrupted keys both for $\mathsf{O}_{\mathsf{ReKeyGen}}$ and for $\mathsf{O}^{\mathsf{CPA}}_{\mathsf{ReEnc}}$. This uses at most $Q_{RK} + Q_{RE\perp}$ simulated update tokens in total. Therefore, by Theorem VI.6, conclude that pcBV-PRE is $(2(\kappa + (Q_{RK} + Q_{RE\perp})(\lfloor \log_2(q)/r \rfloor + 1) + 1) \cdot \epsilon_{\mathsf{RLWE}}, \kappa)$-adaptively PRE-IND-CPA-secure, as required. □

VI: 8.2 Adaptive IND-HRA security

We begin by demonstrating that pcBV-PRE is re-encryption replaceable. We derive the following result from proof of Theorem V.6.

Theorem VI.16. *For all* PPT *adversaries* $\mathcal{A}$*, there exists an RLWE distinguisher* $\mathcal{D}$ *such that:*

$$\Pr\left[\mathsf{Replace}^{\mathsf{pcBV\text{-}PRE}}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \frac{1}{2} + (\kappa + 2Q_H) \cdot \mathbf{Adv}^{RLWE_{\phi,q,\chi_e}}_{\mathcal{D}}(1^\lambda),$$

where κ *is the number of keys, and* Q_H *is the number of ciphertexts created during* $\mathsf{Replace}^{\mathsf{pcBV\text{-}PRE}}_{\mathcal{A}}(1^\lambda)$*.*

Because $\mathcal{A}$ *is* PPT*,* κ *and* Q_H *must also be polynomial, meaning the overall advantage of winning* $\mathsf{Replace}^{\mathsf{pcBV\text{-}PRE}}_{\mathcal{A}}(1^\lambda)$ *is negligible, by the RLWE assumption. Therefore,* pcBV-PRE *has re-encryption replaceability.*

Proof sketch. We use the function used to replace honest re-encryptions in the proof of Theorem V.6. Specifically:

```
pcBV-PRE.ReEncRep$(\mathsf{pk}_i, \mathsf{pk}_j, m, c, (\tilde{v}, \tilde{e_0}, \tilde{e_1}))$

if no replacement for $\mathsf{sk}_i$ has yet been sampled:
    $s_i{}', e_i{}' \xleftarrow{\$} \chi_e$
    $(a_i, b_i) \leftarrow \mathsf{pk}_i$
    $\mathsf{pk}_i' = (a_i, a_i \cdot s_i{}' + p e_i{}')$
$\bar{c} \leftarrow$ det-pcBV-PRE.Enc$(\mathsf{pk}_i', m; (\tilde{v}, \tilde{e_0}, \tilde{e_1}))$
$\Delta_{i',j} \xleftarrow{\$}$ pcBV-PRE.ReKeyGen$(s_i{}', \mathsf{pk}_j)$
$c'^{*} \xleftarrow{\$}$ pcBV-PRE.ReEnc$(\Delta_{i',j}, \bar{c})$
return $c'^{*}$
```

For a challenge $c_b' \overset{(\$)}{\rightarrow} \mathsf{Chal}_{\mathsf{ReEncRep}}(i, j, c)$, the adversary can either later corrupt sk_j, or it can learn sk_i and/or some $\Delta_{i,j}$. In the former case, indistinguishability is demonstrated as described in proof of Theorem V.6, which relies on applying on the indistinguishability of the resampled pk_i used in the creation of the input ciphertext, c. As c could either be a fresh encryption or a re-encryption (or replacement), this would call the $\mathsf{Chal}_{\mathsf{RRLWE}}$ at most Q_H times. In the latter case, as the target public key pk_j will not be corrupted, we can use the indistinguishability of the resample used in pcBV-PRE.ReEnc (or pcBV-PRE.ReEncRep). Since this is an honest re-encryption in the game, it is already counted in Q_H, so this incurs no further security loss. Using Lemma V.14 gives the result.

Theorem VI.17. *For all* PPT *adversaries $\mathcal{A}$, there exists an RLWE distinguisher $\mathcal{D}$ such that:*

$$\Pr\left[\mathsf{a\text{-}IND\text{-}HRA}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \\ (3\kappa + 2(Q_{RK} + Q_{RE\perp}^{\mathsf{chal}})(\lfloor \log_2(q)/r \rfloor + 1) + 2Q_H + 2) \cdot \mathbf{Adv}_{\mathcal{D}}^{RLWE_{\phi,q,\chi_e}}(1^\lambda),$$

where κ is the number of keys, Q_{RK} is the number of calls $\mathcal{A}$ makes to $\mathsf{O}_{\mathsf{ReKeyGen}}$, $Q_{RE\perp}^{\mathsf{chal}}$ is the number of calls to $\mathsf{O}_{\mathsf{ReEnc}}$ for a challenge ciphertext when $\Delta_{i,j} = \perp$, and Q_H is the number of non-challenge ciphertexts created during $\mathsf{IND\text{-}HRA}_{\mathcal{A}}^{\mathsf{pcBV\text{-}PRE}}(1^\lambda)$.
Because $\mathcal{A}$ is PPT*, we note that $\kappa, Q_{RK}, Q_{RE\perp}^{\mathsf{chal}}$ and Q_H are also all polynomial, meaning that the overall advantage is negligible, by the RLWE assumption. Therefore,* pcBV-PRE *is adaptively IND-HRA-secure.*

Proof. This follows from Theorems VI.13, VI.15 and VI.16. We note we only need to use ReEncRep to replace the re-encryptions of non-challenge ciphertexts when c is an honest, non-challenge ciphertext and sk_i is unknown. We therefore only need to use $\mathsf{Chal}^{\mathsf{Replace}}_{\mathsf{ReEnc}}$ less than Q_H times. Using Theorem VI.16 gives the result. □

VI: 8.3 Adaptive PCS

Theorem VI.18. *For all* PPT *adversaries $\mathcal{A}$, there exists an RLWE distinguisher $\mathcal{D}$ such that:*

$$\Pr\left[\mathsf{a\text{-}PostComp}^{\mathsf{pcBV\text{-}PRE}}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \\ (3\kappa + 2(Q_{RK} + Q^{\mathsf{chal}}_{RE\perp})(\lfloor \log_2(q)/r \rfloor + 1) + 2Q_H + 2) \cdot \mathbf{Adv}^{RLWE_{\phi,q,\chi_e}}_{\mathcal{D}}(1^\lambda),$$

where κ is the number of calls $\mathcal{A}$ makes to $\mathsf{O}_{\mathsf{ReKeyGen}}$, $Q^{\mathsf{chal}}_{RE\perp}$ is the number of calls to $\mathsf{O}_{\mathsf{ReEnc}}$ for a challenge ciphertext when $\Delta_{i,j} = \perp$, and Q_H is the number of non-challenge ciphertexts created during $\mathsf{IND\text{-}HRA}^{\mathsf{pcBV\text{-}PRE}}_{\mathcal{A}}(1^\lambda)$.
Because $\mathcal{A}$ is PPT*, we note that κ, Q_{RK} and Q_H are also all polynomial, meaning that the overall advantage is negligible, by the RLWE assumption. Therefore,* pcBV-PRE *is adaptive PCS.*

Theorem VI.18 can be proven analogously to Theorem VI.17. More specifically, we can use Theorem VI.14 together with Theorems V.5 and VI.16 and lemma VI.11 to demonstrate that pcBV-PRE is adaptive PCS.

