Ballot secrecy: Security definition, sufficient conditions, and analysis of Helios

Ben Smyth

School of Computer Science, University of Birmingham, UK

Abstract. We propose a definition of ballot secrecy as an indistinguishability game in the computational model of cryptography. Our definition improves upon earlier definitions to ensure ballot secrecy is preserved in the presence of an adversary that controls ballot collection. We also propose a definition of ballot independence as an adaptation of an indistinguishability game for asymmetric encryption. We prove relations between our definitions. In particular, we prove ballot independence is sufficient for ballot secrecy in voting systems with zero-knowledge tallying proofs. Moreover, we prove that building systems from non-malleable asymmetric encryption schemes suffices for ballot secrecy, thereby eliminating the expense of ballot-secrecy proofs for a class of encryption-based voting systems. We demonstrate applicability of our results by analysing the Helios voting system and its mixnet variant. Our analysis reveals that Helios does not satisfy ballot secrecy in the presence of an adversary that controls ballot collection. The vulnerability cannot be detected by earlier definitions of ballot secrecy, because they do not consider such adversaries. We adopt non-malleable ballots as a fix and prove that the fixed system satisfies ballot secrecy.

Keywords: Anonymity, Democracy, Elections, Helios, Independence, Non-Malleability, Privacy, Secrecy, Voting

1. Introduction

An election is a decision-making procedure to choose representatives [1–4]. Choices should be made by voters with equal influence, and this must be ensured by voting systems, as prescribed by the United Nations [5], the Organisation for Security & Cooperation in Europe [6], and the Organization of American States [7]. Historically, “Americans [voted] with their voices – *viva voce* – or with their hands or with their feet. Yea or nay. Raise your hand. All in favor of Jones, stand on this side of the town common; if you support Smith, line up over there" [8]. Thus, ensuring that only voters voted and did so with equal influence was straightforward. Indeed, the election outcome could be determined by anyone present, simply by considering at most one vote per voter and disregarding non-voters. Yet, voting systems must also ensure choices are made freely, as prescribed by the aforementioned organisations [5–7]. Mill eloquently argues that choices cannot be expressed freely in public: “The unfortunate voter is in the power of some opulent man; the opulent man informs him how he must vote. Conscience, virtue, moral obligation, religion, all cry to him, that he ought to consult his own judgement, and faithfully follow its dictates. The consequences of pleasing, or offending the opulent man, stare him in the face...the moral obligation is disregarded, a faithless, a prostitute, a pernicious vote is given" [9].

The need for free-choice started a movement towards voting as a private act, i.e., “when numerous social constraints in which citizens are routinely and universally enmeshed – community of religious allegiances, the patronage of big men, employers or notables, parties, ‘political machines’ – are kept at bay," and “this idea has become the current *doxa* of democracy-builders worldwide" [10]. The most widely used embodiment of this idea is the Australian system, which demands that votes be marked on uniform ballots in polling booths and deposited into ballot boxes. Uniformity is intended to enable free-choice during distribution, collection and tallying of ballots, and the isolation of polling booths is intended to facilitate free-choice whilst marking.[1] Moreover, the Australian system can assure that only voters vote and do so with equal influence. Indeed, observers can check that ballots are only distributed to voters and at most one ballot is deposited by each voter. Furthermore, observers can check that spoiled ballots are discarded and that votes expressed in the remaining ballots correspond to the election outcome. Albeit, assurance is limited by an observer's ability to monitor [12–14] and the ability to transfer that assurance is limited to the observer's "good word or sworn testimony" [15].

Since the paper-based Australian system's introduction, electronic voting systems have emerged. Unfortunately, these electronic systems are routinely broken in ways that violate free-choice, e.g., [16–21], or permit undue influence, e.g., [16, 18, 22–24]. Breaks can be avoided by proving that systems satisfy formal notions of voters voting freely and of detecting undue influence. Universal verifiability formalises the latter notion, and we propose a definition of ballot secrecy that formalises the former. Our definition is presented in the computational, game-based model of cryptography, whereby a benign challenger, a malicious adversary and a voting system engage in a series of interactions which task the adversary to break security.

Ballot secrecy formalises a notion of free-choice,[2] assuming voters' ballots are constructed and tallied in the prescribed manner.

- Ballot secrecy. A voter's vote is not revealed to anyone.

We capture ballot secrecy as a game that proceeds as follows. First, the adversary picks a pair of votes v_0 and v_1. Secondly, the challenger constructs a ballot for vote v_β (in the manner prescribed by the voting system), where β is a bit chosen uniformly at random. That ballot is given to the adversary. The adversary and challenger repeat the process to construct further ballots, using the same bit β. Thirdly, the adversary constructs a set of ballots, which may include ballots constructed by the adversary and ballots constructed by the challenger. Thus, the game captures a setting where the adversary casts ballots on behalf of some voters and controls the votes cast by the remaining voters. Fourthly, the challenger tallies the set of ballots (in the manner prescribed by the voting system) to determine the election outcome, which is given to the adversary. Finally, the adversary is tasked with determining if $\beta = 0$ or $\beta = 1$. To avoid trivial distinctions, we require that the aforementioned votes (controlled by the adversary) remain constant regardless of whether $\beta = 0$ or $\beta = 1$. If the adversary wins the game, then a voter's vote can be revealed, otherwise, it cannot, i.e., the voting system provides ballot secrecy. Our game improves upon games by Bernhard *et al.* [25–29] to ensure ballot secrecy is preserved in the presence of an adversary that controls ballot collection (i.e., the bulletin board and the communication channel), whereas games by Bernhard *et al.* do not.

Beyond ballot secrecy, voting systems should satisfy properties including universal verifiability, which requires systems to produce evidence that can be checked to determine whether votes expressed in ballots correspond to the election outcome, thereby enabling the detection of undue influence. Smyth, Frink & Clarkson [30] capture universal verifiability as a game that tasks the adversary to falsify evidence that causes checks to succeed when the outcome does not correspond to the votes expressed in collected ballots, or that cause checks to fail when the outcome does correspond to the votes expressed. Thus,

[1] Earlier systems merely required ballots to be marked in polling booths and deposited into ballot boxes, which permitted non-uniform ballots, including ballots of different colours and sizes, that could be easily identified as party tickets [11].

[2] *Ballot secrecy* and *privacy* occasionally appear as synonyms in the literature. We favour ballot secrecy to avoid confusion with other privacy notions, such as receipt-freeness and coercion resistance, which we will briefly discuss in Section 8.

winning the game signifies the existence of a scenario in which a spurious outcome will be accepted or a legitimate outcome rejected. By comparison, when no winning adversary exists, anyone can determine whether the election outcome is correct. Universal verifiability and ballot secrecy are orthogonal properties of voting systems that can be studied, formulated and analysed independently. We shall largely avoid discussion of verifiability, except to introduce general concepts, to highlight features needed solely for verifiability, and to simplify proofs.

We introduce two voting systems to demonstrate how ballot secrecy and universal verifiability can be achieved. The first (Nonce) instructs each voter to cast a ballot comprising of their vote paired with a nonce (which is collected and stored on a bulletin board) and instructs the tallier to publish the election outcome corresponding to votes (stored on that board). The second (Enc2Vote) instructs voters to cast asymmetric encryptions of their votes and instructs the tallier to decrypt the encrypted votes and publish the outcome corresponding to those votes. Universal verifiability is ensured by the former system, because anyone can recompute the election outcome to check that it corresponds to votes expressed in collected ballots. But, ballot secrecy is not, because voters' votes are revealed. By comparison, secrecy is ensured by the latter system, because asymmetric encryption can ensure that votes cannot be recovered from ballots and the tallying procedure ensures that individual votes are not revealed. But, universal verifiability is not ensured. Indeed, spurious election outcomes need not correspond to the encrypted votes. Thus, Enc2Vote ensures secrecy not verifiability, and Nonce achieves the reverse. More advanced voting systems must simultaneously satisfy both secrecy and verifiability, and we will consider the Helios voting system.

Helios is an open-source, web-based electronic voting system [31], which has been used in binding elections. In particular, the International Association of Cryptologic Research (IACR) has used Helios annually since 2010 to elect board members [32, 33],[3] the Association for Computing Machinery (ACM) used Helios for their 2014 general election [34], the Catholic University of Louvain used Helios to elect their university president in 2009 [31], and Princeton University has used Helios since 2009 to elect student governments.[4] Helios is intended to satisfy universal verifiability whilst maintaining ballot secrecy. For ballot secrecy, each voter is instructed to encrypt their vote using an asymmetric homomorphic encryption scheme. Encrypted votes are homomorphically combined and the homomorphic combination is decrypted to reveal the outcome. Alternatively, a mixnet is applied to the encrypted votes and the mixed encrypted votes are decrypted to reveal the outcome [35, 36]. We continue to refer to the former voting system as Helios and, henceforth, refer to the latter variant as *Helios Mixnet*. For universal verifiability, the encryption step is accompanied by a non-interactive zero-knowledge proof demonstrating correct computation. This ensures homomorphic combinations of encrypted votes and mixed encrypted votes can be decrypted, hence, the outcome can be recovered. Helios additionally requires proof that ciphertexts encrypt votes. This prevents an adversarial voter crafting a ciphertext that could be combined with others to derive an election outcome in the voter's favour. (E.g., votes might be switched between candidates.) The decryption step is similarly accompanied by a non-interactive zero-knowledge proof to prevent spurious outcomes.

Contribution and structure. Contributions are summarised in the bullet points below and described in detail by the surrounding text:

[3] https://www.iacr.org/elections/, accessed 29 Mar 2019.

[4] http://heliosvoting.wordpress.com/2009/10/13/helios-deployed-at-princeton/ and https://princeton.heliosvoting.org/, accessed 29 Mar 2019.

- A definition of ballot secrecy that overcomes limitations of previous works by considering an adversary that controls ballot collection.

Section 3 briefly explains the pitfalls of existing ballot secrecy definitions, introduces our game-based definition of ballot secrecy, adapts formalisations of non-malleability and indistinguishability for asymmetric encryption to derive two equivalent game-based definitions of ballot independence, and proves relations between definitions. In particular, ballot independence is shown to be sufficient for ballot secrecy in a class of voting systems with zero-knowledge tallying proofs, and it is shown to be necessary, but not sufficient, in general.

- A Helios case study that identifies an attack, which cannot be detected by previous work, and a proof that secrecy is satisfied after applying a fix.

Section 4 shows that our definition of ballot secrecy can be used to identify a known vulnerability in Helios, explains why earlier definitions of ballot secrecy by Bernhard *et al.* cannot detect that vulnerability, discusses non-malleable ballots as a fix, and uses our sufficient condition to prove that secrecy is satisfied when the fix is applied.

- A proof that voting systems built from non-malleable encryption satisfy ballot secrecy if tallying is additive, which trivialises secrecy proofs.

Section 5 proves that ballot independence cannot be harmed by tallying, if all ballots are tallied correctly; shows that universally-verifiable voting systems tally ballots correctly; proves Enc2Vote satisfies ballot independence, assuming the underlying asymmetric encryption scheme is non-malleable; and combines those results to show that proofs of ballot secrecy are trivial for a class of universally-verifiable, encryption-based voting systems.

- A Helios Mixnet case study that demonstrates the triviality of ballot secrecy proofs for systems built from non-malleable encryption.

Section 6 presents an analysis of Helios Mixnet and demonstrates that our results do indeed make proofs of ballot secrecy trivial, by showing that the combination of universal verifiability and non-malleable encryption suffice for ballot secrecy in Helios Mixnet. The remaining sections present syntax (§2), discussion, limitations, and directions for further research (§7), related work (§8), and a brief conclusion (§9); Sidebar 1 introduces game-based security definitions and recalls notation; and the appendices define cryptographic primitives and relevant security definitions (Appendix A) and present further supplementary material. (Readers familiar with games might like to skip Sidebar 1, and some readers might like to study the related work before our definition of ballot secrecy.)

2. Election scheme syntax

We recall syntax (Definition 1) for voting systems that consist of the following three steps. First, a tallier generates a key pair. Secondly, each voter constructs and casts a ballot for their vote. These ballots are collected and recorded on a bulletin board. Finally, the tallier tallies the collected ballots and announces the outcome as a frequency distribution of votes. The chosen representative is derived from this distribution, e.g., as the candidate with the most votes.[5]

[5] Smyth, Frink & Clarkson use the syntax to model first-past-the-post voting systems [30] and Smyth shows ranked-choice voting systems can be modelled too [37].

Sidebar 1 Preliminaries: Games and notation

A game formulates a series of interactions between a benign challenger, a malicious adversary, and a cryptographic scheme. The adversary wins by completing a task that captures an execution of the scheme in which security is broken, i.e., winning captures what should be unachievable. Tasks can generally be expressed as *indistinguishability* or *reachability* requirements. For example, universal verifiability can be expressed as the inability to reach a state that causes a voting system's checks to succeed for invalid election outcomes, or fail for valid outcomes. Moreover, ballot secrecy can be expressed as the inability to distinguish between an instance of a voting system in which voters cast some votes, from another instance in which the voters cast a permutation of those votes.

Formally, games are probabilistic algorithms that output booleans. We let $A(x_1, \ldots, x_n; r)$ denote the output of probabilistic algorithm A on inputs $x_1, \ldots, x_n$ and coins r, and we let $A(x_1, \ldots, x_n)$ denote $A(x_1, \ldots, x_n; r)$, where coins r are chosen uniformly at random from the coin space of algorithm A. Moreover, we let $x \leftarrow T$ denote assignment of T to x, and $x \leftarrow_R S$ denote assignment to x of an element chosen uniformly at random from set S. Using our notation, we can formulate game $\mathsf{Exp}(H, S, \mathcal{A})$ – which tasks an adversary $\mathcal{A}$ to distinguish between a function H and a simulator S – as follows: $m \leftarrow \mathcal{A}(); \beta \leftarrow_R \{0,1\};$ **if** $\beta = 0$ **then** $x \leftarrow H(m);$ **else** $x \leftarrow S(m);$ $g \leftarrow \mathcal{A}(x);$ **return** $g = \beta$. Adversaries are *stateful*, i.e., information persists across invocations of an adversary in a game. For instance, adversaries can access earlier assignments, e.g., the adversary's second instantiation in game Exp has access to any assignments made during its first instantiation. An adversary *wins* a game by causing it to output true ($\top$) and the adversary's *success* in a game $\mathsf{Exp}(\cdot)$, denoted $\mathsf{Succ}(\mathsf{Exp}(\cdot))$, is the probability that the adversary wins, that is, $\mathsf{Succ}(\mathsf{Exp}(\cdot)) = \Pr[\mathsf{Exp}(\cdot) = \top]$. We focus on computational security, rather than information-theoretic security, and tolerate breaks by adversaries in non-polynomial time and breaks with negligible success, since such breaks are infeasible in practice.

Game Exp captures a single interaction between the challenger and the adversary. We can extend games with oracles to capture arbitrarily many interactions. For instance, we can formulate a strengthening of Exp as follows: $\beta \leftarrow_R \{0,1\}; g \leftarrow \mathcal{A}^{\mathcal{O}}(x);$ **return** $g = \beta$, where $\mathcal{A}^{\mathcal{O}}$ denotes $\mathcal{A}$'s access to oracle $\mathcal{O}$ and $\mathcal{O}(m)$ computes **if** $\beta = 0$ **then** $x \leftarrow H(m);$ **else** $x \leftarrow S(m);$ **return** x. Oracles may access game parameters such as bit β.

Beyond the above notation, we let $x[i]$ denote component i of vector x and let $|x|$ denote the length of vector x. Moreover, we write $(x_1, \ldots, x_{|T|}) \leftarrow T$ for $x \leftarrow T; x_1 \leftarrow x[1]; \ldots; x_{|T|} \leftarrow x[|T|]$, when T is a vector, and $x, x' \leftarrow_R S$ for $x \leftarrow_R S; x' \leftarrow_R S$.

Definition 1 (Election scheme [30])**.** *An* election scheme *is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *such that:*

Setup*, denoted* $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$*, is run by the tallier. The algorithm 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*.*

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

Tally*, denoted* $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$*, is run by the tallier. The algorithm takes as input a private key* sk*, a bulletin board* $\mathfrak{bb}$*, some number of candidates* nc*, and a security parameter* κ*, where* $\mathfrak{bb}$ *is a set. The algorithm outputs an election outcome* $\mathfrak{v}$ *and a non-interactive tallying proof* pf*, where* $\mathfrak{v}$ *is a vector of length* nc *and each index* v *of that vector should indicate the number of votes for candidate* v*.*

Election schemes must satisfy correctness*, that is, 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, given a zero-filled vector $\mathfrak{v}$ of length nc, we have:*

```
$\Pr[(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa);$
  for $1 \leqslant i \leqslant nb$ do
      $b_i \leftarrow \mathsf{Vote}(pk, v_i, nc, \kappa);$
      $\mathfrak{v}[v_i] \leftarrow \mathfrak{v}[v_i] + 1;$
  $(\mathfrak{v}', pf) \leftarrow \mathsf{Tally}(sk, \{b_1, \ldots, b_{nb}\}, nc, \kappa) : nb \leqslant mb \wedge nc \leqslant mc \Rightarrow \mathfrak{v} = \mathfrak{v}'] > 1 - \mathsf{negl}(\kappa).$
```

The syntax provides a language to model voting systems and the correctness condition ensures that such systems function, i.e., election outcomes correspond to votes expressed in ballots, when ballots are constructed and tallied in the prescribed manner.

The syntax focuses on minimalism, rather than generality: Algorithm Setup naturally inputs a security parameter and outputs a key pair. In addition, the algorithm outputs bounds on the number of ballots *mb*, respectively candidates *mc*. These bounds broaden the correctness definition's scope. Indeed, Enc2Vote requires *mc* to be less than or equal to the size of the underlying encryption scheme's message space (otherwise decryption might not reveal voters' votes). Helios additionally requires the same constraint on *mb* (otherwise decryption of the homomorphic combination might not reveal the outcome). Algorithm Vote naturally inputs a public key, a vote, and a security parameter, and outputs a ballot. The vote is expressed as an integer, rather than alphanumeric strings, for brevity. Beyond those inputs, the algorithm inputs a number of candidates *nc*, which is necessary in voting systems such as Helios, because ballot construction depends on the number of candidates. Tallying is similarly dependent on the number of candidates in Helios, hence the inputs and outputs to algorithm Tally are all natural. Finally, the syntax restricts bulletin boards to sets, rather than multisets or lists. This is a limitation: Sets preclude the construction of schemes vulnerable to attacks that arise due to duplicate ballots [38, §2.1 & §4.3]. Systems vulnerable to such attacks cannot be modelled using the syntax: Analysts must not abstract away the details of lists or multisets and model them as sets, since any ensuing analysis will miss attacks. Analysts may abstractly model lists or multisets as sets with some additional effort.

Election schemes may also include an algorithm Verify, which is used to audit an election. We omit that algorithm from Definition 1, because we focus on ballot secrecy, rather than verifiability: Ballot secrecy must be upheld independently of auditing, since voters' votes should not be revealed regardless of whether an election outcome is correct (i.e., regardless of whether auditing succeeds). Hence, algorithm Verify must not be used to achieve privacy and can be omitted. Nonetheless, algorithm Tally must output the tallying proof (that would be input to algorithm Verify), since tallying proofs can harm privacy. (For instance, as an extreme example, a tallying proof may include the private key, which can used to violate every voter's privacy.)

Voting systems must ensure that outcomes only include votes cast by voters, as opposed to non-voters, and that only one vote of each voter has influence [39]. The former can be achieved by authentication. Traditionally, the latter was achieved by permitting each voter to cast at most one ballot, whereas more recent systems permit multiple ballots and count only the last. Our syntax is compatible with voting systems, such as Helios, that rely on trusted external authentication services to ensure that collected ballots satisfy these properties (i.e., collected ballots were cast by voters and contain at most one ballot per voter), hence, election schemes need not achieve these properties directly. (An extension of the syntax is compatible with voting systems that introduce voter credentials and use cryptography to achieve these properties directly [30, 40, 41].) Here be dragons: The security of external authentication services must be verified.

We will use our syntax to express a privacy property of election schemes, moreover, we will model and analyse voting systems, including Helios (§4) and Helios Mixnet (§6). Beyond those systems, it also captures Helios-C [42] and the system by Juels, Catalano & Jakobsson [43] (once extended to include voter credentials), so Belenios and Civitas should be included as well. These schemes are rather dominant in the literature, so the class is interesting and rather general. Notable voting systems that are excluded include: systems relying on features implemented with paper (e.g., [44–46]); systems which require partially private ballots (e.g., [47]); and schemes relying on partially honest bulletin boards (e.g., [48]). We also exclude distributed tallying. These limitations are further discussed in Section 7.

3. Privacy

Some scenarios inevitably reveal voters' votes: Unanimous outcomes reveal how everyone voted and, more generally, outcomes can be coupled with partial knowledge of voters' votes to deduce voters' votes. For example, suppose Alice, Bob and Mallory participate in a referendum and the outcome has frequency two for 'yes' and one for 'no.' Mallory and Alice can deduce Bob's vote by pooling knowledge of their own votes. Similarly, Mallory and Bob can deduce Alice's vote. Furthermore, Mallory can deduce that Alice and Bob both voted yes, if she voted no. For simplicity, our informal definition of ballot secrecy (§1) deliberately omitted side-conditions which exclude these inevitable revelations and which are necessary for satisfiability.[6] We now refine that definition as follows:

> A voter's vote is not revealed to anyone, except when the vote can be deduced from the election outcome and any partial knowledge of voters' votes.

This refinement ensures the aforementioned examples are not violations of ballot secrecy. By comparison, if Mallory votes yes and she can deduce the vote of Alice, without knowledge of Bob's vote, then ballot secrecy is violated.

We could formulate ballot secrecy as the following game: First, the adversary picks a pair of votes v_0 and v_1. Secondly, the challenger constructs a ballot b_1 for vote v_β and a second ballot b_2 for $v_{1-\beta}$, where β is a bit chosen uniformly at random. Those ballots are given to the adversary. Thirdly, the adversary constructs ballots $b_3,\dots,b_n$. Fourthly, the challenger tallies all the ballots (i.e., $b_1,\dots,b_n$) to the determine the election outcome, which the adversary is given. Finally, the adversary is tasked with determining bit β. This game challenges the adversary to determine if the first ballot is for v_0 and the second is for v_1, or vice-versa. Intuitively, a losing adversary cannot distinguish ballots; seemingly suggesting that Alice voting 'yes' is indistinguishable from Bob voting 'no.'

The first release of Helios is not secure with respect to the aforementioned game, due to a vulnerability identified by Cortier & Smyth [53, 54]. Indeed, an adversary can observe a ballot constructed by the challenger, compute a meaningfully related ballot (from a malleable Helios ballot), and exploit the relation to win the game. This vulnerability can be attributed to tallying meaningfully related ballots; omitting such ballots from tallying, i.e., *ballot weeding*, is postulated as a defence [27, 29, 53–57]. Variants of Helios with ballot weeding seem secure with respect to this game. Unfortunately, ballot weeding

[6] Voting systems that announce chosen representatives (e.g., [49–52]), rather than frequency distributions of votes, could offer stronger notions of privacy.

mechanisms can be subverted by intercepting ballots or by re-ordering ballots. Given that current definitions cannot detect such vulnerabilities (§8), we should conclude that they are unsuitable. Indeed, the challenger tallying *all* ballots introduces an implicit trust assumption: ballots are *recorded-as-cast*, i.e., cast ballots are preserved with integrity through the ballot collection process.[7] Thus, vulnerabilities that manipulate the ballot collection process cannot be detected, including vulnerabilities that can be exploited to distinguish Alice voting ‘yes’ from Bob voting ‘no.’ To overcome this shortcoming, we formulate a new definition of ballot secrecy in which the adversary controls the ballot collection process, i.e., the bulletin board and the communication channel.

3.1. Ballot secrecy

We formalise ballot secrecy (Definition 2) as a game that tasks the adversary to: select two lists of votes; construct a bulletin board from ballots for votes in one of those lists, which list is decided by a coin flip; and (non-trivially) determine the result of the coin flip from the resulting election outcome and tallying proof. That is, the game tasks the adversary to distinguish between an instance of the voting system for one list of votes, from another instance with the other list of votes, when the votes cast from each list are permutations of each other (hence, the distinction is non-trivial). The game proceeds as follows: The challenger generates a key pair (Line 1), the adversary chooses some number of candidates (Line 2), and the challenger flips a coin (Line 3) and initialises a set to record lists of votes (Line 4). The adversary computes a bulletin board from ballots for votes in one of two possible lists (Line 5), where the lists are chosen by the adversary, the choice between lists is determined by the coin flip, and the ballots (for votes in one of the lists) are constructed by an left-right oracle (further ballots may be constructed by the adversary).[8] The challenger tallies the bulletin board to derive the election outcome and tallying proof (Line 6), which are given to the adversary and the adversary is tasked with determining the result of the coin flip (Line 7 & 8).

Definition 2 (Ballot-Secrecy)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* Ballot-Secrecy *be the following game.*

```
Ballot-Secrecy$(\Gamma, \mathcal{A}, \kappa)$ =
1 $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
2 $nc \leftarrow \mathcal{A}(pk, \kappa)$;
3 $\beta \leftarrow_R \{0,1\}$;
4 $L \leftarrow \emptyset$;
5 $\mathfrak{bb} \leftarrow \mathcal{A}^{\mathcal{O}}()$;
6 $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$;
7 $g \leftarrow \mathcal{A}(\mathfrak{v}, pf)$;
8 return $g = \beta \wedge balanced(\mathfrak{bb}, nc, L) \wedge 1 \leqslant nc \leqslant mc \wedge |\mathfrak{bb}| \leqslant mb$;
```

Predicate balanced and oracle $\mathcal{O}$ *are defined as follows:*

- $balanced(\mathfrak{bb}, nc, L)$ *holds if for all votes* $v \in \{1, \dots, nc\}$ *we have* $|\{b \mid b \in \mathfrak{bb} \wedge \exists v_1 \,.\, (b, v, v_1) \in L\}| = |\{b \mid b \in \mathfrak{bb} \wedge \exists v_0 \,.\, (b, v_0, v) \in L\}|$*; and*

[7] The recorded-as-cast notion was introduced by Adida & Neff [58, §2].

[8] Bellare *et al.* introduced left-right oracles in the context of symmetric encryption [59] and Bellare & Rogaway provide a tutorial on their use [60].

- *$\mathcal{O}(v_0, v_1)$ computes $b \leftarrow \mathsf{Vote}(pk, v_\beta, nc, \kappa); L \leftarrow L \cup \{(b, v_0, v_1)\}$ and outputs b, where $v_0, v_1 \in \{1, ..., nc\}$.*

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

An election scheme satisfies Ballot-Secrecy when algorithm Vote outputs ballots that do not reveal votes and algorithm Tally outputs election outcomes and proofs that do not reveal the relation between votes expressed in collected ballots and the outcome.

Game Ballot-Secrecy tasks the adversary to compute a bulletin board, from ballots constructed by a left-right oracle for votes in one of two possible lists, and determine which list was used from the election outcome and proof generated from tallying that board. The choice between lists is determined by the result β of a coin flip, and the left-right oracle outputs a ballot for vote v_β on input of a pair of votes v_0, v_1. Hence, the left-right oracle constructs ballots for votes in one of two possible lists, where the lists are chosen by the adversary, and the bulletin board may contain those ballots in addition to ballots constructed by the adversary.

