Authentication with weaker trust assumptions for voting systems

Elizabeth A. Quaglia[1] and Ben Smyth[2]

[1] Information Security Group - Royal Holloway, University of London, UK
[2] Interdisciplinary Centre for Security, Reliability and Trust,
University of Luxembourg, Luxembourg

Abstract. Some voting systems are reliant on external authentication services. Others use cryptography to implement their own. We combine digital signatures and non-interactive proofs to derive a generic construction for voting systems with their own authentication mechanisms, from systems that rely on external authentication services. We prove that our construction produces systems satisfying ballot secrecy and election verifiability, assuming the underlying voting system does. Moreover, we observe that works based on similar ideas provide neither ballot secrecy nor election verifiability. Finally, we demonstrate applicability of our results by applying our construction to the Helios voting system.

1 Introduction

An election is a decision-making procedure to choose representatives [22, 30, 17]. Choices should be made freely by voters with equal influence, and this must be ensured by voting systems [42, 24, 25]. Some voting systems rely on *external* authentication services to ensure choices are made by voters. E.g., Helios [2, 26] supports authentication via Facebook, Google and Yahoo using OAuth.[3] Other voting systems use cryptography to implement their own authentication mechanisms. E.g., the voting system by Juels, Catalano & Jakobsson uses a combination of encrypted nonces and plaintext equality tests for authentication [20]. We combine digital signatures and non-interactive proofs to derive a construction for voting systems with their own authentication mechanisms from systems that rely on external service providers. Our construction produces voting systems which require less trust, since systems built upon cryptography are typically preferable to systems trusting external service providers.

Many voting systems rely on art, rather than science, to ensure that choices are made freely by voters with equal influence. Such systems build upon creativity and skill, rather than scientific foundations, and are typically broken in ways that compromise free choice, e.g., [16, 43, 44, 39], or permit adversaries to unduly influence the outcome, e.g., [19, 10]. By contrast, we prove that our construction produces voting systems that satisfy rigorous and precise security definitions

[3] Meyer & Smyth describe the application of OAuth in Helios [23].

of *ballot secrecy* and *election verifiability* that capture voters voting freely with equal influence.[4]

We demonstrate applicability of our construction by deriving voting systems with their own authentication mechanisms from Helios. Moreover, we compare those systems to Helios-C [13], a variant of Helios for two-candidate elections in which ballots are digitally signed. Our comparison reveals some subtle distinctions and we show that Helios-C does not satisfy our security definition, whereas our construction produces voting systems that do.

Structure. Section 2 recalls election scheme syntax. Section 3 presents our construction. Section 4 proves that our construction produces systems satisfying ballot secrecy. Section 5 proves that election verifiability is also satisfied. Section 6 demonstrates the application of our construction to the Helios voting system and compares the resulting systems to Helios-C. We conclude in Section 7. The appendices recall security definitions for voting systems and present proofs. Definitions of cryptographic primitives and associated security definitions are deferred to an accompanying technical report [28].

2 Election scheme syntax

We recall syntax by Smyth, Frink & Clarkson [36] for a class of voting systems that consist of the following four steps. First, a tallier[5] generates a key pair and (optionally) a registrar generates credentials for voters. Secondly, each voter constructs and casts a ballot for their vote. These ballots are recorded on a bulletin board. Thirdly, the tallier tallies the recorded ballots and announces an outcome, i.e., a distribution of votes. Finally, voters and other interested parties check that the outcome corresponds to votes expressed in recorded ballots.

Definition 1 (Election scheme [36]). An *election scheme with external authentication* is a tuple of efficient algorithms $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ and an *election scheme with internal authentication* is a tuple of efficient algorithms $(\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$, such that:[6]

Setup, denoted $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$, is run by the tallier. Setup takes a security parameter κ as input and outputs a key pair pk, sk, a maximum number of ballots mb, and a maximum number of candidates mc.

[4] Quaglia & Smyth [27] provide a tutorial-style introduction to definitions of ballot secrecy and election verifiability, and Smyth [33] provides a technical introduction.

[5] Some voting systems permit the tallier's role to be distributed amongst several talliers. For simplicity, we consider only a single tallier in this paper.

[6] Let $A(x_1, \ldots, x_n; r)$ denote the output of probabilistic algorithm A on inputs $x_1, \ldots, x_n$ and random coins r. Let $A(x_1, \ldots, x_n)$ denote $A(x_1, \ldots, x_n; r)$, where r is chosen uniformly at random. And let $\leftarrow$ denote assignment. Moreover, let $\langle x \rangle$ denote an optional input and $\mathbf{v}[v]$ denote component v of vector $\mathbf{v}$.

Register, denoted $(pd, d) \leftarrow \mathsf{Register}(pk, \kappa)$, is run by the registrar. It takes as input the public key pk of the tallier and a security parameter κ, and it outputs a *credential pair* (pd, d), where pd is a public credential and d is a private credential.

Vote, denoted $b \leftarrow \mathsf{Vote}(\langle d \rangle, pk, nc, v, \kappa)$, is run by voters. Vote takes as input a private credential d (optional), a public key pk, some number of candidates nc, a voter's vote v, and a security parameter κ. The vote should be selected from a sequence $1, \ldots, nc$ of candidates. Vote outputs a ballot b or error symbol $\perp$.

Tally, denoted $(\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk, nc, \mathfrak{bb}, \langle L \rangle, \kappa)$, is run by the tallier. Tally takes as input a private key sk, some number of candidates nc, a bulletin board $\mathfrak{bb}$, an electoral roll L (optional), and a security parameter κ, where $\mathfrak{bb}$ is a set. It outputs an election outcome $\mathbf{v}$ and a non-interactive proof pf that the outcome is correct. An election outcome is a vector $\mathbf{v}$ of length nc such that $\mathbf{v}[v]$ indicates the number of votes for candidate v.

Verify, denoted $s \leftarrow \mathsf{Verify}(pk, nc, \mathfrak{bb}, \langle L \rangle, \mathbf{v}, pf, \kappa)$, is run to audit an election. It takes as input a public key pk, some number of candidates nc, a bulletin board $\mathfrak{bb}$, an electoral roll L (optional), an election outcome $\mathbf{v}$, a proof pf, and a security parameter κ. It outputs a bit s, which is 1 if the election verifies successfully and 0 otherwise.

Election schemes with internal authentication must always use optional inputs, whereas election schemes with external authentication must not. Both schemes must satisfy *correctness*: there exists a negligible function negl, such that for all security parameters κ, integers nb and nc, and votes $v_1, \ldots, v_{nb} \in \{1, \ldots, nc\}$, it holds that if $\mathbf{v}$ is a vector of length nc whose components are all 0, then

$$\begin{aligned}
&\Pr[(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa);\\
&\quad \textbf{for } 1 \le i \le nb \textbf{ do}\\
&\quad\quad (pd_i, d_i) \leftarrow \mathsf{Register}(pk, \kappa);\\
&\quad\quad b_i \leftarrow \mathsf{Vote}(\langle d_i \rangle, pk, nc, v_i, \kappa);\\
&\quad\quad \mathbf{v}[v_i] \leftarrow \mathbf{v}[v_i] + 1;\\
&\quad (\mathbf{v}', pf) \leftarrow \mathsf{Tally}(sk, nc, \{b_1, \ldots, b_{nb}\}, \langle \{pd_1, \ldots, pd_{nb}\} \rangle, \kappa)\\
&\quad : nb \le mb \wedge nc \le mc \Rightarrow \mathbf{v} = \mathbf{v}'] > 1 - \mathsf{negl}(\kappa),
\end{aligned}$$

where algorithm Register is only applied for election scheme with internal authentication and optional inputs are only used for election scheme with internal authentication.

3 Our construction

Election schemes with internal authentication can be derived from schemes with external authentication using a digital signature scheme and a non-interactive proof system: Each voter publishes a ballot constructed using the underlying scheme with external authentication, along with a signature on that ballot and

a proof that they constructed both the ballot and the signature. Signatures and proofs are used to ensure that each tallied vote was cast by an authorised voter.

Our construction is formal described in Definition 3. It is parameterised by an election scheme with external authentication, a digital signature scheme, and a non-interactive proof system, derived from an underlying sigma protocol and a hash function, using the Fiat-Shamir transformation.[7] Hence, we denote election schemes derived using our construction as $\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H})$, where the underlying election scheme, signature scheme, sigma protocol and hash function are Γ, Ω, Σ and $\mathcal{H}$, respectively. To ensure our construction produces election schemes with internal authentication, the non-interactive proof system must be defined for a suitable relation, and we define such a relation as follows.

Definition 2. Given an election scheme with external authentication $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ and a digital signature scheme $\Omega = (\mathsf{Gen}_\Omega, \mathsf{Sign}_\Omega, \mathsf{Verify}_\Omega)$, we define binary relation $R(\Gamma, \Omega)$ over vectors of length 6 and vectors of length 4 such that $((pk, b, \sigma, nc, \kappa), (v, r, d, r')) \in R(\Gamma, \Omega) \Leftrightarrow b = \mathsf{Vote}(pk, nc, v, \kappa; r) \wedge \sigma = \mathsf{Sign}_\Omega(d, b; r')$.