VI: 9 Summary

In this chapter, we related the fixed, selective, and adaptive models of key corruption in PRE. We discussed a discrepancy in the literature in terms of how selective security is defined, and presented arguments as to why the two variants can be considered equivalent. We then showed how adaptive security can be proven for PRE schemes by generalising many of the techniques used in Chapter V, identifying security properties which imply that a PRE scheme with these properties that is secure in the selective model is also secure in the adaptive model. In doing so, we demonstrated that pcBV-PRE is both adaptively IND-HRA-secure and has adaptive PCS. Our results achieve a significantly smaller security loss than existing work, and apply to all re-encryption graphs, and to the traditional security model where the adversary adaptively determines the re-encryption graph through its queries to $\mathsf{O}_{\mathsf{ReKeyGen}}$ and $\mathsf{O}_{\mathsf{ReEnc}}$. We also introduced re-encryption replaceability an alternative to source-hiding, which can prove HRA security from CPA security. This is preferable to source-hiding at is does not depend on inefficient methods and is met by a larger number of PRE schemes.

Chapter VII

Concluding remarks

We summarise our contributions and present ideas for future work.

In this thesis, we have examined applications of PRE and presented security definitions that reflect those applications. In doing so, we have explored recent trends in re-encryption primitives and explored the new possibilities presented by lattice-based constructions.

In Chapter IV we scrutinised the client's ability to control re-encryptions. We extended ciphertext dependence to token robustness, which implies that a ciphertext cannot be re-encrypted to new keys without knowledge of the appropriate secret key. We defined unidirectionality in this context and discussed reversal attacks, where an adversary can retain some information about a previous ciphertext, and use this to try to reverse the re-encryption of a ciphertext. We also introduced Ciphertext Origin Authentication – a mechanism where it is possible to identify the party responsible for the ciphertext being under its current key. Interesting future work in this area would be to create a PRE scheme based on lattices that has COA, and to investigate the possible advantages that such schemes offer over the ElGamal-based scheme presented in Chapter IV.

In Chapter V we motivated the need to define Post-Compromise Security to identify whether a PRE scheme is suitable for enforcing access revocation and key rotation, as well as to extend the PCS of messages in transit [CCG16] to backups of messaging history stored in the cloud. We then formally defined PCS for PRE. Our definition is strong in the sense that only the current secret key must be kept secret to ensure that it is not possible to tell which ciphertext has been re-encrypted. We note that this is stronger than the reversal attacks considered in Chapter IV. We then discussed how PCS can be proven using a number of existing security properties, before giving an explicit PRE scheme, pcBV-PRE, that meets this definition, based on BV-PRE– a practical PRE scheme originally presented in [PRSV17]. A particular advantage of our scheme is that it does not rely on sub-optimal parameter choices to prove PCS. A useful extension to this work would be to construct alternative PRE schemes with PCS that permit an arbitrary number of re-encryptions.

In Chapter VI we explored stronger adversaries that can adaptively corrupt keys. We presented proofs for the most adaptive model to date, where the adversary can both corrupt keys and build the re-encryption graph adaptively. We identified a number of properties that imply that a PRE scheme secure in the selective model is also secure in the adaptive model. Our proofs demonstrate a sub-exponential security loss and do not rely on inefficient methods. In doing so, we proved that pcBV-PRE is adaptively secure. Helpful follow-up work could investigate whether it is possible to emulate our results using an alternative method that does not rely on token-source privacy. As most existing PRE schemes are not token-source private, such an extension may be applicable to a larger number of existing schemes, making adaptive security more easily achievable. Another extension is to investigate whether the techniques presented in Chapter VI for proving adaptive security can be applied to symmetric and/or bidirectional PRE schemes. Results in this area may also be applicable to updatable encryption schemes. For adaptive security, one possible avenue for achieving this might be to apply recent techniques in proving IND-CCA security for updatable encryption by Klooß, Lehmann and Rupp [KLR19a].

We finish by suggesting some general extensions to our work. One possible continuation is to extend all our contributions to the CCA model. Another future avenue could explore whether the challenge graph observation can be deployed to areas outside PRE. This may enable other cryptographic primitives to be proven secure with respect to adaptive key corruptions. Finally, we note that it would also be useful to explore methods for re-encrypting longer messages without sacrificing security, as this would make the scheme more useful in practice. One of the main challenges here is for the resulting construction to have PCS, as this is not the case for the key encapsulation method. It would be especially desirable for such a scheme to have the majority of the computation performed by the proxy as opposed to the client, as this better aligns with many reasons for outsourcing re-encryption. We believe that these advancements will lead to PRE being more widely deployed in practice.

Bibliography

[ABC$^+$05] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 205–222, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany.

[ABG$^+$97] James P. Anderson, Sheila Brand, Li Gong, Thomas Haigh, Steven B. Lipner, Teresa F. Lunt, Ruth Nelson, William Neugent, Hilarie Orman, Marcus J. Ranum, Roger R. Schell, and Eugene H. Spafford. Firewalls: An expert roundtable. *IEEE Software*, 14(5):60–66, September/October 1997. `http://dlib.computer.org/so/books/so1997/pdf/s5060.pdf`.

[ABH09] Giuseppe Ateniese, Karyn Benson, and Susan Hohenberger. Key-private proxy re-encryption. In Marc Fischlin, editor, *Topics in Cryptology – CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 279–294, San Francisco, CA, USA, April 20–24, 2009. Springer, Heidelberg, Germany.

[AFGH05] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *ISOC Network and Distributed System Security Symposium – NDSS 2005*, San Diego, CA, USA, February 3–4, 2005. The Internet Society.

[AKSX04] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM*

SIGMOD international conference on Management of data, pages 563–574. ACM, 2004.

[Ama17] Amazon Web Services. Protecting data using client-side encryption, 2017. Online; accessed 26-November-2018, `http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingClientSideEncryption.html`.

[AMP19] Navid Alamati, Hart Montgomery, and Sikhar Patranabis. Symmetric primitives with structured secrets. Cryptology ePrint Archive, Report 2019/608, 2019. `https://eprint.iacr.org/2019/608`.

[BBB$^+$12] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. Recommendation for key management part 1: General (revision 3). *NIST special publication*, 800(57):1–147, 2012.

[BBC16] Dropbox hack 'affected 68 million users', 31 August 2016. Online; accessed 6-November-2019; `https://www.bbc.co.uk/news/technology-37232635`.