Election schemes reveal the number of votes for each candidate (i.e., the election outcome). Hence, to avoid trivial distinctions in game Ballot-Secrecy, we require that runs of the game are *balanced*: "left" and "right" inputs to the left-right oracle are equivalent, when the corresponding outputs appear on the bulletin board. For example, suppose the inputs to the left-right oracle are $(v_{1,0}, v_{1,1}), \ldots, (v_{n,0}, v_{n,1})$ and the corresponding outputs are $b_1, \ldots, b_n$, further suppose the bulletin board is $\{b_1, \ldots, b_\ell\}$ such that $\ell \leqslant n$. That game is balanced if the "left" inputs $v_{1,0}, \ldots, v_{\ell,0}$ are a permutation of the "right" inputs $v_{1,1}, \ldots, v_{\ell,1}$. The balanced condition prevents trivial distinctions.[9] For instance, an adversary that computes a bulletin board containing only the ballot output by a left-right oracle query with input $(1, 2)$ cannot win the game, because it is unbalanced. Albeit, that adversary could trivially determine whether $\beta = 0$ or $\beta = 1$, given the tally of that board.

Intuitively, if the adversary wins game Ballot-Secrecy, then there exists a strategy to distinguish ballots. Indeed, such an adversary can distinguish between an instance of the voting system in which voters cast some votes, from another instance in which voters cast a permutation of those votes, thus, voters' votes are revealed. Otherwise, the adversary is unable to distinguish between a voter casting a ballot for vote v_0 and another voter casting a ballot for vote v_1, hence, voters' votes cannot be revealed.

Proving ballot secrecy is time consuming. Indeed, a proof for the simple Enc2Vote scheme consumes around four and a half pages [61, Appendix C.6]. To reduce the expense of such proofs, we introduce ballot independence (§3.2) and prove that it suffices for Ballot-Secrecy (§3.3). Using this sufficient condition, the four and a half page proof can be reduced to around one page [62, §4.3].

3.2. Ballot independence

Ballot independence [53, 63, 64] is seemingly related to ballot secrecy.

- *Ballot independence.* Observing another voter's interaction with the voting system does not allow a voter to cast a meaningfully related vote.

[9] A weaker balanced condition might be sufficient for alternative formalisations of election schemes. For instance, voting systems which only announce the winning candidate could be analysed using a balanced condition asserting that the winning candidate was input on both the "left" and "right."

Our informal definition essentially states that an adversary is unable to construct a ballot meaningfully related to a non-adversarial ballot, i.e., ballots are non-malleable. Hence, we can formalise ballot independence as a straightforward adaptation of the non-malleability definition for asymmetric encryption by Bellare & Sahai [65].[10] Such a formalisation captures an intuitive notion of ballot independence, but it is complex and proofs of non-malleability are relatively difficult. Bellare & Sahai observe similar complexities and show that their definition is equivalent to a simpler, indistinguishability notion. In a similar direction, we derive a definition of ballot independence, called indistinguishability under chosen vote attack (IND-CVA), as a straightforward adaptation of their indistinguishability notion for asymmetric encryption.

Definition 3 (IND-CVA). *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be the security parameter, and* IND-CVA *be the following game.*

```
IND-CVA$(\Gamma, \mathcal{A}, \kappa)$ =
    $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
    $(v_0, v_1, nc) \leftarrow \mathcal{A}(pk, \kappa)$;
    $\beta \leftarrow_R \{0,1\}$;
    $b \leftarrow \mathsf{Vote}(pk, v_\beta, nc, \kappa)$;
    $\mathfrak{bb} \leftarrow \mathcal{A}(b)$;
    $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$;
    $g \leftarrow \mathcal{A}(\mathfrak{v})$;
    return $g = \beta \wedge b \notin \mathfrak{bb} \wedge 1 \leqslant v_0, v_1 \leqslant nc \leqslant mc \wedge |\mathfrak{bb}| \leqslant mb$;
```

We say Γ *satisfies* indistinguishability under chosen vote attack (IND-CVA)*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such that for all security parameters* κ*, we have* $\mathsf{Succ}(\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{A}, \kappa)) \leqslant \frac{1}{2} + \mathsf{negl}(\kappa)$.

Game IND-CVA is satisfied if the adversary cannot determine whether the challenge ballot b is for one of two possible votes v_0 and v_1. In addition to the challenge ballot, the adversary is given the election outcome derived by tallying a bulletin board computed by the adversary. To avoid trivial distinctions, the adversary's bulletin board should not contain the challenge ballot. Intuitively, the adversary wins if there exists a strategy to construct related ballots, since this strategy enables the adversary to construct a ballot b', related to the challenge ballot b, and determine if b is for v_0 or v_1 from the outcome derived by tallying a bulletin board containing b'.

Comparing IND-CVA *and* IND-PA0. The main distinction between indistinguishability for asymmetric encryption (IND-PA0) and indistinguishability for election schemes (IND-CVA) is as follows: IND-PA0 performs a parallel decryption, whereas IND-CVA performs a single tally. Hence, indistinguishability for encryption reveals plaintexts corresponding to ciphertexts, whereas indistinguishability for elections reveals the number of votes for each candidate.

We present an alternative definition of ballot independence (Appendix B), based upon the definition of non-malleability for asymmetric encryption by Bellare & Sahai, and prove that the definition is equivalent to IND-CVA.

[10] Non-malleability was first formalised by Dolev, Dwork & Naor in the context of asymmetric encryption [66, 67]; the definition by Bellare & Sahai builds upon their results and results by Bellare *et al.* [68].

3.3. Secrecy and independence coincide (for zero-knowledge tallying proofs)

The main distinctions between our ballot secrecy (Ballot-Secrecy) and ballot independence (IND-CVA) games are as follows.

(1) The challenger produces one challenge ballot for the adversary in game IND-CVA, whereas the left-right oracle produces arbitrarily many challenge ballots for the adversary in game Ballot-Secrecy.

(2) The adversary in game Ballot-Secrecy has access to a tallying proof, but the adversary in game IND-CVA does not.

(3) The winning condition in game Ballot-Secrecy requires the bulletin board to be balanced, whereas the bulletin board must not contain the challenge ballot in game IND-CVA.

The second point distinguishes our games and shows Ballot-Secrecy is at least as strong as IND-CVA. Hence, non-malleable ballots are necessary in election schemes satisfying Ballot-Secrecy. (Non-malleable ballots are not generally necessary: Privacy definitions are sound but incomplete, Section 7.)

Theorem 1 (Ballot-Secrecy $\Rightarrow$ IND-CVA)**.** *Given an election scheme Γ satisfying* Ballot-Secrecy*, we have Γ satisfies* IND-CVA*.*

A proof of Theorem 1 and all further proofs, except where otherwise stated, appear in Appendix C.

Tallying proofs may reveal voters' votes. For example, a variant of Enc2Vote might define tallying proofs that map ballots to votes. Since proofs are available to the adversary in game Ballot-Secrecy, but not in game IND-CVA, it follows that Ballot-Secrecy is strictly stronger than IND-CVA.

Proposition 2 (IND-CVA $\not\Rightarrow$ Ballot-Secrecy)**.** *An election scheme may satisfy* IND-CVA*, but not* Ballot-Secrecy*.*

Proposition 2 follows from our informal reasoning and we omit a formal proof.

Game Ballot-Secrecy is generally (strictly) stronger than game IND-CVA. Nonetheless, we show that our games coincide for election schemes without tallying proofs (Definition 4), assuming tallying is additive (Definition 5), that is, the sum of election outcomes derived by tallying distinct subsets of a bulletin board is equivalent to the election outcome derived by tallying the union of those subsets.

Definition 4. *An election scheme $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ is* without tallying proofs*, if there exists a constant symbol ϵ such that for all private keys sk, sets $\mathfrak{bb}$, integers nc, security parameters κ, and computations $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$, we have $pf = \epsilon$.*

Definition 5 (Additivity)**.** *Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ be an election scheme, $\mathcal{A}$ be an adversary, κ be a security parameter, and* Additivity *be the following game.*

```
$\mathsf{Additivity}(\Gamma, \mathcal{A}, \kappa) =$
    $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
    $(nc, \mathfrak{bb}, \mathfrak{bb}') \leftarrow \mathcal{A}(pk, \kappa)$;
    $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$;
    $(\mathfrak{v}_0, pf_0) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb} \setminus \mathfrak{bb}', nc, \kappa)$;
    $(\mathfrak{v}_1, pf_1) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb} \cap \mathfrak{bb}', nc, \kappa)$;
    **return** $\mathfrak{v} = \mathfrak{v}_0 + \mathfrak{v}_1 \wedge nc \leqslant mc \wedge |\mathfrak{bb}| \leqslant mb$;
```

We say Γ *satisfies* Additivity*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such that for all security parameters* κ*, we have* $\mathsf{Succ}(\mathsf{Additivity}(\Gamma, \mathcal{A}, \kappa)) > 1 - \mathsf{negl}(\kappa)$.

Proposition 3 (Ballot-Secrecy = IND-CVA, without proofs)**.** *Let* Γ *be an election scheme without tallying proofs. Suppose* Γ *satisfies* Additivity*. We have* Γ *satisfies* Ballot-Secrecy *iff* Γ *satisfies* IND-CVA.

Our equivalence result generalises to election schemes with *zero-knowledge tallying proofs* (Definition 19), i.e., schemes that compute proofs using non-interactive zero-knowledge proof systems.

Theorem 4 (Ballot-Secrecy = IND-CVA, with ZK proofs)**.** *Let* Γ *be an election scheme with zero-knowledge tallying proofs. Suppose* Γ *satisfies* Additivity*. We have* Γ *satisfies* Ballot-Secrecy *iff* Γ *satisfies* IND-CVA.

Proof sketch. Game Ballot-Secrecy computes the election outcome from ballots constructed by the oracle and ballots constructed by the adversary. Intuitively, such an outcome can be equivalently computed as follows:

```
$(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\}, nc, \kappa)$;
$(\mathfrak{v}', pf') \leftarrow \mathsf{Tally}(sk, \mathfrak{bb} \cap \{b \mid (b, v_0, v_1) \in L\}, nc, \kappa)$;
$\mathfrak{v} \leftarrow \mathfrak{v} + \mathfrak{v}'$;
```

Yet, a poorly designed tallying algorithm might not ensure equivalence. In particular, ballots constructed by the adversary can cause the algorithm to behave unexpectedly. (Such algorithms are nonetheless compatible with our correctness requirement, because correctness does not consider an adversary.) Our Additivity property ensures equivalence. Moreover, our Additivity property ensures the above computation is equivalent to the following:

```
$(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\}, nc, \kappa)$;
for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ do
    $(\mathfrak{v}', pf') \leftarrow \mathsf{Tally}(sk, \{b\}, nc, \kappa)$;
    $\mathfrak{v} \leftarrow \mathfrak{v} + \mathfrak{v}'$;
```

Furthermore, by correctness of the election scheme, the above for-loop can be equivalently computed as follows:

```
for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ do
    $\mathfrak{v}[v_\beta] \leftarrow \mathfrak{v}[v_\beta] + 1$;
```

Indeed, for each $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$, we have b is an output of $\mathsf{Vote}(pk, v_\beta, nc, \kappa)$, hence, $\mathsf{Tally}(sk, \{b\}, nc, \kappa)$ outputs $(\mathfrak{v}, pf)$ such that $\mathfrak{v}$ is a zero-filled vector, except for index v_β which contains one, and this suffices to ensure equivalence. In addition, for any adversary that wins game Ballot-Secrecy, we are assured that $balanced(\mathfrak{bb}, nc, L)$ holds, hence, the above for-loop can be computed as

```
for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ do
    $\mathfrak{v}[v_0] \leftarrow \mathfrak{v}[v_0] + 1$;
```

or

```
for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ do
    $\mathfrak{v}[v_1] \leftarrow \mathfrak{v}[v_1] + 1$;
```

without weakening the game. Thus, perhaps surprisingly, tallying ballots constructed by the oracle does not provide the adversary with an advantage (in determining whether $\beta = 0$ or $\beta = 1$) and we can omit such ballots from tallying in game Ballot-Secrecy. Thus, game Ballot-Secrecy is equivalent to game BS (Definition 20), which modifies the tallying procedure as described and simulates tallying proofs. Thereafter, we proceed with a hybrid argument (for which Shoup presents a brief tutorial [69]). □

Additivity is implied by universal verifiability (Lemmata 8 & 28). Thus, a special case of Theorem 4 requires the election scheme to satisfy universal verifiability, which is useful to simplify its application. Indeed, we exploit this result in the following section to prove Ballot-Secrecy.

4. Case study I: Helios

Helios can be informally modelled as the following election scheme (further details appear in Sidebar 2):

Setup generates a key pair for an asymmetric additively-homomorphic encryption scheme, proves correct key generation in zero-knowledge, and outputs the key pair and proof.

Vote enciphers the vote's bitstring encoding to a tuple of ciphertexts, proves in zero-knowledge that each ciphertext is correctly constructed and that the vote is selected from the sequence of candidates, and outputs the ciphertexts coupled with the proofs.

Tally selects ballots from the bulletin board for which proofs hold, homomorphically combines the ciphertexts in those ballots, decrypts the homomorphic combination to reveal the election outcome, and announces the outcome, along with a zero-knowledge proof of correct decryption.

Helios was first released in 2009 as *Helios 2.0*,[11] and the current release is *Helios 3.1.4*.[12]

4.1. Helios 2.0 & Helios 3.1.4

Cortier & Smyth show that Helios 2.0 does not satisfy ballot secrecy (§3) and neither does Helios 3.1.4.[13] Thus, we would not expect our definition of ballot secrecy to hold. Indeed, we adopt formal descriptions of Helios 2.0 and Helios 3.1.4 by Smyth, Frink & Clarkson [30] (Appendix D) and use those descriptions to prove that Ballot-Secrecy is not satisfied.

Theorem 5. *Neither Helios 2.0 nor Helios 3.1.4 satisfy* Ballot-Secrecy.

Cortier & Smyth attribute the vulnerability to tallying meaningfully related ballots. Indeed, Helios uses malleable ballots: Given a ballot $c_1, \ldots, c_{nc-1}, \sigma_1, \ldots, \sigma_{nc}$, we have $c_{\chi(1)}, \ldots, c_{\chi(nc-1)}, \sigma_{\chi(1)}, \ldots, \sigma_{\chi(nc-1)}, \sigma_{nc}$ is a ballot for all permutations χ on $\{1, \ldots, nc - 1\}$. Thus, ballots are malleable, which is incompatible with Ballot-Secrecy (§3.3).

[11] https://github.com/benadida/helios/releases/tag/2.0, released 25 Jul 2009, accessed 29 Mar 2019.

[12] https://github.com/benadida/helios-server/releases/tag/v3.1.4, released 10 Mar 2011, last patched 27 Oct 2017, accessed 29 Mar 2019.

[13] Helios 3.1.4 mitigates against a universal-verifiability vulnerability in Helios 2.0, by checking that tallied ballots are constructed from suitable cryptographic parameters [70, §4.1]. The vulnerabilities described herein use well-formed parameters, hence, the additional checks do not preclude vulnerabilities and we refer the reader to the original description for details.

Sidebar 2 Helios: Ballot construction and tallying

Algorithm Vote inputs a vote v selected from candidates $1, \ldots, nc$ and computes ciphertexts $c_1, \ldots, c_{nc-1}$ such that if $v \neq nc$, then ciphertext c_v contains plaintext 1 and the remaining ciphertexts contain plaintext 0, otherwise, all ciphertexts contain plaintext 0. The algorithm also computes zero-knowledge proofs $\sigma_1, \ldots, \sigma_{nc}$ demonstrating correct computation. Proof σ_j demonstrates that ciphertext c_j contains 0 or 1, where $1 \leqslant j \leqslant nc-1$, and proof σ_{nc} demonstrates that the homomorphic combination of ciphertexts $c_1 \otimes \cdots \otimes c_{nc-1}$ contains 0 or 1. The algorithm outputs the ciphertexts and proofs.

Algorithm Tally inputs a bulletin board $\mathfrak{bb}$; selects all the ballots $b_1, \ldots, b_k \in \mathfrak{bb}$ for which proofs hold, i.e., ballots $b_i = \mathsf{Enc}(pk, m_{i,1}), \ldots, \mathsf{Enc}(pk, m_{i,nc-1}), \sigma_{i,1}, \ldots, \sigma_{i,nc}$ such that proofs $\sigma_{i,1}, \ldots, \sigma_{i,nc}$ hold, where $1 \leqslant i \leqslant k$; forms a matrix of the encapsulated ciphertexts, i.e.,

$$\begin{array}{ccc} \mathsf{Enc}(pk, m_{1,1}), & \ldots, & \mathsf{Enc}(pk, m_{1,nc-1}) \\ \vdots & & \vdots \\ \mathsf{Enc}(pk, m_{k,1}), & \ldots, & \mathsf{Enc}(pk, m_{k,nc-1}); \end{array}$$

homomorphically combines the ciphertexts in each column to derive the encrypted outcome, i.e.,

$$\mathsf{Enc}(pk, \Sigma_{i=1}^{k} m_{i,1}), \quad \ldots, \quad \mathsf{Enc}(pk, \Sigma_{i=1}^{k} m_{i,nc-1});$$

decrypts the homomorphic combinations to reveal the frequency of votes $1, \ldots, nc-1$, i.e.,

$$\Sigma_{i=1}^{k} m_{i,1}, \quad \ldots, \quad \Sigma_{i=1}^{k} m_{i,nc-1};$$

computes the frequency of vote nc by subtracting the frequency of any other vote from the number of ballots for which proofs hold, i.e., $k - \sum_{j=1}^{nc-1} \sum_{i=1}^{k} m_{i,j}$; and announces the outcome as those frequencies, along with a zero-knowledge proof demonstrating correctness of decryption.

Proof sketch. Suppose an adversary queries the left-right oracle with (distinct) inputs $v_0, v_1 \in \{1, \ldots, nc-1\}$ to derive a ballot for v_β, where integer $nc \geqslant 3$ is chosen by the adversary and bit β is chosen by the challenger. Further suppose the adversary picks a permutation χ on $\{1, \ldots, nc-1\}$, abuses malleability to derive a related ballot b for $\chi(v_\beta)$, and outputs bulletin board $\{b\}$. The board is balanced, because it does not contain the ballot output by the oracle. Suppose the adversary performs the following computation on input of election outcome $\mathfrak{v}$: if $\mathfrak{v}[\chi(v_0)] = 1$, then output 0, otherwise, output 1. Since b is a ballot for $\chi(v_\beta)$, it follows by correctness that $\mathfrak{v}[\chi(v_0)] = 1$ iff $\beta = 0$, and $\mathfrak{v}[\chi(v_1)] = 1$ iff $\beta = 1$, hence, the adversary wins the game. $\square$

For simplicity, our proof sketch considers an adversary that omits ballots from the bulletin board. Voters might detect such an adversary, because Helios satisfies individual verifiability, hence, voters can discover if their ballot is omitted. Detection does not invalidate Theorem 5, since the ability to detect an attack after the fact does not eliminate the possibility of an attack. Our proof sketch can be extended to avoid such detection: Let b_1 be the ballot output by the left-right oracle in the proof sketch and suppose b_2 is the ballot output by a (second) left-right oracle query with inputs v_1 and v_0. Further suppose the adversary outputs (the balanced) bulletin board $\{b, b_1, b_2\}$ and performs the following computation on input of election outcome $\mathfrak{v}$: if $\mathfrak{v}$ corresponds to votes $v_0, v_1, \chi(v_0)$, then output 0, otherwise, output 1, where χ is the permutation chosen by the adversary. Hence, the adversary wins the game.

Notions of ballot secrecy used by Bernhard, Pereira & Warinschi [71], Bernhard [72, §6.11] and Bernhard *et al.* [57, §D.3] cannot detect the known Helios 2.0 vulnerability (in the presence of an adversary that controls ballot collection), because interception is not possible when ballots are recorded-as-cast.[14]

Beyond ballot secrecy, Bernhard, Pereira & Warinschi show that Helios 3.1.4 does not satisfy universal verifiability [71].[15] They attribute vulnerabilities to application of the Fiat-Shamir transformation without inclusion of statements in hashes (i.e., weak Fiat-Shamir), and propose including statements in hashes (i.e., applying the Fiat-Shamir transformation) as a defence.

4.2. Helios'16

We have seen that non-malleable ballots are necessary for Ballot-Secrecy (§3.3); future Helios releases should adopt non-malleable ballots. The Fiat-Shamir transformation alone is insufficient to ensure non-malleability, because permutations can be applied to a ballot's ciphertexts: An ordering over ciphertexts must be proved.[16] Smyth, Frink & Clarkson formalise this idea in Helios'16 [30], a variant of Helios 3.1.4 that uses the Fiat-Shamir transformation and includes a ciphertext's position (relative to other ciphertexts) in hashes, which is intended, but not proven, to ensure non-malleable ballots. We recall their formal description in Appendix D, and using that formalisation we prove that Helios'16 delivers secrecy.

Theorem 6. *Helios'16 satisfies* Ballot-Secrecy.

Proof sketch. We prove that Helios'16 has zero-knowledge tallying proofs and, since universal verifiability is satisfied [30], we have Additivity too (Lemmata 8 & 28). Hence, by Theorem 4, it suffices to show that Helios'16 satisfies IND-CVA, which we prove by reduction to the security of the underlying encryption scheme, using an extractor that exists for the proof system used to construct ballots. □

A formal proof of Theorem 6 appears in Appendix D.1, assuming the random oracle model [75]. This proof, coupled with the proof of verifiability by Smyth, Frink & Clarkson [30], provides strong motivation for future Helios releases being based upon Helios'16, since it is the only variant of Helios which is proven to satisfy both ballot secrecy and verifiability.[17] A new release of Helios, which we'll call *Helios 4.0*, has been long-planned.[18] That version goes beyond the idea of merely proving an ordering over ciphertexts, to include far more information in hashes, which should be useful when sharing keys between elections (to prevent ballots from one election being cast in another, which could violate ballot secrecy). Security results are summarised in Table 1. Once Helios 4.0 is finalised, future work could consider whether our ballot secrecy proof and verifiability proofs by Smyth, Frink, & Clarkson can be adapted to that scheme.

[14] This observation suggests that recorded-as-cast is unsatisfiable: An adversary that can intercept ballots can always prevent the collection of ballots. Nevertheless, the definition of recorded-as-cast is informal, thus ambiguity should be expected and some interpretation of the definition should be satisfiable.

[15] Beyond secrecy and verifiability, eligibility is not satisfied [39, 73, 74].

[16] I do not recall the exact origin of this idea: Email communication suggests both I (17 Dec 2010) and Olivier Pereira (27 Dec 2010) each independently conceived the need for a proven ordering over ciphertexts.

[17] Earlier versions of Helios have been shown to satisfy definitions of ballot secrecy by Bernhard *et al.*, but not notions of verifiability (the analysis by Küsters et al. [76] does not detect vulnerabilities identified by Bernhard et al. [71] and Chang-Fong & Essex [70], possibly because their analysis "does not formalize all the cryptographic primitives used by Helios" [30, §9]).

[18] https://web.archive.org/web/20171026064140/http://documentation.heliosvoting.org/verification-specs/helios-v4, published c. 2012, accessed 29 Mar 2019.

	Helios 2.0	Helios 3.1.4	Helios'16	Helios 4.0
Secrecy	✗	✗	✓	?
Verifiability	✗	✗	✓	?

Cortier & Smyth identify a secrecy vulnerability in Helios 2.0 and Helios 3.1.4 [53]. Bernhard, Pereira & Warinschi [71] and Bernhard *et al.* [57, §D.3] show that Helios 4.0 satisfies various notions of ballot secrecy in two candidate elections – a general result is unknown. We have proved that Helios'16 satisfies ballot secrecy (Theorem 6). Bernhard, Pereira & Warinschi identify universal-verifiability vulnerabilities in Helios 2.0 and Helios 3.1.4 [71] and Chang-Fong & Essex identify vulnerabilities in Helios 2.0 [70]. Smyth, Frink, & Clarkson prove that Helios'16 satisfies individual and universal verifiability.

Table 1

Summary of Helios security results

4.3. Ballot weeding considered harmful

Ballot weeding mechanisms (that omit meaningfully related ballots from tallying) have been proposed, e.g., [27, 29, 53–57, 77], to mitigate against the vulnerabilities in Helios 3.1.4.[19] One candidate mechanism omits any ballot containing a previously observed hash from the tallying procedure. Another omits any ballot containing a previously observed hash from the bulletin board.[20] (More precisely, the mechanism stores the hashes used by non-interactive zero-knowledge proofs in a hashtable and any ballot containing a previously stored hash is omitted from the bulletin board.) These mechanisms can be subverted by excluding ballots (Remark 7). Moreover, similarly to our extended proof sketch of Theorem 5 (§4.1), we can extend our proof sketch of Remark 7 to avoid voter detection, because the former mechanism includes all ballots on the bulletin board and (silently) omits ballots during tallying, and the latter can be disregarded by an adversary that controls ballot collection (hence, the bulletin board).

Remark 7. *Variants of Helios 3.1.4 with the Fiat-Shamir transformation (rather than weak Fiat-Shamir) and ballot weeding do not satisfy* Ballot-Secrecy.

Proof sketch. Neither ballot weeding nor the Fiat-Shamir transformation eliminate the vulnerability we identified in Helios 3.1.4, because related ballots need not be tallied, as shown in the proof sketch of Theorem 5. Hence, we conclude by the proof of that theorem. □

We derive a new exploit (as the following example demonstrates) by extrapolating from the proof sketch of Theorem 5 and an attack by Cortier & Smyth that asserts the following: given a ballot b for vote v, we can abuse malleability to derive a ballot b' for vote v' [53, §3.2.2]. Suppose Alice, Bob and Charlie are voters, and Mallory is an adversary that wants to convince herself that Alice did not vote for a candidate v. Further suppose Alice casts a ballot b_1 for vote v_1, Bob casts a ballot b_2, and Charlie casts a ballot b_3. Moreover, suppose that either Bob or Charlie vote for v. (Thereby avoiding scenarios without any votes for candidate v, i.e., scenarios which inevitably permit Mallory to convince herself that Alice did not vote for candidate v.) Let us assume that votes for v' are not expected. Mallory proceeds as follows: she intercepts ballot b_1, abuses malleability to derive a ballot b such that $v = v_1$ implies b is a vote for v', and casts ballot b. It follows that the tallier will compute the election outcome from bulletin board $\{b, b_2, b_3\}$. (Omitting meaningfully related ballots before tallying does not eliminate the

[19] Cf. https://github.com/benadida/helios-server/issues/8 and https://github.com/benadida/helios-server/issues/35, accessed 29 Mar 2019.

[20] David Bernhard, email communication, c. 2014 and 19 Sep 2017.

vulnerability, because none of the tallied ballots are related.) If the outcome does not contain any votes for v', then Mallory is convinced that Alice did not vote for v.

The exploit is reliant on a particular candidate not receiving any votes. This is trivial to capture in the context of game $\mathsf{Ballot\text{-}Secrecy}$, because the bulletin board is computed by an adversary that casts ballots on behalf of some voters and controls the votes cast by the remaining voters. Beyond the game, candidates will presumably vote for themselves. Thus, for first-past-the-post elections, the exploit's success is probably limited to elections in which voters vote in constituencies and each polling station announces its own outcome (cf. Cortier & Smyth [53, §3.3]).