Definition 3 (Construction). Suppose $\Gamma = (\mathsf{Setup}_\Gamma, \mathsf{Vote}_\Gamma, \mathsf{Tally}_\Gamma, \mathsf{Verify}_\Gamma)$ is an election scheme with external authentication, $\Omega = (\mathsf{Gen}_\Omega, \mathsf{Sign}_\Omega, \mathsf{Verify}_\Omega)$ is a digital signature scheme, Σ is a sigma protocol for a binary relation $R(\Gamma, \Omega)$, and $\mathcal{H}$ is a hash function. Let $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{Prove}_\Sigma, \mathsf{Verify}_\Sigma)$. We define $\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ such that:

- $\mathsf{Setup}(\kappa)$ computes $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}_\Gamma(\kappa)$ and outputs (pk, sk, mb, mc).
- $\mathsf{Register}(pk, \kappa)$ computes $(pd, d) \leftarrow \mathsf{Gen}_\Omega(\kappa)$ and outputs $(pd, (pd, d))$.
- $\mathsf{Vote}(d', pk, nc, v, \kappa)$ parses d' as (pd, d) and outputs $\perp$ if parsing fails, selects coins r and r' uniformly at random, computes
  $b \leftarrow \mathsf{Vote}_\Gamma(pk, nc, v, \kappa; r)$;
  $\sigma \leftarrow \mathsf{Sign}_\Omega(d, b; r')$;
  $\tau \leftarrow \mathsf{Prove}_\Sigma((pk, b, \sigma, nc, \kappa), (v, r, d, r'), \kappa)$,
  and outputs (pd, b, σ, τ).
- $\mathsf{Tally}(sk, nc, \mathfrak{bb}, L, \kappa)$ computes $(\mathbf{v}, pf) \leftarrow \mathsf{Tally}_\Gamma(sk, \mathsf{auth}(\mathfrak{bb}, L), nc, \kappa)$ and outputs $(\mathbf{v}, pf)$.
- $\mathsf{Verify}(pk, nc, \mathfrak{bb}, L, \mathbf{v}, pf, \kappa)$ computes $s \leftarrow \mathsf{Verify}_\Gamma(pk, \mathsf{auth}(\mathfrak{bb}, L), nc, \mathbf{v}, pf, \kappa)$ and outputs s.

Set $\mathsf{auth}(\mathfrak{bb}, L) = \{b \mid (pd, b, \sigma, \tau) \in \mathfrak{bb} \wedge \mathsf{Verify}_\Omega(pd, b, \sigma) = 1 \wedge \mathsf{Verify}_\Sigma((pk, b, nc, \kappa), \tau, \kappa) = 1 \wedge pd \in L \wedge (pd, b', \sigma', \tau') \notin \mathfrak{bb} \setminus \{(pd, b, \sigma, \tau)\} \wedge \mathsf{Verify}_\Omega(pd, b', \sigma') = 1\}$.

Our construction uses function auth to ensure tallied ballots are authorised and to discard ballots submitted under the same credential (i.e., if there is more

[7] Let $\mathsf{FS}(\Sigma, \mathcal{H})$ denote the non-interactive proof system derived by application of the Fiat-Shamir transformation to sigma protocol Σ and hash function $\mathcal{H}$.

than one ballot submitted with a private credential, then all ballots submitted under that credential are discarded). Since election schemes with internal authentication must satisfy correctness, the underlying digital signature scheme must ensure that key pairs are distinct. Hence, correctness of our construction depends on security of the underlying digital signature scheme, albeit in a tedious manner. Since we exploit strong unforgeability of the signature scheme for results in the following sections, we assume the same property here (to ensure key pairs are distinct). Weaker conditions could be used for generality.
The proof of Lemma 1 appears in our companion technical report [28].

Lemma 1. *Let Γ be an election scheme with external authentication, Ω be a digital signature scheme, Σ be a sigma protocol for relation $R(\Gamma, \Omega)$, and $\mathcal{H}$ be a random oracle. Suppose Ω satisfies strong unforgeability. We have* $\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H})$ *is an election scheme with internal authentication.*

4 Our construction ensures ballot secrecy

We adopt the definition of ballot secrecy for election schemes with external authentication (Ballot-Secrecy-Ext) by Smyth [32]. That definition appears to be the most suitable in the literature, because it detects the largest class of attacks [32, §7]. In particular, it detects attacks that arise when the adversary controls the bulletin board or the communications channel, whereas other definitions, e.g., [6, 8, 7, 35, 11, 12, 5], fail to detect such attacks. A definition of ballot secrecy for election schemes with internal authentication (Ballot-Secrecy-Int) can be derived from Smyth's definition by a natural, straightforward extension that takes credentials into account. Both definitions are presented in Appendix A. The definition of ballot secrecy we recall challenges an adversary, who has access to the election outcome, to distinguish between ballots.

We can prove that our construction ensures ballot secrecy (a formal proof of Theorem 2 appears in Appendix A), assuming the underlying election scheme satisfies ballot secrecy and the underlying sigma protocol satisfies special soundness and special honest verifier zero-knowledge.

Theorem 2. *Let Γ be an election scheme with external authentication, Ω be a digital signature scheme, Σ be a sigma protocol for relation $R(\Gamma, \Omega)$, and $\mathcal{H}$ be a random oracle. Suppose Γ satisfies* Ballot-Secrecy-Ext*, Σ satisfies special soundness and special honest verifier zero-knowledge, and Ω satisfies strong unforgeability. Election scheme with internal authentication* $\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H})$ *satisfies* Ballot-Secrecy-Int*.*

Proof sketch. Ballot secrecy of election scheme $\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H})$ follows from secrecy of the underlying scheme Γ, because signatures and non-interactive zero-knowledge proofs do not leak information. (Special soundness and special honest verifier zero-knowledge ensure proof system $\mathsf{FS}(\Sigma, \mathcal{H})$ is zero-knowledge [7].) □

We demonstrate applicability of Theorem 2 using a construction for election schemes from asymmetric encryption.[8]

Definition 4 (Enc2Vote [29]). Given a perfectly correct asymmetric encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ satisfying IND-CPA, election scheme with external authentication $\mathsf{Enc2Vote}(\Pi)$ is defined as follows:

- $\mathsf{Setup}(\kappa)$ computes $(pk, sk) \leftarrow \mathsf{Gen}(\kappa)$ and outputs $(pk, sk, poly(\kappa), |\mathfrak{m}|)$.
- $\mathsf{Vote}(pk, nc, v, \kappa)$ computes $b \leftarrow \mathsf{Enc}(pk, v)$ and outputs b if $1 \leq v \leq nc \leq |\mathfrak{m}|$ and $\perp$ otherwise.
- $\mathsf{Tally}(sk, nc, \mathfrak{bb}, \kappa)$ initialises vector $\mathbf{v}$ of length nc, computes **for** $b \in \mathfrak{bb}$ **do** $v \leftarrow \mathsf{Dec}(sk, b)$; **if** $1 \leq v \leq nc$ **then** $\mathbf{v}[v] \leftarrow \mathbf{v}[v] + 1$, and outputs $(\mathbf{v}, \epsilon)$.
- $\mathsf{Verify}(pk, nc, \mathfrak{bb}, \mathbf{v}, pf, \kappa)$ outputs 1.

Algorithm Setup requires *poly* to be a polynomial function, algorithms Setup and Vote require $\mathfrak{m} = \{1, \ldots, |\mathfrak{m}|\}$ to be the encryption scheme's plaintext space, and algorithm Tally requires ϵ to be a constant symbol.

Intuitively, given a non-malleable asymmetric encryption scheme Π,[9] $\mathsf{Enc2Vote}(\Pi)$ derives ballot secrecy from Π until tallying and tallying maintains ballot secrecy by returning only the number of votes for each candidate.

Proposition 3 ([29, 32]). *Let Π be an encryption scheme with perfect correctness. If Π satisfies IND-PA0, then election scheme with external authentication* $\mathsf{Enc2Vote}(\Pi)$ *satisfies* Ballot-Secrecy-Ext.

Hence, by Theorem 2, we have the following result.

Corollary 4. *Let Π be an asymmetric encryption scheme with perfect correctness, Ω be a digital signature scheme, Σ be a sigma protocol for relation $R(\mathsf{Enc2Vote}(\Pi), \Omega)$, and $\mathcal{H}$ be a random oracle. Suppose Π satisfies IND-PA0, Σ satisfies special soundness and special honest verifier zero-knowledge, and Ω satisfies strong unforgeability. Election scheme with internal authentication* $\mathsf{Ext2Int}(\mathsf{Enc2Vote}(\Pi), \Omega, \Sigma, \mathcal{H})$ *satisfies* Ballot-Secrecy-Int.

Clearly election scheme Enc2Vote does not satisfy universal verifiability, because it will accept any election outcome.

5 Our construction ensures election verifiability

We adopt definitions of individual (Exp-IV-Ext) and universal (Exp-UV-Ext) verifiability for election schemes with external authentication from Smyth, Frink, & Clarkson [36]. We also adopt their definitions of individual (Exp-IV-Int), universal

[8] We omit a formal definition of asymmetric encryption for brevity.

[9] We adopt the formal definition of comparison based non-malleability under chosen plaintext attack, which coincides with indistinguishability under a parallel chosen-ciphertext attack (IND-PA0) [3]. We omit formal security definitions for brevity.

(Exp-UV-Int) and eligibility (Exp-EV-Int) verifiability for schemes with internal authentication. Those definitions seem to be the most suitable in the literature, because they detect the largest class of attacks. In particular, they detect collusion and biasing attacks [36, §7], whereas other definitions, e.g., [20, 13, 21], fail to detect such attacks. The definitions are presented in Appendix B.

The definitions by Smyth, Frink, & Clarkson work as follows: Individual verifiability challenges the adversary to generate a collision from algorithm Vote. Universal verifiability challenges the adversary to concoct a scenario in which either: Verify accepts, but the election outcome is not correct, or Tally produces an election outcome that Verify rejects. Hence, universal verifiability requires algorithm Verify to accept if and only if the election outcome is correct. Finally, eligibility verifiability challenges an adversary, which can corrupt voters, to generate a valid ballot under a non-corrupt voter's private credential.

We can prove that our construction ensures election verifiability. Individual and eligibility verifiability of $\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H})$ follow from security of the underlying signature scheme, and universal verifiability follows from universal verifiability of the underlying election scheme Γ.