[BBC18] Uber pays $148m over data breach cover-up, 27 September 2018. Online; accessed 6-November-2019; `https://www.bbc.co.uk/news/technology-45666280`.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582, Gold Coast, Australia, December 9–13, 2001. Springer, Heidelberg, Germany.

[BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 535–552, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Heidelberg, Germany.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany.

[BCLO09] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 224–241, Cologne, Germany, April 26–30, 2009. Springer, Heidelberg, Germany.

[BCO11] Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 578–595, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[BS16] Jean-François Biasse and Fang Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In Robert Krauthgamer, editor, *27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 893–902, Arlington, VA, USA, January 10–12, 2016. ACM-SIAM.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.

[CCG16] Katriel Cohn-Gordon, Cas J. F. Cremers, and Luke Garratt. On post-compromise security. In *IEEE 29th Computer Security Foundations Sym-*

posium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016, pages 164–178. IEEE Computer Society, 2016.

[CCL$^+$14] Nishanth Chandran, Melissa Chase, Feng-Hao Liu, Ryo Nishimaki, and Keita Xagawa. Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices. In Hugo Krawczyk, editor, *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 95–112, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany.

[CCV12] Nishanth Chandran, Melissa Chase, and Vinod Vaikuntanathan. Functional re-encryption and collusion-resistant obfuscation. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 404–421, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Heidelberg, Germany.

[CDPR16] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 559–585, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[CDW17] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short stickelberger class relations and application to ideal-SVP. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 324–348, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

[CH07] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 2007: 14th Conference on Computer and Communications Security*, pages 185–194, Alexandria, Virginia, USA, October 28–31, 2007. ACM Press.

[CJJ+14] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *ISOC Network and Distributed System Security Symposium – NDSS 2014*, San Diego, CA, USA, February 23–26, 2014. The Internet Society.

[Coh17] Aloni Cohen. What about Bob? The inadequacy of CPA security for proxy reencryption. Cryptology ePrint Archive, Report 2017/785, 2017. `http://eprint.iacr.org/2017/785`.

[Coh19] Aloni Cohen. What about Bob? The inadequacy of CPA security for proxy reencryption. In Dongdai Lin and Kazue Sako, editors, *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 11443 of *Lecture Notes in Computer Science*, pages 287–316, Beijing, China, April 14–17, 2019. Springer, Heidelberg, Germany.

[Coo17] James Cook. A lawyer explains what you should do if your private photos are posted online, *Business Insider*, 25 March 2017. Online; accessed 12-September-2019.

[CWYD10] Sherman S. M. Chow, Jian Weng, Yanjiang Yang, and Robert H. Deng. Efficient unidirectional proxy re-encryption. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 10: 3rd International Conference on Cryptology in Africa*, volume 6055 of *Lecture Notes in Computer Science*, pages 316–332, Stellenbosch, South Africa, May 3–6, 2010. Springer, Heidelberg, Germany.

[DDLM19a] Alex Davidson, Amit Deo, Ela Lee, and Keith Martin. Strong post-compromise secure proxy re-encryption. In Julian Jang-Jaccard and Fuchun Guo, editors, *ACISP 19: 24th Australasian Conference on Information Security and Privacy*, volume 11547 of *Lecture Notes in Computer Science*, pages 58–77, Christchurch, New Zealand, July 3–5, 2019. Springer, Heidelberg, Germany.

[DDLM19b] Alex Davidson, Amit Deo, Ela Lee, and Keith Martin. Strong post-compromise secure proxy re-encryption. Cryptology ePrint Archive, Report 2019/368, 2019. `https://eprint.iacr.org/2019/368`.

[ElG86] Taher ElGamal. On computing logarithms over finite fields. In Hugh C. Williams, editor, *Advances in Cryptology – CRYPTO'85*, volume 218 of *Lecture Notes in Computer Science*, pages 396–402, Santa Barbara, CA, USA, August 18–22, 1986. Springer, Heidelberg, Germany.

[EPRS17] Adam Everspaugh, Kenneth G. Paterson, Thomas Ristenpart, and Samuel Scott. Key rotation for authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 98–129, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[FKKP18] Georg Fuchsbauer, Chethan Kamath, Karen Klein, and Krzysztof Pietrzak. Adaptively secure proxy re-encryption. Cryptology ePrint Archive, Report 2018/426, 2018. `https://eprint.iacr.org/2018/426`.

[FKKP19] Georg Fuchsbauer, Chethan Kamath, Karen Klein, and Krzysztof Pietrzak. Adaptively secure proxy re-encryption. In Dongdai Lin and Kazue Sako, editors, *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 11443 of *Lecture Notes in Computer Science*, pages 317–346, Beijing, China, April 14–17, 2019. Springer, Heidelberg, Germany.

[FL16] Xiong Fan and Feng-Hao Liu. Various proxy re-encryption schemes from lattices. Cryptology ePrint Archive, Report 2016/278, 2016. `http://eprint.iacr.org/2016/278`.

[GA07] Matthew Green and Giuseppe Ateniese. Identity-based proxy re-encryption. In Jonathan Katz and Moti Yung, editors, *ACNS 07: 5th International Conference on Applied Cryptography and Network Security*, volume 4521 of *Lecture Notes in Computer Science*, pages 288–306, Zhuhai, China, June 5–8, 2007. Springer, Heidelberg, Germany.

[GDP16] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (General Data Protection Regulation). *Official Journal of the European Union*, L119:1—88, 2016.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 465–482, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GN08] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 31–51, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[HRsV07] Susan Hohenberger, Guy N. Rothblum, abhi shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 233–252, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany.

[ID03] Anca Ivan and Yevgeniy Dodis. Proxy cryptography revisited. In *ISOC Network and Distributed System Security Symposium – NDSS 2003*, San Diego, CA, USA, February 5–7, 2003. The Internet Society.

[JKK$^+$17] Zahra Jafargholi, Chethan Kamath, Karen Klein, Ilan Komargodski, Krzysztof Pietrzak, and Daniel Wichs. Be adaptive, avoid overcommitting. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 133–163, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[Jon13] Josh Jones. The Enigma machine: How Alan Turing helped break the unbreakable Nazi code, 2013. Online; accessed 16-September-2019.

[Kee10] John Keegan. *Intelligence in War.* Random House, 2010.

[Ker83] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, Vol. IX:5–38, January 1883.

[Kir14] Elena Kirshanova. Proxy re-encryption from lattices. In Hugo Krawczyk, editor, *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 77–94, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography.* Chapman and Hall/CRC, 2014.

[KLR19a] Michael Klooß, Anja Lehmann, and Andy Rupp. (R)CCA secure updatable encryption with integrity protection. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 68–99, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.

[KLR19b] Michael Klooß, Anja Lehmann, and Andy Rupp. (r)cca secure updatable encryption with integrity protection. Cryptology ePrint Archive, Report 2019/222, 2019. `https://eprint.iacr.org/2019/222`.