We have seen that an attack is feasible. Consequently, we cannot prove $\mathsf{Ballot\text{-}Secrecy}$. Whether attacks are likely is subjective, and instead of offering an opinion, let us consider a tale of when an attack might be used in the real-world: Alice meets Charlie, who Alice's family believe is a supporter of a undesirable party. Alice is torn; her love for Charlie is hurting her family. She devises a plan to ensure Charlie is welcomed with open arms: She uses the attack to prove that Charlie did not support the undesirable candidate in the town's election. The concerns of Alice's family are alleviated, Charlie is welcomed, and Alice is finally content. (Content in love at least, she still has numerous manuscripts to finalise, a stack of papers to review, and several meetings that-should-have-been-emails to attend.) Those that believe in the existence of a real-world Alice will surely accept the attack as practical, because observation of a real-world attack surely suffices to demonstrate practicality.

5. Simplifying ballot-secrecy proofs

We have seen that our definitions of ballot secrecy and ballot independence coincide when tallying is additive and tallying proofs are zero-knowledge (Theorem 4). Building upon this result and Proposition 9, we show that tallying cannot harm secrecy when ballots are tallied correctly and tallying proofs are zero-knowledge. That is, $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ satisfies $\mathsf{Ballot\text{-}Secrecy}$ if and only if $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}')$ does, assuming algorithms Tally and Tally' both tally ballots correctly and tallying proofs are zero-knowledge.[21]

Smyth, Frink & Clarkson capture the notion of tallying ballots correctly using function *correct-outcome* [30]. That function uses a *counting quantifier*: A predicate $(\exists^{=\ell} x : P(x))$ that holds exactly when there are ℓ distinct values for x such that $P(x)$ is satisfied [78].[22] Using the counting quantifier, function *correct-outcome* is defined such that $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)[v] = \ell$ iff $\exists^{=\ell} b \in \mathfrak{bb} \setminus \{\perp\} : \exists r : b = \mathsf{Vote}(pk, v, nc, \kappa; r)$, where $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)$ is a vector of length nc and $1 \leqslant v \leqslant nc$. Hence, component v of vector $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)$ equals ℓ iff there exist ℓ ballots for vote v on the bulletin board. The function requires ballots be interpreted for only one candidate, which can be ensured by injectivity.

Definition 6 ($\mathsf{HK\text{-}Injectivity}$)**.** *An election scheme* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *satisfies* honest-key injectivity ($\mathsf{HK\text{-}Injectivity}$)*, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, security parameters κ and computations* $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa); (nc, v, v') \leftarrow \mathcal{A}(pk, \kappa); b \leftarrow \mathsf{Vote}(pk, nc, v, \kappa); b' \leftarrow \mathsf{Vote}(pk, nc, v', \kappa)$ *such that $v \neq v' \wedge b \neq \perp \wedge b' \neq \perp$, we have $b \neq b'$.*

[21] A more general result also holds: $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ satisfies $\mathsf{Ballot\text{-}Secrecy}$ iff $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}')$ satisfies $\mathsf{Ballot\text{-}Secrecy}$, assuming algorithms Tally and Tally' are indistinguishable, in particular, they tally ballots in the same way. However, election schemes that tally ballots incorrectly are not useful, so we forgo generality for practicality.

[22] Variable x is bound by the quantifier and integer ℓ is free.

Equipped with notions of injectivity and of tallying ballots correctly, we formalise a soundness condition asserting that an election scheme tallies ballots correctly (Definition 7), which allows us to formally state that tallying cannot harm ballot independence (Proposition 9).

Definition 7 (Tally-Soundness)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* Tally-Soundness *be the following game.*

```
Tally-Soundness$(\Gamma, \mathcal{A}, \kappa)$ =
    $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
    $(nc, \mathfrak{bb}) \leftarrow \mathcal{A}(pk, \kappa)$;
    $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$;
    return $\mathfrak{v} = correct\text{-}outcome(pk, nc, \mathfrak{bb}, \kappa) \wedge nc \leqslant mc \wedge |\mathfrak{bb}| \leqslant mb$;
```

We say Γ *satisfies* tally soundness (Tally-Soundness)*, if* Γ *satisfies* HK-Injectivity *and for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such that for all security parameters* κ*, we have* $\mathsf{Succ}(\mathsf{Tally\text{-}Soundness}(\Gamma, \mathcal{A}, \kappa)) > 1 - \mathsf{negl}(\kappa)$.

Lemma 8. Tally-Soundness *implies* Additivity.

Proposition 9. *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *and* $\Gamma' = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}')$ *be election schemes. Suppose* Γ *and* Γ' *satisfy* Tally-Soundness. *We have* Γ *satisfies* IND-CVA *iff* Γ' *satisfies* IND-CVA.

Proof. Tally soundness assures us that algorithms Tally and Tally′ produce indistinguishable election outcomes, thus they are interchangeable in the context of game IND-CVA. □

It follows from Proposition 9 that tally soundness suffices for ballot independence of scheme (Setup, Vote, Tally), if there exists an algorithm Tally′ such that (Setup, Vote, Tally′) is an election scheme satisfying tally soundness and ballot independence. We demonstrate the existence of such an algorithm with respect to election scheme Enc2Vote,[23] thereby showcasing the applicability of Proposition 9 for a class of encryption-based election schemes.

Definition 8 (Enc2Vote)**.** *Given an asymmetric encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, we define* $\mathsf{Enc2Vote}(\Pi) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *such that:*

- $\mathsf{Setup}(\kappa)$ *computes* $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa); pk' \leftarrow (pk, \mathfrak{m}); sk' \leftarrow (pk, sk)$*, derives mc as the largest integer such that* $\{0, \dots, mc\} \subseteq \{0\} \cup \mathfrak{m}$ *and for all* $m_0, m_1 \in \{1, \dots, mc\}$ *we have* $|m_0| = |m_1|$*, and outputs* $(pk', sk', p(\kappa), mc)$*, where p is a polynomial function.*
- $\mathsf{Vote}(pk', v, nc, \kappa)$ *parses* pk' *as pair* $(pk, \mathfrak{m})$*, outputting* $\perp$ *if parsing fails or* $v \notin \{1, \dots, nc\} \vee \{1, \dots, nc\} \nsubseteq \mathfrak{m}$*, computes* $b \leftarrow \mathsf{Enc}(pk, v)$*, and outputs b.*
- $\mathsf{Tally}(sk', \mathfrak{bb}, nc, \kappa)$ *initialises* $\mathfrak{v}$ *as a zero-filled vector of length nc, parses* sk' *as pair* (pk, sk)*, outputting* $(\mathfrak{v}, \perp)$ *if parsing fails, computes* **for** $b \in \mathfrak{bb}$ **do** $v \leftarrow \mathsf{Dec}(sk, b)$; **if** $1 \leqslant v \leqslant nc$ **then** $\mathfrak{v}[v] \leftarrow \mathfrak{v}[v] + 1$*, and outputs* $(\mathfrak{v}, \epsilon)$*, where* ϵ *is a constant symbol.*

[23] Our presentation of Enc2Vote extends the presentation by Quaglia & Smyth [61, Definition 7] to make the plaintext space explicit. We also embed the public key inside the private key. (Quaglia & Smyth's formalisation of Enc2Vote builds upon constructions by Bernhard *et al.* [25–28].)

To ensure $\mathsf{Enc2Vote}(\Pi)$ is an election scheme, we require asymmetric encryption scheme Π to produce distinct ciphertexts with overwhelming probability [62, Lemma 3]. Hence, we must restrict the class of asymmetric encryption schemes used to instantiate Enc2Vote. We consider a broad class of schemes that produce distinct ciphertexts with overwhelming probability.

Lemma 10. *Given an asymmetric encryption scheme Π satisfying* IND-CPA*, we have* $\mathsf{Enc2Vote}(\Pi)$ *is an election scheme. Moreover, if Π has perfect correctness, then* $\mathsf{Enc2Vote}(\Pi)$ *satisfies* HK-Injectivity.

A proof of Lemma 10 follows from Smyth's correctness proof [62, Lemma 4] and Quaglia & Smyth's proof of a slightly stronger notion of HK-Injectivity [61, Lemma 2].

Intuitively, given a non-malleable asymmetric encryption scheme Π, election scheme $\mathsf{Enc2Vote}(\Pi)$ derives ballot secrecy from Π until tallying and tallying maintains ballot secrecy by returning only the number of votes for each candidate. Smyth presents a formal proof of ballot secrecy [62, Proposition 5], hence, ballot independence is satisfied too (Theorem 1).

Corollary 11. *Given an asymmetric encryption scheme Π satisfying* IND-PA0*, we have* $\mathsf{Enc2Vote}(\Pi)$ *satisfies* IND-CVA.

The reverse implication of Corollary 11 does not hold. Indeed, we have the following (stronger) result.

Proposition 12. *There exists an asymmetric encryption scheme Π such that* $\mathsf{Enc2Vote}(\Pi)$ *satisfies* Ballot-Secrecy*, but Π does not satisfy* IND-PA0.

To capitalise on Proposition 9, we demonstrate that Enc2Vote produces election schemes satisfying tallying soundness (Lemma 13), assuming "ill-formed" ciphertexts are distinguishable from "well-formed" ciphertexts, and combine our results to derive Theorem 14.

Definition 9. *Given an asymmetric encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, we say Π satisfies* well-definedness*, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function* negl*, such that for all security parameters κ, we have* $\Pr[(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa); c \leftarrow \mathcal{A}(pk, \mathfrak{m}, \kappa) : \mathsf{Dec}(sk, c) \neq \perp \Rightarrow \exists m, r \,.\, m \in \mathfrak{m} \wedge c = \mathsf{Enc}(pk, m; r) \wedge c \neq \perp] > 1 - \mathsf{negl}(\kappa)$.

Tally-Soundness demands that every counted ballot be output by algorithm Vote, i.e., only "well-formed" ballots are counted, "ill-formed" ballots are not. Election scheme $\mathsf{Enc2Vote}(\Pi)$ defines ballots as ciphertexts, hence, when Π satisfies well-definedness, we are assured that only "well-formed" ballots are counted, as Tally-Soundness demands.

Lemma 13. *Given a perfectly-correct asymmetric encryption scheme Π satisfying well-definedness and* IND-CPA*, we have* $\mathsf{Enc2Vote}(\Pi)$ *satisfies* Tally-Soundness.

Theorem 14. *Let Π be an asymmetric encryption scheme,* $\mathsf{Enc2Vote}(\Pi) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$*, and* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}')$ *for some algorithm* Tally' *such that Γ is an election scheme with zero-knowledge tallying proofs. Suppose Π is perfectly correct and satisfies* IND-PA0 *and well-definedness. Further suppose Γ satisfies* Tally-Soundness. *We have Γ satisfies* Ballot-Secrecy.

Proof. Election scheme $\mathsf{Enc2Vote}(\Pi)$ satisfies Tally-Soundness (Lemma 13) and IND-CVA (Corollary 11). Thus, Γ satisfies IND-CVA (Proposition 9) and Ballot-Secrecy (Theorem 4 & Lemma 8). □

We show that tally soundness is implied by universal verifiability in Appendix E. Thus, a special case of Theorem 14 requires universal verifiability rather than tally soundness. It follows that $\mathsf{Ballot\text{-}Secrecy}$ is satisfied by verifiable election schemes that produce ballots by encrypting votes with asymmetric encryption schemes satisfying well-definedness and $\mathsf{IND\text{-}PA0}$. Thereby making proofs of ballot secrecy trivial for a class of encryption-based election schemes. Indeed, we exploit this result in the following section to show that the combination of non-malleable encryption and universal verifiability achieve ballot secrecy.

6. Case study II: Helios Mixnet

Helios Mixnet can be informally modelled as the following election scheme:

Setup generates a key pair for an asymmetric homomorphic encryption scheme, proves correct key generation in zero-knowledge, and outputs the key pair and proof.

Vote enciphers the vote to a ciphertext, proves correct ciphertext construction in zero-knowledge, and outputs the ciphertext coupled with the proof.

Tally selects ballots from the bulletin board for which proofs hold, mixes the ciphertexts in those ballots, decrypts the ciphertexts output by the mix to reveal the election outcome (i.e., the frequency distribution of votes), and announces that outcome, along with zero-knowledge proofs demonstrating correct decryption.

Neither Adida [35] nor Bulens, Giry & Pereira [36] have released an implementation of Helios Mixnet.[24] Tsoukalas *et al.* [79] released *Zeus* as a fork of Helios spliced with mixnet code to derive an implementation,[25] and Yingtong Li released *helios-server-mixnet* as an extension of Zeus with threshold asymmetric encryption and some other minor changes.[26,27]

We can derive Helios Mixnet from $\mathsf{Enc2Vote}(\Pi)$ by replacing its tallying algorithm, and by using an asymmetric encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where algorithm Gen proves correct key generation, and algorithm Enc verifies such proofs, enciphers plaintexts to ciphertexts using a second encryption scheme, proves correct ciphertext construction, and outputs the ciphertext coupled with the proof. However, a blight arises when $\mathsf{Enc2Vote}$ is instantiated with encryption schemes that prove correct key generation.[28] To avoid this blight, we extend $\mathsf{Enc2Vote}$ with such proofs and show that results in Section 5 still hold (Appendix F). This leads us to treat Helios Mixnet as an election scheme built from asymmetric encryption schemes $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and $\Pi_0 = (\mathsf{Gen}, \mathsf{Enc}', \mathsf{Dec}')$ such that:

[24] The planned implementation of Helios Mixnet (https://web.archive.org/web/20160912182802/https://documentation.heliosvoting.org/verification-specs/mixnet-support, published c. 2010, accessed 29 Mar 2019, & https://web.archive.org/web/20110119223848/http://documentation.heliosvoting.org/verification-specs/helios-v3-1, published Dec 2010, accessed 29 Mar 2019) has not been released.

[25] https://github.com/grnet/zeus, accessed 29 Mar 2019.

[26] https://github.com/RunasSudo/helios-server-mixnet, accessed 29 Mar 2019.

[27] The problem of malleable Helios ballots (§4) was discussed with the developers of Zeus and helios-server-mixnet, and they explained that their systems use non-malleable ballots (email communication, Oct & Dec 2017).

[28] Helios Mixnet defines public keys as triples comprising a public key and message space defined by the underlying asymmetric encryption scheme, along with a proof of correct key generation. By comparison, $\mathsf{Enc2Vote}$ defines public keys as pairs comprising a public key and message space defined by the underlying asymmetric encryption scheme (wherein the public key defined by the encryption scheme incorporates some proof of correct key generation). Hence, $\mathsf{Enc2Vote}$ cannot be instantiated to immediately derive Helios Mixnet – we have a slight mismatch of parameters, a minor blight.

- $\mathsf{Setup}(\kappa)$ selects coins s uniformly at random, computes $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa; s)$ and a proof ρ of correct key generation using sk and s as the witness, derives mc as the largest integer such that $\{0, \dots, mc\} \subseteq \{0\} \cup \mathfrak{m}$, computes $pk' \leftarrow (pk, \mathfrak{m}, \rho); sk' \leftarrow (pk, sk)$, and outputs $(pk', sk', p(\kappa), mc)$, where p is a polynomial function.
- $\mathsf{Vote}(pk, v, nc, \kappa)$ parses pk' as a vector $(pk, \mathfrak{m}, \rho)$, outputting $\perp$ if parsing fails, ρ does not verify, $v \notin \{1, \dots, nc\}$, or $\{0, \dots, nc\} \nsubseteq \mathfrak{m}$, computes $b \leftarrow \mathsf{Enc}(pk, v)$, and outputs b.

where

- $\mathsf{Enc}(pk, v)$ selects coins r uniformly at random, computes $c \leftarrow \mathsf{Enc}'(pk, v; r)$ and a proof σ of correct ciphertext construction using v and r as the witness, and outputs (c, σ).
- $\mathsf{Dec}(sk, b)$ parses b as a pair (c, σ), outputting $\perp$ if parsing fails or σ does not verify, computes $v \leftarrow \mathsf{Dec}'(sk, c)$, and outputs v.

It follows that our results (§5) can be applied to trivialise a ballot-secrecy proof: It is known that Π is a non-malleable encryption scheme [71, Theorem 2], assuming the proof system used by algorithm Enc satisfies simulation sound extractability and Π_0 satisfies IND-CPA. Moreover, we have Π satisfies well-definedness, by the former assumption. Furthermore, Smyth has shown that universal verifiability is satisfied [80], hence, Tally-Soundness is satisfied too. Thus, Ballot-Secrecy is satisfied. Thereby providing evidence that our results do indeed make ballot-secrecy proofs trivial.

To formally state our ballot secrecy result, we adopt a set HeliosM'17 of election schemes that includes Helios Mixnet and prove ballot secrecy for every scheme in that set.

Theorem 15. *Each election scheme in set HeliosM'17 satisfies* Ballot-Secrecy.

A proof of Theorem 15 along with a definition of HeliosM'17 appear in Appendix G.

7. Discussion, limitations, and directions for further research

This article advances our understanding of ballot secrecy, establishing a foundation for discussion, debate, and development, especially in terms of the following limitations.

Authentication, re-voting, and unforgeability. Voting systems have traditionally permitted voters to cast at most one ballot. More recent systems permit multiple ballots [31, 39, 41, 43, 81–83], with only the voter's last having influence, enabling voters to change their vote, providing flexibility and aiding education (since voters can "*ask the help of anyone for submitting a random ballot, and then re-voting privately afterwards*" [31, §3.3]). Election schemes are reliant on external authentication services to ensure that only the last ballot of each voter has influence. (E.g., Helios supports authentication via Facebook, Google and Yahoo using OAuth [39].) Such services are assumed to be trusted, they are not intended to exclude attacks that arise when services are subverted (to authenticate non-voter ballots or to assert a ballot other than a voter's last should have influence, for instance). Evaluating whether trust is merited (including determining whether external authentication services ensure that only the last choice of each voter has influence) is beyond the scope of our privacy definitions. Definitions could be extended with an ideal authentication service and external authentication services could be proven equivalent. Ultimately, a reliance on external authentication services introduces a trust assumption. That assumption is eliminated by voting systems that use cryptography to implement their own authentication mechanisms, rendering authorised ballots unforgeable. (E.g., the voting system by Juels, Catalano &

Jakobsson uses a combination of encrypted nonces and plaintext equality tests for authentication [43].) Extending definitions to include voting systems with their own cryptography-based authentication mechanisms and considering the interplay between privacy and re-voting, without trusting an authentication service, would be an interesting direction for further research. The aforementioned extensions necessitate changes to our balanced condition such that only the last choice of each voter is considered. This could be achieved by keeping track of the number of ballots cast by each voter, for example, $\mathcal{O}(id, v_0, v_1)$ could compute $L \leftarrow L \cup \{(id, b, n, v_0, v_1)\}$, where id is a voter identifier, b is the constructed ballot, and $n = |\{b' \mid (id, b', i, v'_0, v'_1) \in L\}|$, i.e., the number n of ballots previously cast with identifier id. And by applying predicate *balanced* to a set containing only voters' last votes, for instance, using $balanced(\mathfrak{bb}, nc, \{b \mid (id, b, i, v_0, v_1) \in L \wedge \forall (id, b', i', v'_0, v'_1) \in L \wedge i \geqslant i'\})$ on the final line of game Ballot-Secrecy. Further investigation of the details would be prudent.

Ballot secrecy notions and interplay with individual verifiability. We know some scenarios inevitably reveal voters' votes, which is why an exception is included in our informal definition of ballot secrecy (a voter's vote is not revealed to anyone, except when the vote can be deduced from the election outcome and any partial knowledge of voters' votes).[29] That exception tolerates revelations in the following example, namely, Alice, Bob and Mallory participate in a referendum, Mallory (who controls ballot collection) discards Bob's ballot, and the outcome has frequency one for each of 'yes' and 'no,' allowing Mallory to deduce Alice's vote. Such revelations are undesirable, but are tolerated by many voting systems and our definition of ballot secrecy. It is open to debate as to whether such voting systems should be branded secret and whether my definition is sound. (I believe so: My definition should only tolerate revelations deducible from outcomes and partial knowledge of votes. As I've asserted, it is open to debate as to whether such tolerance is too broad.) Individual verifiability allows voters to check whether their ballots are collected, allowing detection of some troublesome scenarios. Checks cannot, in themselves, preclude revelations. (Indeed, Bob discovering their ballot is absent, does not preclude revelation of Alice's vote.) Additional properties are necessary. The ability to attribute malice to a particular party (cf. accountability [84]) and to recover from such malice may suffice. (Demanding inclusion of Bob's ballot can avoid revelation of Alice's vote.) However, even with additional properties, there is a reliance on voters performing checks, which few voters do [85, §2.1.6]. Regardless, ballot secrecy should be delivered without regard for privilege, without dependence on actions few are willing and able to perform. Future research may consider such possibilities. Demanding delegatable individual-verifiability checks may provide a fruitful advancement: Envisage voters leaving polling stations and handing over material for checks to voting advocates, for instance.

Distributed tallying. Ballot secrecy does not provide assurances when deviations from the prescribed tallying procedure are possible. Indeed, ballots can be tallied individually to reveal votes. Hence, the tallier must be trusted. Consequently, an election scheme that leaks the ballot-vote relation during tallying can satisfy ballot secrecy, because a trusted tallier will not disclose mappings. E.g., construction Enc2Vote produces election schemes satisfying ballot secrecy, despite revealing such a map during tallying. Additionally, the tallier cannot collude with the adversary, in particularly, they cannot post to the bulletin board. We can design election schemes that distribute the tallier's role amongst several talliers and ensure free-choice assuming at least one tallier tallies ballots in the prescribed manner, other talliers can collude with the adversary. Extending our results in this direction is an opportunity for further

[29] Some revelations are an artefact of announcing the outcome as a frequency distribution of votes (a functional requirement of many nations, which comes at a privacy cost), rather than revealing less information, e.g., just the winning candidate [49, 51, 52].

research; as it stands, the definition is limited. This is a design decision: Distributed tallying adds complexity, increasing the possibility of error; getting the "right" ballot-secrecy definition for a single tallier seems the safer, preferable first step. Ultimately, we would prefer not to trust talliers. Unfortunately, this is only known to be possible for decentralised voting systems, e.g., [86–91], which are designed such that ballots cannot be tallied individually, but are unsuitable for large-scale elections.

Privacy definitions are incomplete: Controlled malleability is tolerable. We have seen that tallying meaningfully related ballots introduces a vulnerability that can be exploited to violate secrecy. Non-malleable ballots serve as a solution and are necessary in election schemes satisfying our definition. More generally, non-malleable ballots are not necessary. Indeed, given an election scheme Γ with non-malleable ballots, let Γ' be derived from Γ by prepending a bit to Γ-ballots and disregarding any ballot that is identical to another except for the first bit before tallying. Intuitively, election scheme Γ' satisfies ballot secrecy, if Γ does. Malleability cannot be exploited in a manner that causes meaningfully related ballots to be tallied; related ballots can be tolerated, as long as they are not counted. Albeit, scheme Γ' cannot satisfy universal verifiability, because voters' votes may be discarded when related ballots are present. This makes a case for adopting non-malleable ballots. Nonetheless, further research could consider a weakening of our ballot secrecy definition that is compatible with such malleability. Without such a weakening, our definition is too strong: it will incorrectly brand some election schemes (with malleable ballots) as insecure, as Γ' demonstrates.

8. Related work

Discussion of ballot secrecy originates from Chaum [92] and the earliest definitions of ballot secrecy are due to Benaloh *et al.* [49, 93, 94].[30] More recently, Bernhard *et al.* propose a series of ballot secrecy definitions: They consider election schemes without tallying proofs [25, 26] and, subsequently, schemes with tallying proofs [27–29, 71]. The definition by Bernhard, Pereira & Warinschi computes tallying proofs using algorithm Tally or a simulator [71], but that definition is too weak and some strengthening is required [29, §3.4 & §4]. (Cortier *et al.* [96, 97] propose a variant of the ballot secrecy definition by Bernhard, Pereira & Warinschi, which is also too weak [29].) By comparison, the definition by Smyth & Bernhard computes tallying proofs using only algorithm Tally [27], but their definition is too strong [29, §3.5] and a weakening is required [28]. We prove that our ballot secrecy definition is strictly stronger than the weakened definition by Smyth & Bernhard (Appendix H). The relative merits of definitions by Smyth & Bernhard [28, Definition 5] and by Bernhard *et al.* [29, Definition 7] are unknown, in particular, it is unknown whether one definition is stronger than the other.

Discussion of ballot independence originates from Gennaro [63] and the apparent relationship between ballot secrecy and ballot independence has been considered. In particular, Benaloh shows that a simplified version of his voting system allows the administrator's private key to be recovered by an adversary who casts a ballot as a function of other voters' ballots [94, §2.9]. More generally, Sako & Kilian [98, §2.4], Michels & Horster [99, §3], Wikström [100–102] and Cortier & Smyth [53, 54] discuss how malleable ballots can be abused to compromise ballot secrecy. The first definition of ballot independence seems to be due to Smyth & Bernhard [27, 28]. Moreover, Smyth & Bernhard formally prove relations between their definitions of ballot secrecy and ballot independence. Independence has also been studied

[30] Quaglia & Smyth present a tutorial-style introduction to modelling ballot secrecy [95], and Bernhard *et al.* survey ballot secrecy definitions [29, 57].

beyond elections, e.g., [64], and the possibility of compromising security in the absence of independence has been considered, e.g., [66, 67, 103–106].

All of the ballot secrecy definitions by Bernhard *et al.* [25–29, 71] and the ballot independence definition by Smyth & Bernhard [27, 28] focus on detecting vulnerabilities exploitable by adversaries that control some voters. Vulnerabilities that require control of the bulletin board or the communication channel are not detected, i.e., the bulletin board is implicitly assumed to operate in accordance with the election scheme's rules and the communication channel is implicitly assumed to be secure. This introduces a trust assumption. Under this assumption, Smyth & Bernhard prove that non-malleable ballots are not necessary for ballot secrecy [27, §4.3]. By comparison, we prove that non-malleable ballots are necessary for ballot secrecy without this trust assumption. Thus, our definition of ballot secrecy improves upon definitions by Bernhard *et al.* by detecting more vulnerabilities.

Confidence in our ballot secrecy definition might be improved by proving equivalence with a simulation-based definition of ballot secrecy. However, it is unclear how to formulate a suitable simulation-based definition. Bernhard *et al.* propose an ideal functionality that "collects all votes from the voters, then computes and announces the [election outcome]" [29, §1],[31] but a voting system satisfying ballot secrecy need not be equivalent, because ballot secrecy does not guarantee correct computation of the election outcome. Equivalence can perhaps be shown between their ideal functionality and voting systems satisfying ballot secrecy and some soundness condition (e.g., Tally-Soundness). Albeit, voting systems that bound the number of ballots or candidates, e.g., Helios, may not be equivalent, because soundness conditions (such as Tally-Soundness) need only provide guarantees when operating within the aforementioned bounds. Thus, bounds need to be taken into consideration. Moreover, no voting system is equivalent to the ideal functionality by Bernhard *et al.* in the presence of an adversary that controls the bulletin board, because such an adversary can discard ballots, which creates a trivial distinction.[32] Nonetheless, equivalence can perhaps be shown for verifiable voting systems in cases where the number of ballots and candidates are bounded, and all voters successfully verify the presence of their ballot, but such a result has no practical relevance, because it is well-known that many voters do not perform checks necessary for verifiability [85, §2.1.6]. Alternatively, the ideal functionality could be weakened such that a discarded ballot in the real functionality corresponds to a discarded vote in the ideal functionality. We leave further exploration of simulation-based definitions of ballot secrecy as a possible extension for future work.