Theorem 5. *Let Γ be an election scheme with external authentication, Ω be a digital signature scheme, Σ be a sigma protocol for relation $R(\Gamma, \Omega)$, and $\mathcal{H}$ be a random oracle. Suppose Ω satisfies strong unforgeability, Σ satisfies special soundness and special honest verifier zero-knowledge, and Γ satisfies* Exp-UV-Ext. *Election scheme with internal authentication* $\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H})$ *satisfies* Exp-IV-Int, Exp-EV-Int, *and* Exp-UV-Int.

Proof sketch. Individual verifiability is satisfied because voters can check that their signatures appear on the bulletin board. Universal verifiability is satisfied because the underlying voting scheme does, and the properties of Ω and Σ ensure only authorised ballots are tallied. And eligibility verifiability is satisfied because anyone can check that signatures belong to registered voters. □

A formal proof of Theorem 5 follows immediately from our proofs of individual, universal and eligibility verifiability, which we defer to Appendix B (Lemmata 10–12).

We demonstrate applicability of our results for election schemes from nonces.

Definition 5 (Nonce [36]). Election scheme with external authentication Nonce is defined as follows:

- $\mathsf{Setup}(\kappa)$ outputs $(\perp, \perp, p_1(\kappa), p_2(\kappa))$, where p_1 and p_2 may be any polynomial functions.
- $\mathsf{Vote}(pk, nc, v, \kappa)$ selects a nonce r uniformly at random from $\mathbb{Z}_{2^\kappa}$ and outputs (r, v).
- $\mathsf{Tally}(sk, nc, \mathfrak{bb}, \kappa)$ computes a vector $\mathbf{v}$ of length nc, such that $\mathbf{v}$ is a tally of the votes on $\mathfrak{bb}$ for which the nonce is in $\mathbb{Z}_{2^\kappa}$, and outputs $(\mathbf{v}, \perp)$.
- $\mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathbf{v}, pf, \kappa)$ outputs 1 if $(\mathbf{v}, pf) = \mathsf{Tally}(\perp, nc, \mathfrak{bb}, \kappa)$, and 0 otherwise.

Intuitively, election scheme Nonce ensures verifiability because voters can use their nonce to check that their ballot is recorded (individual verifiability) and anyone can recompute the election outcome to check that it corresponds to votes expressed in recorded ballots (universal verifiability).

Proposition 6 ([36]). *Election scheme with external authentication* Nonce *satisfies* Exp-IV-Ext *and* Exp-UV-Ext*.*

Hence, by Theorem 5, we have the following result.

Corollary 7. *Let Ω be a digital signature scheme, Σ be a sigma protocol for relation R(Nonce, Ω), and $\mathcal{H}$ be a random oracle. Suppose Ω satisfies strong unforgeability and Σ satisfies special soundness and special honest verifier zero-knowledge. Election scheme with internal authentication* Ext2Int(Nonce, $\Omega, \Sigma, \mathcal{H}$) *satisfies* Exp-IV-Int*,* Exp-UV-Int*, and* Exp-EV-Int*.*

Clearly election scheme Nonce does not satisfy ballot secrecy.

6 Case study: A secret, verifiable election scheme with internal authentication

Helios is an open-source, web-based electronic voting system which has been used in binding elections. The International Association of Cryptologic Research has used Helios annually since 2010 to elect board members [4, 18], the ACM used Helios for their 2014 general election [40], the Catholic University of Louvain used Helios to elect the university president in 2009 [2], and Princeton University has used Helios since 2009 to elect student governments. Informally, Helios can be modelled as the following election scheme with external authentication:

- Setup generates a key pair for an asymmetric homomorphic encryption scheme, proves correct key generation in zero-knowledge, and outputs the public key coupled with the proof.
- Vote encrypts the vote, proves correct ciphertext construction and that the vote is selected from the sequence of candidates (both in zero-knowledge), and outputs the ciphertext coupled with the proof.
- Tally proceeds as follows. First, any ballots on the bulletin board for which proofs do not hold are discarded. Secondly, the ciphertexts in the remaining ballots are homomorphically combined, the homomorphic combination is decrypted to reveal the election outcome, and correctness of decryption is proved in zero-knowledge. Finally, the election outcome and proof of correct decryption are output.
- Verify recomputes the homomorphic combination, checks the proofs, and outputs 1 if these checks succeed and 0 otherwise.

The original scheme [2] is known to be vulnerable to attacks against ballot secrecy and verifiability,[10] and defences against those attacks have been proposed [15, 7, 35, 32]. We adopt the formal definition of a Helios variant by Smyth,

[10] Beyond secrecy and verifiability, attacks against eligibility are also known [38, 23].

Frink & Clarkson [36], which adopts non-malleable ballots [37, 32] and uses the Fiat–Shamir transformation with statements in hashes [7] to defend against those attacks. Henceforth, we write *Helios'16* to refer to that formalisation.

Using our construction we derive an election scheme with internal authentication from Helios'16 and prove privacy and verifiability using our results.

Theorem 8. *Let Ω be a digital signature scheme, Σ be a sigma protocol for relation $R(\textit{Helios'16}, \Omega)$, and $\mathcal{H}$ be a random oracle. Suppose Ω satisfies strong unforgeability and Σ satisfies special soundness and special honest verifier zero-knowledge. Election scheme with internal authentication* $\mathsf{Ext2Int}(\textit{Helios'16}, \Omega, \Sigma, \mathcal{H})$ *satisfies* Ballot-Secrecy-Int*,* Exp-IV-Int*,* Exp-UV-Int*, and* Exp-EV-Int*.*

Proof. Helios'16 satisfies Ballot-Secrecy-Ext, Exp-IV-Ext, and Exp-UV-Ext [36, 32], $\mathsf{FS}(\Sigma, \mathcal{H})$ satisfies zero-knowledge [7], and we conclude by Theorems 2 & 5. □

Comparison with Helios-C. Schemes derived from Helios using our construction are similar to Helios-C [13, 14]. Indeed, they use ballots that include a Helios ballot and a signature on that Helios ballot. The schemes derived by our construction also include proofs of correct construction, unlike Helios-C. We will see that this distinction is crucial to ensure ballot secrecy.

Cortier *et al.* [13, §5] analysed Helios-C using the definition of ballot secrecy by Bernhard et al. [7]. That definition assumes "ballots are *recorded-as-cast*, i.e., cast ballots are preserved with integrity through the ballot collection process" [32, §7]. Unfortunately, ballot secrecy is not satisfied without this assumption, because Helios-C uses malleable ballots.

Remark 9. Helios-C does not satisfy Ballot-Secrecy-Int.

Proof sketch. An adversary can observe and block a voter's ballot,[11] extract the underlying Helios ballot, sign that ballot, and post the ballot and signature on the bulletin board. The adversary can then exploit the relation between ballots to recover the voter's vote from the election outcome. (Cf. [15].) □

$\mathsf{Ext2Int}(\text{Helios'16}, \Omega, \Sigma, \mathcal{H})$ ballots extend non-malleable Helios'16 ballots with a signature and a proof demonstrating construction of both the embedded Helios'16 ballot and signature, thus, $\mathsf{Ext2Int}(\text{Helios'16}, \Omega, \Sigma, \mathcal{H})$ uses non-malleable ballots, so it is not similarly effected.

Beyond secrecy, Smyth, Frink & Clarkson [36] have shown that Helios-C does not satisfy Exp-UV-Int. Hence, we improve upon Helios-C by satisfying Ballot-Secrecy-Ext and Exp-UV-Int.

Our results can also be applied to the variant of Helios that applies a mixnet to encrypted votes and decrypts the mixed encrypted votes to reveal the outcome [1, 9], rather than homomorphically combining encrypted votes and decrypting the homomorphic combination to reveal the outcome. Tsoukalas *et al.* [41] released *Zeus* as a fork of Helios spliced with mixnet code to derive an implementation of

[11] Ballot blocking violates the recorded-as-cast assumption used in Cortier *et al.*'s proof.

that variant, and Yingtong Li released *helios-server-mixnet* as an extension of Zeus with threshold asymmetric encryption.[12] We could use our construction to derive an election scheme with internal authentication from the mixnet variant of Helios and use our privacy and verifiability results to prove security. Since the ideas remain the same, we do not pursue further details.

7 Conclusion

This work was initiated by a desire to eliminate trust assumptions placed upon the operators of external authentication services. Cortier *et al.* made progress in this direction with Helios-C, which builds upon Helios by signing ballots. We discovered that Helios-C does not satisfy ballot secrecy in the presence of an adversary that controls the bulletin board or the communication channel, and it is known that verifiability is not satisfied either. We realised that proving correct construction of both the Helios ballot and the signature suffices for non-malleability. This prompted the design of our construction and led to the accompanying security proofs that it produces voting systems satisfying ballot secrecy and verifiability. Finally, we demonstrated the applicability of our results by applying our construction to the Helios voting system. The next step would be to select a suitable sigma protocol and signature scheme to instantiate our construction concretely. And an interesting and useful direction for future work will be to consider, in general, the practical challenges of implementing our construction efficiently.

Acknowledgements

In the context of [36], Smyth conceived the fundamental ideas of our construction for election schemes with internal authentication. In addition, Smyth discovered that Helios-C does not satisfy ballot secrecy, whilst analysing election verifiability. Smyth and his co-authors, Frink & Clarkson, decided not to publish these results. This paper builds upon those unpublished results and we are grateful to Frink & Clarkson for their part in inspiring this line of work.

A Ballot privacy: Definitions and proofs

We recall Smyth's definition of ballot secrecy for election schemes with external authentication (Definition 6), and present a natural, straightforward extension of that definition to capture ballot secrecy for election schemes with internal authentication (Definition 7). Our definitions both use predicate *balanced* such that $\mathit{balanced}(\mathfrak{bb}, nc, B)$ holds when: for all votes $v \in \{1, \ldots, nc\}$ we have $|\{b \mid b \in \mathfrak{bb} \wedge \exists v_1 \,.\, (b, v, v_1) \in B\}| = |\{b \mid b \in \mathfrak{bb} \wedge \exists v_0 \,.\, (b, v_0, v) \in B\}|$. Intuitively, the definitions challenge an adversary to determine whether the left-right

[12] Smyth [34] shows that vulnerabilities in Helios cause vulnerabilities in implementations of the mixnet variant and proves verifiability is satisfied when a fix is applied.

oracle produces ballots for "left" or "right" inputs, by giving the adversary the oracle's outputs, as well as the election outcome and tallying proof. The definitions prevent the adversary from trivially distinguishing ballots by requiring predicate *balanced* to hold.