[LCLS09] Xiaohui Liang, Zhenfu Cao, Huang Lin, and Jun Shao. Attribute based proxy re-encryption with delegating capabilities. In Wanqing Li, Willy Susilo, Udaya Kiran Tupakula, Reihaneh Safavi-Naini, and Vijay Varadharajan, editors, *ASIACCS 09: 4th ACM Symposium on Information, Computer and Communications Security*, pages 276–286, Sydney, Australia, March 10–12, 2009. ACM Press.

[Lee17a] Ela Lee. Improved security notions for proxy re-encryption to enforce access control. In Tanja Lange and Orr Dunkelman, editors, *Progress in Cryptology - LATINCRYPT 2017: 5th International Conference on Cryptology and*

Information Security in Latin America, volume 11368 of *Lecture Notes in Computer Science*, pages 66–85, Havana, Cuba, September 20–22, 2017. Springer, Heidelberg, Germany.

[Lee17b] Seungkwang Lee. A masked white-box cryptographic implementation for protecting against differential computation analysis. Cryptology ePrint Archive, Report 2017/267, 2017. `http://eprint.iacr.org/2017/267`.

[Leo11] Woody Leonhard. Dropbox caught with its finger in the cloud cookie jar, *InfoWorld.com*, May 2011. Online; accessed 26-November-2018.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339, San Francisco, CA, USA, February 14–18, 2011. Springer, Heidelberg, Germany.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 35–54, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.

[LT18] Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 685–716, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th Annual ACM Symposium on Theory of Computing*, pages 1219–1234, New York, NY, USA, May 19–22, 2012. ACM Press.

[LV08a] Benoît Libert and Damien Vergnaud. Multi-use unidirectional proxy re-signatures. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008: 15th Conference on Computer and Communications Security*, pages 511–520, Alexandria, Virginia, USA, October 27–31, 2008. ACM Press.

[LV08b] Benoît Libert and Damien Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In Ronald Cramer, editor, *PKC 2008: 11th International Workshop on Theory and Practice in Public Key Cryptography*, volume 4939 of *Lecture Notes in Computer Science*, pages 360–379, Barcelona, Spain, March 9–12, 2008. Springer, Heidelberg, Germany.

[Mic10] Daniele Micciancio. Duality in lattice cryptography. In *Public key cryptography*, page 2, 2010. Invited talk.

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages 372–381, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press.

[MS17] Steven Myers and Adam Shull. Efficient hybrid proxy re-encryption for practical revocation and key rotation. Cryptology ePrint Archive, Report 2017/833, 2017. `http://eprint.iacr.org/2017/833`.

[NAL15] David Nuñez, Isaac Agudo, and Javier Lopez. NTRUReEncrypt: An efficient proxy re-encryption scheme based on NTRU. In Feng Bao, Steven Miller, Jianying Zhou, and Gail-Joon Ahn, editors, *ASIACCS 15: 10th ACM Symposium on Information, Computer and Communications Security*, pages 179–189, Singapore, April 14–17, 2015. ACM Press.

[NK16] Harvey Nash and KPMG. Harvey Nash / KPMG CIO survey 2016: The creative CIO, 2016. Online; accessed 10-December 2019.

[OWA18] OWASP. Cryptographic storage cheat sheet. `https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet`, 2018. Online; accessed 9-October 2018.

[PCI18] PCI Security Standards Council 2018. Payment Card Industry (PCI) data security standard (version 3.2.1). 2018.

[PRSV17] Yuriy Polyakov, Kurt Rohloff, Gyana Sahu, and Vinod Vaikuntanathan. Fast proxy re-encryption for publish/subscribe systems. *ACM Transactions on Privacy and Security (TOPS)*, 20(4):14, 2017.

[PWA$^+$16] Le Trieu Phong, Lihua Wang, Yoshinori Aono, Manh Ha Nguyen, and Xavier Boyen. Proxy re-encryption schemes with key privacy from LWE. Cryptology ePrint Archive, Report 2016/327, 2016. `http://eprint.iacr.org/2016/327`.

[Rog04] Phillip Rogaway. On the role definitions in and beyond cryptography. In *Annual Asian computing science conference*, pages 13–32. Springer, 2004.

[Sav16] Vlad Savov. Pixel 'phone by Google' announced, *The Verge*, 4 October 2016. Online; accessed 10-December-2019.

[SC09] Jun Shao and Zhenfu Cao. CCA-secure proxy re-encryption without pairings. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 357–376, Irvine, CA, USA, March 18–20, 2009. Springer, Heidelberg, Germany.

[Sin00] Simon Singh. *The Code Book: The Secret History of Codes and Code-breaking.* Fourth Estate, 2000.

[SPS14] Emil Stefanov, Charalampos Papamanthou, and Elaine Shi. Practical dynamic searchable encryption with small leakage. In *ISOC Network and Distributed System Security Symposium – NDSS 2014*, San Diego, CA, USA, February 23–26, 2014. The Internet Society.

[Swe17] Mark Sweney. Film and TV streaming and downloads overtake DVD sales for first time. *The Guardian*, Jan 2017.

[TYM14] Isamu Teranishi, Moti Yung, and Tal Malkin. Order-preserving encryption secure beyond one-wayness. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 42–61, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany.

[vJO$^+$12] Marten van Dijk, Ari Juels, Alina Oprea, Ronald L. Rivest, Emil Stefanov, and Nikos Triandopoulos. Hourglass schemes: how to prove that cloud files are encrypted. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012: 19th Conference on Computer and Communications Security*, pages 265–280, Raleigh, NC, USA, October 16–18, 2012. ACM Press.

[Wea00] Patrick D Weadon. The Battle of Midway: AF is short of water, 2000. Fort Meade, MD: National Security Agency, Center for Cryptologic History.

[Wil18] Hannah Williams. The history of cloud computing: A timeline of key moments from the 1960s to now, *Computerworld*, 13 March 2018. Online; accessed 9-November-2019;.

[YCPC18] Yinan Yu, Hailiang Chen, Chih Hung Peng, and Patrick YK Chau. The causal effect of video streaming on DVD sales: Evidence from a natural experiment, *Working Paper, The University of Hong Kong.* 2018.

[Zhe97] Yuliang Zheng. Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption). In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Heidelberg, Germany.

Appendix A

Other re-encryption primitives

Updatable encryption is a similar primitive to PRE in that it also re-encrypts ciphertexts from one key to another key. The main difference is that, as updatable encryption was created as a means of enforcing key rotation, keys are associated with *epochs*, and re-encryptions are performed sequentially from one epoch to the next. This means the resulting re-encryption graph is a chain (or collection of chains). The following definition is taken from [LT18], although we note that an alternative definition that is more similar to PRE is given in [KLR19a]. We give the original definition as it better fits the descriptions of updatable encryption schemes and security models discussed in this thesis.