Bulens, Giry & Pereira pose the investigation of voting systems which allow casting of meaningfully related ballots as a means for voters to delegate candidate selection, whilst preserving ballot secrecy, as an interesting research question [36, §3.2]. Desmedt & Chaidos claim to provide such a system [107]. Smyth & Bernhard critique their work and argue that the security results do not support their claims [27, §5.1]. We have shown that non-malleable ballots are necessary to satisfy Ballot-Secrecy (§3.3), providing a negative result for the question posed by Bulens, Giry & Pereira.

Some of the ideas presented in this article previously appeared in my technical report [108] and an extended version of that technical report by Bernhard & Smyth [38]. In particular, the limitations of ballot secrecy definitions by Bernhard *et al.* were identified and Definition 2 is based upon my earlier definition [108]. The main distinction between Definition 2 and the earlier definition [108, Definition 3]

[31] In the context of voting systems that announce the chosen representative (rather than the frequency distribution of votes), a stronger ideal functionality might announce the chosen representative.

[32] The real functionality by Bernhard *et al.* does not capture adversaries that control ballot collection. Thus, the relation they prove between their game-based and simulation-based definitions of ballot secrecy does not preclude vulnerabilities exploitable by such adversaries. Indeed, proving such relations does not guarantee the absence of vulnerabilities.

is syntax: This article adopts syntax for election schemes from Smyth, Frink & Clarkson [30], whereas the earlier definition adopts syntax by Smyth & Bernhard [27, 28]. The change in syntax is motivated by the superiority of syntax by Smyth, Frink & Clarkson. Moreover, we can capitalise upon results by Smyth, Frink & Clarkson [30] and Quaglia & Smyth [61].

Following the initial release of these results [109, 110], Cortier *et al.* [111] presented a machine-checked proof that variants of Helios satisfy the notion of ballot secrecy by Bernhard *et al.* [29]. As discussed above, that notion is too weak: It cannot detect vulnerabilities that require control of ballot collection. Thus, our proof is more appropriate. Nonetheless, their proof builds upon similar ideas. In particular, their proof is dependent upon non-malleable ballots and zero-knowledge tallying proofs.

Further to their earlier work [112], Smyth & Hanatani show that Helios'16-ballots are essentially non-malleable ciphertexts and build upon this article to further simplify ballot-secrecy proofs [113], presenting a proof in just two pages, compared to the five pages used here.

Quaglia & Smyth [40] extend game Ballot-Secrecy to syntax with voter credentials [30, Definition 6]. Moreover, they define a transformation to that syntax from our syntax (Definition 1) and prove that their transformation preserves secrecy and verifiability. Furthermore, they apply their transformation to Helios. The resulting scheme is similar Helios-C [42], albeit Quaglia & Smyth observe that Helios-C does not satisfy ballot secrecy when the adversary controls ballot collection, whereas the scheme derived by applying their transformation does.

Beyond the computational model of security, Delaune, Kremer & Ryan formulate a definition of ballot secrecy in the applied pi calculus [114, 115] and Smyth *et al.* show that this definition is amenable to automated reasoning [116–120]. An alternative definition is proposed by Cremers & Hirschi, along with sufficient conditions which are also amenable to automated reasoning [121]. Albeit, the scope of automated reasoning is limited by analysis tools (e.g., ProVerif [122]), because the function symbols and equational theory used to model cryptographic primitives might not be suitable for automated analysis (cf. [123–125]).

Ballot secrecy formalises a notion of free-choice assuming ballots are constructed and tallied in the prescribed manner. Moreover, our definition of ballot secrecy assumes the adversary's capabilities are limited to casting ballots on behalf of some voters and controlling the votes cast by any remaining voters. We have seen that Helios'16 satisfies our definition, but ballot secrecy does not ensure free-choice when an adversary is able to communicate with voters, nor when voters deviate from the prescribed voting procedure to follow instructions provided by an adversary. Indeed, the coins used for encryption serve as proof of how a voter voted in Helios and the voter may communicate those coins to the adversary. Stronger notions of free-choice, such as receipt-freeness [47, 114, 126–128] and coercion resistance [129–133], are needed in the presence of such adversaries. Appendix I introduces these notions, proves that our syntax cannot be used to construct (interesting) schemes satisfying them, and discusses variants of our syntax that can.

McCarthy, Smyth & Quaglia [134] have shown that auction schemes can be constructed from election schemes, and Quaglia & Smyth [61] provide a generic construction for auction schemes from election schemes. Moreover, Quaglia & Smyth adapt our definition of ballot secrecy to a definition of bid secrecy, and prove that auction schemes produced by their construction satisfy bid secrecy. (Similarly, they adapt the definition of verifiability by Smyth, Frink & Clarkson [30] to a definition of verifiability for auctions, and prove that their construction produces schemes satisfying verifiability.) Thus, this research has applications beyond voting.

9. Conclusion

This work was initiated by a desire to eliminate the trust assumptions placed upon the bulletin board and the communication channel in definitions of ballot secrecy by Bernhard *et al.* and the definition of ballot independence by Smyth & Bernhard. This necessitated the introduction of new security definitions. The definition of ballot secrecy was largely constructed from intuition, with inspiration from indistinguishability games for asymmetric encryption and existing definitions of ballot secrecy. Moreover, the definition was guided by the desire to strengthen existing definitions of ballot secrecy. The definition of ballot independence was inspired by the realisation that independence requires non-malleable ballots, which enabled definitions to be constructed as straightforward adaptations of non-malleability and indistinguishability definitions for asymmetric encryption. The former adaptation being a more natural formulation of ballot independence and the latter being simpler.

Relationships between security definitions aid our understanding and offer insights that facilitate the construction of secure schemes. This prompted the study of relations between our definitions of ballot secrecy and ballot independence, resulting in a proof that non-malleable ballots are necessary for our definitions. We also proved non-malleable ballots suffice for our definition of ballot secrecy in election schemes with zero-knowledge tallying proofs. Moreover, we established a separation result demonstrating that our ballot secrecy definition is strictly stronger than our ballot independence definition.

We proved that Helios'16 uses non-malleable ballots and a proof that Helios'16 satisfies ballot secrecy follows from our results. This proof is particularly significant due to the use of Helios in binding elections, and we encourage developers to base future releases on this variant, since it is the only variant of Helios which is proven to satisfy both ballot secrecy and verifiability.

Proving ballot secrecy is expensive: It requires a significant devotion of time by experts. Indeed, Cortier *et al.* devoted one person-year to their machine-checked proof. Thus, sufficient conditions for ballot secrecy are highly desirable. We have established that non-malleable ballots are sufficient for our definition of ballot secrecy in election schemes with zero-knowledge tallying proofs and this simplified our ballot-secrecy proof for Helios'16. We have also established that building election schemes from non-malleable asymmetric encryption schemes suffices for our definition of ballot secrecy if tallying is additive (a condition implied by verifiability), and this trivialised our ballot-secrecy proof for Helios Mixnet. Thereby demonstrating the possibility of simple, inexpensive proofs.

This article delivers a definition of ballot secrecy that has been useful in detecting subtle vulnerabilities in voting systems, and has led to the development of election schemes that are proven secure. Thereby demonstrating the necessity of appropriate security definitions and accompanying analysis to ensure security of voting systems, especially those used in binding elections. I hope this article will simplify future proofs of ballot secrecy and, ultimately, aid democracy-builders in deploying their systems securely. There is still work to be done, especially in seeking definitions that consider distributed tallying.

Acknowledgements

I am grateful to David Bernhard and to Elizabeth Quaglia for extensive discussions, feedback, and, more generally, their help in extending my knowledge of cryptography. In addition, I am grateful to Constantin Cătălin Drăgan for explaining subtleties of his work and to Maxime Meyer for his careful proofreading. I am also grateful to JCS reviewers and editors: it's clear from reviews that this article has been thoroughly studied. I appreciate the detail that reviewers have gone to. I also appreciate the actionable points listed by my associate editor, which expedited time required for revision. All efforts have helped improve this article and any remaining issues are my own. This work received financial support from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC project *CRYSP* (259639) and from the Luxembourg National Research Fund (FNR) under the FNR-INTER-VoteVerif project (10415467). Work was partly performed at INRIA and the University of Luxembourg.

References

[1] A. Lijphart and B. Grofman, *Choosing an electoral system: Issues and Alternatives*, Praeger, 1984.

[2] T. Saalfeld, On Dogs and Whips: Recorded Votes, in: *Parliaments and Majority Rule in Western Europe*, H. Döring, ed., St. Martin's Press, 1995, Chapter 16.

[3] A. Gumbel, *Steal This Vote: Dirty Elections and the Rotten History of Democracy in America*, Nation Books, 2005.

[4] R.M. Alvarez and T.E. Hall, *Electronic Elections: The Perils and Promises of Digital Democracy*, Princeton University Press, 2010.

[5] United Nations, *Universal Declaration of Human Rights*, 1948.

[6] Organization for Security and Co-operation in Europe, *Document of the Copenhagen Meeting of the Conference on the Human Dimension of the CSCE*, 1990.

[7] Organization of American States, *American Convention on Human Rights, "Pact of San Jose, Costa Rica"*, 1969.

[8] J. Lepore, Rock, Paper, Scissors: How we used to vote, *Annals of Democracy, The New Yorker* (2008).

[9] J. Mill, The Ballot, in: *The Westminster Review*, Vol. 13, Robert Heward, 1830.

[10] R. Bertrand, J.-L. Briquet and P. Pels, Introduction: Towards a Historical Ethnography of Voting, in: *The Hidden History of the Secret Ballot*, Indiana University Press, 2007.

[11] P. Brent, The Australian ballot: Not the secret ballot, *Australian Journal of Political Science* **41**(1) (2006), 39–50.

[12] E.C. Bjornlund, *Beyond Free and Fair: Monitoring Elections and Building Democracy*, Woodrow Wilson Center Press / Johns Hopkins University Press, 2004.

[13] J.G. Kelley, *Monitoring Democracy: When International Election Observation Works, and Why It Often Fails*, Princeton University Press, 2012.

[14] P. Norris, *Why Elections Fail*, Cambridge University Press, 2015.

[15] C.A. Neff and J. Adler, Verifiable e-Voting: Indisputable electronic elections at polling places, Technical Report, Vote-Here, 2003.

[16] T. Kohno, A. Stubblefield, A.D. Rubin and D.S. Wallach, Analysis of an Electronic Voting System, in: *S&P'04: 25th Security and Privacy Symposium*, IEEE Computer Society, 2004, pp. 27–40.

[17] R. Gonggrijp and W.-J. Hengeveld, Studying the Nedap/Groenendaal ES3B Voting Computer: A Computer Security Perspective, in: *EVT'07: Electronic Voting Technology Workshop*, USENIX Association, 2007.

[18] D. Bowen, Secretary of State Debra Bowen Moves to Strengthen Voter Confidence in Election Security Following Top-to-Bottom Review of Voting Systems, 2007, California Secretary of State, press release DB07:042.

[19] S. Wolchok, E. Wustrow, J.A. Halderman, H.K. Prasad, A. Kankipati, S.K. Sakhamuri, V. Yagati and R. Gonggrijp, Security Analysis of India's Electronic Voting Machines, in: *CCS'10: 17th ACM Conference on Computer and Communications Security*, ACM Press, 2010, pp. 1–14.

[20] S. Wolchok, E. Wustrow, D. Isabel and J.A. Halderman, Attacking the Washington, D.C. Internet Voting System, in: *FC'12: 16th International Conference on Financial Cryptography and Data Security*, LNCS, Vol. 7397, Springer, 2012, pp. 114–128.

[21] D. Springall, T. Finkenauer, Z. Durumeric, J. Kitcat, H. Hursti, M. MacAlpine and J.A. Halderman, Security Analysis of the Estonian Internet Voting System, in: *CCS'14: 21st ACM Conference on Computer and Communications Security*, ACM Press, 2014, pp. 703–715.

[22] UK Electoral Commission, *Key issues and conclusions: May 2007 electoral pilot schemes*, 2007.

[23] Bundesverfassungsgericht (Germany's Federal Constitutional Court), *Use of voting computers in 2005 Bundestag election unconstitutional*, 2009, Press release 19/2009.

[24] D.W. Jones and B. Simons, *Broken Ballots: Will Your Vote Count?*, CSLI Lecture Notes, Vol. 204, Center for the Study of Language and Information, Stanford University, 2012.

[25] D. Bernhard, V. Cortier, O. Pereira, B. Smyth and B. Warinschi, Adapting Helios for provable ballot privacy, in: *ESORICS'11: 16th European Symposium on Research in Computer Security*, LNCS, Vol. 6879, Springer, 2011, pp. 335–354.

[26] D. Bernhard, O. Pereira and B. Warinschi, On Necessary and Sufficient Conditions for Private Ballot Submission, 2012, Cryptology ePrint Archive, Report 2012/236 (version 20120430:154117b).

[27] B. Smyth and D. Bernhard, Ballot secrecy and ballot independence coincide, in: *ESORICS'13: 18th European Symposium on Research in Computer Security*, LNCS, Vol. 8134, Springer, 2013, pp. 463–480.

[28] B. Smyth and D. Bernhard, Ballot secrecy and ballot independence: definitions and relations, 2014, Cryptology ePrint Archive, Report 2013/235 (version 20141010:082554).

[29] D. Bernhard, V. Cortier, D. Galindo, O. Pereira and B. Warinschi, SoK: A comprehensive analysis of game-based ballot privacy definitions, in: *S&P'15: 36th Security and Privacy Symposium*, IEEE Computer Society, 2015, pp. 499–516.

[30] B. Smyth, S. Frink and M.R. Clarkson, Election Verifiability: Cryptographic Definitions and an Analysis of Helios and JCJ, 2017, Cryptology ePrint Archive, Report 2015/233 (version 20170213:132559).

[31] B. Adida, O. Marneffe, O. Pereira and J. Quisquater, 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.

[32] J. Benaloh, S. Vaudenay and J. Quisquater, Final Report of IACR Electronic Voting Committee, 2010, International Association for Cryptologic Research. http://www.iacr.org/elections/eVoting/finalReportHelios_2010-09-27.html.

[33] S. Haber, J. Benaloh and S. Halevi, The Helios e-Voting Demo for the IACR, 2010, International Association for Cryptologic Research. http://www.iacr.org/elections/eVoting/heliosDemo.pdf.

[34] C. Staff, ACM's 2014 General Election: Please Take This Opportunity to Vote, *Communications of the ACM* **57**(5) (2014), 9–17.

[35] B. Adida, Helios: Web-based Open-Audit Voting, in: *USENIX Security'08: 17th USENIX Security Symposium*, USENIX Association, 2008, pp. 335–348.

[36] P. Bulens, D. Giry and O. Pereira, Running Mixnet-Based Elections with Helios, in: *EVT/WOTE'11: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*, USENIX Association, 2011.

[37] B. Smyth, First-past-the-post suffices for ranked voting, 2017, https://bensmyth.com/publications/2017-FPTP-suffices-for-ranked-voting/.

[38] D. Bernhard and B. Smyth, Ballot secrecy with malicious bulletin boards, 2015, Cryptology ePrint Archive, Report 2014/822 (version 20150413:170300).

[39] M. Meyer and B. Smyth, Exploiting re-voting in the Helios election system, *Information Processing Letters* (2019), 14–19.

[40] E.A. Quaglia and B. Smyth, Authentication with weaker trust assumptions for voting systems, in: *AFRICACRYPT'18: 10th International Conference on Cryptology in Africa*, LNCS, Springer, 2018.

[41] B. Smyth, Athena: A verifiable, coercion-resistant voting system with linear complexity, 2019.

[42] V. Cortier, D. Galindo, S. Glondu and M. Izabachène, Election Verifiability for Helios under Weaker Trust Assumptions, in: *ESORICS'14: 19th European Symposium on Research in Computer Security*, LNCS, Vol. 8713, Springer, 2014, pp. 327–344.

[43] A. Juels, D. Catalano and M. Jakobsson, Coercion-Resistant Electronic Elections, in: *Towards Trustworthy Elections: New Directions in Electronic Voting*, D. Chaum, M. Jakobsson, R.L. Rivest and P.Y. Ryan, eds, LNCS, Vol. 6000, Springer, 2010, pp. 37–63.

[44] C.A. Neff, Practical High Certainty Intent Verification for Encrypted Votes, Technical Report, VoteHere, 2004.

[45] D. Chaum, P.Y. Ryan and S. Schneider, A Practical Voter-Verifiable Election Scheme, in: *ESORICS'05: 10th European Symposium On Research In Computer Security*, LNCS, Vol. 3679, Springer, 2005, pp. 118–139.

[46] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R.L. Rivest, P.Y.A. Ryan, E. Shen and A.T. Sherman, Scantegrity II: End-to-end Verifiability for Optical Scan Election Systems Using Invisible Ink Confirmation Codes, in: *EVT'08: Electronic Voting Technology Workshop*, USENIX Association, 2008.

[47] T. Moran and M. Naor, Receipt-Free Universally-Verifiable Voting with Everlasting Privacy, in: *CRYPTO'06: 26th International Cryptology Conference*, LNCS, Vol. 4117, Springer, 2006, pp. 373–392.

[48] M. Clarkson, B. Hay, M. Inge, abhi shelat, D. Wagner and A. Yasinsac, Software Review and Security Analysis of Scytl Remote Voting Software, http://election.dos.state.fl.us/voting-systems/pdf/FinalReportSept19.pdf.

[49] J. Benaloh and M. Yung, Distributing the Power of a Government to Enhance the Privacy of Voters, in: *PODC'86: 5th Principles of Distributed Computing Symposium*, ACM Press, 1986, pp. 52–62.

[50] A. Hevia and M.A. Kiwi, Electronic Jury Voting Protocols, in: *LATIN'02: Theoretical Informatics*, LNCS, Vol. 2286, Springer, 2002, pp. 415–429.

[51] A. Hevia and M.A. Kiwi, Electronic jury voting protocols, *Theoretical Computer Science* **321**(1) (2004), 73–94.

[52] Y. Desmedt and K. Kurosawa, Electronic Voting: Starting Over?, in: *ISC'05: International Conference on Information Security*, LNCS, Vol. 3650, Springer, 2005, pp. 329–343.

[53] V. Cortier and B. Smyth, Attacking and fixing Helios: An analysis of ballot secrecy, *Journal of Computer Security* **21**(1) (2013), 89–148.

[54] V. Cortier and B. Smyth, Attacking and fixing Helios: An analysis of ballot secrecy, in: *CSF'11: 24th Computer Security Foundations Symposium*, IEEE Computer Society, 2011, pp. 297–311.

[55] B. Smyth and V. Cortier, A note on replay attacks that violate privacy in electronic voting schemes, Technical Report, RR-7643, INRIA, 2011.

[56] B. Smyth, Replay attacks that violate ballot secrecy in Helios, 2012, Cryptology ePrint Archive, Report 2012/185.

[57] D. Bernhard, V. Cortier, D. Galindo, O. Pereira and B. Warinschi, A comprehensive analysis of game-based ballot privacy definitions, 2015, Cryptology ePrint Archive, Report 2015/255 (version 20150319:100626).

[58] B. Adida and C.A. Neff, Ballot Casting Assurance, in: *EVT'06: Electronic Voting Technology Workshop*, USENIX Association, 2006.

[59] M. Bellare, A. Desai, E. Jokipii and P. Rogaway, A Concrete Security Treatment of Symmetric Encryption, in: *FOCS'97: 38th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, 1997, pp. 394–403.

[60] M. Bellare and P. Rogaway, Symmetric Encryption, in: *Introduction to Modern Cryptography*, 2005, Chapter 4. http://cseweb.ucsd.edu/~mihir/cse207/w-se.pdf.

[61] E.A. Quaglia and B. Smyth, Secret, verifiable auctions from elections, *Theoretical Computer Science* **730** (2018), 44–92.

[62] B. Smyth, A foundation for secret, verifiable elections, 2018, Cryptology ePrint Archive, Report 2018/225 (version 20180301:164045).

[63] R. Gennaro, Achieving independence efficiently and securely, in: *PODC'95: 14th Principles of Distributed Computing Symposium*, ACM Press, 1995, pp. 130–136.

[64] B. Chor, S. Goldwasser, S. Micali and B. Awerbuch, Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults, in: *FOCS'85: 26th Foundations of Computer Science Symposium*, IEEE Computer Society, 1985, pp. 383–395.

[65] M. Bellare and A. Sahai, Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization, in: *CRYPTO'99: 19th International Cryptology Conference*, LNCS, Vol. 1666, Springer, 1999, pp. 519–536.

[66] D. Dolev, C. Dwork and M. Naor, Non-Malleable Cryptography, in: *STOC'91: 23rd Theory of computing Symposium*, ACM Press, 1991, pp. 542–552.

[67] D. Dolev, C. Dwork and M. Naor, Nonmalleable Cryptography, *Journal on Computing* **30**(2) (2000), 391–437.

[68] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, Relations Among Notions of Security for Public-Key Encryption Schemes, in: *CRYPTO'98: 18th International Cryptology Conference*, LNCS, Vol. 1462, Springer, 1998, pp. 26–45.

[69] V. Shoup, Sequences of games: a tool for taming complexity in security proofs, 2004.

[70] N. Chang-Fong and A. Essex, The Cloudier Side of Cryptographic End-to-end Verifiable Voting: A Security Analysis of Helios, in: *ACSAC'16: 32nd Annual Conference on Computer Security Applications*, ACM Press, 2016, pp. 324–335.

[71] D. Bernhard, O. Pereira and B. Warinschi, 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, Springer, 2012, pp. 626–643.

[72] D. Bernhard, Zero-Knowledge Proofs in Theory and Practice., PhD thesis, Department of Computer Science, University of Bristol, 2014.

[73] B. Smyth and A. Pironti, 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.).

[74] B. Smyth and A. Pironti, Truncating TLS Connections to Violate Beliefs in Web Applications, Technical Report, hal-01102013, INRIA, 2015.

[75] M. Bellare and P. Rogaway, Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols, in: *CCS'93: 1st ACM Conference on Computer and Communications Security*, ACM, 1993, pp. 62–73. ISBN 0-89791-629-8.

[76] R. Küsters, T. Truderung and A. Vogt, Clash Attacks on the Verifiability of E-Voting Systems, in: *S&P'12: 33rd IEEE Symposium on Security and Privacy*, IEEE Computer Society, 2012, pp. 395–409.

[77] D. Bernhard and B. Warinschi, Cryptographic Voting — A Gentle Introduction, in: *Foundations of Security Analysis and Design VII*, LNCS, Vol. 8604, Springer, 2014, pp. 167–211.

[78] N. Schweikardt, Arithmetic, First-order Logic, and Counting Quantifiers, *ACM Transactions on Computational Logic* **6**(3) (2005), 634–671.

[79] G. Tsoukalas, K. Papadimitriou, P. Louridas and P. Tsanakas, From Helios to Zeus, *Journal of Election Technology and Systems* **1**(1) (2013).

[80] B. Smyth, Verifiability of Helios Mixnet, in: *Voting'18: 3rd Workshop on Advances in Secure Electronic Voting*, LNCS, Springer, 2018.

[81] E. Maaten, Towards remote e-voting: Estonian case, *Electronic Voting in Europe-Technology, Law, Politics and Society* **47** (2004), 83–100.

[82] K. Gjøsteen, *The Norwegian Internet Voting Protocol*, in: *VoteID'11: 3rd international conference on e-voting and identity*, Springer, 2012, pp. 1–18.

[83] G.V. Post, Using re-voting to reduce the threat of coercion in elections, *Electronic Government, an International Journal* **7**(2) (2010), 168–182.

[84] R. Küsters, T. Truderung and A. Vogt, Accountability: Definition and Relationship to Verifiability, in: *CCS'10: 17th ACM Conference on Computer and Communications Security*, ACM Press, 2010, pp. 526–535.

[85] M. Bernhard, J. Benaloh, J.A. Halderman, R.L. Rivest, P.Y.A. Ryan, P.B. Stark, V. Teague, P.L. Vora and D.S. Wallach, Public Evidence from Secret Ballots, in: *E-Vote-ID'17: 10th International Conference for Electronic Voting*, LNCS, Springer, 2017, pp. 84–109.

[86] B. Schoenmakers, A Simple Publicly Verifiable Secret Sharing Scheme and its Application to Electronic Voting, in: *CRYPTO'99: 19th International Cryptology Conference*, LNCS, Vol. 1666, Springer, 1999, pp. 148–164.

[87] A. Kiayias and M. Yung, Self-tallying Elections and Perfect Ballot Secrecy, in: *PKC'01: 3rd International Workshop on Practice and Theory in Public Key Cryptography*, LNCS, Vol. 2274, Springer, 2002, pp. 141–158.

[88] J. Groth, Efficient Maximal Privacy in Boardroom Voting and Anonymous Broadcast, in: *FC'04: 8th International Conference on Financial Cryptography*, LNCS, Vol. 3110, Springer, 2004, pp. 90–104.

[89] F. Hao, P.Y.A. Ryan and P. Zieliński, Anonymous voting by two-round public discussion, *Journal of Information Security* **4**(2) (2010), 62–67.

[90] D. Khader, B. Smyth, P.Y.A. Ryan and F. Hao, A Fair and Robust Voting System by Broadcast, in: *EVOTE'12: 5th International Conference on Electronic Voting*, Lecture Notes in Informatics, Vol. 205, Gesellschaft für Informatik, 2012, pp. 285–299.

[91] S. Khazaei and M. Rezaei-Aliabadi, A rigorous security analysis of a decentralized electronic voting protocol in the universal composability framework, *Journal of Information Security and Applications* **43** (2018), 99–109.

[92] D.L. Chaum, Untraceable electronic mail, return addresses, and digital pseudonyms, *Communications of the ACM* **24** (1981), 84–90.

[93] J.C. Benaloh and D. Tuinstra, Receipt-free secret-ballot elections, in: *STOC'94: 26th Theory of computing Symposium*, ACM Press, 1994, pp. 544–553.

[94] J. Benaloh, Verifiable Secret-Ballot Elections, PhD thesis, Department of Computer Science, Yale University, 1996.

[95] E.A. Quaglia and B. Smyth, A short introduction to secrecy and verifiability for elections, 2018, arXiv, Report 1702.03168.

[96] V. Cortier, D. Galindo, S. Glondu and M. Izabachene, A generic construction for voting correctness at minimum cost - Application to Helios, 2013, Cryptology ePrint Archive, Report 2013/177 (version 20130521:145727).

[97] V. Cortier, D. Galindo, S. Glondu and M. Izabachene, Distributed ElGamal à la Pedersen: Application to Helios, in: *WPES'13: Workshop on Privacy in the Electronic Society*, ACM Press, 2013, pp. 131–142.

[98] K. Sako and J. Kilian, Receipt-Free Mix-Type Voting Scheme: A practical solution to the implementation of a voting booth, in: *EUROCRYPT'95: 12th International Conference on the Theory and Applications of Cryptographic Techniques*, LNCS, Vol. 921, Springer, 1995, pp. 393–403.