Definition 6 (Ballot-Secrecy-Ext [32]). Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ be an election scheme with external authentication, $\mathcal{A}$ be an adversary, κ be a security parameter, and $\mathsf{Ballot\text{-}Secrecy\text{-}Ext}(\Gamma, \mathcal{A}, \kappa)$ be the following game.

$\mathsf{Ballot\text{-}Secrecy\text{-}Ext}(\Gamma, \mathcal{A}, \kappa) =$

$(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
$nc \leftarrow \mathcal{A}(pk, \kappa)$;
$\beta \leftarrow_R \{0,1\}; B \leftarrow \emptyset$;
$\mathfrak{bb} \leftarrow \mathcal{A}^{\mathcal{O}}()$;
$(\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk, nc, \mathfrak{bb}, \kappa)$;
$g \leftarrow \mathcal{A}(\mathbf{v}, pf)$;
if $g = \beta \wedge \mathit{balanced}(\mathfrak{bb}, nc, B) \wedge 1 \leq nc \leq mc \wedge |\mathfrak{bb}| \leq mb$ **then**
  return 1
else
  return 0

Oracle $\mathcal{O}$ is defined as follows:[13]

- $\mathcal{O}(v_0, v_1)$ computes **if** $v_0, v_1 \in \{1, ..., nc\}$ **then** $b \leftarrow \mathsf{Vote}(pk, nc, v_\beta, \kappa); B \leftarrow B \cup \{(b, v_0, v_1)\}$; **return** b.

We say Γ satisfies Ballot-Secrecy-Ext, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function negl, such that for all security parameters κ, we have $\mathsf{Succ}(\mathsf{Ballot\text{-}Secrecy\text{-}Ext}(\Gamma, \mathcal{A}, \kappa)) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$.

Definition 7 (Ballot-Secrecy-Int). Let $\Gamma = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ be an election scheme with internal authentication, $\mathcal{A}$ be an adversary, κ be a security parameter, and $\mathsf{Ballot\text{-}Secrecy\text{-}Int}(\Gamma, \mathcal{A}, \kappa)$ be the following game.

$\mathsf{Ballot\text{-}Secrecy\text{-}Int}(\Gamma, \mathcal{A}, \kappa) =$

$(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
$nv \leftarrow \mathcal{A}(pk, \kappa)$;
for $1 \leq i \leq nv$ **do**
  $(pd_i, d_i) \leftarrow \mathsf{Register}(pk, \kappa)$;
$nc \leftarrow \mathcal{A}(pd_1, \ldots, pd_{nv})$;
$\beta \leftarrow_R \{0,1\}; B \leftarrow \emptyset; R \leftarrow \emptyset$;
$\mathfrak{bb} \leftarrow \mathcal{A}^{\mathcal{O}}()$;
$(\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk, nc, \mathfrak{bb}, \{pd_1, \ldots, pd_{nv}\}, \kappa)$;
$g \leftarrow \mathcal{A}(\mathbf{v}, pf)$;
if $g = \beta \wedge \mathit{balanced}(\mathfrak{bb}, nc, B) \wedge 1 \leq nc \leq mc \wedge |\mathfrak{bb}| \leq mb$ **then**
  return 1
else
  return 0

[13] Oracles may access game parameters, e.g., pk.

Oracle $\mathcal{O}$ is defined as follows:

- $\mathcal{O}(i, v_0, v_1)$ computes **if** $v_0, v_1 \in \{1, ..., nc\} \wedge i \notin R$ **then** $b \leftarrow \mathsf{Vote}(d_i, pk, nc, v_\beta, \kappa); B \leftarrow B \cup \{(b, v_0, v_1)\}; R \leftarrow R \cup \{i\}$; **return** b; and
- $\mathcal{O}(i)$ computes **if** $i \notin R$ **then** $R \leftarrow R \cup \{i\}$; **return** d_i.

We say Γ satisfies Ballot-Secrecy-Int, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function negl, such that for all security parameters κ, we have $\mathsf{Succ}(\text{Ballot-Secrecy-Int}(\Gamma, \mathcal{A}, \kappa)) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$.

Game Ballot-Secrecy-Int extends Ballot-Secrecy-Ext to take credentials into account. In particular, the challenger constructs nv credentials, where nv is chosen by the adversary. These credentials are used to construct ballots and for tallying. Public and private credentials are available to the adversary. Albeit, the oracle will only reveal a private credential if it has not used it to construct a ballot. Moreover, the oracle may only use a private credential to construct a ballot if it has not revealed it nor constructed a previous ballot with it.

Proof of Theorem 2. Suppose Ballot-Secrecy-Int is not satisfied by $\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H})$, i.e., there exists a adversary $\mathcal{A}$ such that for all negligible functions negl there exists a security parameter κ and $\mathsf{Succ}(\text{Ballot-Secrecy-Int}(\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H}), \mathcal{A}, \kappa)) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$. We construct an adversary $\mathcal{B}$ against Γ from $\mathcal{A}$.

Let $\Gamma = (\mathsf{Setup}_\Gamma, \mathsf{Vote}_\Gamma, \mathsf{Tally}_\Gamma, \mathsf{Verify}_\Gamma)$, $\Omega = (\mathsf{Gen}_\Omega, \mathsf{Sign}_\Omega, \mathsf{Verify}_\Omega)$, $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{Prove}_\Sigma, \mathsf{Verify}_\Sigma)$, and $\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$. By [7, Theorem 1], non-interactive proof system $(\mathsf{Prove}_\Sigma, \mathsf{Verify}_\Sigma)$ satisfies zero-knowledge, i.e., there exists a simulator for $(\mathsf{Prove}_\Sigma, \mathsf{Verify}_\Sigma)$. Let $\mathcal{S}$ be such a simulator. We define $\mathcal{B}$ as follows:

- $\mathcal{B}(pk, \kappa)$ computes $nv \leftarrow \mathcal{A}(pk, \kappa)$; **for** $1 \leq i \leq nv$ **do** $(pd_i, d_i) \leftarrow \mathsf{Register}(pk, \kappa)$; $nc \leftarrow \mathcal{A}(pd_1, \ldots, pd_{nv})$ and outputs nc.
- $\mathcal{B}()$ computes $R \leftarrow \emptyset; \mathfrak{bb} \leftarrow \mathcal{A}^{\mathcal{O}}(); \mathfrak{bb} \leftarrow \mathsf{auth}(\mathfrak{bb}, \{pd_1, \ldots, pd_{nv}\})$ and outputs $\mathfrak{bb}$, handling oracle calls from $\mathcal{A}$ as follows. Given an oracle call $\mathcal{O}(i, v_0, v_1)$ such that $v_0, v_1 \in \{1, ..., nc\} \wedge i \notin R$, adversary $\mathcal{B}$ computes $b \leftarrow \mathcal{O}(v_0, v_1); \sigma \leftarrow \mathsf{Sign}_\Omega(d_i, b); \tau \leftarrow \mathcal{S}((pk, b, \sigma, nc, \kappa), \kappa); R \leftarrow R \cup \{i\}$ and returns (pd_i, b, σ, τ) to $\mathcal{A}$. Moreover, given an oracle call $\mathcal{O}(i)$ such that $i \notin R$, adversary $\mathcal{B}$ computes $R \leftarrow R \cup \{i\}$ and returns d_i to $\mathcal{A}$.
- $\mathcal{B}(\mathbf{v}, pf)$ computes $g \leftarrow \mathcal{A}(\mathbf{v}, pf)$ and outputs g.

We prove that $\mathcal{B}$ wins Ballot-Secrecy-Ext against Γ.

Suppose (pk, sk, mb, mc) is an output of $\mathsf{Setup}_\Gamma(\kappa)$ and nc is an output of $\mathcal{B}(pk, \kappa)$. It is trivial to see that $\mathcal{B}(pk, \kappa)$ simulates $\mathcal{A}$'s challenger to $\mathcal{A}$. Let β be a bit. Suppose $\mathfrak{bb}$ is an output of $\mathcal{B}()$. Since $\mathcal{S}$ is a simulator for $(\mathsf{Prove}_\Sigma, \mathsf{Verify}_\Sigma)$, we have $\mathcal{B}()$ simulates $\mathcal{A}$'s challenger to $\mathcal{A}$. In particular, $\mathcal{B}()$ simulates oracle calls $\mathcal{O}(i, v_0, v_1)$. Indeed, adversary $\mathcal{B}$ computes $b \leftarrow \mathcal{O}(v_0, v_1); \sigma \leftarrow \mathsf{Sign}_\Omega(d_i, b); \tau \leftarrow \mathcal{S}((pk, b, \sigma, nc, \kappa), \kappa)$, which, by definition of $\mathcal{B}$'s oracle, is equivalent to $b \leftarrow \mathsf{Vote}_\Gamma(pk, nc, v_\beta, \kappa); \sigma \leftarrow \mathsf{Sign}_\Omega(d_i, b); \tau \leftarrow \mathcal{S}((pk, b, \sigma, nc, \kappa), \kappa)$. And $\mathcal{A}$'s oracle computes $b \leftarrow \mathsf{Vote}(d_i, pk, nc, v_\beta, \kappa)$, i.e., $b \leftarrow \mathsf{Vote}_\Gamma(pk, nc, v_\beta, \kappa; r); \sigma \leftarrow$