Definition A.1. *An* updatable encryption *scheme consists of the following algorithms:*

- $\mathsf{Setup}(1^\lambda) \xrightarrow{\$} (\text{params}, k_0)$*: A probabilistic algorithm that outputs a set of parameters and an initial symmetric key* k_0*.*

  As usual params *is input to every subsequent algorithm, but we leave it out for compactness of notation.*
- $\mathsf{Next}(k_e) \xrightarrow{\$} (k_{e+1}, \Delta_{e,e+1})$*: A probabilistic algorithm that takes a symmetric key* k_e *for epoch* e*, it outputs a new key* k_{e+1} *for epoch* $e+1$ *and an update token* $\Delta_{e,e+1}$ *from epoch* e *to epoch* $e+1$*.*
- $\mathsf{Enc}(k_e, m) \xrightarrow{\$} c_e$*: A probabilistic algorithm that takes a message* m *and epoch key* k_e *and produces a ciphertext* c_e*.*
- $\mathsf{Dec}(k_e, c_e) \rightarrow \{m', \perp\}$*: A deterministic algorithm that takes a ciphertext* c_e *and epoch key* k_e *and produces either an element of the message space* m' *or an error symbol* $\perp$*.*
- $\mathsf{Upd}(\Delta_{e-1,e}, c_e) \xrightarrow{(\$)} c_{e+1}$*: An algorithm that is either probabilistic or deterministic that takes an update token* $\Delta_{e-1,e}$ *and ciphertext* c_e *and outputs an updated ciphertext*

c_{e+1}.

An updatable encryption scheme is correct *if the underlying encryption scheme given by* $\mathcal{SE} = \{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *is correct, and if for all messages* $m \in \mathcal{M}$*, initial keys* $k_0 \overset{\$}{\leftarrow} \mathsf{Setup}(1^\lambda)$*, subsequent keys and update tokens* $\{(k_e, \Delta_{e-1,e}) \overset{\$}{\leftarrow} \mathsf{Next}(k_{e-1})\}_{e=1}$*, all fresh ciphertexts* $c_0 \overset{\$}{\leftarrow} \mathsf{Enc}(k_0, m)$*, and all re-encrypted ciphertexts* $c_j \overset{(\$)}{\leftarrow} \mathsf{Upd}(\Delta_{j-1,j}, c_{j-1})$ *for* $1 \leq j \leq e$*:*

$$\mathsf{Dec}(k_j, c_j) \to m.$$

We note that it is named as an explicit goal of updatable encryption that re-encrypted ciphertexts should be as secure as fresh ones, meaning that all components of a ciphertext should be updated upon re-encryption. To our knowledge, the only other re-encryption schemes are *key rotation schemes* [EPRS17], which are essentially the same as updatable encryption schemes, and so we do not give an explicit definition here. Interestingly, security for key rotation schemes is modelled for arbitrary re-encryptions as opposed to sequential ones

Appendix B

Confidentiality of symmetric PRE

Here we give definitions for the symmetric setting for easier comparison with other re-encryption schemes defined with symmetric keys. We leave out some security games where the change is analogous, but explicitly give those which contains lists to clarify how those lists are updated.

```
$\mathsf{symIND\text{-}CPA}^{\mathcal{SE}}_{\mathcal{A}}(1^\lambda)$
$b \xleftarrow{\$} \{0,1\}$
$\kappa = 0$
$b' \xleftarrow{(\$)} \mathcal{A}^{\mathsf{O}_{\mathsf{KeyGen}},\mathsf{O}_{\mathsf{Enc}},\mathsf{Chal}_{\mathsf{Enc}}}(1^\lambda)$
**return** $b' = b$
```

```
$\mathsf{Chal}_{\mathsf{Enc}}(i, m_0, m_1) \xrightarrow{(\$)} c^*_b$
**if** $|m_0| \neq |m_1|$ :
    **return** $\perp$
$c^*_0 \xleftarrow{\$} \mathsf{Enc}(k_i, m_0)$
$c^*_1 \xleftarrow{\$} \mathsf{Enc}(k_i, m_1)$
**return** $c^*_b$
```

Figure B.1: symIND-CPA, the symmetric variant of the IND-CPA game. Again, we give a multi-key, multi-challenge variant.

We present oracles commonly used in games for symmetric PRE schemes in Figure B.2.

```
$\mathsf{O}_{\mathsf{KeyGen}}(1^\lambda)$
$\kappa = \kappa + 1$
$k_\kappa \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$\mathcal{DRG}.\mathsf{add}\{v_\kappa\}$
```

```
$\mathsf{O}_{\mathsf{Corrupt}}(i) \to k_i$
$\mathcal{K}_{\mathsf{corrupt}}.\mathsf{add}\{k_i\}$
**return** $k_i$
```

```
$\mathsf{O}_{\mathsf{Enc}}(i, m) \xrightarrow{\$} c$
$c \xleftarrow{\$} \mathsf{Enc}(k_i, m)$
$\boxed{\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(i, c)\}}$
**return** $c$
```

```
$\mathsf{O}_{\mathsf{ReKeyGen}}(i, j) \xrightarrow{(\$)} \Delta_{i,j}$
$\Delta_{i,j} \xleftarrow{(\$)} \mathsf{ReKeyGen}(k_i, k_j)$
$\mathcal{T}_{\mathsf{honest}}.\mathsf{add}\{(i, j, \Delta_{i,j})\}$
$\mathcal{DRG}.\mathsf{add}\{\vec{e}_{i,j}\}$
**return** $\Delta_{i,j}$
```

Figure B.2: Common oracles used in security games for symmetric PRE. $\mathsf{O}_{\mathsf{ReEnc}}$ is described in the specific security games, as its formulation differs depending on the game.

Appendix C

Symmetric PCS

Definition C.1. *An encryption scheme* $\mathcal{SE}$ *is* ϵ-Indistinguishable against Chosen Plaintext Attacks (ϵ-SE-IND-CPA-secure) *if for all* PPT *adversaries* $\mathcal{A}$*:*

$$\left|\Pr\left[\mathsf{symIND\text{-}CPA}_{\mathcal{A}}^{\mathcal{SE}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{symIND\text{-}CPA}_{\mathcal{A}}^{\mathcal{SE}}(1^\lambda) = 1\right]\right| \leq \epsilon,$$

where $\mathsf{symIND\text{-}CPA}_{\mathcal{A}}^{b\mathcal{SE}}(1^\lambda)$ *is defined in Figure B.1. If* ϵ *is negligible as parameterised by the security parameter* λ*, then we say the scheme is*Indistinguishable against Chosen Plaintext Attacks (sym-CPA-secure).