[99] M. Michels and P. Horster, Some Remarks on a Receipt-Free and Universally Verifiable Mix-Type Voting Scheme, in: *ASIACRYPT'96: International Conference on the Theory and Application of Cryptology and Information Security*, LNCS, Vol. 1163, Springer, 1996, pp. 125–132.

[100] D. Wikström, Simplified Submission of Inputs to Protocols, 2006, Cryptology ePrint Archive, Report 2006/259.

[101] D. Wikström, Simplified Submission of Inputs to Protocols, in: *SCN'08: 6th International Conference on Security and Cryptography for Networks*, LNCS, Vol. 5229, Springer, 2008, pp. 293–308.

[102] D. Wikström, *Verificatum: How to Implement a Stand-alone Verifier for the Verificatum Mix-Net (VMN Version 3.0.2)*, 2016, http://www.verificatum.com/files/vmnum-3.0.2.pdf.

[103] B. Chor and M.O. Rabin, Achieving Independence in Logarithmic Number of Rounds, in: *PODC'87: 6th Principles of Distributed Computing Symposium*, ACM Press, 1987, pp. 260–268.

[104] B. Pfitzmann and A. Pfitzmann, How to Break the Direct RSA-Implementation of Mixes, in: *EUROCRYPT'89: 6th International Conference on the Theory and Applications of Cryptographic Techniques*, LNCS, Vol. 434, Springer, 1989, pp. 373–381.

[105] B. Pfitzmann, Breaking Efficient Anonymous Channel, in: *EUROCRYPT'94: 11th International Conference on the Theory and Applications of Cryptographic Techniques*, LNCS, Vol. 950, Springer, 1994, pp. 332–340.

[106] R. Gennaro, A Protocol to Achieve Independence in Constant Rounds, *IEEE Transactions on Parallel and Distributed Systems* **11**(7) (2000), 636–647.

[107] Y. Desmedt and P. Chaidos, Applying Divertibility to Blind Ballot Copying in the Helios Internet Voting System, in: *ESORICS'12: 17th European Symposium on Research in Computer Security*, LNCS, Vol. 7459, Springer, 2012, pp. 433–450.

[108] B. Smyth, Ballot secrecy with malicious bulletin boards, 2014, Cryptology ePrint Archive, Report 2014/822 (version 20141012:004943).

[109] B. Smyth, Secrecy and independence for election schemes, 2015, Cryptology ePrint Archive, Report 2015/942 (version 20150928:195428).

[110] B. Smyth, Secrecy and independence for election schemes, 2016, Cryptology ePrint Archive, Report 2015/942 (version 20160713:142934).

[111] V. Cortier, B. Schmidt, C.C. Drăgan, P.-Y. Strub, F. Dupressoir and B. Warinschi, Machine-Checked Proofs of Privacy for Electronic Voting Protocols, in: *S&P'17: 37th IEEE Symposium on Security and Privacy*, IEEE Computer Society, 2017.

[112] B. Smyth, Y. Hanatani and H. Muratani, NM-CPA secure encryption with proofs of plaintext knowledge, in: *IWSEC'15: 10th International Workshop on Security*, LNCS, Vol. 9241, Springer, 2015, pp. 115–134.

[113] B. Smyth and Y. Hanatani, Non-malleable encryption with proofs of plaintext knowledge and applications to voting, *International Journal of Security and Networks* **14**(4) (2019), 191–204.

[114] S. Delaune, S. Kremer and M. Ryan, Coercion-Resistance and Receipt-Freeness in Electronic Voting, in: *CSFW'06: 19th Computer Security Foundations Workshop*, IEEE Computer Society, 2006, pp. 28–42.

[115] S. Delaune, S. Kremer and M.D. Ryan, Verifying privacy-type properties of electronic voting protocols, *Journal of Computer Security* **17**(4) (2009), 435–487.

[116] S. Delaune, M.D. Ryan and B. Smyth, Automatic verification of privacy properties in the applied pi-calculus, in: *IFIPTM'08: 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security*, International Federation for Information Processing (IFIP), Vol. 263, Springer, 2008, pp. 263–278.

[117] P. Klus, B. Smyth and M.D. Ryan, ProSwapper: Improved equivalence verifier for ProVerif, 2010, http://www.bensmyth.com/proswapper.php.

[118] B. Smyth, Formal verification of cryptographic protocols with automated reasoning, PhD thesis, School of Computer Science, University of Birmingham, 2011.

[119] B. Blanchet and B. Smyth, Automated reasoning for equivalences in the applied pi calculus with barriers, in: *CSF'16: 29th Computer Security Foundations Symposium*, IEEE Computer Society, 2016, pp. 310–324.

[120] B. Blanchet and B. Smyth, Automated reasoning for equivalences in the applied pi calculus with barriers, *Journal of Computer Security* **26**(3) (2018), 367–422.

[121] C. Cremers and L. Hirschi, Improving Automated Symbolic Analysis for E-voting Protocols: A Method Based on Sufficient Conditions for Ballot Secrecy, 2017, arXiv, Report 1709.00194.

[122] B. Blanchet, B. Smyth, V. Cheval and M. Sylvestre, *ProVerif 1.96: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial*, 2016.

[123] S. Delaune, S. Kremer, M.D. Ryan and G. Steel, Formal analysis of protocols based on TPM state registers, in: *CSF'11: 24th Computer Security Foundations Symposium*, IEEE Computer Society, 2011, pp. 66–80.

[124] M. Paiola and B. Blanchet, Verification of Security Protocols with Lists: From Length One to Unbounded Length, in: *POST'12: First Conference on Principles of Security and Trust*, LNCS, Vol. 7215, Springer, 2012, pp. 69–88.

[125] M. Arapinis, S. Bursuc and M. Ryan, Reduction of Equational Theories for Verification of Trace Equivalence: Re-encryption, Associativity and Commutativity, in: *POST'12: First Conference on Principles of Security and Trust*, LNCS, Vol. 7215, Springer, 2012, pp. 169–188.

[126] A. Kiayias, T. Zacharias and B. Zhang, 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, Springer, 2015, pp. 468–498.

[127] P. Chaidos, V. Cortier, G. Fuschbauer and D. Galindo, BeleniosRF: A Non-interactive Receipt-Free Electronic Voting Scheme, in: *CCS'16: 23rd ACM Conference on Computer and Communications Security*, ACM Press, 2016, pp. 1614–1625.

[128] A. Fraser, E.A. Quaglia and B. Smyth, A critique of game-based definitions of receipt-freeness for voting, in: *ProveSec'19: 13th International Conference on Provable and Practical Security*, LNCS, Springer, 2019.

[129] A. Juels, D. Catalano and M. Jakobsson, Coercion-Resistant Electronic Elections, in: *WPES'05: 4th Workshop on Privacy in the Electronic Society*, ACM Press, 2005, pp. 61–70.

[130] R.W. Gardner, S. Garera and A.D. Rubin, Coercion Resistant End-to-end Voting, in: *FC'09: 13th International Conference on Financial Cryptography and Data Security*, LNCS, Vol. 5628, Springer, 2009, pp. 344–361.

[131] D. Unruh and J. Müller-Quade, Universally Composable Incoercibility, in: *CRYPTO'10: 30th International Cryptology Conference*, LNCS, Vol. 6223, Springer, 2010, pp. 411–428.

[132] R. Küsters, T. Truderung and A. Vogt, A Game-Based Definition of Coercion-Resistance and its Applications, *Journal of Computer Security* **20**(6) (2012), 709–764.

[133] T. Haines and B. Smyth, Surveying definitions of coercion resistance, 2020.

[134] A. McCarthy, B. Smyth and E.A. Quaglia, Hawk and Aucitas: e-auction schemes from the Helios and Civitas e-voting schemes, in: *FC'14: 18th International Conference on Financial Cryptography and Data Security*, LNCS, Vol. 8437, Springer, 2014, pp. 51–63.

[135] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, Chapman & Hall/CRC, 2007.

[136] A. Fiat and A. Shamir, How To Prove Yourself: Practical Solutions to Identification and Signature Problems, in: *CRYPTO'86: 6th International Cryptology Conference*, LNCS, Vol. 263, Springer, 1987, pp. 186–194.

[137] J. Groth, Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures, in: *ASIACRYPT'02: 12th International Conference on the Theory and Application of Cryptology and Information Security*, LNCS, Vol. 4284, Springer, 2006, pp. 444–459.

[138] R. Cramer, R. Gennaro and B. Schoenmakers, A Secure and Optimally Efficient Multi-Authority Election Scheme, in: *EUROCRYPT'97: 16th International Conference on the Theory and Applications of Cryptographic Techniques*, LNCS, Vol. 1233, Springer, 1997, pp. 103–118.

[139] D. Chaum, J. Evertse, J. van de Graaf and R. Peralta, Demonstrating Possession of a Discrete Logarithm Without Revealing It, in: *CRYPTO'86: 6th International Cryptology Conference*, LNCS, Vol. 263, Springer, 1987, pp. 200–212.

[140] R. Cramer, M.K. Franklin, B. Schoenmakers and M. Yung, Multi-Authority Secret-Ballot Elections with Linear Work, in: *EUROCRYPT'96: 15th International Conference on the Theory and Applications of Cryptographic Techniques*, LNCS, Vol. 1070, Springer, 1996, pp. 72–83.

[141] D. Chaum and T.P. Pedersen, Wallet Databases with Observers, in: *CRYPTO'92: 12th International Cryptology Conference*, LNCS, Vol. 740, Springer, 1993, pp. 89–105.

[142] NIST, Secure Hash Standard (SHS), FIPS PUB, 180-4, Information Technology Laboratory, National Institute of Standards and Technology, 2012.

[143] Y. Tsiounis and M. Yung, On the Security of ElGamal Based Encryption, in: *PKC'98: First International Workshop on Practice and Theory in Public Key Cryptography*, LNCS, Vol. 1431, Springer, 1998, pp. 117–134.

[144] B. Smyth, Verifiability of Helios Mixnet, 2018, Cryptology ePrint Archive, Report 2018/017.

[145] J. Heather and S. Schneider, A formal framework for modelling coercion resistanc and receipt freeness, in: *FM'12: 18th International Symposium on Formal Methods*, LNCS, Springer, 2012, pp. 217–231.

Appendix A. Cryptographic primitives

A.1. Asymmetric encryption

Definition 10 (Asymmetric encryption scheme [135]). *An* asymmetric encryption scheme *is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, such that:*[33]

- **Gen**, *denoted* $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa)$*, inputs a security parameter* κ *and outputs a key pair* (pk, sk) *and message space* $\mathfrak{m}$.
- **Enc**, *denoted* $c \leftarrow \mathsf{Enc}(pk, m)$*, inputs a public key* pk *and message* $m \in \mathfrak{m}$*, and outputs a ciphertext* c.
- **Dec**, *denoted* $m \leftarrow \mathsf{Dec}(sk, c)$*, inputs a private key* sk *and ciphertext* c*, and outputs a message* m *or an error symbol. We assume* Dec *is deterministic.*

Moreover, the scheme must be correct*: there exists a negligible function* negl*, such that for all security parameters* κ *and messages* m*, we have* $\Pr[(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa); c \leftarrow \mathsf{Enc}(pk, m) : m \in \mathfrak{m} \Rightarrow \mathsf{Dec}(sk, c) = m] > 1 - \mathsf{negl}(\kappa)$*. A scheme has* perfect correctness *if the probability is* 1.

Definition 11 (Homomorphic encryption [30]). *An asymmetric encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* homomorphic*, with respect to ternary operators* $\odot$*,* $\oplus$*, and* $\otimes$*,*[34] *if there exists a negligible function* negl*, such that for all security parameters* κ*, we have the following.*[35] *First, for all messages* m_1 *and* m_2 *we have* $\Pr[(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa); c_1 \leftarrow \mathsf{Enc}(pk, m_1); c_2 \leftarrow \mathsf{Enc}(pk, m_2) : m_1, m_2 \in \mathfrak{m} \Rightarrow \mathsf{Dec}(sk, c_1 \otimes_{pk} c_2) = \mathsf{Dec}(sk, c_1) \odot_{pk} \mathsf{Dec}(sk, c_2)] > 1 - \mathsf{negl}(\kappa)$*. Secondly, for all messages* m_1 *and*

[33] Our definition differs from Katz and Lindell's original definition [135, Definition 10.1] in that we formally state the plaintext space.

[34] For brevity, we write Π *is a homomorphic asymmetric encryption scheme* as opposed to the more verbose Π *is a homomorphic asymmetric encryption scheme, with respect to ternary operators* $\odot$*,* $\oplus$*, and* $\otimes$*.*

[35] We write $X \circ_{pk} Y$ for the application of ternary operator $\circ$ to inputs X, Y, and pk. We occasionally abbreviate $X \circ_{pk} Y$ as $X \circ Y$, when pk is clear from the context.

m_2, and all coins r_1 and r_2, we have $\Pr[(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa) : m_1, m_2 \in \mathfrak{m} \Rightarrow \mathsf{Enc}(pk, m_1; r_1) \otimes_{pk} \mathsf{Enc}(pk, m_2; r_2) = \mathsf{Enc}(pk, m_1 \odot_{pk} m_2; r_1 \oplus_{pk} r_2)] > 1 - \mathsf{negl}(\kappa)$*. We say Π is* additively homomorphic*, if for all security parameters κ, key pairs pk, sk, and message spaces $\mathfrak{m}$, such that there exists coins r and $(pk, sk, \mathfrak{m}) = \mathsf{Gen}(\kappa; r)$, we have $\odot_{pk}$ is the addition operator in group $(\mathfrak{m}, \odot_{pk})$.*

Definition 12 (IND-CPA [68])**.** *Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an asymmetric encryption scheme, $\mathcal{A}$ be an adversary, κ be the security parameter, and* IND-CPA *be the following game.*[36]

```
$\mathsf{IND\text{-}CPA}(\Pi, \mathcal{A}, \kappa) =$
  $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa);$
  $(m_0, m_1) \leftarrow \mathcal{A}(pk, \mathfrak{m}, \kappa);$
  $\beta \leftarrow_R \{0, 1\};$
  $c \leftarrow \mathsf{Enc}(pk, m_\beta);$
  $g \leftarrow \mathcal{A}(c);$
  return $g = \beta;$
```

In the above game, we require $m_0, m_1 \in \mathfrak{m}$ and $|m_0| = |m_1|$. We say Π satisfies IND-CPA*, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function* negl*, such that for all security parameters κ, we have* $\mathsf{Succ}(\mathsf{IND\text{-}CPA}(\Pi, \mathcal{A}, \kappa)) \leqslant \frac{1}{2} + \mathsf{negl}(\kappa)$*.*

Definition 13 (IND-PA0 [65])**.** *Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an asymmetric encryption scheme, $\mathcal{A}$ be an adversary, κ be the security parameter, and* IND-PA0 *be the following game.*

```
$\mathsf{IND\text{-}PA0}(\Pi, \mathcal{A}, \kappa) =$
  $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa);$
  $(m_0, m_1) \leftarrow \mathcal{A}(pk, \mathfrak{m}, \kappa);$
  $\beta \leftarrow_R \{0, 1\};$
  $c \leftarrow \mathsf{Enc}(pk, m_\beta);$
  $\mathbf{c} \leftarrow \mathcal{A}(c);$
  $\mathbf{m} \leftarrow (\mathsf{Dec}(sk, \mathbf{c}[1]), \ldots, \mathsf{Dec}(sk, \mathbf{c}[|\mathbf{c}|]);$
  $g \leftarrow \mathcal{A}(\mathbf{m});$
  return $g = \beta \wedge \bigwedge_{1 \leqslant i \leqslant |\mathbf{c}|} c \neq \mathbf{c}[i];$
```

In the above game, we require $m_0, m_1 \in \mathfrak{m}$ and $|m_0| = |m_1|$. We say Π satisfies IND-PA0*, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function* negl*, such that for all security parameters κ, we have* $\mathsf{Succ}(\mathsf{IND\text{-}PA0}(\Pi, \mathcal{A}, \kappa)) \leqslant \frac{1}{2} + \mathsf{negl}(\kappa)$*.*

A.2. Proof systems

Definition 14 (Non-interactive proof system [30])**.** *A* non-interactive proof system *for a relation R is a tuple of algorithms* (Prove, Verify)*, such that:*

- **Prove***, denoted $\sigma \leftarrow \mathsf{Prove}(s, w, \kappa)$, is executed by a prover to prove $(s, w) \in R$.*

[36] Our definition of an asymmetric encryption scheme explicitly defines the plaintext space, whereas Bellare *et al.* [68] leave the plaintext space implicit; this change is reflected in our definition of IND-CPA. Moreover, we provide the adversary with the message space and security parameter. We adapt IND-PA0 similarly.

- **Verify**, *denoted* $v \leftarrow \mathsf{Verify}(s, \sigma, \kappa)$, *is executed by anyone to check the validity of a proof. We assume* Verify *is deterministic.*

Moreover, the system must be complete*: there exists a negligible function* negl*, such that for all statement and witnesses* $(s, w) \in R$ *and security parameters* κ*, we have* $\Pr[\sigma \leftarrow \mathsf{Prove}(s, w, \kappa) : \mathsf{Verify}(s, \sigma, \kappa) = 1] > 1 - \mathsf{negl}(\kappa)$*. A system has* perfect completeness *if the probability is* 1.

Definition 15 (Fiat-Shamir transformation [136])**.** *Given a sigma protocol* $\Sigma = (\mathsf{Comm}, \mathsf{Chal}, \mathsf{Resp}, \mathsf{Verify}_\Sigma)$ *for relation* R *and a hash function* $\mathcal{H}$*, the* Fiat-Shamir transformation*, denoted* $\mathsf{FS}(\Sigma, \mathcal{H})$*, is the non-interactive proof system* $(\mathsf{Prove}, \mathsf{Verify})$*, defined as follows:*

```
$\mathsf{Prove}(s, w, \kappa) =$
    $(\mathsf{comm}, t) \leftarrow \mathsf{Comm}(s, w, \kappa)$;
    $\mathsf{chal} \leftarrow \mathcal{H}(\mathsf{comm}, s)$;
    $\mathsf{resp} \leftarrow \mathsf{Resp}(\mathsf{chal}, t, \kappa)$;
    return $(\mathsf{comm}, \mathsf{resp})$;
```

```
$\mathsf{Verify}(s, (\mathsf{comm}, \mathsf{resp}), \kappa) =$
    $\mathsf{chal} \leftarrow \mathcal{H}(\mathsf{comm}, s)$;
    return $\mathsf{Verify}_\Sigma(s, (\mathsf{comm}, \mathsf{chal}, \mathsf{resp}), \kappa)$;
```

A string m *can be included in the hashes computed by algorithms* Prove *and* Verify*. That is, the hashes are computed in both algorithms as* $\mathsf{chal} \leftarrow \mathcal{H}(\mathsf{comm}, s, m)$*. We write* $\mathsf{Prove}(s, w, m, \kappa)$ *and* $\mathsf{Verify}(s, (\mathsf{comm}, \mathsf{resp}), m, k)$ *for invocations of* Prove *and* Verify *which include string* m*.*

Definition 16 (Zero-knowledge [61])**.** *Let* $\Delta = (\mathsf{Prove}, \mathsf{Verify})$ *be a non-interactive proof system for a relation* R*, derived by application of the Fiat-Shamir transformation [136] to a random oracle* $\mathcal{H}$ *and a sigma protocol. Moreover, let* $\mathcal{S}$ *be an algorithm,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* ZK *be the following game.*

```
$\mathsf{ZK}(\Delta, \mathcal{A}, \mathcal{H}, \mathcal{S}, \kappa) =$
    $\beta \leftarrow_R \{0, 1\}$;
    $g \leftarrow \mathcal{A}^{\mathcal{H}, \mathcal{P}}(\kappa)$;
    return $g = \beta$;
```

Oracle $\mathcal{P}$ *is defined on inputs* $(s, w) \in R$ *as follows:*

- $\mathcal{P}(s, w)$ *computes* **if** $\beta = 0$ **then** $\sigma \leftarrow \mathsf{Prove}(s, w, \kappa)$ **else** $\sigma \leftarrow \mathcal{S}(s, \kappa)$ *and outputs* σ*.*

And algorithm $\mathcal{S}$ *can patch random oracle* $\mathcal{H}$*.*[37] *We say* Δ *satisfies* zero-knowledge*, if there exists a probabilistic polynomial-time algorithm* $\mathcal{S}$*, such that for all probabilistic polynomial-time algorithm adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, and for all security parameters* κ*, we have* $\mathsf{Succ}(\mathit{ZK}(\Delta, \mathcal{A}, \mathcal{H}, \mathcal{S}, \kappa)) \leqslant \frac{1}{2} + \mathsf{negl}(\kappa)$*. An algorithm* $\mathcal{S}$ *for which zero-knowledge holds is called a* simulator *for* $(\mathsf{Prove}, \mathsf{Verify})$.

[37] Random oracles can be *programmed* or *patched*. We will not need the details of how patching works, so we omit them here; see Bernhard et al. [71] for a formalisation.

Definition 17 (Simulation sound extractability [30, 71, 137])**.** *Suppose Σ is a sigma protocol for relation R, $\mathcal{H}$ is a random oracle, and* $(\mathsf{Prove}, \mathsf{Verify})$ *is a non-interactive proof system, such that* $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{Prove}, \mathsf{Verify})$. *Further suppose $\mathcal{S}$ is a simulator for* $(\mathsf{Prove}, \mathsf{Verify})$ *and $\mathcal{H}$ can be patched by $\mathcal{S}$. Proof system* $(\mathsf{Prove}, \mathsf{Verify})$ *satisfies* simulation sound extractability *if there exists a probabilistic polynomial-time algorithm $\mathcal{K}$, such that for all probabilistic polynomial-time adversaries $\mathcal{A}$ and coins r, there exists a negligible function* negl*, such that for all security parameters κ, we have:*[38]

$$\begin{aligned}
&\Pr[\mathbf{P} \leftarrow (); \mathbf{Q} \leftarrow \mathcal{A}^{\mathcal{H},\mathcal{P}}(—; r); \mathbf{W} \leftarrow \mathcal{K}^{\mathcal{A}'}(\mathbf{H}, \mathbf{P}, \mathbf{Q}) :\\
&|\mathbf{Q}| \neq |\mathbf{W}| \vee \exists j \in \{1, \ldots, |\mathbf{Q}|\} \,.\, (\mathbf{Q}[j][1], \mathbf{W}[j]) \notin R \wedge\\
&\qquad\qquad\qquad \forall (s, \sigma) \in \mathbf{Q}, (t, \tau) \in \mathbf{P} \,.\, \mathsf{Verify}(s, \sigma, \kappa) = 1 \wedge \sigma \neq \tau] \leqslant \mathsf{negl}(\kappa)
\end{aligned}$$

where $\mathcal{A}(—; r)$ denotes running adversary $\mathcal{A}$ with an empty input and coins r, where $\mathbf{H}$ is a transcript of the random oracle's input and output, and where oracles $\mathcal{A}'$ and $\mathcal{P}$ are defined below:

- $\mathcal{A}'()$. *Computes $\mathbf{Q}' \leftarrow \mathcal{A}(—; r)$, forwarding any of $\mathcal{A}$'s oracle queries to $\mathcal{K}$, and outputs $\mathbf{Q}'$. By running $\mathcal{A}(—; r)$, $\mathcal{K}$ is rewinding the adversary.*
- $\mathcal{P}(s)$. *Computes $\sigma \leftarrow \mathcal{S}(s, \kappa); \mathbf{P} \leftarrow (\mathbf{P}[1], \ldots, \mathbf{P}[|\mathbf{P}|], (s, \sigma))$ and outputs σ.*

Algorithm $\mathcal{K}$ is an extractor *for* $(\mathsf{Prove}, \mathsf{Verify})$.

Theorem 16 (from [71])**.** *Let Σ be a sigma protocol for relation R, and let $\mathcal{H}$ be a random oracle. Suppose Σ satisfies special soundness and special honest verifier zero-knowledge. Non-interactive proof system* $\mathsf{FS}(\Sigma, \mathcal{H})$ *satisfies zero-knowledge and simulation sound extractability.*

The Fiat-Shamir transformation may include a string in the hashes computed by functions Prove and Verify. Simulators can be generalised to include such a string too. We write $\mathcal{S}(s, m, \kappa)$ for invocations of simulator $\mathcal{S}$ which include string m. And remark that Theorem 16 can be extended to this generalisation.

Appendix B. Ballot independence: Non-malleability game

We formalise an alternative definition of ballot independence as a non-malleability game, called comparison based non-malleability under chosen vote attack (CNM-CVA), as a straightforward adaptation of the non-malleability definition for asymmetric encryption by Bellare & Sahai [65].

Definition 18 (CNM-CVA)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *be an election scheme, $\mathcal{A}$ be an adversary, κ be a security parameter, and* cnm-cva *and* cnm-cva-\$ *be the following games.*

```
cnm-cva(Γ, A, κ) =
    $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
    $(V, nc) \leftarrow \mathcal{A}(pk, \kappa)$;
    $v \leftarrow_R V$;
    $b \leftarrow \mathsf{Vote}(pk, v, nc, \kappa)$;
    $(R, \mathfrak{bb}) \leftarrow \mathcal{A}(b)$;
    $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$;
    return $R(v, \mathfrak{v}) \wedge b \notin \mathfrak{bb} \wedge V \subseteq \{1, \ldots, nc\}$
      $\wedge 1 \leqslant nc \leqslant mc \wedge |\mathfrak{bb}| \leqslant mb$;
```

[38] We extend set membership notation to vectors: we write $x \in \mathbf{x}$ if x is an element of the set $\{\mathbf{x}[i] : 1 \leqslant i \leqslant |\mathbf{x}|\}$.

```
$\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{A}, \kappa) =$
    $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa);$
    $(V, nc) \leftarrow \mathcal{A}(pk, \kappa);$
    $v, v' \leftarrow_R V;$
    $b \leftarrow \mathsf{Vote}(pk, v', nc, \kappa);$
    $(R, \mathfrak{bb}) \leftarrow \mathcal{A}(b);$
    $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa);$
    $\textbf{return } R(v, \mathfrak{v}) \wedge b \notin \mathfrak{bb} \wedge V \subseteq \{1, \ldots, nc\}$
      $\wedge\ 1 \leqslant nc \leqslant mc \wedge |\mathfrak{bb}| \leqslant mb;$
```

In the above games, we require that relation R is computable in polynomial time. We say Γ satisfies comparison based non-malleability under chosen vote attack (CNM-CVA)*, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function* negl*, such that for all security parameters κ, we have* $\mathsf{Succ}(\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{A}, \kappa)) - \mathsf{Succ}(\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{A}, \kappa)) \leqslant \mathsf{negl}(\kappa)$.