$\mathsf{Sign}_\Omega(d_i, b; r'); \tau \leftarrow \mathsf{Prove}_\Sigma((pk, b, \sigma, nc, \kappa), (v_\beta, r, d_i, r'), \kappa)$, where r and r' are coins chosen uniformly at random. Hence, computations of b, σ and τ by $\mathcal{B}$ and $\mathcal{A}$'s oracle are equivalent, with overwhelming probability. Suppose $(\mathbf{v}, pf)$ is an output of $\mathsf{Tally}_\Gamma(sk, \mathfrak{bb}, nc, \kappa)$ and g is an output of $\mathcal{B}(\mathbf{v}, pf)$. We have $\mathcal{B}(\mathbf{v}, pf)$ simulates $\mathcal{A}$'s challenger to $\mathcal{A}$, because outputs of $\mathsf{Tally}_\Gamma(sk', \mathsf{auth}(\mathfrak{bb}', L), nc', \kappa')$ and $\mathsf{Tally}(sk', nc', \mathfrak{bb}', L, \kappa')$ are indistinguishable for all sk', $\mathfrak{bb}'$, L, nc', and κ'. Indeed, Tally computes $(\mathbf{v}', pf') \leftarrow \mathsf{Tally}_\Gamma(sk', \mathsf{auth}(\mathfrak{bb}', L), nc', \kappa')$ and outputs $(\mathbf{v}', pf')$. Since adversary $\mathcal{B}$ simulates $\mathcal{A}$'s challenger, with overwhelming probability. It follows that $\mathcal{B}$ determines β correctly with the same success as $\mathcal{A}$ with overwhelming probability. Hence, $\mathcal{B}$ wins $\mathsf{Ballot\text{-}Secrecy\text{-}Ext}(\Gamma, \mathcal{A}, \kappa)$, with overwhelming probability, deriving a contradiction and concluding our proof. □

B Election verifiability: Definitions and proofs

B.1 Individual verifiability

Definition 8 (Exp-IV-Ext [36]). Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ be an election scheme with external authentication, $\mathcal{A}$ be an adversary, κ be a security parameter, and $\mathsf{Exp\text{-}IV\text{-}Ext}(\Gamma, \mathcal{A}, \kappa)$ be the following game. $\mathsf{Exp\text{-}IV\text{-}Ext}(\Gamma, \mathcal{A}, \kappa) =$

$(pk, nc, v, v') \leftarrow \mathcal{A}(\kappa)$;
$b \leftarrow \mathsf{Vote}(pk, nc, v, \kappa)$;
$b' \leftarrow \mathsf{Vote}(pk, nc, v', \kappa)$;
if $b = b' \wedge b \neq \perp \wedge b' \neq \perp$ **then**
  return 1
else
  return 0

We say Γ satisfies Exp-IV-Ext, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function negl, such that for all security parameters κ, we have $\mathsf{Succ}(\mathsf{Exp\text{-}IV\text{-}Ext}(\Gamma, \mathcal{A}, \kappa)) \leq \mathsf{negl}(\kappa)$.

Definition 9 (Exp-IV-Int [36]). Let $\Gamma = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ be an election scheme with external authentication, $\mathcal{A}$ be an adversary, κ be a security parameter, and $\mathsf{Exp\text{-}IV\text{-}Int}(\Pi, \mathcal{A}, \kappa)$ be the following game.

$\mathsf{Exp\text{-}IV\text{-}Int}(\Pi, \mathcal{A}, \kappa) =$

$(pk, nv) \leftarrow \mathcal{A}(\kappa)$;
for $1 \leq i \leq nv$ **do** $(pd_i, d_i) \leftarrow \mathsf{Register}(pk, \kappa)$;
$L \leftarrow \{pd_1, \ldots, pd_{nv}\}$;
$Crpt \leftarrow \emptyset$;
$(nc, v, v', i, j) \leftarrow \mathcal{A}^C(L)$;
$b \leftarrow \mathsf{Vote}(d_i, pk, nc, v, \kappa)$;
$b' \leftarrow \mathsf{Vote}(d_j, pk, nc, v', \kappa)$;
if $b = b' \wedge b \neq \perp \wedge b' \neq \perp \wedge i \neq j \wedge d_i \notin Crpt \wedge d_j \notin Crpt$ **then**
  return 1
else
  return 0

Oracle C is defined such that $C(i)$ computes $Crpt \leftarrow Crpt \cup \{d_i\}$ and outputs d_i, where $1 \leq i \leq nv$.

We say Γ satisfies $\mathsf{Exp\text{-}IV\text{-}Int}$, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function negl, such that for all security parameters κ, we have $\mathsf{Succ}(\mathsf{Exp\text{-}IV\text{-}Int}(\Pi, \mathcal{A}, \kappa)) \leq \mathsf{negl}(\kappa)$.

Lemma 10. *Let $\Gamma = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ be an election scheme with external authentication, $\Omega = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a digital signature scheme, Σ be a sigma protocol for relation $R(\Gamma, \Omega)$, and $\mathcal{H}$ be a hash function. Suppose Ω satisfies strong unforgeability. We have $\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H})$ satisfies $\mathsf{Exp\text{-}IV\text{-}Int}$.*

Proof. Suppose $\mathsf{Ext2Int}(\Gamma, \Pi, \Sigma, \mathcal{H})$ does not satisfy $\mathsf{Exp\text{-}IV\text{-}Int}$. Hence, there exists a PPT adversary $\mathcal{A}$, such that for all negligible functions negl, there exists a security parameter κ and $\mathsf{negl}(\kappa) < \mathsf{Succ}(\mathsf{Exp\text{-}IV\text{-}Int}(\mathsf{Ext2Int}(\Gamma, \Pi, \Sigma, \mathcal{H}), \mathcal{A}, \kappa))$. We construct the following adversary $\mathcal{B}$ against strong unforgeability from $\mathcal{A}$:

$\mathcal{B}(pd, \kappa) =$
  $(pk, nv) \leftarrow \mathcal{A}(\kappa)$;
  $i^* \leftarrow_R \{1, \ldots, nv\}$;
  for $i \in \{1, \ldots, nv\} \setminus \{i^*\}$ **do** $(pd_i, d_i) \leftarrow \mathsf{Register}(pk, \kappa)$;
  $(nc, v, v', j, k) \leftarrow \mathcal{A}^C(\{pd_1, \ldots, pd_{i^*-1}, pd, pd_{i^*+1}, \ldots, pd_{nv}\})$;
  if $i^* = k$ **then**
    $(pd_j, b, \sigma, \tau) \leftarrow \mathsf{Vote}(d_j, pk, nc, v, \kappa)$;
    return (σ, b);
  else if $i^* = j$ **then**
    $(pd_k, b, \sigma, \tau) \leftarrow \mathsf{Vote}(d_k, pk, nc, v', \kappa)$;
    return (σ, b);
  else
    abort;

where $C(i)$ outputs d_i if $i \neq i^*$ and aborts otherwise. We prove that $\mathcal{B}$ wins strong unforgeability against Ω.

Since adversary $\mathcal{B}$ chooses i^* uniformly at random and independently of adversary $\mathcal{A}$, and since $\mathcal{A}$ is a winning adversary, hence, does not corrupt at least two distinct credentials, we have that $\mathcal{B}$ aborts with a probability upper-bounded by $\frac{nv-2}{nv}$. Let us consider the probability that $\mathcal{B}$ wins, when there is no abort. Suppose (pd, d) is an output of $\mathsf{Gen}(\kappa)$, (pk, nv) is an output of $\mathcal{A}(\kappa)$, and i^* is chosen uniformly at random from $\{1, \ldots, nv\}$. Further suppose (pd_i, d_i) is an output of $\mathsf{Register}(pk, \kappa)$ for each $i \in \{1, \ldots, nv\} \setminus \{i^*\}$. It is straightforward to see that $\mathcal{B}$ simulates the challenger and oracle in $\mathsf{Exp\text{-}IV\text{-}Int}$ to $\mathcal{A}$. Suppose (nc, v, v', j, k) is an output of $\mathcal{A}^C(\{pd_1, \ldots, pd_{i^*-1}, pd, pd_{i^*+1}, \ldots, pd_{nv}\})$. Since $\mathcal{A}$ is a winning adversary, outputs of $\mathsf{Vote}(d_j, pk, nc, v, \kappa)$ and $\mathsf{Vote}(d_k, pk, nc, v', \kappa)$ collide with non-negligible probability. Hence, if $i^* = k$, then $\mathsf{Vote}(d_j, pk, nc, v, \kappa)$ outputs (pd_j, b, σ, τ) such that σ is a signature on b with respect to private key d_{i^*}, otherwise $(i^* = j)$, $\mathsf{Vote}(d_k, pk, nc, v', \kappa)$ outputs (pd_k, b, σ, τ) such that σ is a signature on b with respect to private key d_{i^*}. Thus, $\mathsf{Succ}(\mathsf{Exp\text{-}StrongSign}(\Gamma,$

$\mathcal{B}, \kappa))$ is at least $\frac{2}{nv} \cdot \mathsf{Succ}(\mathsf{Exp\text{-}IV\text{-}Int}(\mathsf{Ext2Int}(\Gamma, \Pi, \Sigma, \mathcal{H}), \mathcal{A}, \kappa))$, which is non-negligible. □

B.2 Universal verifiability

External authentication Algorithm Verify is required to accept iff the election outcome is correct. The notion of a correct outcome is captured using function *correct-outcome*, which is defined such that for all pk, nc, $\mathfrak{bb}$, κ, ℓ, and $v \in \{1, \ldots, nc\}$, we have $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)[v] = \ell$ iff $\exists^{=\ell} b \in \mathfrak{bb} \setminus \{\perp\} : \exists r : b = \mathsf{Vote}(pk, nc, v, \kappa; r)$,[14] and the produced vector is of length nc. Hence, component v of vector $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)$ equals ℓ iff there exist ℓ ballots on the bulletin board that are votes for candidate v. The function requires ballots to be interpreted for only one candidate, which can be ensured by injectivity.