A symmetric PRE scheme $s\mathcal{PRE}$ *is* (ϵ-sym-CPA-secure) *if the encryption scheme given by* $s\mathcal{PRE} = \{s\mathcal{PRE}.\mathsf{KeyGen}, s\mathcal{PRE}.\mathsf{Enc}, s\mathcal{PRE}.\mathsf{Dec}\}$ *is* ϵ*-SE-IND-CPA-secure.*

```
$\mathsf{symSH}_{\mathcal{A}}^{s\mathcal{PRE}}(1^\lambda)$
$b \xleftarrow{\$} \{0,1\}$
$k_0 \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda), k_1 \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$\Delta_{0,1} \xleftarrow{\$} \mathsf{ReKeyGen}(k_0, k_1)$
$b' \xleftarrow{(\$)} \mathcal{A}^{\mathsf{Chal}^{\mathsf{symSH}}_{\mathsf{ReEnc/Enc}}}(1^\lambda, k_0, k_1, \Delta_{0,1})$
return $b' = b$
```

```
$\mathsf{Chal}^{\mathsf{symSH}}_{\mathsf{ReEnc/Enc}}(m^*, \ell^*) \xrightarrow{(\$)} (c, c'^{(b)})$
if $\ell^* > L - 1$ : return $\perp$
$c \leftarrow \mathsf{Enc}(k_0, m^*, \ell^*)$
$c'^{(0)} \xleftarrow{(\$)} \mathsf{ReEnc}(\Delta_{0,1}, c)$
$c'^{(1)} \xleftarrow{\$} \mathsf{Enc}(k_1, m^*, \ell^* + 1)$
return $(c, c'^{(b)})$
```

Figure C.1: Experiments for the symmetric source-hiding property. L is the number of times a ciphertext can be re-encrypted without breaking the correctness conditions.

Definition C.2. *An symmetric PRE scheme* $s\mathcal{PRE}$ *is* ϵ-source-hiding *if for all* PPT *adversaries* $\mathcal{A}$*:*

$$\Pr\left[\mathsf{symIND\text{-}CPA}_{\mathcal{A}}^{s\mathcal{PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon,$$

where SH *is defined in Figure C.1. If ϵ is negligible as parameterised by the security parameter λ, then we say $s\mathcal{PRE}$ is* source-hiding.

```
symPostComp_A^{sPRE}(1^λ)
    $b \xleftarrow{\$} \{0,1\}$
    $\mathcal{K}_{\mathsf{chal}}, \mathcal{K}_{\mathsf{corrupt}}, \mathcal{C}_{\mathsf{honest}}, \mathcal{C}_{\mathsf{chal}}, \mathcal{T}_{\mathsf{honest}}, \mathcal{DRG} = \emptyset$
    $\kappa = 0, \mathsf{called} = \mathsf{false}$
    $state \xleftarrow{(\$)} \mathcal{A}_0^{\mathsf{O_{KeyGen}}, \mathsf{O_{Corrupt}}}(1^\lambda)$
    $b' \xleftarrow{(\$)} \mathcal{A}_1^{\mathsf{O_{Enc}}, \mathsf{O_{ReKeyGen}}, \mathsf{O_{ReEnc}}, \mathsf{Chal_{ReEnc}}}(1^\lambda, state)$
    $\mathcal{K}_{\mathsf{chal}} \leftarrow \mathsf{UpdateChallengeKeys}(\mathcal{K}_{\mathsf{chal}}, \mathcal{DRG})$
    if $\mathcal{K}_{\mathsf{chal}} \cap \mathcal{K}_{\mathsf{corrupt}} \neq \emptyset$ : return $\perp$
    return $b' = b$

O_ReEnc(c, i, j, [Δ_{i,j}]) →($) c'
    if $\Delta_{i,j}$ given :
        if $(i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}$ : return $\perp$
    else : $\Delta_{i,j} \xleftarrow{(\$)} \mathsf{ReKeyGen}(k_i, k_j)$
    if $(i, c) \notin \mathcal{C}_{\mathsf{honest}}$ : return $\perp$
    $c' \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, c)$
    $\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(j, c')\}$
    if $(i, c) \in \mathcal{C}_{\mathsf{chal}}$ :
        $\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\{(j, c')\}, \mathcal{K}_{\mathsf{chal}}.\mathsf{add}\{k_j\}$
    return $c'$

Chal_ReEnc(i, j, [Δ_{i,j}, ]c_0, c_1) →($) c'_b
    if $|c_0| \neq |c_1|$ OR $\mathsf{called} = \mathsf{true}$ : return $\perp$
    if $(i, c_0), (i, c_1) \notin \mathcal{C}_{\mathsf{honest}}$ OR $(i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}}$ : return $\perp$
    $c'_0 \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, c_0)$
    $c'_1 \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, c_1)$
    $\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(j, c'_b)\}, \mathcal{C}_{\mathsf{chal}}.\mathsf{add}\{(j, c'_b)\}, \mathcal{K}_{\mathsf{chal}}.\mathsf{add}\{k_j\}$
    $\mathsf{called} \leftarrow \mathsf{true}$
    return $c'_b$
```

Figure C.2: The symmetric variant of selective PCS game.

Definition C.3. *A symmetric PRE scheme $s\mathcal{PRE}$ is said to be* ϵ-post-compromise secure (ϵ-PCS) *if for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*:*

$$\Pr\left[\mathsf{symPostComp}_{\mathcal{A}}^{s\mathcal{PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon,$$

where $\mathsf{symPostComp}_{\mathcal{A}}^{s\mathcal{PRE}}(1^\lambda)$ *is defined in Figure C.2.*

If ϵ is negligible as parameterised by the security parameter, then we say the scheme is Post-Compromise Security (PCS).

C: 1 RISE

The PCS updatable encryption scheme RISE [LT18] is given in Figure C.3.

```
$\mathsf{Setup}(1^\lambda) \xrightarrow{\$} (\text{params}, k_0)$
    Select a $\lambda$-bit prime $q$
    Select a generator $g$ of the group $\mathbb{Z}_q$
    params $= (\mathbb{Z}_q, g, q)$
    $x \xleftarrow{\$} \mathbb{Z}_q^*$
    $k_0 := (x, g^x)$
    return (params, $k_0$)

$\mathsf{Next}(k_e) \xrightarrow{\$} (k_{e+1}, \Delta_{e,e+1})$
    $(x, y) \leftarrow k_e$
    $x' \xleftarrow{\$} \mathbb{Z}_q^*$
    $\Delta_{e,e+1} \leftarrow (x'/x, g^{x'})$
    $k_{e+1} := (x', g^{x'})$
    return $(k_{e+1}, \Delta_{e,e+1})$

$\mathsf{Enc}(k_e, m) \xrightarrow{\$} c_e$
    $(x, y) \leftarrow k_e$
    $r \xleftarrow{\$} \mathbb{Z}_q$
    $c_e \leftarrow (y^r, g^r \cdot m)$
    return $c_e$

$\mathsf{Dec}(k_e, c_e) \rightarrow m'$
    $(x, y) \leftarrow k_e$
    $(c_0, c_1) \leftarrow c_e$
    $m' \leftarrow c_1 \cdot c_1^{-1/x}$
    return $m'$

$\mathsf{ReEnc}(\Delta_{e,e+1}, c_e) \xrightarrow{\$} c_{e+1}$
    $(\Delta, y') \leftarrow \Delta_{e,e+1}$
    $(c_0, c_1) \leftarrow c_e$
    $r' \xleftarrow{\$} \mathbb{Z}_p$
    $c_0' \leftarrow c_0^\Delta \cdot y'^{r'}$
    $c_1' \leftarrow c_1 \cdot g^{r'}$
    return $c_{e+1} = (c_0', c_1')$
```

Figure C.3: RISE, an updatable encryption scheme with PCS as defined in [LT18].