CNM-CVA is satisfied if no adversary can distinguish between games cnm-cva and cnm-cva-\$. That is, for all adversaries, the adversary wins cnm-cva iff the adversary looses cnm-cva-\$, with negligible probability. The first three steps of games cnm-cva and cnm-cva-\$ are identical, thus, these steps cannot be distinguished. Then, game cnm-cva-\$ performs an additional step: the challenger samples a second vote v' from vote space V. Thereafter, game cnm-cva, respectively game cnm-cva-\$, proceeds as follows: the challenger constructs a challenge ballot b for v, respectively v'; the adversary is given ballot b and computes a relation R and bulletin board $\mathfrak{bb}$; and the challenger tallies $\mathfrak{bb}$ to derive election outcome $\mathfrak{v}$ and evaluates whether $R(v, \mathfrak{v})$ holds. Hence, CNM-CVA is satisfied if there is no advantage of the relation computed by an adversary given a challenge ballot for v, over the relation computed by the adversary given a challenge ballot for v'. That is, for all adversaries, we have with negligible probability that the relation evaluated by the challenger in cnm-cva holds iff the relation evaluated in cnm-cva-\$ does not hold. It follows that no adversary can meaningfully relate ballots. On the other hand, if CNM-CVA is not satisfied, then there exists a strategy to construct related ballots.

CNM-CVA avoids crediting the adversary for trivial and unavoidable relations which hold if the challenge ballot appears on the bulletin board. For example, suppose the adversary is given a challenge ballot for v in cnm-cva, respectively v' in cnm-cva-\$. This adversary could output a bulletin board containing only the challenge ballot and a relation R such that $R(v, \mathfrak{v})$ holds if $\mathfrak{v}[v] = 1$, hence, the relation evaluated in cnm-cva holds, whereas the relation evaluated in cnm-cva-\$ does not hold, but the adversary looses in both games because the challenge ballot appears on the bulletin board. By contrast, if the adversary can derive a ballot meaningfully related to the challenge ballot, then the adversary can win the game. For instance, Cortier & Smyth [53, 54] identify a class of vulnerabilities against voting systems, which can be exploited as follows: an adversary observes a voter's ballot, casts a meaningfully related ballot, and abuses the relation to recover the voter's vote from the election outcome.

Comparing CNM-CVA *and* CNM-CPA.. Unsurprisingly, the distinction between non-malleability for asymmetric encryption (CNM-CPA) and non-malleability for election schemes (CNM-CVA) is similar to the distinction between indistinguishability for asymmetric encryption and indistinguishability for election schemes (§3.2), namely, CNM-CPA performs a parallel decryption, whereas CNM-CVA performs a single tally.

Our ballot independence games are adaptations of standard security definitions for asymmetric encryption: CNM-CVA is based on non-malleability for encryption and IND-CVA is based on indistinguishability for encryption. Bellare & Sahai [65] have shown that non-malleability is equivalent to indistinguishability for encryption and their proof can be adapted to show that CNM-CVA and IND-CVA are equivalent.

Theorem 17 (CNM-CVA = IND-CVA)**.** *Given an election scheme Γ, we have Γ satisfies* CNM-CVA *iff Γ satisfies* IND-CVA.

Proof. For the *if* implication, suppose Γ does not satisfy CNM-CVA, hence, there exists a probabilistic polynomial-time adversary $\mathcal{A}$, such that for all negligible functions negl, there exists a security parameter κ and $\mathsf{Succ}(\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{A}, \kappa)) - \mathsf{Succ}(\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{A}, \kappa)) > \mathsf{negl}(\kappa)$. We construct an adversary $\mathcal{B}$ against game IND-CVA from $\mathcal{A}$.

- $\mathcal{B}(pk, \kappa)$ computes $(V, nc) \leftarrow \mathcal{A}(pk, \kappa); v, v' \leftarrow_R V$ and outputs (v, v', nc).
- $\mathcal{B}(b)$ computes $(R, \mathfrak{bb}) \leftarrow \mathcal{A}(b)$ and outputs $\mathfrak{bb}$.
- $\mathcal{B}(\mathfrak{v})$ outputs 0 if $R(v, \mathfrak{v})$ holds and 1 otherwise.

If the challenger selects $\beta = 0$ in game IND-CVA, then adversary $\mathcal{B}$ simulates $\mathcal{A}$'s challenger to $\mathcal{A}$ in cnm-cva and $\mathcal{B}$'s success (which requires $R(v, \mathfrak{v})$ to hold) is $\mathsf{Succ}(\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{A}, \kappa))$. Otherwise ($\beta = 1$), adversary $\mathcal{B}$ simulates $\mathcal{A}$'s challenger to $\mathcal{A}$ in cnm-cva-\$ and, since $\mathcal{B}$ will evaluate $R(v, \mathfrak{v})$, $\mathcal{B}$'s success (which requires $R(v, \mathfrak{v})$ not to hold) is $1 - \mathsf{Succ}(\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{A}, \kappa))$. It follows that $\mathsf{Succ}(\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{B}, \kappa)) = \frac{1}{2} \cdot (\mathsf{Succ}(\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{A}, \kappa)) + 1 - \mathsf{Succ}(\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{A}, \kappa)))$, therefore, $2 \cdot \mathsf{Succ}(\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{B}, \kappa)) - 1 = \mathsf{Succ}(\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{A}, \kappa)) - \mathsf{Succ}(\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{A}, \kappa))$. Thus, $\mathsf{Succ}(\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{B}, \kappa)) > \frac{1}{2} + \frac{1}{2} \cdot \mathsf{negl}(\kappa)$, concluding our proof of the *if* implication.

For the *only if* implication, suppose Γ does not satisfy IND-CVA, hence, there exists a probabilistic polynomial-time adversary $\mathcal{A}$, such that for all negligible functions negl, there exists a security parameter κ and $\mathsf{Succ}(\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{A}, \kappa)) > \frac{1}{2} + \mathsf{negl}(\kappa)$. We construct an adversary $\mathcal{B}$ against CNM-CVA from $\mathcal{A}$.

- $\mathcal{B}(pk, \kappa)$ computes $(v_0, v_1, nc) \leftarrow \mathcal{A}(pk, \kappa)$ and outputs $(\{v_0, v_1\}, nc)$.
- $\mathcal{B}(b)$ computes $\mathfrak{bb} \leftarrow \mathcal{A}(b)$, picks coins r uniformly at random, derives a relation R such that $R(v, \mathfrak{v})$ holds if there exists a bit g such that $v = v_g \wedge g = \mathcal{A}(\mathfrak{v}; r)$ and fails otherwise, and outputs $(R, \mathfrak{bb})$.

Adversary $\mathcal{B}$ simulates $\mathcal{A}$'s challenger to $\mathcal{A}$ in game IND-CVA. Indeed, the challenge ballot is equivalently computed. As is the election outcome. The computation $\mathcal{A}(\mathfrak{v}; r)$ is not black-box, but this does not matter: it is still invoked exactly once in the game. Let us consider adversary $\mathcal{B}$'s success against cnm-cva and cnm-cva-\$.

- Game cnm-cva samples a single vote v from V. By inspection of cnm-cva and IND-CVA, we have $\mathsf{Succ}(\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{B}, \kappa)) = \mathsf{Succ}(\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{A}, \kappa))$, hence, $\mathsf{Succ}(\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{B}, \kappa)) - \frac{1}{2} > \mathsf{negl}(\kappa)$.
- Game cnm-cva-\$ samples votes v and v' from V. Vote v is independent of $\mathcal{A}$'s perspective, indeed, an equivalent formulation of cnm-cva-\$ could sample v after $\mathcal{A}$ has terminated and immediately before evaluating the adversary's relation. By inspection of cnm-cva-\$ and IND-CVA, we have $\mathsf{Succ}(\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{B}, \kappa)) = \frac{1}{2} \cdot \mathsf{Succ}(\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{A}, \kappa)) + \frac{1}{2} \cdot (1 - \mathsf{Succ}(\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{A}, \kappa))) = \frac{1}{2}$.

It follows that $\mathsf{Succ}(\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{B}, \kappa)) - \mathsf{Succ}(\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{B}, \kappa)) > \mathsf{negl}(\kappa)$. □

Appendix C. Proofs

C.1. Proof of Theorem 1

Suppose Γ does not satisfy ballot independence, hence, there exists a probabilistic polynomial-time adversary $\mathcal{A}$, such that for all negligible functions negl, there exists a security parameter κ and $\mathsf{Succ}(\mathsf{IND\text{-}CVA}) > \frac{1}{2} + \mathsf{negl}(\kappa)$. We construct an adversary $\mathcal{B}$ against $\mathsf{Ballot\text{-}Secrecy}$ from $\mathcal{A}$.

- $\mathcal{B}(pk,\kappa)$ computes $(v_0,v_1,nc) \leftarrow \mathcal{A}(pk,\kappa)$ and outputs nc.
- $\mathcal{B}()$ computes $b \leftarrow \mathcal{O}(v_0,v_1)$; $\mathfrak{bb} \leftarrow \mathcal{A}(b)$ and outputs $\mathfrak{bb}$.
- $\mathcal{B}(\mathfrak{v},pf)$ computes $g \leftarrow \mathcal{A}(\mathfrak{v})$ and outputs g.

Adversary $\mathcal{B}$ simulates $\mathcal{A}$'s challenger to $\mathcal{A}$. Indeed, the challenge ballot and election outcome are equivalently computed. Moreover, the challenge ballot does not appear on the bulletin board, hence, the bulletin board is balanced. It follows that $\mathsf{Succ}(\mathsf{IND\text{-}CVA}(\Gamma,\mathcal{A},\kappa)) = \mathsf{Succ}(\mathsf{Ballot\text{-}Secrecy}(\Gamma,\mathcal{B},\kappa))$, hence, $\mathsf{Succ}(\mathsf{Ballot\text{-}Secrecy}(\Gamma,\mathcal{B},\kappa)) > \frac{1}{2} + \mathsf{negl}(\kappa)$, concluding our proof. □

C.2. Proof of Proposition 3

In essence, the proof follows from Theorem 4. Albeit, formally, a few extra steps are required. In particular, the definition of an election scheme with zero-knowledge proofs demands that tallying proofs must be computed by a zero-knowledge non-interactive proof system, but an election scheme without tallying proofs need not compute proofs with such a system. Thus, we must introduce an election scheme with zero-knowledge proofs and prove that it is equivalent to the election scheme without proofs. This is trivial, so we do not pursue the details. □

C.3. Proof of Theorem 4

Definition 19 (Zero-knowledge tallying proofs)**.** *An election scheme* $\Gamma = (\mathsf{Setup},\mathsf{Vote},\mathsf{Tally})$ *has* zero-knowledge tallying proofs, *if there exists a non-interactive zero-knowledge proof system* $(\mathsf{Prove},\mathsf{Verify})$*, such that for all security parameters* κ*, integers nc, bulletin boards* $\mathfrak{bb}$*, outputs* (pk,sk,mb,mc) *of* $\mathsf{Setup}(\kappa)$*, and outputs* $(\mathfrak{v},pf)$ *of* $\mathsf{Tally}(sk,\mathfrak{bb},nc,\kappa)$*, we have pf is an output of algorithm* Prove *parameterised with statement* $(pk,\mathfrak{bb},nc,\mathfrak{v})$ *and coins chosen uniformly at random.*

(Beyond the statement and coins, algorithm Prove additionally inputs some witness and security parameter.)

Definition 20. *Let* $\Gamma = (\mathsf{Setup},\mathsf{Vote},\mathsf{Tally})$ *be an election scheme with zero-knowledge tallying proofs,* $\mathcal{A}$ *be an adversary, and* κ *be a security parameter. Moreover, let* $\mathcal{S}$ *be the simulator for the non-interactive zero-knowledge proof system used by algorithm* Tally *to compute tallying proofs. We define game* BS *as follows.*

```
$\mathsf{BS}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa) =$
    $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa);$
    $nc \leftarrow \mathcal{A}(pk, \kappa);$
    $\beta \leftarrow_R \{0,1\};$
    $L \leftarrow \emptyset;$
    $\mathfrak{bb} \leftarrow \mathcal{A}^{\mathcal{O}}();$
    $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\}, nc, \kappa);$
    for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ do
        $\mathfrak{v}[v_0] \leftarrow \mathfrak{v}[v_0] + 1;$
    $pf' \leftarrow \mathcal{S}((pk, nc, \mathfrak{bb}, \mathfrak{v}), \kappa);$
    $g \leftarrow \mathcal{A}(\mathfrak{v}, pf');$
    return $g = \beta \wedge balanced(\mathfrak{bb}, nc, L) \wedge 1 \leqslant nc \leqslant mc \wedge |\mathfrak{bb}| \leqslant mb;$
```

Predicate balanced and oracle $\mathcal{O}$ are defined as per game Ballot-Secrecy.

Lemma 18. *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *be an election scheme with zero-knowledge tally proofs,* $\mathcal{S}$ *be the simulator for that proof system,* $\mathcal{A}$ *be an adversary, and* κ *be a security parameter. If* Γ *satisfies* Additivity*, then* $|\mathsf{Succ}(\mathsf{BS}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) - \mathsf{Succ}(\mathsf{Ballot\text{-}Secrecy}(\Gamma, \mathcal{A}, \kappa))| \leqslant \mathsf{negl}(\kappa)$*, for all negligible functions* negl *and some security parameter* κ*.*

Proof. The challengers in games BS and Ballot-Secrecy both compute public keys using the same algorithm and provide those keys, along with the security parameter, as input to the first adversary call, thus, these inputs and corresponding outputs are equivalent. Moreover, the left-right oracle is the same in both BS and Ballot-Secrecy, hence, the bulletin board output by the second adversary call is equivalent in both games. Furthermore, predicate *balanced* is satisfied in BS iff it is satisfied in Ballot-Secrecy, hence, if predicate *balanced* is not satisfied, then $\mathsf{Succ}(\mathsf{BS}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) = \mathsf{Succ}(\mathsf{Ballot\text{-}Secrecy}(\Gamma, \mathcal{A}, \kappa))$, concluding our proof. Otherwise, it suffices to show that the outcome and tallying proof are equivalently computed in BS and Ballot-Secrecy, since this ensures the inputs to the third adversary call are equivalent, thus the corresponding outputs are equivalent too, which suffices to conclude.

In Ballot-Secrecy, the outcome is computed by tallying the bulletin board. By comparison, in BS, the outcome is computed by tallying the ballots on the bulletin board that were constructed by the adversary (i.e., ballots in $\mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\}$, where $\mathfrak{bb}$ is the bulletin board and L is the set constructed by the oracle) and by simulating the tallying of any remaining ballots (i.e., ballots constructed by the oracle, namely, ballots in $\mathfrak{bb} \cap \{b \mid (b, v_0, v_1) \in L\}$). Suppose (pk, sk, mb, mc) is an output of $\mathsf{Setup}(\kappa)$ and nc is an integer. Since Γ satisfies Additivity, computing $\mathfrak{v}$ as

```
    $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa);$
```

is equivalent to computing $\mathfrak{v}$ as

```
    $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\}, nc, \kappa);$
    $(\mathfrak{v}', pf') \leftarrow \mathsf{Tally}(sk, \mathfrak{bb} \cap \{b \mid (b, v_0, v_1) \in L\}, nc, \kappa);$
    $\mathfrak{v} \leftarrow \mathfrak{v} + \mathfrak{v}';$
```

and as

```
$(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\}, nc, \kappa)$;
$(\mathfrak{v}', pf') \leftarrow \mathsf{Tally}(sk, \emptyset, nc, \kappa)$;
for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ do
    $(\mathfrak{v}'', pf'') \leftarrow \mathsf{Tally}(sk, \{b\}, nc, \kappa)$;
    $\mathfrak{v}' \leftarrow \mathfrak{v}' + \mathfrak{v}''$;
$\mathfrak{v} \leftarrow \mathfrak{v} + \mathfrak{v}'$;
```

Moreover, by correctness of Γ, we have $\mathsf{Tally}(sk, \emptyset, nc, \kappa)$ outputs $(\mathfrak{v}', pf')$ such that $\mathfrak{v}'$ is a zero-filled vector. Hence, the above computation is equivalent to

```
$(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\}, nc, \kappa)$;
for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ do
    $(\mathfrak{v}', pf') \leftarrow \mathsf{Tally}(sk, \{b\}, nc, \kappa)$;
    $\mathfrak{v} \leftarrow \mathfrak{v} + \mathfrak{v}'$;
```

Thus, to prove the outcome is computed equivalently in Ballot-Secrecy and BS, it suffices to prove that the simulations are valid, i.e., computing the above for-loop is equivalent to computing **for** $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ **do** $\mathfrak{v}[v_0] \leftarrow \mathfrak{v}[v_0] + 1$.

In BS, we have for all $(b, v_0, v_1) \in L$ that b is an output of $\mathsf{Vote}(pk, v_\beta, nc, \kappa)$ such that $v_0, v_1 \in \{1, ..., nc\}$, where β is the bit chosen by the challenger. Moreover, by correctness of Γ, we have $\mathsf{Tally}(sk, \{b\}, nc, \kappa)$ outputs $(\mathfrak{v}', pf')$ such that $\mathfrak{v}'$ is a zero-filled vector, except for index v_β, which contains one, hence, computing $\mathfrak{v} \leftarrow \mathfrak{v} + \mathfrak{v}'$ inside the for-loop is equivalent to computing $\mathfrak{v}[v_\beta] \leftarrow \mathfrak{v}[v_\beta] + 1$ inside the loop. Furthermore, since predicate *balanced* holds in BS, we have for all $v \in \{1, \ldots, nc\}$ that $|\{b \mid b \in \mathfrak{bb} \wedge \exists v_1 \,.\, (b, v, v_1) \in L\}| = |\{b \mid b \in \mathfrak{bb} \wedge \exists v_0 \,.\, (b, v_0, v) \in L\}|$. Hence, in BS, computing

```
for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ do $\mathfrak{v}[v_\beta] \leftarrow \mathfrak{v}[v_\beta] + 1$;
```

is equivalent to computing

```
for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ do $\mathfrak{v}[v_0] \leftarrow \mathfrak{v}[v_0] + 1$;
```

Thus, the simulation is valid in BS.

In Ballot-Secrecy, the tallying proof is computed by tallying the bulletin board. By comparison, in BS, the tallying proof is computed by simulator $\mathcal{S}$. Since Γ has zero-knowledge tallying proofs, there exists a non-interactive proof system $(\mathsf{Prove}, \mathsf{Verify})$ such that for all $(\mathfrak{v}, pf)$ output by $\mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$, we have pf is an output of algorithm Prove parameterised with statement $(pk, \mathfrak{bb}, nc, \mathfrak{v})$ and coins chosen uniformly at random. Moreover, since $\mathcal{S}$ is a simulator for $(\mathsf{Prove}, \mathsf{Verify})$, proofs output by algorithm Prove are indistinguishable from outputs of simulator $\mathcal{S}$. Thus, tallying proofs are equivalently computed in Ballot-Secrecy and BS, thereby concluding our proof. □

Let BS-0, respectively BS-1, be the game derived from BS by replacing $\beta \leftarrow_R \{0,1\}$ with $\beta \leftarrow 0$, respectively $\beta \leftarrow 1$. These games are trivially related to BS, namely, $\mathsf{Succ}(\mathsf{BS}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) = \frac{1}{2} \cdot \mathsf{Succ}(\mathsf{BS\text{-}0}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) + \frac{1}{2} \cdot \mathsf{Succ}(\mathsf{BS\text{-}1}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa))$. Moreover, let BS-1:0 be the game derived from BS-1 by replacing $g = \beta$ with $g = 0$ and let G_j be the game derived from BS-1:0 by removing $\beta \leftarrow 1$

and redefining oracle $\mathcal{O}$ such that $\mathcal{O}(v_0, v_1)$ computes, on inputs $v_0, v_1 \in \{1, ..., nc\}$, the following:

```
**if** $|L| < j$ **then**
    $b \leftarrow \mathsf{Vote}(pk, v_1, nc, \kappa)$;
**else**
    $b \leftarrow \mathsf{Vote}(pk, v_0, nc, \kappa)$;
$L \leftarrow L \cup \{(b, v_0, v_1)\}$;
**return** $b$;
```

Games $\mathsf{G}_0, \mathsf{G}_1, \ldots$ are distinguished from games BS-0 and BS-1:0 by their left-right oracles. In particular, the first j left-right oracle queries in G_j construct ballots for the oracle's "right" input and any remaining queries construct ballots for the oracle's "left" input, whereas the left-right oracle in BS-0, respectively BS-1:0, always constructs ballots for the oracle's "left," respectively "right," input. We relate game BS-1:0 to BS-1, games BS-0 and BS-1:0 to the hybrid games $\mathsf{G}_0, \mathsf{G}_1, \ldots$, and we prove Theorem 4 using these relations.

Lemma 19. *Let Γ be an election scheme with zero-knowledge tally proofs, $\mathcal{S}$ be the simulator for that proof system, $\mathcal{A}$ be an adversary, and κ be a security parameter. If adversary $\mathcal{A}$ wins game* Ballot-Secrecy *against election scheme Γ, then* $\mathsf{Succ}(\text{BS-1}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) = 1 - \mathsf{Succ}(\text{BS-1:0}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa))$.

Lemma 20. *Let Γ be an election scheme with zero-knowledge tally proofs, $\mathcal{S}$ be the simulator for that proof system, $\mathcal{A}$ be an adversary, and κ be a security parameter. If Γ satisfies* Additivity*, then* $\mathsf{Succ}(\text{BS-0}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) = \mathsf{Succ}(\mathsf{G}_0(\Gamma, \mathcal{A}, \mathcal{S}, \kappa))$ *and* $\mathsf{Succ}(\text{BS-1:0}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) = \mathsf{Succ}(\mathsf{G}_q(\Gamma, \mathcal{A}, \mathcal{S}, \kappa))$*, where q is an upper-bound on $\mathcal{A}$'s left-right oracle queries.*

Proof. Games BS-0 and G_0, respectively BS-1:0 and G_q, are identical up to the oracle, except BS-0 computes $\beta \leftarrow 0$ and checks $g = \beta$, which is equivalent to G_0 checking $g = 0$, respectively BS-1:0 computes $\beta \leftarrow 1$ and checks $g = 0$, which is equivalent to G_q checking $g = 0$. Thus, it suffices to show that oracle outputs in BS-0 are equivalent to oracle outputs in G_0, and similarly for BS-1:0 and G_q. Left-right oracles queries $\mathcal{O}(v_0, v_1)$ in games BS-0 and G_0 output ballots for vote v_0, hence, outputs are equivalent in both games. Oracle outputs in BS-1:0 and G_q are similarly equivalent, in particular, left-right oracles queries $\mathcal{O}(v_0, v_1)$ in both games output ballots for vote v_1, because q is an upper-bound on the left-right oracle queries, therefore, $|L| < q$ in G_q, concluding our proof. □

Proof of Theorem 4. By Theorem 1, it suffices to prove that ballot independence implies ballot secrecy. Suppose Γ does not satisfy ballot secrecy, hence, there exists a probabilistic polynomial-time adversary $\mathcal{A}$, such that for all negligible functions negl, there exists a security parameter κ and

$$\frac{1}{2} + \mathsf{negl}(\kappa) < \mathsf{Succ}(\text{Ballot-Secrecy}(\Gamma, \mathcal{A}, \kappa))$$

Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$. Since Γ has zero-knowledge tallying proofs, tallying proofs output by Tally are computed by a non-interactive zero-knowledge proof system. Let algorithm $\mathcal{S}$ be the simulator for that proof system. By Lemma 18, we have

$$\approx \mathsf{Succ}(\text{BS}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa))$$

By definition of BS-0 and BS-1, we have

$$= \frac{1}{2} \cdot (\mathsf{Succ}(\mathsf{BS}\text{-}0(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) + \mathsf{Succ}(\mathsf{BS}\text{-}1(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)))$$

And, by Lemma 19, we have

$$= \frac{1}{2} \cdot (\mathsf{Succ}(\mathsf{BS}\text{-}0(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) + 1 - \mathsf{Succ}(\mathsf{BS}\text{-}1{:}0(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)))$$
$$= \frac{1}{2} + \frac{1}{2} \cdot (\mathsf{Succ}(\mathsf{BS}\text{-}0(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) - \mathsf{Succ}(\mathsf{BS}\text{-}1{:}0(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)))$$

Let q be an upper-bound on $\mathcal{A}$'s left-right oracle queries. Hence, by Lemma 20, we have

$$= \frac{1}{2} + \frac{1}{2} \cdot (\mathsf{Succ}(\mathsf{G}_0(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) - \mathsf{Succ}(\mathsf{G}_q(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)))$$

which can be rewritten as the telescoping series

$$= \frac{1}{2} + \frac{1}{2} \cdot \sum_{1 \leqslant j \leqslant q} \mathsf{Succ}(\mathsf{G}_{j-1}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) - \mathsf{Succ}(\mathsf{G}_j(\Gamma, \mathcal{A}, \mathcal{S}, \kappa))$$

Let $j \in \{1, \ldots, q\}$ be such that $\mathsf{Succ}(\mathsf{G}_{j-1}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) - \mathsf{Succ}(\mathsf{G}_j(\Gamma, \mathcal{A}, \mathcal{S}, \kappa))$ is the largest term in that series. Hence,

$$\leqslant \frac{1}{2} + \frac{1}{2} \cdot q \cdot (\mathsf{Succ}(\mathsf{G}_{j-1}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) - \mathsf{Succ}(\mathsf{G}_j(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)))$$

Thus,

$$\frac{1}{2} + \frac{1}{q} \cdot \mathsf{negl}(\kappa) \leqslant \frac{1}{2} + \frac{1}{2} \cdot (\mathsf{Succ}(\mathsf{G}_{j-1}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) - \mathsf{Succ}(\mathsf{G}_j(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)))$$

From $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ against IND-CVA:

- $\mathcal{B}(pk, \kappa)$ computes $nc \leftarrow \mathcal{A}(pk, \kappa); L \leftarrow \emptyset$ and runs $\mathcal{A}^{\mathcal{O}}()$, handling $\mathcal{A}$'s oracle queries $\mathcal{O}(v_0, v_1)$ as follows: if $|L| < j$, then compute $b \leftarrow \mathsf{Vote}(pk, v_1, nc, \kappa); L \leftarrow L \cup \{b, v_0, v_1\}$ and return b to $\mathcal{A}$, otherwise, assign $v_0^c \leftarrow v_0; v_1^c \leftarrow v_1$, and output (v_0, v_1, nc).
- $\mathcal{B}(b)$ assigns $L \leftarrow L \cup \{(b, v_0^c, v_1^c)\}$; returns b to $\mathcal{A}$ and handles any further oracle queries $\mathcal{O}(v_0, v_1)$ as follows, namely, compute $b \leftarrow \mathsf{Vote}(pk, v_0, nc, \kappa); L \leftarrow L \cup \{(b, v_0, v_1)\}$ and return b to $\mathcal{A}$; assigns $\mathcal{A}$'s output to $\mathfrak{bb}$; and outputs $\mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\}$.

- $\mathcal{B}(\mathfrak{v})$ computes

```
**for** $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ **do**
    $\mathfrak{v}[v_0] \leftarrow \mathfrak{v}[v_0] + 1$;
$pf \leftarrow \mathcal{S}((pk, \mathfrak{bb}, nc, \mathfrak{v}), \kappa)$;
$g \leftarrow \mathcal{A}(\mathfrak{v}, pf)$;
```

  and outputs g.

We prove that $\mathcal{B}$ wins IND-CVA with success of at least $\frac{1}{2} + \frac{1}{2} \cdot (\mathsf{Succ}(\mathsf{G}_{j-1}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) - \mathsf{Succ}(\mathsf{G}_j(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)))$.