The *if* requirement of universal verifiability is captured by Completeness, which stipulates that election outcomes produced by algorithm Tally will actually be accepted by algorithm Verify. And the *only if* requirement is captured by Soundness, which challenges an adversary to concoct a scenario in which algorithm Verify accepts, but the election outcome is not correct.

Definition 10 ([36]). An election scheme with external authentication (Setup, Vote, Tally, Verify) satisfies *Soundness*, if the scheme satisfies Injectivity [36] and for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function negl, such that for all security parameters κ, we have $\Pr[(pk, nc, \mathfrak{bb}, \mathbf{v}, pf) \leftarrow \mathcal{A}(\kappa); \mathbf{return}\ \mathbf{v} \neq \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa) \wedge \mathsf{Verify}(pk, nc, \mathfrak{bb}, \mathbf{v}, pf, \kappa) = 1] \leq \mathsf{negl}(\kappa)$.

An election scheme with external authentication satisfies Exp-UV-Ext, if Injectivity, Completeness and Soundness are satisfied, where formal definitions of Injectivity and Completeness appear in [36].

Internal authentication Function *correct-outcome* is now modified to tally only authorised ballots: let function *correct-outcome* now be defined such that for all pk, nc, $\mathfrak{bb}$, M, κ, ℓ, and $v \in \{1, \ldots, nc\}$, we have $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, M, \kappa)[v] = \ell$ iff $\exists^{=\ell} b \in \mathit{authorized}(pk, nc, (\mathfrak{bb} \setminus \{\perp\}), M, \kappa) : \exists d, r : b = \mathsf{Vote}(d, pk, nc, v, \kappa; r)$. A ballot is *authorised* if it is constructed with a private credential from M, and that private credential was not used to construct any other ballot on $\mathfrak{bb}$. Let *authorized* be defined as follows: $\mathit{authorized}(pk, nc, \mathfrak{bb}, M, \kappa) = \{b : \ b \in \mathfrak{bb} \wedge \exists pd, d, v, r : b = \mathsf{Vote}(d, pk, nc, v, \kappa; r) \wedge (pd, d) \in M \wedge \neg\exists b', v', r' : b' \in (\mathfrak{bb} \setminus \{b\}) \wedge b' = \mathsf{Vote}(d, pk, nc, v', \kappa; r')\}$.

Definition 11 ([36]). An election scheme with internal authentication (Setup, Register, Vote, Tally, Verify) satisfies *Soundness*, if the scheme satisfies Injectivity [36] and for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a

[14] Function *correct-outcome* uses a *counting quantifier* [31] denoted $\exists^{=}$. Predicate $(\exists^{=\ell} x : P(x))$ holds exactly when there are ℓ distinct values for x such that $P(x)$ is satisfied. Variable x is bound by the quantifier, whereas ℓ is free.

negligible function negl, such that for all security parameters κ, we have $\Pr[(pk, nv) \leftarrow \mathcal{A}(\kappa);\ \textbf{for}\ 1 \leq i \leq nv\ \textbf{do}\ (pd_i, d_i) \leftarrow \mathsf{Register}(pk, \kappa);\ \ L \leftarrow \{pd_1, \ldots, pd_{nv}\}; M \leftarrow \{(pd_1, d_1), \ldots, (pd_{nv}, d_{nv})\}; (\mathfrak{bb}, nc, \mathbf{v}, pf) \leftarrow \mathcal{A}(M); \textbf{return}\ \mathbf{v} \neq \textit{correct-outcome}(pk, nc, \mathfrak{bb}, M, \kappa) \wedge \mathsf{Verify}(pk, nc, \mathfrak{bb}, L, \mathbf{v}, pf, \kappa) = 1] \leq \mathsf{negl}(\kappa)$.

An election scheme with internal authentication satisfies Exp-UV-Int, if Injectivity, Completeness and Soundness are satisfied.

Lemma 11. *Let $\Gamma = (\mathsf{Setup}_\Gamma, \mathsf{Vote}_\Gamma, \mathsf{Tally}_\Gamma, \mathsf{Verify}_\Gamma)$ be an election scheme with external authentication, $\Omega = (\mathsf{Gen}_\Omega, \mathsf{Sign}_\Omega, \mathsf{Verify}_\Omega)$ be a perfectly correct digital signature scheme, Σ be a sigma protocol for relation $R(\Gamma, \Omega)$, and $\mathcal{H}$ be a random oracle. Moreover, let $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{Prove}_\Sigma, \mathsf{Verify}_\Sigma)$. Suppose Γ satisfies* Exp-UV-Ext*, Ω satisfies strong unforgeabilityand Σ satisfies perfect special soundness and special honest verifier zero-knowledge. Election scheme with internal authentication* $\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *satisfies* Exp-UV-Int.

Proof. We prove that $\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H})$ satisfies Injectivity, Completeness and Soundness: The proofs for Injectivity and Completeness are quite straightforward and can be found in our technical report [28].

Soundness. We prove that $\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H})$ satisfies Soundness by contradiction. Suppose $\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H})$ does not satisfy Soundness, i.e., there exists an adversary $\mathcal{A}$ such that for all negligible functions negl there exists a security parameter κ and the probability defined in Definition 11 is greater than $\mathsf{negl}(\kappa)$. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that wins the Soundness game against Γ.

$\mathcal{B}(\kappa) =$
$(pk, nv) \leftarrow \mathcal{A}(\kappa)$;
for $1 \leq i \leq nv$ **do**
  $(pd_i, d_i) \leftarrow \mathsf{Register}(pk, \kappa)$;
$L = \{pd_1, \ldots, pd_{nv}\}$;
$M \leftarrow \{(pd_1, d_1), \ldots, (pd_{nv}, d_{nv})\}$;
$(\mathfrak{bb}, nc, \mathbf{v}, pf) \leftarrow \mathcal{A}(M)$;
return $(pk, nc, \mathsf{auth}(\mathfrak{bb}, L), \mathbf{v}, pf)$

We prove that $\mathcal{B}$ wins the Soundness game against Γ.

Suppose (pk, nv) is an output of $\mathcal{A}(\kappa)$ and $(pd_1, d_1), \ldots, (pd_{nv}, d_{nv})$ are outputs of $\mathsf{Register}(pk, \kappa)$. Let $L = \{pd_1, \ldots, pd_{nv}\}$ and $M = \{(pd_1, d_1), \ldots, (pd_{nv}, d_{nv})\}$. Suppose $(\mathfrak{bb}, nc, \mathbf{v}, pf)$ is an output of $\mathcal{A}(M)$. Further suppose $(pk, nc, \mathsf{auth}(\mathfrak{bb}, L), \mathbf{v}, pf)$ is an output of $\mathcal{B}(\kappa)$. Since $\mathcal{A}$ is a winning adversary, we have $\mathsf{Verify}(pk, nc, \mathfrak{bb}, L, \mathbf{v}, pf, \kappa) = 1$, with non-negligible probability. By inspection of algorithm Verify, we have $\mathsf{Verify}(pk, nc, \mathfrak{bb}, L, \mathbf{v}, pf, \kappa) = 1$ implies $\mathsf{Verify}_\Gamma(pk, \mathsf{auth}(\mathfrak{bb}, L), nc, \mathbf{v}, pf, \kappa) = 1$. Hence, it remains to show $\mathbf{v} \neq \textit{correct-outcome}(pk, nc, \mathsf{auth}(\mathfrak{bb}, L), \kappa)$, with probability greater than $\mathsf{negl}(\kappa)$.

By definition of function *correct-outcome*, we have $\mathbf{v}$ is a vector of length nc such that

$$\begin{aligned} &\mathit{correct\text{-}outcome}(pk, nc, \mathsf{auth}(\mathfrak{bb}, L), \kappa)[v] = \ell \\ &\quad \Leftrightarrow \exists^{=\ell} b \in \mathsf{auth}(\mathfrak{bb}, L) \setminus \{\bot\} : \exists r : b = \mathsf{Vote}(pk, nc, v, \kappa; r) \end{aligned}$$

Since $\mathcal{A}$ is a winning adversary, it suffices to derive

$$\begin{aligned} &\Leftrightarrow \exists^{=\ell} b \in \mathit{authorized}(pk, nc, (\mathfrak{bb} \setminus \{\bot\}), M, \kappa) \\ &\qquad\qquad : \exists d, r : b = \mathsf{Vote}(d, pk, nc, v, \kappa; r) \end{aligned} \tag{1}$$

Let set $\mathit{auth}^*(pk, nc, \mathfrak{bb}, M, \kappa) = \{b^* | (pd, b^*, \sigma, \tau) \in \mathit{authorized}(pk, nc, \mathfrak{bb}, M, \kappa)\}$. To prove (1), it suffices to show $\mathsf{auth}(\mathfrak{bb}, L) \setminus \{\bot\} = \mathit{auth}^*(pk, nc, \mathfrak{bb}, M, \kappa) \setminus \{\bot\}$, since this would imply that *correct-outcome* is computed on sets of corresponding ballots in both the external and internal authentication setting.