Re-encrypted ciphertexts have the form

$$\begin{aligned} c' &= ((y^r)^{x'/x} \cdot y^{r'}, g^r \cdot m \cdot g^{r'}) \\ &= ((y')^{r+r'}, g^{r+r'} \cdot m). \end{aligned}$$

In other words, it effectively creates a fresh encryption using randomness $\bar{r} = r + r'$.

We give the equivalent lemma to Lemma V.6 for symmetric PRE, using RISE as an example in our proof.

Lemma C.1. *sIND-UPD $\not\Longrightarrow$ symmetric PCS.*

Proof. For this proof, we use RISE as a counterexample. We note that RISE [LT18] is proven to be sIND-UPD. Because RISE is not unidirectional, it does not have PCS by Lemma V.2. To see this, we consider the trivial adaptation of RISE for general proxy re-encryption[1] as this fits more with our notation, but we observe that the original construction defined as an updatable encryption scheme is also sufficient for the proof.

- $\mathsf{RISE'.KeyGen}(1^\lambda) : x \xleftarrow{\$} \mathbb{Z}_q^*, (\mathsf{pk}, \mathsf{sk}) =, (x, g^x)$

[1]We allow for re-encryptions between arbitrary keys as opposed to sequentially from k_i to k_{i+1}.

- $\mathsf{RISE}'.\mathsf{Enc}(\mathsf{pk}, m) : r \xleftarrow{\$} \mathbb{Z}_q^*, c \leftarrow (\mathsf{pk}^r, g^r \cdot m)$
- $\mathsf{RISE}'.\mathsf{Dec}(\mathsf{sk}, m) : m' \leftarrow c_1 \cdot c_0^{-1/\mathsf{sk}}$
- $\mathsf{RISE}'.\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_j) : \Delta_{i,j} = (\mathsf{sk}_j/\mathsf{sk}_i, \mathsf{pk}_j) = (x_j/x_i, g^{x_j})$
- $\mathsf{RISE}'.\mathsf{ReEnc}(\Delta_{i,j}, c) : (\Delta, y') = \Delta_{i,j}, (c_0, c_1) \leftarrow c, r' \xleftarrow{\$} \mathbb{Z}_q^*, c' \leftarrow (c_0^{\Delta} \cdot y'^{r'}, c_1 \cdot g^{r'})$

Given $\Delta_{i,j} = (x_j/x_i, g^{x_j})$, and x_i, $\mathcal{A}$ can compute x_j and use this to decrypt the challenge ciphertext. $\square$

Appendix D

Asymmetric sIND-UPD

IND-UPD [LT18, definition 3] is a post-compromise security notion for updatable encryption schemes, which are a variant of symmetric PCS schemes. The main difference in updatable encryption is that key updates happen sequentially from k_i to k_{i+1}, meaning the re-encryption graph is always a chain. Another notable difference between updatable encryption schemes and PRE schemes is that they contain an algorithm Next, which generates a new key and an update token from the old key to the new one as opposed to defining these functions separately.

Here we adapt the selective definition that we presented in Definition D.1 for the public-key setting.

Definition D.1. *A PRE scheme $\mathcal{PRE}$ is said to be* ϵ-pksIND-UPD-secure (selectively ϵ-pkIND-UPD-secure) *if for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*:*

$$\Pr\left[\mathsf{pksIND\text{-}UPD}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon,$$

where $\mathsf{pksIND\text{-}UPD}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ *is defined in Figure D.1.*

If ϵ is negligible as parameterised by the security parameter, then we say the scheme is (selectively) IND-UPD-secure*.*

```
$\mathsf{pksIND\text{-}UPD}^{\mathcal{PRE}}_{\mathcal{A}}(1^\lambda)$
    $b \xleftarrow{\$} \{0,1\}$
    $e := 0, e^* := \perp, \mathcal{DRG} := \{v_0\}$
    $\mathcal{K}_{\mathsf{corrupt}}, \mathcal{K}_{\mathsf{chal}}, \mathcal{C}_{\mathsf{honest}}, \mathcal{C}_{\mathsf{chal}} = \emptyset$
    $(\mathsf{pk}_0, \mathsf{sk}_0) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
    $(c_0, c_1, state) \xleftarrow{(\$)} \mathcal{A}_0^{\mathsf{O_{Enc}},\mathsf{O_{Next}},\mathsf{O_{ReEnc}},\mathsf{O_{Corrupt}},\mathsf{O_{LearnTok}}}(1^\lambda)$
    if $\left((e, c_0) \notin \mathcal{C}_{\mathsf{honest}}\right) \vee \left((e, c_1) \notin \mathcal{C}_{\mathsf{honest}}\right) \vee (|c_0| \neq |c_1|)$ :
        return $\perp$
    $e^* := e, \bar{c}_0 := c_0, \bar{c}_1 := c_1$
    $c^*_e \xleftarrow{(\$)} \mathsf{ReEnc}(\Delta_{e-1,e}, c_b)$
    $\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\{(e, c^*_e)\}, \mathcal{K}_{\mathsf{chal}}.\mathsf{add}\{k_e\}$
    $b' \xleftarrow{(\$)} \mathcal{A}_1^{\mathsf{O_{Enc}},\mathsf{O_{Next}},\mathsf{O_{ReEnc}},\mathsf{O_{LearnTok}},\mathsf{O_{ReEnc}}}(1^\lambda, state)$
    $\mathcal{K}_{\mathsf{chal}} \leftarrow \mathsf{UpdateChallengeKeys}(\mathcal{K}_{\mathsf{chal}}, \mathcal{DRG})$
    if $\mathcal{K}_{\mathsf{chal}} \cap \mathcal{K}_{\mathsf{corrupt}} \neq \emptyset$ : return $\perp$
    return $b' = b$

$\mathsf{O_{Enc}}(m) \xrightarrow{\$} c_e$
    $c_e \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_e, m)$
    $\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(e, c_e)\}$
    return $c_e$

$\mathsf{O_{Next}}() \xrightarrow{(\$)} \mathsf{pk}_e$
    $e \leftarrow e + 1$
    $\mathcal{DRG}.\mathsf{add}\{v_e\}$
    $k_e \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
    $\Delta_{e-1,e} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_{e-1}, \mathsf{pk}_e)$
    if $e^* \neq \perp$ :
        $c^*_e \leftarrow \mathsf{ReEnc}(\Delta_{e-1,e}, c^*_{e-1})$
        $\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\{(e, c^*_e)\}$
    return $\mathsf{pk}_e$

$\mathsf{O_{ReEnc}}(c_{e-1}) \xrightarrow{(\$)} c_e$
    if $(e-1, c_{e-1}) \notin \mathcal{C}_{\mathsf{honest}}$ :
        return $\perp$
    if (ReEnc is deterministic) $\wedge$ $(e = e^*)$ :
        if $(c_{e-1} = \bar{c}_0) \vee (c_{e-1} = \bar{c}_1)$ :
            return $\perp$
    $c_e \xleftarrow{(\$)} \mathsf{ReEnc}(\Delta_{e-1,e}, c_{e-1})$
    $\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(e, c_e)\}$
    return $c_e$

$\mathsf{O_{Corrupt}}(i) \rightarrow \mathsf{sk}_i$
    $\mathcal{K}_{\mathsf{corrupt}}.\mathsf{add}\{\mathsf{sk}_i\}$
    return $\mathsf{sk}_i$

$\mathsf{O_{LearnTok}}(i) \rightarrow \Delta_{i-1,i}$
    if $(i > e) \vee (i = e^*)$ : return $\perp$
    if $\mathcal{PRE}$ is unidirectional :
        $\mathcal{DRG}.\mathsf{add}\{\overrightarrow{e}_{i-1,i}\}$ // directed edge
    else ($\mathcal{PRE}$ is bidirectional) :
        $\mathcal{DRG}.\mathsf{add}\{e_{i-1,i}\}$ // undirected edge
    return $\Delta_{i-1,i}$

$\mathsf{O_{ReEncChal}}() \xrightarrow{(\$)} c^*_e$
    $\mathcal{K}_{\mathsf{chal}}.\mathsf{add}\{\mathsf{sk}_e\}$
    return $c^*_e$
```