Suppose (pk, sk, mb, mc) is an output of $\mathsf{Setup}(\kappa)$. Further suppose we run $\mathcal{B}(pk, \kappa)$. It is straightforward to see that $\mathcal{B}$ simulates the challenger and oracle in both G_{j-1} and G_j to $\mathcal{A}$. In particular, $\mathcal{B}$ simulates query $\mathcal{O}(v_0, v_1)$ by computing $b \leftarrow \mathsf{Vote}(pk, v_1, nc, \kappa)$ for the first $j - 1$ queries. Since G_{j-1} and G_j are equivalent to adversaries that make fewer than j left-right oracle queries, adversary $\mathcal{A}$ must make at least j queries to ensure $\mathsf{Succ}(\mathsf{G}_{j-1}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) - \mathsf{Succ}(\mathsf{G}_j(\Gamma, \mathcal{A}, \mathcal{S}, \kappa))$ is non-negligible. Hence, $\mathcal{B}(pk, \kappa)$ terminates with non-negligible probability. Suppose adversary $\mathcal{B}$ terminates by outputting (v_0, v_1, nc), where v_0, v_1 correspond to the inputs of the jth oracle query by $\mathcal{A}$. Further suppose b is an output of $\mathsf{Vote}(pk, v_\beta, nc, \kappa)$, where β is a bit. If $\beta = 0$, then $\mathcal{B}(b)$ simulates the oracle in G_{j-1} to $\mathcal{A}$, otherwise, $\mathcal{B}(b)$ simulates the oracle in G_j to $\mathcal{A}$. In particular, $\mathcal{B}(b)$ responds to the jth oracle query with ballot b for v_β, thus, simulating the challenger in G_{j-1} when $\beta = 0$, respectively G_j when $\beta = 1$. And $\mathcal{B}(b)$ responds to any further oracle queries $\mathcal{O}(v_0, v_1)$ with ballots for v_0. Suppose $\mathfrak{bb}$ is an output of $\mathcal{A}$, thus $\mathcal{B}(b)$ outputs $\mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\}$. Further suppose $(\mathfrak{v}, pf)$ is an output of $\mathsf{Tally}(sk, \mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\}, nc, \kappa)$ and g is an output of $\mathcal{B}(\mathfrak{v})$. It is trivial to see that $\mathcal{B}(\mathfrak{v})$ simulates $\mathcal{A}$'s challenger. Thus, either

(1) $\beta = 0$ and $\mathcal{B}$ simulates G_{j-1} to $\mathcal{A}$, thus, $g = \beta$ with at least the probability that $\mathcal{A}$ wins G_{j-1}; or
(2) $\beta = 1$ and $\mathcal{B}$ simulates G_j to $\mathcal{A}$, thus, $g \neq 0$ with at least the probability that $\mathcal{B}$ looses G_j and, since $\mathcal{A}$ wins game Ballot-Secrecy, we have g is a bit, hence, $g = \beta$.

It follows that the success of adversary $\mathcal{B}$ is at least $\frac{1}{2} \cdot \mathsf{Succ}(\mathsf{G}_{j-1}(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)) + \frac{1}{2} \cdot (1 - \mathsf{Succ}(\mathsf{G}_j(\Gamma, \mathcal{A}, \mathcal{S}, \kappa)))$, thus we conclude our proof. □

C.4. Proof of Lemma 8

Suppose (pk, sk, mb, mc) is an output of $\mathsf{Setup}(\kappa)$ and $(nc, \mathfrak{bb}, \mathfrak{bb}')$ is an output of $\mathcal{A}(pk, \kappa)$ such that $nc \leqslant mc$ and $|\mathfrak{bb}| \leqslant mb$. Further suppose $(\mathfrak{v}, pf)$ is an output $\mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$, $(\mathfrak{v}_0, pf_0)$ is an output of $\mathsf{Tally}(sk, \mathfrak{bb} \setminus \mathfrak{bb}', nc, \kappa)$, and $(\mathfrak{v}_1, pf_1)$ is an output of $\mathsf{Tally}(sk, \mathfrak{bb} \cap \mathfrak{bb}', nc, \kappa)$. Let $\mathfrak{v}^+ = \mathfrak{v}_0 + \mathfrak{v}_1$. Moreover, let $\mathfrak{v}^* = \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)$, $\mathfrak{v}_0^* = \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb} \setminus \mathfrak{bb}', \kappa)$, and $\mathfrak{v}_1^* = \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb} \cap \mathfrak{bb}', \kappa)$. By definition of function *correct-outcome*, we have $\mathfrak{v}^* = \mathfrak{v}_0^* + \mathfrak{v}_1^*$. Moreover, by Tally-Soundness, we have $\mathfrak{v} = \mathfrak{v}^*$, $\mathfrak{v}_0 = \mathfrak{v}_0^*$, and $\mathfrak{v}_1 = \mathfrak{v}_1^*$, with overwhelming probability. It follows by transitivity that $\mathfrak{v} = \mathfrak{v}^+$, with overwhelming probability.

C.5. Proof of Proposition 12

We present a construction (Definition 21) for encryption schemes (Lemma 21) which are clearly not secure (Lemma 22). Nevertheless, the construction produces encryption schemes that are sufficient for ballot secrecy (Lemma 23). The proof of Proposition 12 follows from Lemmata 21–23.

Definition 21. *Given an asymmetric encryption scheme* $\Pi = (\mathsf{Gen}_\Pi, \mathsf{Enc}_\Pi, \mathsf{Dec}_\Pi)$ *and a constant symbol* ω*, let* $\mathsf{Leak}(\Pi, \omega) = (\mathsf{Gen}_\Pi, \mathsf{Enc}_\Pi, \mathsf{Dec})$ *such that* $\mathsf{Dec}(sk, c)$ *proceeds as follows: if* $c = \omega$*, then output sk, otherwise, compute* $m \leftarrow \mathsf{Dec}_\Pi(sk, c)$ *and output m.*

Lemma 21. *Given an asymmetric encryption scheme* Π *and a constant symbol* ω *such that* Π*'s ciphertext space does not contain* ω*, we have* $\mathsf{Leak}(\Pi, \omega)$ *is an asymmetric encryption scheme.*

Proof sketch. The proof follows immediately from correctness of the underlying encryption scheme, because constant symbol ω does not appear in the scheme's ciphertext space. □

Lemma 22. *Given an asymmetric encryption scheme* Π *and a constant symbol* ω *such that* Π*'s ciphertext space does not contain* ω *and* Π*'s message space is larger than one for some security parameter, we have* $\mathsf{Leak}(\Pi, \omega)$ *does not satisfy* IND-PA0.

Proof sketch. The proof is trivial: an adversary can output two distinct messages and a vector containing constant symbol ω during the first two adversary calls, learn the private key from the parallel decryption, and use the key to recover the plaintext from the challenge ciphertext, which allows the adversary to win the game. □

Lemma 23. *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an asymmetric encryption scheme and* ω *be a constant symbol. Suppose* Π*'s ciphertext space does not contain* ω *and* Π*'s message space is smaller than the private key. Further suppose* $\mathsf{Enc2Vote}(\Pi)$ *satisfies* Ballot-Secrecy. *We have* $\mathsf{Enc2Vote}(\mathsf{Leak}(\Pi, \omega))$ *satisfies* Ballot-Secrecy.

Proof. Let $\mathsf{Enc2Vote}(\Pi) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ and let $\mathsf{Enc2Vote}(\mathsf{Leak}(\Pi, \omega)) = (\mathsf{Setup}', \mathsf{Vote}', \mathsf{Tally}')$. By definition of $\mathsf{Enc2Vote}(\Pi)$ and Leak, we have $\mathsf{Setup} = \mathsf{Setup}'$ and $\mathsf{Vote} = \mathsf{Vote}'$. Suppose $\mathfrak{m}$ is Π's message space. By definition of Leak, we have $\mathfrak{m}$ is $\mathsf{Leak}(\Pi, \omega)$'s message space too. Moreover, since $|\mathfrak{m}|$ is smaller than the private key, we have for all security parameters κ, bulletin boards $\mathfrak{bb}$, and number of candidates nc, that $nc \leqslant |\mathfrak{m}|$ implies

$$\Pr[(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa); (\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa);$$
$$(\mathfrak{v}', pf') \leftarrow \mathsf{Tally}'(sk, \mathfrak{bb}, nc, \kappa) : \mathfrak{v} = \mathfrak{v}' \wedge pf = pf'] = 1,$$

because $\mathsf{Enc2Vote}(\Pi)$ ensures that $\mathfrak{v}'$ is not influenced by decrypting ω (witness that decrypting ω outputs sk such that $sk > |\mathfrak{m}| \geqslant nc$) and pf is a constant symbol. It follows for all adversaries $\mathcal{A}$ and security parameters κ that games $\mathsf{Ballot\text{-}Secrecy}(\mathsf{Enc2Vote}(\Pi), \mathcal{A}, \kappa)$ and $\mathsf{Ballot\text{-}Secrecy}(\mathsf{Enc2Vote}(\mathsf{Leak}(\Pi, \omega)), \mathcal{A}, \kappa)$ are equivalent, hence, we have $\mathsf{Succ}(\mathsf{Ballot\text{-}Secrecy}(\mathsf{Enc2Vote}(\Pi), \mathcal{A}, \kappa)) = \mathsf{Succ}(\mathsf{Ballot\text{-}Secrecy}(\mathsf{Enc2Vote}(\mathsf{Leak}(\Pi, \omega), \mathcal{A}, \kappa))$. Moreover, since $\mathsf{Enc2Vote}(\Pi)$ satisfies Ballot-Secrecy, it follows that $\mathsf{Enc2Vote}(\mathsf{Leak}(\Pi, \omega))$ satisfies Ballot-Secrecy too. □

Proof of Proposition 12. Let Π be an asymmetric encryption scheme and ω be a constant symbol. Suppose Π's ciphertext space does not contain ω. Further suppose Π's message space is larger than one for some security parameter, but smaller than the private key. We have $\mathsf{Enc2Vote}(\mathsf{Leak}(\Pi, \omega))$ is an asymmetric encryption scheme (Lemma 21) such that $\mathsf{Enc2Vote}(\mathsf{Leak}(\Pi, \omega))$ satisfies Ballot-Secrecy (Lemma 23), but $\mathsf{Leak}(\Pi, \omega)$ does not satisfy IND-PA0 (Lemma 22), concluding our proof. □

C.6. Proof of Lemma 13

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and $\mathsf{Enc2Vote}(\Pi) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$. Election scheme $\mathsf{Enc2Vote}(\Pi)$ satisfies HK-Injectivity (Lemma 10). Suppose $\mathsf{Enc2Vote}(\Pi)$ does not satisfy Tally-Soundness, hence, there exists a probabilistic polynomial-time adversary $\mathcal{A}$, such that for all negligible functions negl, there exists a security parameter κ and $\mathsf{Succ}(\mathsf{Tally\text{-}Soundness}(\mathsf{Enc2Vote}(\Pi), \mathcal{A}, \kappa)) \leqslant 1 - \mathsf{negl}(\kappa)$. Further suppose (pk', sk, mb, mc) is an output of $\mathsf{Setup}(\kappa)$, $(nc, \mathfrak{bb})$ is an output of $\mathcal{A}(pk, \kappa)$, and $(\mathfrak{v}, pf)$ is an output of $\mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$. By definition of algorithm Setup, we have pk' is a pair $(pk, \mathfrak{m})$ such that $(pk, sk, \mathfrak{m})$ is an output of $\mathsf{Gen}(\kappa)$, and mc is the largest integer such that $\{0, \dots, mc\} \subseteq \{0\} \cup \mathfrak{m}$. Moreover, since $\mathcal{A}$ is a winning adversary, we have $nc \leqslant mc$. By definition of algorithm Tally, we have $\mathfrak{v}$ is initialised as a zero-filled vector of length nc and updated by computing **for** $b \in \mathfrak{bb}$ **do** $v \leftarrow \mathsf{Dec}(sk, b)$; **if** $1 \leqslant v \leqslant nc$ **then** $\mathfrak{v}[v] \leftarrow \mathfrak{v}[v] + 1$. Since Π satisfies well-definedness and error symbol $\perp$ is not an integer, that computation is equivalent to

```
for $b \in \mathfrak{bb} \wedge \exists m, r \,.\, m \in \mathfrak{m} \wedge b = \mathsf{Enc}(pk, m; r) \wedge b \neq \perp$ do
    $v \leftarrow \mathsf{Dec}(sk, b)$;
    if $1 \leqslant v \leqslant nc$ then
        $\mathfrak{v}[v] \leftarrow \mathfrak{v}[v] + 1$;
```

with overwhelming probability. Although each ciphertext $\mathsf{Enc}(pk, m; r) \in \mathfrak{bb}$ may not have been computed using coins r chosen uniformly at random, we nonetheless have $\mathsf{Dec}(sk, \mathsf{Enc}(pk, m; r)) = m$, because Π is perfectly correct. Hence, the above computation is equivalent to

```
for $b \in \mathfrak{bb} \wedge \exists v, r \,.\, v \in \mathfrak{m} \wedge b = \mathsf{Enc}(pk, v; r) \wedge b \neq \perp$ do
    if $1 \leqslant v \leqslant nc$ then
        $\mathfrak{v}[v] \leftarrow \mathfrak{v}[v] + 1$;
```

Thus, for all $v \in \{1, \dots, nc\}$, we have $\mathfrak{v}[v] = \ell$ if and only if $\exists^{=\ell} b \in \mathfrak{bb} \setminus \{\perp\} : \exists r : b = \mathsf{Enc}(pk, v; r)$, with overwhelming probability. It follows by definition of Vote that for all $v \in \{1, \dots, nc\}$ we have

$$\mathfrak{v}[v] = \ell \text{ iff } \exists^{=\ell} b \in \mathfrak{bb} \setminus \{\perp\} : \exists r : b = \mathsf{Vote}(pk, v, nc, \kappa; r)$$

with overwhelming probability. Thereby contradicting our assumption that $\mathcal{A}$ is a winning adversary, since $\mathfrak{v} = \textit{correct-outcome}(pk, nc, \mathfrak{bb}, \kappa)$, with overwhelming probability, which concludes our proof. □

Appendix D. Helios

Smyth, Frink & Clarkson [30] formalise a generic construction for Helios-like election schemes (Definition 23), which is parameterised on the choice of homomorphic encryption scheme and sigma protocols for the relations introduced in the following definition.

Definition 22 (from [30]). *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a homomorphic asymmetric encryption scheme and* Σ *be a sigma protocol for a binary relation R.*[39]

[39] Given a binary relation R, we write $((s_1, \dots, s_l), (w_1, \dots, w_k)) \in R \Leftrightarrow P(s_1, \dots, s_l, w_1, \dots, w_k)$ for $(s, w) \in R \Leftrightarrow P(s_1, \dots, s_l, w_1, \dots, w_k) \wedge s = (s_1, \dots, s_l) \wedge w = (w_1, \dots, w_k)$, hence, R is only defined over pairs of vectors of lengths l and k.

- Σ proves correct key generation *if a* $((\kappa, pk, \mathfrak{m}), (sk, s)) \in R \Leftrightarrow (pk, sk, \mathfrak{m}) = \mathsf{Gen}(\kappa; s)$.

Further, suppose that $(pk, sk, \mathfrak{m})$ *is the output of* $\mathsf{Gen}(\kappa; s)$*, for some security parameter* κ *and coins* s*.*

- Σ proves plaintext knowledge in a subspace *if* $((pk, c, \mathfrak{m}'), (m, r)) \in R \Leftrightarrow c = \mathsf{Enc}(pk, m; r) \wedge m \in \mathfrak{m}' \wedge \mathfrak{m}' \subseteq \mathfrak{m}$.
- Σ proves correct decryption *if* $((pk, c, m), sk) \in R \Leftrightarrow m = \mathsf{Dec}(sk, c)$.

Definition 23 (Generalised Helios [30])**.** *Suppose* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is an additively homomorphic asymmetric encryption scheme,* Σ_1 *is a sigma protocol that proves correct key generation,* Σ_2 *is a sigma protocol that proves plaintext knowledge in a subspace,* Σ_3 *is a sigma protocol that proves correct decryption, and* $\mathcal{H}$ *is a hash function. Let* $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$*,* $\mathsf{FS}(\Sigma_2, \mathcal{H}) = (\mathsf{ProveCiph}, \mathsf{VerCiph})$*, and* $\mathsf{FS}(\Sigma_3, \mathcal{H}) = (\mathsf{ProveDec}, \mathsf{VerDec})$*. We define election scheme* generalised Helios*, denoted* $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$*, as follows.*[40]

- $\mathsf{Setup}(\kappa)$*. Select coins* s *uniformly at random, compute* $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa; s); \rho \leftarrow \mathsf{ProveKey}((\kappa, pk, \mathfrak{m}), (sk, s), \kappa); pk' \leftarrow (pk, \mathfrak{m}, \rho); sk' \leftarrow (pk, sk)$*, let* m *be the largest integer such that* $\{0, \dots, m\} \subseteq \{0\} \cup \mathfrak{m}$*, and output* (pk', sk', m, m)*.*
- $\mathsf{Vote}(pk', v, nc, \kappa)$*. Parse* pk' *as a vector* $(pk, \mathfrak{m}, \rho)$*. Output* $\perp$ *if parsing fails or* $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) \neq 1 \vee v \notin \{1, \dots, nc\}$*. Select coins* $r_1, \dots, r_{nc-1}$ *uniformly at random and compute:*

```
for $1 \leqslant j \leqslant nc - 1$ do
    if $j = v$ then $m_j \leftarrow 1$; else $m_j \leftarrow 0$;
    $c_j \leftarrow \mathsf{Enc}(pk, m_j; r_j)$;
    $\sigma_j \leftarrow \mathsf{ProveCiph}((pk, c_j, \{0,1\}), (m_j, r_j), j, \kappa)$;
$c \leftarrow c_1 \otimes \cdots \otimes c_{nc-1}$;
$m \leftarrow m_1 \odot \cdots \odot m_{nc-1}$;
$r \leftarrow r_1 \oplus \cdots \oplus r_{nc-1}$;
$\sigma_{nc} \leftarrow \mathsf{ProveCiph}((pk, c, \{0,1\}), (m, r), nc, \kappa)$;
```

  Output ballot $(c_1, \dots, c_{nc-1}, \sigma_1, \dots, \sigma_{nc})$*.*
- $\mathsf{Tally}(sk', \mathfrak{bb}, nc, \kappa)$*. Initialise vectors* $\mathfrak{v}$ *of length nc and pf of length* $nc - 1$*. Compute* **for** $1 \leqslant j \leqslant nc$ **do** $\mathfrak{v}[j] \leftarrow 0$*. Parse* sk' *as a pair* (pk, sk)*. Output* $(\mathfrak{v}, pf)$ *if parsing fails. Let* $\{b_1, \dots, b_\ell\}$ *be the largest subset of* $\mathfrak{bb}$ *such that* $b_1 < \cdots < b_\ell$ *and for all* $1 \leqslant i \leqslant \ell$ *we have* b_i *is a vector of length* $2 \cdot nc - 1$ *and* $\bigwedge_{j=1}^{nc-1} \mathsf{VerCiph}((pk, b_i[j], \{0,1\}), b_i[j + nc - 1], j, \kappa) = 1 \wedge \mathsf{VerCiph}((pk, b_i[1] \otimes \cdots \otimes b_i[nc-1], \{0,1\}), b_i[2 \cdot nc - 1], nc, \kappa) = 1$*.If* $\{b_1, \dots, b_\ell\} = \emptyset$*, then output* $(\mathfrak{v}, pf)$*, otherwise,*

[40] We omit algorithm Verify for brevity.

compute:

```
for $1 \leqslant j \leqslant nc - 1$ do
    $c \leftarrow b_1[j] \otimes \cdots \otimes b_\ell[j]$;
    $\mathfrak{v}[j] \leftarrow \mathsf{Dec}(sk, c)$;
    $pf[j] \leftarrow \mathsf{ProveDec}((pk, c, \mathfrak{v}[j]), sk, \kappa)$;
$\mathfrak{v}[nc] \leftarrow \ell - \sum_{j=1}^{nc-1} \mathfrak{v}[j]$;
```

Output $(\mathfrak{v}, pf)$.

The above algorithms assume $nc > 1$*. Smyth, Frink & Clarkson define special cases of* Vote *and* Tally *when* $nc = 1$*. We omit those cases for brevity and, henceforth, assume nc is always greater than one.*

The generic construction can be instantiated to derive Helios 2.0 and Helios'16.

Definition 24 (Weak Fiat-Shamir transformation [71]). *The* weak Fiat-Shamir transformation *is a function* wFS *that is identical to* FS*, except that it excludes statement s in the hashes computed by* Prove *and* Verify*, as follows:* $\mathsf{chal} \leftarrow \mathcal{H}(\mathsf{comm})$.

Definition 25 (Helios 2.0 [30]). *Let* $\widehat{\mathsf{Helios}}$ *be* Helios *after replacing all instances of the Fiat-Shamir transformation with the weak Fiat-Shamir transformation and excluding the (optional) messages input to* ProveCiph*, i.e.,* ProveCiph *should be used as a ternary function.* Helios 2.0 *is* $\widehat{\mathsf{Helios}}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$*, where* Π *is additively homomorphic El Gamal [138, §2],* Σ_1 *is the sigma protocol for proving knowledge of discrete logarithms by Chaum et al. [139, Protocol 2],* Σ_2 *is the sigma protocol for proving knowledge of disjunctive equality between discrete logarithms by Cramer et al. [140, Figure 1],* Σ_3 *is the sigma protocol for proving knowledge of equality between discrete logarithms by Chaum and Pedersen [141, §3.2], and* $\mathcal{H}$ *is SHA-256 [142].*

Definition 26 (Helios 3.1.4 [30]). *Election scheme* Helios 3.1.4 *is* Helios 2.0 *after modifying the sigma protocols to perform the checks proposed by Chang-Fong & Essex [70, §4].*

Definition 27 (Helios'16 [30]). *Election scheme* Helios'16 *is* $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$*, where* Π *is additively homomorphic El Gamal [138, §2],* Σ_1 *is the sigma protocol for proving knowledge of discrete logarithms by Chaum et al. [139, Protocol 2],* Σ_2 *is the sigma protocol for proving knowledge of disjunctive equality between discrete logarithms by Cramer et al. [140, Figure 1],* Σ_3 *is the sigma protocol for proving knowledge of equality between discrete logarithms by Chaum & Pedersen [141, §3.2],* $\mathcal{H}$ *is a random oracle, and the sigma protocols are modified to perform the checks proposed by Chang-Fong & Essex [70, §4].*

Although Helios actually uses SHA-256 [142], we assume that $\mathcal{H}$ is a random oracle to prove Theorem 6. Moreover, we assume the sigma protocols used by Helios'16 satisfy the preconditions of generalised Helios, that is, [139, Protocol 2] is a sigma protocol for proving correct key generation, [140, Figure 1] is a sigma protocol for proving plaintext knowledge in a subspace, and [141, §3.2] is a sigma protocol for proving decryption. We leave formally proving this assumption as future work.

D.1. Proof of Theorem 6

The construction for Helios-like schemes produces election schemes with zero-knowledge tallying proofs (Lemma 24) that satisfy universal verifiability [30] and, thus, $\mathsf{Additivity}$ (Lemma 28). They also satisfy ballot independence (Proposition 25). Hence, they satisfy ballot secrecy too (Theorem 4). We show that Helios'16 satisfies ballot secrecy.

Henceforth, we assume Π, Σ_1, Σ_2 and Σ_3 satisfy the preconditions of Definition 23, and $\mathcal{H}$ is a random oracle. Let $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ and $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Moreover, let $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$, $\mathsf{FS}(\Sigma_2, \mathcal{H}) = (\mathsf{ProveCiph}, \mathsf{VerCiph})$, and $\mathsf{FS}(\Sigma_3, \mathcal{H}) = (\mathsf{ProveDec}, \mathsf{VerDec})$.

Lemma 24. *If* $(\mathsf{ProveDec}, \mathsf{VerDec})$ *is zero-knowledge, then* $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ *has zero-knowledge tallying proofs.*

Proof sketch. Suppose $\mathcal{A}$ is an adversary and κ is a security parameter. Further suppose (pk, sk, mb, mc) is an output of $\mathsf{Setup}(\kappa)$, $(nc, \mathfrak{bb})$ is an output of $\mathcal{A}(pk, \kappa)$, and $(\mathfrak{v}, pf)$ is an output of $\mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$, such that $|\mathfrak{bb}| \leqslant mb \wedge nc \leqslant mc$. By inspection of algorithm Tally, tallying proof pf is a vector of proofs produced by $\mathsf{ProveDec}$. Thus, there trivially exists a non-interactive proof system that could compute pf, moreover, that proof system is zero-knowledge because $(\mathsf{ProveDec}, \mathsf{VerDec})$ is zero-knowledge, which concludes our proof. □

Proposition 25. *Suppose* Π *is perfectly correct and satisfies* $\mathsf{IND\text{-}CPA}$. *Further suppose* $(\mathsf{ProveKey}, \mathsf{VerKey})$ *and* $(\mathsf{ProveCiph}, \mathsf{VerCiph})$ *satisfy special soundness and special honest verifier zero-knowledge. We have* $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ *satisfies* $\mathsf{IND\text{-}CVA}$.

Proof. By Theorem 16, the proof systems have extractors and simulators. Let $\mathsf{SimProveKey}$, respectively $\mathsf{SimProveCiph}$, be the simulator for $(\mathsf{ProveKey}, \mathsf{VerKey})$, respectively $(\mathsf{ProveCiph}, \mathsf{VerCiph})$. And let $\mathsf{ExtProveCiph}$ be the extractor for $(\mathsf{ProveCiph}, \mathsf{VerCiph})$.

Let $\mathsf{IND\text{-}CPA}^*$ be a variant of $\mathsf{IND\text{-}CPA}$ in which: 1) the adversary outputs two vectors of messages $\mathbf{m_0}$ and $\mathbf{m_1}$ such that $|\mathbf{m_0}| = |\mathbf{m_1}|$ and for all $1 \leqslant i \leqslant |\mathbf{m_0}|$ we have $|\mathbf{m_0}[i]| = |\mathbf{m_1}[i]|$ and $\mathbf{m_0}[i]$ and $\mathbf{m_1}[i]$ are from the encryption scheme's message space, and 2) the challenger computes $c_1 \leftarrow \mathsf{Enc}(pk, \mathbf{m}_\beta[1]); \ldots; c_{|\mathbf{m}_\beta|} \leftarrow \mathsf{Enc}(pk, \mathbf{m}_\beta[|\mathbf{m}_\beta|])$ and inputs $c_1, \ldots, c_{|\mathbf{m}_\beta|}$ to the adversary. We have Π satisfies $\mathsf{IND\text{-}CPA}^*$ [135, §10.2.2].