- $\mathit{auth}^*(pk, nc, \mathfrak{bb}, M, \kappa) \setminus \{\bot\} \subseteq \mathsf{auth}(\mathfrak{bb}, L) \setminus \{\bot\}$
  If $b^* \in \mathit{auth}^*(pk, nc, \mathfrak{bb}, M, \kappa)$, then $b^* \neq \bot$ and there exists $b \in \mathit{authorized}(pk, nc, \mathfrak{bb}, M, \kappa)$ such that (i) $b \in \mathfrak{bb}$; (ii) $\exists pd, d, v, r, r', r'' : b = (pd, b^*, \sigma, \tau)$, $b^* = \mathsf{Vote}_\Gamma(pk, nc, v, \kappa; r)$, $\sigma = \mathsf{Sign}_\Omega(d, b^*; r')$, and $\tau = \mathsf{Prove}_\Sigma((pk, b^*, \sigma, nc, \kappa), (v, r, d, r'), \kappa; r'')$, which – by correctness of Ω and completeness of Σ – implies $\mathsf{Verify}_\Omega(pd, b^*, \sigma) = 1$ and $\mathsf{Verify}_\Sigma((pk, b^*, nc, \kappa), \tau, \kappa)) = 1$; (iii) $(pd, d) \in M$, which implies $pd \in L$ by construction; and (iv) $\neg \exists b', v', r, r', r'' : b' \in (\mathfrak{bb} \setminus \{b\}) \wedge b' = (pd, b^{*'}, \sigma', \tau')$, $b^{*'} = \mathsf{Vote}_\Gamma(pk, nc, v', \kappa; r)$, $\sigma' = \mathsf{Sign}_\Omega(d, b^{*'}; r')$, and $\tau' = \mathsf{Prove}_\Sigma((pk, b^{*'}, \sigma', nc, \kappa), (v', r, d, r'), \kappa; r'')$, which, by correctness of Ω, implies $\mathsf{Verify}_\Omega(pd, b^{*'}, \sigma') = 1$. It follows by (i)–(iv) that $b^* \in \mathit{auth}^*(pk, nc, \mathfrak{bb}, M, \kappa)$ implies $b^* \in \mathsf{auth}(\mathfrak{bb}, L) \setminus \{\bot\}$.
- $\mathsf{auth}(\mathfrak{bb}, L) \setminus \{\bot\} \subseteq \mathit{auth}^*(pk, nc, \mathfrak{bb}, M, \kappa) \setminus \{\bot\}$
  If $b^* \in \mathsf{auth}(\mathfrak{bb}, L) \setminus \{\bot\}$, then $b^* \neq \bot$ such that (i) $(pd, b^*, \sigma, \tau) \in \mathfrak{bb}$; (ii) $\mathsf{Verify}_\Omega(pd, b^*, \sigma) = 1$ and $\mathsf{Verify}_\Sigma((pk, b^*, nc, \kappa), \tau, \kappa)) = 1$, which – by the security of Ω and Σ – implies $\exists pd, d, v, r, r', r'' : b^* = \mathsf{Vote}_\Gamma(pk, nc, v, \kappa; r)$, $\sigma = \mathsf{Sign}_\Omega(d, b^*; r')$, and $\tau = \mathsf{Prove}_\Sigma((pk, b^*, \sigma, nc, \kappa), (v, r, d, r'), \kappa; r'')$. Indeed, suppose this is not true, i.e., such values do not exist. Then (b^*, σ) and $((pk, b^*, nc, \kappa), \tau)$ could be used by adversaries to break the unforgeability property of Ω and the special soundness and special honest verifier zero-knowledge property of Σ, respectively. Furthermore, we have (iii) $pd \in L$, which implies $(pd, d) \in M$ by construction; and (iv) $b' = (pd, b^{*'}, \sigma', \tau') \notin (\mathfrak{bb} \setminus \{(pd, b^*, \sigma, \tau)\}) \wedge \mathsf{Verify}_\Omega(pd, b^{*'}, \sigma') = 1$, which implies $\neg \exists b', v', r, r', r'' : b' \in (\mathfrak{bb} \setminus \{b\}) \wedge b' = (pd, b^{*'}, \sigma', \tau')$, $b^{*'} = \mathsf{Vote}_\Gamma(pk, nc, v', \kappa; r)$, $\sigma' = \mathsf{Sign}_\Omega(d, b^{*'}; r')$, and $\tau' = \mathsf{Prove}_\Sigma((pk, b^{*'}, \sigma', nc, \kappa), (v', r, d, r'), \kappa; r'')$, as per definition of *authorized*, concluding our proof. □

B.3 Eligibility verifiability

Definition 12 (Eligibility verifiability[36]). Let $\Gamma = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ be an election scheme with internal authentication, $\mathcal{A}$ be an adversary, κ be a security parameter, and $\mathsf{Exp\text{-}EV\text{-}Int}(\Pi, \mathcal{A}, \kappa)$ be the following game.

$\mathsf{Exp\text{-}EV\text{-}Int}(\Pi, \mathcal{A}, \kappa) =$

$(pk, nv) \leftarrow \mathcal{A}(\kappa)$;
for $1 \leq i \leq nv$ **do** $(pd_i, d_i) \leftarrow \mathsf{Register}(pk, \kappa)$;
$L \leftarrow \{pd_1, \ldots, pd_{nv}\}$;
$Crpt \leftarrow \emptyset$; $Rvld \leftarrow \emptyset$;
$(nc, v, i, b) \leftarrow \mathcal{A}^{C,R}(L)$;
if $\exists r : b = \mathsf{Vote}(d_i, pk, nc, v, \kappa; r) \wedge b \neq \perp \wedge b \notin Rvld \wedge d_i \notin Crpt$ **then**
  return 1
else
  return 0

Oracle C is the same oracle as in $\mathsf{Exp\text{-}IV\text{-}Int}$, and oracle R is defined such that $R(i, v, nc)$ computes $b \leftarrow \mathsf{Vote}(d_i, pk, nc, v, k)$; $Rvld \leftarrow Rvld \cup \{b\}$ and outputs b.

We say Γ satisfies $\mathsf{Exp\text{-}EV\text{-}Int}$, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function negl, such that for all security parameters κ, we have $\mathsf{Succ}(\mathsf{Exp\text{-}EV\text{-}Int}(\Pi, \mathcal{A}, \kappa)) \leq \mathsf{negl}(\kappa)$.

Lemma 12. *Let* $\Gamma = (\mathsf{Setup}_\Gamma, \mathsf{Vote}_\Gamma, \mathsf{Tally}_\Gamma, \mathsf{Verify}_\Gamma)$ *be an election scheme with external authentication,* $\Omega = (\mathsf{Gen}_\Omega, \mathsf{Sign}_\Omega, \mathsf{Verify}_\Omega)$ *be a digital signature scheme,* Σ *be a sigma protocol for relation* $R(\Gamma, \Omega)$*, and* $\mathcal{H}$ *be a hash function. Suppose* Σ *satisfies special soundness and special honest verifier zero-knowledge, and* Ω *satisfies strong unforgeability. Election scheme with internal authentication* $\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *satisfies* $\mathsf{Exp\text{-}EV\text{-}Int}$.

Proof. Suppose $\mathsf{Ext2Int}(\Gamma, \Omega, \Sigma, \mathcal{H})$ does not satisfy $\mathsf{Exp\text{-}EV\text{-}Int}$, i.e., there exists an adversary $\mathcal{A}$ such that for all negligible functions negl there exists a security parameter κ and $\mathsf{Succ}(\mathsf{Exp\text{-}EV\text{-}Int}(\Pi, \mathcal{A}, \kappa)) > \mathsf{negl}(\kappa)$. We construct the following adversary $\mathcal{B}$ against the strong unforgeability of Ω from $\mathcal{A}$.

$\mathcal{B}(pd, \kappa) =$

$(pk, nv) \leftarrow \mathcal{A}(\kappa)$;
$i^* \leftarrow_R \{1, ..., nv\}$;
for $i \in \{1, ..., nv\} \setminus \{i^*\}$ **do** $(pd_i, d_i) \leftarrow \mathsf{Register}(pk, \kappa)$;
$Rvld \leftarrow \emptyset$; $Crpt \leftarrow \emptyset$;
$(nc, v, i, b) \leftarrow \mathcal{A}^{C,R}(\{pd_1, \ldots, pd_{i^*-1}, pd, pd_{i^*+1}, \ldots, pd_{nv}\})$;
if $b[1] = pd$ **then**
  return $(b[2], b[3])$;
else
  abort;

where oracle calls are handled as follows:

- $C(i)$ computes $Crpt \leftarrow Crpt \cup \{d_i\}$ and returns d_i if $i \neq i^*$, and aborts otherwise.
- $R(i, v, nc)$ distinguishes two cases: If $i = i^*$, then $\mathcal{B}$ computes $b \leftarrow \mathsf{Vote}_\Gamma(pk, nc, v, \kappa); \sigma \leftarrow \mathcal{O}(b); \tau \leftarrow \mathcal{S}((pk, b, \sigma, nc, \kappa), \kappa)$, computes $Rvld \leftarrow Rvld \cup \{(pd, b, \sigma, \tau)\}$, and returns (pd, b, σ, τ), where $\mathcal{S}$ is a simulator for $\mathsf{FS}(\Sigma, \mathcal{H})$

that exists by [7, Theorem 1]. Otherwise, $\mathcal{B}$ computes $b \leftarrow \mathsf{Vote}(d_i, pk, nc, v, \kappa)$, $Rvld \leftarrow Rvld \cup \{b\}$ and returns b.

We prove that $\mathcal{B}$ wins the strong unforgeability game against Ω.

Let κ be a security parameter. Suppose (pd, d) is an output of $\mathsf{Gen}(\kappa)$ and (pk, nv) is an output of $\mathcal{A}(\kappa)$. Let i^* be an integer chosen uniformaly at random from $\{1, ..., nv\}$. Suppose (pd_i, d_i) is an output of $\mathsf{Register}(pk, \kappa)$, for each $i \in \{1, ..., nv\} \setminus \{i^*\}$. Let us consider an execution of $\mathcal{A}(\{pd_1, \ldots, pd_{i^*-1}, pd, pd_{i^*+1}, \ldots, pd_{nv}\})$. Let (nc, v, i, b) be the output of $\mathcal{A}$. By definition of algorithm $\mathsf{Register}$, it is trivial to see that $\mathcal{B}$ simulates $\mathcal{A}$'s challenger to $\mathcal{A}$. Moreover, $\mathcal{B}$ simulates oracle C to $\mathcal{A}$, except when $\mathcal{B}$ aborts. Furthermore, $\mathcal{B}$ simulates oracle R to $\mathcal{A}$ as well. In particular, simulator $\mathcal{S}$ produces proofs that are indistinguishable from proofs constructed by non-interactive proof system $\mathsf{FS}(\Sigma, \mathcal{H})$.