Figure D.1: The pksIND-UPD game, a public-key adaptation of sIND-UPD (Figure V.3). Differences from the original IND-UPD game [LT18] described in Figure V.3 are indicated with shading.

Appendix E

Adaptive Post-Compromise Security

Definition E.1. *A PRE scheme $\mathcal{PRE}$ is said to have* (ϵ, κ)-adaptive Post-Compromise Security (ϵ-adaptive PCS) *if for all* PPT *adversaries $\mathcal{A}$:*

$$\Pr\left[\mathsf{a\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \epsilon,$$

where $\mathsf{a\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ *is defined in Figure E.1 and κ is the maximum number of keys generated in* $\mathsf{a\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$.

If ϵ is negligible and κ is polynomial as determined by the security parameter, then we say the scheme has adaptive Post-Compromise Security (adaptive PCS).

```
$\mathsf{a\text{-}PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$
$b \xleftarrow{\$} \{0,1\}$
$\mathcal{K}_{\mathsf{corrupt}}, \mathcal{T}_{\mathsf{honest}}, \mathcal{C}_{\mathsf{honest}}, \mathcal{C}_{\mathsf{chal}} = \emptyset$
$\kappa := 0, \mathsf{called} \leftarrow \mathsf{false}$
$b' \xleftarrow{(\$)} \mathcal{A}^{\mathsf{O}_{\mathsf{KeyGen}}, \mathsf{O}_{\mathsf{Corrupt}}, \mathsf{O}_{\mathsf{Enc}}, \mathsf{O}_{\mathsf{ReKeyGen}}, \mathsf{O}^{\mathsf{PC}}_{\mathsf{ReEnc}}, \mathsf{Chal}^{\mathsf{PC}}_{\mathsf{ReEnc}}}(1^\lambda)$
$\mathcal{K}_{\mathsf{chal}} \leftarrow \mathsf{UpdateChallengeKeys}(\mathcal{K}_{\mathsf{chal}}, \mathcal{DRG})$
if $\mathcal{K}_{\mathsf{chal}} \cap \mathcal{K}_{\mathsf{corrupt}} \neq \emptyset$ :
  return $\perp$
else :
  return $(b' = b)$
```

```
$\mathsf{O}_{\mathsf{KeyGen}}(1^\lambda) \xrightarrow{\$} \mathsf{pk}$
$\kappa = \kappa + 1$
$(\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
return $\mathsf{pk}_\kappa$
```

```
$\mathsf{O}_{\mathsf{Corrupt}}(i) \rightarrow \mathsf{sk}_i$
$\mathcal{K}_{\mathsf{corrupt}}.\mathsf{add}\{\mathsf{sk}_i\}$
return $\mathsf{sk}_i$
```

```
$\mathsf{O}_{\mathsf{Enc}}(i, m) \xrightarrow{\$} c$
$c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m)$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(i, c)\}$
return $c$
```

```
$\mathsf{O}_{\mathsf{ReKeyGen}}(i, j) \xrightarrow{(\$)} \Delta_{i,j}$
$\Delta_{i,j} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$\mathcal{T}_{\mathsf{honest}}.\mathsf{add}\{i, j, \Delta_{i,j}\}$
$\mathcal{DRG}.\mathsf{add}\{\overrightarrow{e}_{i,j}\}$
return $\Delta_{i,j}$
```

```
$\mathsf{O}^{\mathsf{PC}}_{\mathsf{ReEnc}}(i, j, [\Delta_{i,j}], c)$
if $(i, c) \notin \mathcal{C}_{\mathsf{honest}}$ : return $\perp$
if $((i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}})$ :
  $\Delta_{i,j} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$c' \xleftarrow{(\$)} \mathsf{ReEnc}(\Delta_{i,j}, c)$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{j, c'\}$
if $(i, c) \in \mathcal{C}_{\mathsf{chal}}$ :
  $\mathcal{DRG}.\mathsf{add}\{\overrightarrow{e}_{i,j}\}$
  $\mathcal{C}_{\mathsf{chal}}.\mathsf{add}\{(j, c')\}$
return $c'$
```

```
$\mathsf{Chal}^{\mathsf{PC}}_{\mathsf{ReEnc}}(i, j, [\Delta_{i,j},]\ c_0, c_1)$
if $\mathsf{called} \vee (|c_0| \neq |c_1|)$ :
  return $\perp$
if $\big((i, c_0) \notin \mathcal{C}_{\mathsf{honest}}\big) \vee \big((i, c_1) \notin \mathcal{C}_{\mathsf{honest}}\big)$
  return $\perp$
if $((i, j, \Delta_{i,j}) \notin \mathcal{T}_{\mathsf{honest}})$ :
  $\Delta_{i,j} \xleftarrow{(\$)} \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$
$c'_0 \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, c_0)$
$c'_1 \xleftarrow{\$} \mathsf{ReEnc}(\Delta_{i,j}, c_1)$
$\mathcal{K}_{\mathsf{chal}}.\mathsf{add}\{\mathsf{sk}_i\}$
$\mathcal{C}_{\mathsf{honest}}.\mathsf{add}\{(i, c)\}, \mathcal{C}_{\mathsf{chal}}.\mathsf{add}\{(i, c)\}$
$\mathsf{called} \leftarrow \mathsf{true}$
return $c'_b$
```

Figure E.1: The adaptive PCS game, the adaptive version of $\mathsf{PostComp}_{\mathcal{A}}^{\mathcal{PRE}}(1^\lambda)$ (Figure V.6).