We denote by Good the event that $i = i^*$. Now, let us assess $\mathcal{B}$'s probability not to abort, to determine the success probability of $\mathcal{B}$. Since $\mathcal{A}$ is not allowed to corrupt the credential it finally outputs (as $\mathcal{A}$ is a winning adversary, $d_i \notin Crpt$ must hold), a sufficient condition for $\mathcal{B}$ not to be asked for the unknown private credential d_i is to be lucky when drawing $i^* \leftarrow \{1, ..., nv\}$ at random and have event Good occurring.

This is the case with probability $\Pr[\mathsf{Good}] = \frac{1}{nv}$ since the choice of i^* is completely independent of $\mathcal{A}$'s view. Therefore we have $\mathsf{Succ}(\mathsf{Exp\text{-}EV\text{-}Int}(\Pi, \mathcal{A}, \kappa)) \leq nv \cdot \mathsf{Succ}(\mathsf{Exp\text{-}StrongSign}(\Omega, \mathcal{B}, k))$.

□

References

1. Adida, B.: Helios: Web-based Open-Audit Voting. In: USENIX Security'08: 17th USENIX Security Symposium. pp. 335–348. USENIX Association (2008)
2. Adida, B., Marneffe, O., Pereira, O., Quisquater, J.: Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios. In: EVT/WOTE'09: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections. USENIX Association (2009)
3. Bellare, M., Sahai, A.: Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In: CRYPTO'99: 19th International Cryptology Conference. LNCS, vol. 1666, pp. 519–536. Springer (1999)
4. Benaloh, J., Vaudenay, S., Quisquater, J.: Final Report of IACR Electronic Voting Committee. International Association for Cryptologic Research. `iacr.org/elections/eVoting/finalReportHelios_2010-09-27.html` (Sept 2010)
5. Bernhard, D., Cortier, V., Galindo, D., Pereira, O., Warinschi, B.: SoK: A comprehensive analysis of game-based ballot privacy definitions. In: S&P'15: 36th Security and Privacy Symposium. IEEE Computer Society (2015)
6. Bernhard, D., Cortier, V., Pereira, O., Smyth, B., Warinschi, B.: Adapting Helios for provable ballot privacy. In: ESORICS'11: 16th European Symposium on Research in Computer Security. LNCS, vol. 6879, pp. 335–354. Springer (2011)
7. Bernhard, D., Pereira, O., Warinschi, B.: How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In: ASIACRYPT'12: 18th International Conference on the Theory and Application of Cryptology and Information Security. LNCS, vol. 7658, pp. 626–643. Springer (2012)

8. Bernhard, D., Pereira, O., Warinschi, B.: On Necessary and Sufficient Conditions for Private Ballot Submission. Cryptology ePrint Archive, Report 2012/236 (version 20120430:154117b) (2012)
9. Bulens, P., Giry, D., Pereira, O.: Running Mixnet-Based Elections with Helios. In: EVT/WOTE'11: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections. USENIX Association (2011)
10. Bundesverfassungsgericht (Germany's Federal Constitutional Court): Use of voting computers in 2005 Bundestag election unconstitutional (March 2009), press release 19/2009
11. Cortier, V., Galindo, D., Glondu, S., Izabachene, M.: A generic construction for voting correctness at minimum cost - Application to Helios. Cryptology ePrint Archive, Report 2013/177 (version 20130521:145727) (2013)
12. Cortier, V., Galindo, D., Glondu, S., Izabachene, M.: Distributed elgamal à la pedersen: Application to helios. In: WPES'13: Workshop on Privacy in the Electronic Society. pp. 131–142. ACM Press (2013)
13. Cortier, V., Galindo, D., Glondu, S., Izabachène, M.: Election Verifiability for Helios under Weaker Trust Assumptions. In: ESORICS'14: 19th European Symposium on Research in Computer Security. LNCS, vol. 8713, pp. 327–344. Springer (2014)
14. Cortier, V., Galindo, D., Glondu, S., Izabachène, M.: Election Verifiability for Helios under Weaker Trust Assumptions. Tech. Rep. RR-8555, INRIA (2014)
15. Cortier, V., Smyth, B.: Attacking and fixing Helios: An analysis of ballot secrecy. In: CSF'11: 24th Computer Security Foundations Symposium. pp. 297–311. IEEE Computer Society (2011)
16. Gonggrijp, R., Hengeveld, W.J.: Studying the Nedap/Groenendaal ES3B Voting Computer: A Computer Security Perspective. In: EVT'07: Electronic Voting Technology Workshop. USENIX Association (2007)
17. Gumbel, A.: Steal This Vote: Dirty Elections and the Rotten History of Democracy in America. Nation Books (2005)
18. Haber, S., Benaloh, J., Halevi, S.: The Helios e-Voting Demo for the IACR. International Association for Cryptologic Research. `iacr.org/elections/eVoting/heliosDemo.pdf` (May 2010)
19. Jones, D.W., Simons, B.: Broken Ballots: Will Your Vote Count?, CSLI Lecture Notes, vol. 204. Center for the Study of Language and Information, Stanford University (2012)
20. Juels, A., Catalano, D., Jakobsson, M.: Coercion-Resistant Electronic Elections. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y. (eds.) Towards Trustworthy Elections: New Directions in Electronic Voting, LNCS, vol. 6000, pp. 37–63. Springer (2010)
21. Kiayias, A., Zacharias, T., Zhang, B.: End-to-end verifiable elections in the standard model. In: EUROCRYPT'15: 34th International Conference on the Theory and Applications of Cryptographic Techniques. LNCS, vol. 9057, pp. 468–498. Springer (2015)
22. Lijphart, A., Grofman, B.: Choosing an electoral system: Issues and Alternatives. Praeger (1984)
23. Meyer, M., Smyth, B.: An attack against the helios election system that exploits re-voting. arXiv, Report 1612.04099 (2017)
24. Organization for Security and Co-operation in Europe: Document of the Copenhagen Meeting of the Conference on the Human Dimension of the CSCE (1990)
25. Organization of American States: American Convention on Human Rights, "Pact of San Jose, Costa Rica" (1969)

26. Pereira, O.: Internet Voting with Helios. In: Real-World Electronic Voting: Design, Analysis and Deployment, chap. 11. CRC Press (2016)
27. Quaglia, E.A., Smyth, B.: A short introduction to secrecy and verifiability for elections. arXiv, Report 1702.03168 (2017)
28. Quaglia, E.A., Smyth, B.: Authentication with weaker trust assumptions for voting systems. https://bensmyth.com/publications/2018-voting-authentication/ (2018)
29. Quaglia, E.A., Smyth, B.: Secret, verifiable auctions from elections. Cryptology ePrint Archive, Report 2015/1204 (2018)
30. Saalfeld, T.: On Dogs and Whips: Recorded Votes. In: Döring, H. (ed.) Parliaments and Majority Rule in Western Europe, chap. 16. St. Martin's Press (1995)
31. Schweikardt, N.: Arithmetic, first-order logic, and counting quantifiers. Search Results ACM Transactions on Computational Logic 6(3), 634–671 (Jul 2005)
32. Smyth, B.: Ballot secrecy: Security definition, sufficient conditions, and analysis of Helios. Cryptology ePrint Archive, Report 2015/942 (2018)
33. Smyth, B.: A foundation for secret, verifiable elections (2018), https://bensmyth.com/publications/2018-secrecy-verifiability-elections-tutorial/
34. Smyth, B.: Verifiability of Helios Mixnet. In: Voting'18: 3rd Workshop on Advances in Secure Electronic Voting. LNCS, Springer (2018)
35. Smyth, B., Bernhard, D.: Ballot secrecy and ballot independence coincide. In: ESORICS'13: . LNCS, vol. 8134, pp. 463–480. Springer (2013)
36. Smyth, B., Frink, S., Clarkson, M.R.: Election Verifiability: Cryptographic Definitions and an Analysis of Helios, Helios-C, and JCJ. Cryptology ePrint Archive, Report 2015/233 (2017)
37. Smyth, B., Hanatani, Y., Muratani, H.: NM-CPA secure encryption with proofs of plaintext knowledge. In: IWSEC'15: . LNCS, vol. 9241. Springer (2015)
38. Smyth, B., Pironti, A.: Truncating TLS Connections to Violate Beliefs in Web Applications. In: WOOT'13: 7th USENIX Workshop on Offensive Technologies. USENIX Association (2013), first appeared at Black Hat USA 2013
39. Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., MacAlpine, M., Halderman, J.A.: Security Analysis of the Estonian Internet Voting System. In: CCS'14: 21st ACM Conference on Computer and Communications Security. pp. 703–715. ACM Press (2014)
40. Staff, C.: ACM's 2014 General Election: Please Take This Opportunity to Vote. Communications of the ACM 57(5), 9–17 (May 2014)
41. Tsoukalas, G., Papadimitriou, K., Louridas, P., Tsanakas, P.: From Helios to Zeus. Journal of Election Technology and Systems 1(1) (2013)
42. United Nations: Universal Declaration of Human Rights (1948)
43. Wolchok, S., Wustrow, E., Halderman, J.A., Prasad, H.K., Kankipati, A., Sakhamuri, S.K., Yagati, V., Gonggrijp, R.: Security Analysis of India's Electronic Voting Machines. In: CCS'10: 17th ACM Conference on Computer and Communications Security. pp. 1–14. ACM Press (2010)
44. Wolchok, S., Wustrow, E., Isabel, D., Halderman, J.A.: Attacking the Washington, D.C. Internet Voting System. In: FC'12: 16th International Conference on Financial Cryptography and Data Security. LNCS, vol. 7397, pp. 114–128. Springer (2012)