Suppose $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ does not satisfy $\mathsf{IND\text{-}CVA}$. Hence, there exists a probabilistic polynomial-time adversary $\mathcal{A}$, such that for all negligible functions negl, there exists a security parameter κ and $\frac{1}{2} + \mathsf{negl}(\kappa) < \mathsf{IND\text{-}CVA}(\Gamma, \mathcal{A}, \kappa)$. Since $\mathcal{A}$ is a winning adversary, we have $\mathcal{A}(pk', \kappa)$ outputs (v_0, v_1, nc) such that $v_0 \neq v_1$ with non-negligible probability, hence, either $v_0 < v_1$ or $v_1 < v_0$. For brevity, we suppose $v_0 < v_1$. (Our proof can be adapted to consider cases such that $v_1 < v_0$, but these details provide little value, so we do not pursue them.) We construct the following adversary $\mathcal{B}$ against $\mathsf{IND\text{-}CPA}^*$ from $\mathcal{A}$:

- $\mathcal{B}(pk, \mathfrak{m}, \kappa)$ outputs $((1, 0), (0, 1))$.
- $\mathcal{B}(\mathbf{c})$ proceeds as follows. First, compute:

  $\rho \leftarrow \mathsf{SimProveKey}((\kappa, pk, \mathfrak{m}), \kappa)$;
  $pk' \leftarrow (pk, \mathfrak{m}, \rho)$;
  $(v_0, v_1, nc) \leftarrow \mathcal{A}(pk', \kappa)$;

Secondly, select coins $r_1, \dots, r_{nc-1}$ uniformly at random and compute:

```
for $j \in \{1, \dots, nc-1\} \setminus \{v_0, v_1\}$ do
    $c_j \leftarrow \mathsf{Enc}(pk, 0; r_j)$;
    $\sigma_j \leftarrow \mathsf{ProveCiph}((pk, c_j, \{0,1\}), (0, r_j), j, \kappa)$;
$c_{v_0} \leftarrow \mathbf{c}[1]$;
$\sigma_{v_0} \leftarrow \mathsf{SimProveCiph}((pk, c_{v_0}, \{0,1\}), v_0, \kappa)$;
if $v_1 \neq nc$ then
    $c_{v_1} \leftarrow \mathbf{c}[2]$;
    $\sigma_{v_1} \leftarrow \mathsf{SimProveCiph}((pk, c_{v_1}, \{0,1\}), v_1, \kappa)$;
$c \leftarrow c_1 \otimes \cdots \otimes c_{nc-1}$;
$\sigma_{nc} \leftarrow \mathsf{SimProveCiph}((pk, c, \{0,1\}), nc, \kappa)$;
$b \leftarrow (c_1, \dots, c_{nc-1}, \sigma_1, \dots, \sigma_{nc})$;
$\mathfrak{bb} \leftarrow \mathcal{A}(b)$;
```

Thirdly, compute $\{b_1, \dots, b_\ell\}$ as the largest subset of $\mathfrak{bb}$ satisfying the conditions of algorithm Tally. Fourthly, initialise $\mathbf{H}$ as a transcript of the random oracle's input and output, $\mathbf{P}$ as a transcript of simulated proofs, $\mathbf{Q}$ as a vector of length $nc-1$, and $\mathfrak{v}$ as a zero-filled vector of length nc. Fifthly, compute:

$$\begin{aligned}\mathbf{Q} \leftarrow \Big(&\Big(\big(pk, b_1[1], \{0,1\}\big), b_1[nc]\Big), \dots, \\ &\Big(\big(pk, b_\ell[1], \{0,1\}\big), b_\ell[nc]\Big), \dots, \\ &\Big(\big(pk, b_1[nc-1], \{0,1\}\big), b_1[2 \cdot (nc-1)]\Big), \dots, \\ &\Big(\big(pk, b_\ell[nc-1], \{0,1\}\big), b_\ell[2 \cdot (nc-1)]\Big)\Big);\end{aligned}$$

$$\begin{aligned}&\mathbf{W} \leftarrow \mathsf{ExtProveCiph}(\mathbf{H}, \mathbf{P}, \mathbf{Q}); \\ &\mathfrak{v} \leftarrow (\Sigma_{i=1}^{\ell} \mathbf{W}[i][1],\ \dots,\ \Sigma_{i=\ell\cdot(nc-2)+1}^{\ell\cdot(nc-1)} \mathbf{W}[i][1],\ \ell - \Sigma_{j=1}^{nc-1} \mathfrak{v}[j]); \\ &g \leftarrow \mathcal{A}(\mathfrak{v});\end{aligned}$$

Finally, output g.

We prove that $\mathcal{B}$ wins $\mathsf{IND\text{-}CPA}^*$.

Suppose $(pk, sk, \mathfrak{m})$ is an output of $\mathsf{Gen}(\kappa)$ and $(\mathbf{m_0}, \mathbf{m_1})$ is an output of $\mathcal{B}(pk, \mathfrak{m}, \kappa)$. Let $\beta \in \{0,1\}$. Further suppose c_1 is an output of $\mathsf{Enc}(pk, \mathbf{m}_\beta[1])$ and c_2 is an output of $\mathsf{Enc}(pk, \mathbf{m}_\beta[2])$. Let $\mathbf{c} = (c_1, c_2)$. Moreover, suppose ρ is an output of $\mathsf{SimProveKey}((\kappa, pk, \mathfrak{m}), \kappa)$. Let $pk' = (pk, \mathfrak{m}, \rho)$. Suppose (v_0, v_1, nc) is an output of $\mathcal{A}(pk', \kappa)$. Since $\mathsf{SimProveKey}$ is a simulator for $(\mathsf{ProveKey}, \mathsf{VerKey})$, we have $\mathcal{B}$ simulates the challenger in $\mathsf{IND\text{-}CVA}$ to $\mathcal{A}(pk', \kappa)$. In particular, pk' is a triple containing a public key and corresponding message space generated Gen, and a (simulated) proof of correct key generation. Suppose $\mathcal{B}$ computes b and $\mathfrak{bb}$ is an output of $\mathcal{A}(b)$. Further suppose $\mathcal{B}$ computes $\mathfrak{v}$, and g is an output of $\mathcal{A}(\mathfrak{v})$. The following claims prove that $\mathcal{B}$ simulates the challenger in $\mathsf{IND\text{-}CVA}$ to $\mathcal{A}(b)$ and $\mathcal{A}(\mathfrak{v})$, hence, $g = \beta$, with at least the probability that $\mathcal{A}$ wins $\mathsf{IND\text{-}CVA}$, concluding our proof.

Claim 26. *Adversary $\mathcal{B}$'s computation of b is equivalent to computing b as $b \leftarrow \mathsf{Vote}(pk', v_\beta, nc, \kappa)$.*

Proof of Claim 26. We have pk' parses as a vector $(pk, \mathfrak{m}, \rho)$. Moreover, since $(pk, sk, \mathfrak{m})$ is an output of $\mathsf{Gen}(\kappa)$, there exist coins r such that $(pk, sk, \mathfrak{m}) = \mathsf{Gen}(\kappa; r)$. Hence, (sk, r) is a witness for statement $(\kappa, pk, \mathfrak{m})$. Furthermore, since $\mathsf{SimProveKey}$ is a simulator for $(\mathsf{ProveKey}, \mathsf{VerKey})$ and proofs output by $\mathsf{ProveKey}$ are indistinguishable from outputs of $\mathsf{SimProveKey}$, we have $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa)\kappa, pk, \mathfrak{m}\rho = 1$, with non-negligible probability. In addition, since $\mathcal{B}$ is a winning adversary, we have $v_0, v_1 \in \{1, \ldots, nc\}$, with non-negligible probability. It follows that $\mathsf{Vote}(pk', v_\beta, nc, \kappa)$ does not output $\perp$, with non-negligible probability. Indeed, computation $b \leftarrow \mathsf{Vote}(pk', v_\beta, nc, \kappa)$ is equivalent to the following. Select coins $r_1, \ldots, r_{nc-1}$ uniformly at random and compute:

```
for $1 \leqslant j \leqslant nc - 1$ do
    if $j = v_\beta$ then $m_j \leftarrow 1$; else $m_j \leftarrow 0$;
    $c_j \leftarrow \mathsf{Enc}(pk, m_j; r_j)$;
    $\sigma_j \leftarrow \mathsf{ProveCiph}((pk, c_j, \{0, 1\}), (m_j, r_j), j, \kappa)$;
$c \leftarrow c_1 \otimes \cdots \otimes c_{nc-1}$;
$m \leftarrow m_1 \odot \cdots \odot m_{nc-1}$;
$r \leftarrow r_1 \oplus \cdots \oplus r_{nc-1}$;
$\sigma_{nc} \leftarrow \mathsf{ProveCiph}((pk, c, \{0, 1\}), (m, r), nc, \kappa)$;
$b \leftarrow (c_1, \ldots, c_{nc-1}, \sigma_1, \ldots, \sigma_{nc})$;
```

Since $v_\beta \in \{v_0, v_1\}$, ciphertexts computed by the above for-loop all contain plaintext 0, except (possibly) ciphertext c_{v_0} and, if defined, ciphertext c_{v_1}. (Ciphertext c_{v_1} only exists if $v_1 < nc$.) Given that $v_0 < v_1 \leqslant nc$, ciphertext c_{v_0} contains $1 - \beta$, i.e., if $\beta = 0$, then c_{v_0} contains 1, otherwise $(\beta = 1)$, c_{v_0} contains 0. If $v_1 < nc$, then ciphertext c_{v_1} contains β. Moreover, since $\odot$ is the addition operator in group $(\mathfrak{m}, \odot)$ and 0 is the identity element in that group, if $v_1 = nc$, then plaintext m computed by the above algorithm is $1 - \beta$, otherwise, $m = 1 - \beta \odot \beta = 1$. Hence, the above algorithm is equivalent to selecting coins $r_1, \ldots, r_{nc-1}$ uniformly at random and computing:

```
for $j \in \{1, \ldots, nc - 1\} \setminus \{v_0, v_1\}$ do
    $c_j \leftarrow \mathsf{Enc}(pk, 0; r_j)$;
    $\sigma_j \leftarrow \mathsf{ProveCiph}((pk, c_j, \{0, 1\}), (0, r_j), j, \kappa)$;
$c_{v_0} \leftarrow \mathsf{Enc}(pk, 1 - \beta; r_{v_0})$;
$\sigma_{v_0} \leftarrow \mathsf{ProveCiph}((pk, c_{v_0}, \{0, 1\}), (1 - \beta, r_{v_0}), v_0, \kappa)$;
if $v_1 \neq nc$ then
    $c_{v_1} \leftarrow \mathsf{Enc}(pk, \beta; r_{v_1})$;
    $\sigma_{v_1} \leftarrow \mathsf{ProveCiph}((pk, c_{v_1}, \{0, 1\}), (\beta, r_{v_1}), v_1, \kappa)$;
$c \leftarrow c_1 \otimes \cdots \otimes c_{nc-1}$;
if $v_1 = nc$ then $m \leftarrow 1 - \beta$; else $m \leftarrow 1$;
$r \leftarrow r_1 \oplus \cdots \oplus r_{nc-1}$;
$\sigma_{nc} \leftarrow \mathsf{ProveCiph}((pk, c, \{0, 1\}), (m, r), nc, \kappa)$;
$b \leftarrow (c_1, \ldots, c_{nc-1}, \sigma_1, \ldots, \sigma_{nc})$;
```

Computation $c_{v_0} \leftarrow \mathsf{Enc}(pk, 1 - \beta; r_{v_0})$ is equivalent to $c_{v_0} \leftarrow \mathbf{c}[1]$, because if $\beta = 0$, then $\mathbf{c}[1]$ contains plaintext 1, otherwise $(\beta = 1)$, $\mathbf{c}[1]$ contains plaintext 0. Similarly, if $v_1 \neq nc$, then computation $c_{v_1} \leftarrow \mathsf{Enc}(pk, \beta; r_{v_1})$ is equivalent to $c_{v_1} \leftarrow \mathbf{c}[1]$. Moreover, proof $\mathsf{ProveCiph}((pk, c_{v_0}, \{0, 1\}), (1 - \beta, r_{v_0}), v_0, \kappa)$, respectively $\mathsf{ProveCiph}((pk, c_{v_1}, \{0, 1\}), (\beta, r_{v_1}), v_1, \kappa)$, can be simulated by $\mathsf{SimProveCiph}((pk,$

$c_{v_0}, \{0,1\}), v_0, \kappa)$, respectively $\mathsf{SimProveCiph}((pk, c_{v_1}, \{0,1\}), v_1, \kappa)$. Furthermore,

```
$c \leftarrow c_1 \otimes \cdots \otimes c_{nc-1}$;
if $v_1 = nc$ then $m \leftarrow 1 - \beta$; else $m \leftarrow 1$;
$r \leftarrow r_1 \oplus \cdots \oplus r_{nc-1}$;
$\sigma_{nc} \leftarrow \mathsf{ProveCiph}((pk, c, \{0,1\}), (m, r), nc, \kappa)$;
```

can be simulated by

```
$c \leftarrow c_1 \otimes \cdots \otimes c_{nc-1}$;
$\sigma_{nc} \leftarrow \mathsf{SimProveCiph}((pk, c, \{0,1\}), nc, \kappa)$;
```

Hence, we conclude the proof of this claim.

Claim 27. *Adversary $\mathcal{B}$'s computation of $\mathfrak{v}$ is equivalent to computing $\mathfrak{v}$ as $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk', \mathfrak{bb}, nc, \kappa)$, where $sk' = (pk, sk)$.*

Proof of Claim 27. Let $\{b_1, \dots, b_\ell\}$ be the largest subset of $\mathfrak{bb}$ satisfying the conditions of algorithm Tally. It is trivial to see that the claim holds when $\{b_1, \dots, b_\ell\} = \emptyset$, because $\mathfrak{v}$ is computed as a zero-filled vector of length nc in both cases. We prove the claim also holds when $\{b_1, \dots, b_\ell\} \neq \emptyset$.

By simulation sound extractability, for all $1 \leqslant i \leqslant \ell$ and $1 \leqslant j \leqslant nc - 1$, there exists a message $m_{i,j} \in \{0,1\}$ and coins $r_{i,j}$ and $r_{i,j+nc-1}$ such that $b_i[j] = \mathsf{Enc}(pk, m_{i,j}; r_{i,j})$ and $b_i[j+nc-1] = \mathsf{ProveCiph}((pk, b_i[j], \{0,1\}), (m_{i,j}, r_{i,j}), j, \kappa; r_{i,j+nc-1})$, with overwhelming probability. Suppose $\mathbf{Q}$ and $\mathbf{W}$ are computed by $\mathcal{B}$. We have for all $1 \leqslant i \leqslant \ell$ and $1 \leqslant j \leqslant nc-1$ that $\mathbf{Q}[\ell \cdot (j-1)+i] = ((pk, b_i[j], \{0,1\}), b_i[j+nc-1])$ and $\mathbf{W}[\ell \cdot (j-1)+i]$ is a witness for $(pk, b_i[j], \{0,1\})$, i.e., $(m_{i,j}, r_{i,j})$, and $\mathbf{W}[\ell \cdot (j-1)+i][1] = m_{i,j}$. Hence, adversary $\mathcal{B}$'s computation of $\mathfrak{v}$ is equivalent to computing $\mathfrak{v}$ as:

$$\mathfrak{v} \leftarrow (\Sigma_{i=1}^{\ell} m_{i,1}, \ \dots, \ \Sigma_{i=1}^{\ell} m_{i,nc-1}, \ \ell - \Sigma_{j=1}^{nc-1} \mathfrak{v}[j])$$

Moreover, computing $\mathfrak{v}$ as $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk', \mathfrak{bb}, nc, \kappa)$ is equivalent to initialising $\mathfrak{v}$ as a zero-filled vector of length nc and computing

```
for $1 \leqslant j \leqslant nc - 1$ do
    $c \leftarrow b_1[j] \otimes \cdots \otimes b_\ell[j]$;
    $\mathfrak{v}[j] \leftarrow \mathsf{Dec}(sk, c)$;
$\mathfrak{v}[nc] \leftarrow \ell - \sum_{j=1}^{nc-1} \mathfrak{v}[j]$;
```

Since Π is a homomorphic encryption scheme, we have for all $1 \leqslant j \leqslant nc - 1$ that $b_1[j] \otimes \cdots \otimes b_\ell[j]$ is a ciphertext with overwhelming probability. And although ciphertext $b_1[j] \otimes \cdots \otimes b_\ell[j]$ may not have been computed using coins chosen uniformly at random, we nevertheless have $\mathsf{Dec}(sk, b_1[j] \otimes \cdots \otimes b_\ell[j]) = m_{1,j} \odot \cdots \odot m_{\ell,j}$ with overwhelming probability, because Π is perfectly correct. It follows that $\mathfrak{v} = (m_{1,1} \odot \cdots \odot m_{\ell,1}, \ \dots, \ m_{1,nc-1} \odot \cdots \odot m_{\ell,nc-1}, \ \ell - \sum_{j=1}^{nc-1} \mathfrak{v}[j])$, with overwhelming probability. Let mb be the largest integer such that $\{0, \dots, mb\} \subseteq \mathfrak{m}$. Since $\mathcal{A}$ is a winning adversary, we have $\ell \leqslant mb$. Moreover, since $m_{1,j}, \dots, m_{\ell,j} \in \{0,1\}$ for all $1 \leqslant j \leqslant nc - 1$ and $\odot$ is the addition operator in group $(\mathfrak{m}, \odot)$, we have $m_{1,j} \odot \cdots \odot m_{\ell,j} = \sum_{i=1}^{\ell} m_{i,j}$, which suffices to conclude the proof of this claim. □

For Helios'16, encryption scheme Π is additively homomorphic El Gamal [138, §2]. Moreover, (ProveKey, VerKey), respectively (ProveCiph, VerCiph) and (ProveDec, VerDec), is the non-interactive proof system derived by application of the Fiat-Shamir transformation [136] to a random oracle $\mathcal{H}$ and the sigma protocol for proving knowledge of discrete logarithms by Chaum *et al.* [139, Protocol 2], respectively the sigma protocol for proving knowledge of disjunctive equality between discrete logarithms by Cramer *et al.* [140, Figure 1] and the sigma protocol for proving knowledge of equality between discrete logarithms by Chaum & Pedersen [141, §3.2].

Bernhard, Pereira & Warinschi [71, §4] remark that the sigma protocols underlying non-interactive proof systems $(\mathsf{ProveKey}, \mathsf{VerKey})$ and $(\mathsf{ProveCiph}, \mathsf{VerCiph})$ both satisfy special soundness and special honest verifier zero-knowledge, hence, Theorem 16 is applicable. Bernhard, Pereira & Warinschi also remark that the sigma protocol underlying $(\mathsf{ProveDec}, \mathsf{VerDec})$ satisfies special soundness and "almost special honest verifier zero-knowledge" and argue that "we could fix this[, but] it is easy to see that ... all relevant theorems [including Theorem 16] still hold." We adopt the same position and assume that Theorem 16 is applicable.

Proof of Theorem 6. Helios'16 has zero-knowledge tallying proofs (Lemma 24), subject to the applicability of Theorem 16 to the sigma protocol underlying $(\mathsf{ProveDec}, \mathsf{VerDec})$. Moreover, since Helios'16 satisfies UV [30], we have Helios'16 satisfies $\mathsf{Additivity}(\Gamma, \mathcal{A}, \kappa)$ (Lemma 28). Furthermore, since El Gamal satisfies IND-CPA [135, 143] and is perfectly correct, and since non-interactive proof systems $(\mathsf{ProveKey}, \mathsf{VerKey})$ and $(\mathsf{ProveCiph}, \mathsf{VerCiph})$ satisfy special soundness and special honest verifier zero-knowledge, we have Helios'16 satisfies IND-CVA (Proposition 25). Hence, Helios'16 satisfies Ballot-Secrecy too (Theorem 4). □

Appendix E. Universal verifiability implies tally soundness

We recall the definition of universal verifiability by Smyth, Frink & Clarkson [30] and show that verifiable election schemes satisfy Tally-Soundness (Lemma 28). This is useful to simplify applications of Theorems 4, 14, & 30. Indeed, our ballot-secrecy proofs for Helios and Helios Mixnet make use of this result.

We extend our syntax for election schemes (Definition 1) to include a probabilistic polynomial-time algorithm Verify:

- Verify, denoted $s \leftarrow \mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa)$, is run to audit an election. It takes as input a public key pk, a bulletin board $\mathfrak{bb}$, some number of candidates nc, an election outcome $\mathfrak{v}$, a tallying proof pf, and a security parameter κ. It outputs a bit s, where 1 signifies success and 0 failure.

We previously omitted algorithm Verify, because we did not focus on verifiability in the main body.

For universal verifiability, anyone must be able to check whether the election outcome represents the votes used to construct ballots on the bulletin board. The formal definition of universal verifiability by Smyth, Frink & Clarkson requires algorithm Verify to accept if and only if the election outcome is correct. The *if* requirement is captured by completeness (Definition 28), 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 (Definition 30), which challenges an adversary to concoct a scenario in which algorithm Verify accepts, but the election outcome is not correct.

Definition 28 (Completeness [30])**.** *An election scheme* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *satisfies* completeness*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such* *that for all security parameters* κ*, we have* $\Pr[(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa); (\mathfrak{bb}, nc) \leftarrow \mathcal{A}(pk,\kappa); (\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa) : |\mathfrak{bb}| \leqslant mb \wedge nc \leqslant mc \Rightarrow \mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa) = 1] > 1 - \mathsf{negl}(\kappa)$.

Definition 29 (Injectivity [30, 80])**.** *An election scheme* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *satisfies* injectivity*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, security parameters* κ *and computations* $(pk, nc, v, v') \leftarrow \mathcal{A}(\kappa); b \leftarrow \mathsf{Vote}(pk, v, nc, \kappa); b' \leftarrow \mathsf{Vote}(pk, v', nc, \kappa)$ *such that* $v \neq v' \wedge b \neq \perp \wedge b' \neq \perp$*, we have* $b \neq b'$*.*

Definition 30 (Soundness [30])**.** *An election scheme* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *satisfies* soundness*, if* Γ *satisfies injectivity and for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such that for all security parameters* κ*, we have* $\Pr[(pk, nc, \mathfrak{bb}, \mathfrak{v}, pf) \leftarrow \mathcal{A}(\kappa) : \mathfrak{v} \neq \textit{correct-outcome}(pk, nc, \mathfrak{bb}, \kappa) \wedge \mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa) = 1] \leqslant \mathsf{negl}(\kappa)$.

Definition 31 (UV [30, 80])**.** *An election scheme* Γ *satisfies* universal verifiability (UV)*, if completeness, injectivity and soundness are satisfied.*

Lemma 28. *If election scheme* Γ *satisfies completeness and soundness, then* Γ *satisfies* Tally-Soundness.

Proof. Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$. Suppose there exists a probabilistic polynomial-time adversary $\mathcal{A}$ that wins Tally-Soundness against Γ. We construct an adversary $\mathcal{B}$ against Exp-UV-Ext from $\mathcal{A}$. We define $\mathcal{B}$ such that $\mathcal{B}(\kappa) = (pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa); (nc, \mathfrak{bb}) \leftarrow \mathcal{A}(pk, \kappa); (\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa); \textbf{return}(pk, nc, \mathfrak{bb}, \mathfrak{v}, pf)$. Suppose (pk, sk, mb, mc) is an output of $\mathsf{Setup}(\kappa)$, $(nc, \mathfrak{bb})$ is an output of $\mathcal{A}(pk, \kappa)$, and $(\mathfrak{v}, pf)$ is an output of $\mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$. Since $\mathcal{A}$ is a winning adversary, we have $\mathfrak{v} \neq \textit{correct-outcome}(pk, nc, \mathfrak{bb}, \kappa) \wedge |\mathfrak{bb}| \leqslant mb \wedge nc \leqslant mc$, with non-negligible probability. And, by completeness, we have $\mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa) = 1$, with overwhelming probability. Thereby concluding our proof. □

The reverse implication of Lemma 28 does not hold: Observe that Tally-Soundness only ensures algorithm Tally tallies ballots correctly, whereas UV additionally ensures that anyone can check whether ballots are tallied correctly.

Appendix F. Encryption-based voting systems

We have seen that election scheme $\mathsf{Enc2Vote}(\Pi)$ satisfies HK-Injectivity, if Π is perfectly correct (Lemma 10). But, HK-Injectivity assumes public keys are computed using the key generation algorithm. Thus, perfect correctness is insufficient to ensure injectivity when public keys are controlled by an adversary. Nonetheless, this can be ensured using proofs of correct key generation. A sub-class of schemes generated by Enc2Vote prove correct key generation. Indeed, we can consider schemes $\mathsf{Enc2Vote}(\Pi)$ such that Gen proves correct key generation and Enc verifies such proofs, where $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Alternatively, we can couple Enc2Vote with proofs of correct key generation:

Definition 32 ($\mathsf{Enc2Vote}^{+}$ [80])**.** *Suppose* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is an asymmetric encryption scheme,* Σ *is a sigma protocol that proves correct key generation, and* $\mathcal{H}$ *is a hash function. Let* $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$*. We define* $\mathsf{Enc2Vote}^{+}(\Pi, \Sigma, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *such that:*

- $\mathsf{Setup}(\kappa)$ *selects coins s uniformly at random, computes* $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa; s); \rho \leftarrow \mathsf{ProveKey}((\kappa, pk, \mathfrak{m}), (sk, s), \kappa); pk' \leftarrow (pk, \mathfrak{m}, \rho); sk' \leftarrow (pk, sk)$*, derives mc as the largest integer such that* $\{0, \ldots, mc\} \subseteq \{0\} \cup \mathfrak{m}$ *and for all* $m_0, m_1 \in \{1, \ldots, mc\}$ *we have* $|m_0| = |m_1|$*, and outputs* $(pk', sk', p(\kappa), mc)$*, where p is a polynomial function.*
- $\mathsf{Vote}(pk', v, nc, \kappa)$ *parses pk′ as vector* $(pk, \mathfrak{m}, \rho)$*, outputting* $\perp$ *if parsing fails or* $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) \neq 1 \vee v \notin \{1, \ldots, nc\} \vee \{1, \ldots, nc\} \not\subseteq \mathfrak{m}$*, computes* $b \leftarrow \mathsf{Enc}(pk, v)$*, and outputs b.*
- $\mathsf{Tally}(sk', \mathfrak{bb}, nc, \kappa)$ *initialises* $\mathfrak{v}$ *as a zero-filled vector of length nc, parses sk′ as pair* (pk, sk)*, outputting* $(\mathfrak{v}, \perp)$ *if parsing fails, computes* **for** $b \in \mathfrak{bb}$ **do** $v \leftarrow \mathsf{Dec}(sk, b)$; **if** $1 \leqslant v \leqslant nc$ **then** $\mathfrak{v}[v] \leftarrow \mathfrak{v}[v] + 1$*, and outputs* $(\mathfrak{v}, \epsilon)$*, where* ϵ *is a constant symbol.*

Lemma 29. *Given an asymmetric encryption scheme* Π *satisfying* IND-CPA*, a sigma protocol* Σ *that proves correct key generation, and a hash function* $\mathcal{H}$*, we have* $\mathsf{Enc2Vote}^+(\Pi, \Sigma, \mathcal{H})$ *is an election scheme.*

A proof of Lemma 29 follows from [80].[41]

Although the set of election schemes produced by $\mathsf{Enc2Vote}^+$ is not a subset of the schemes produced by Enc2Vote, there is nonetheless a straightforward mapping from the former to the latter. Thus, the results in Section 5 also hold for $\mathsf{Enc2Vote}^+$:

Theorem 30. *Let* $\mathsf{Enc2Vote}^+(\Pi, \Sigma, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$*, where* Π *is an asymmetric encryption scheme,* Σ *is a sigma protocol that proves correct key generation, and* $\mathcal{H}$ *is a random oracle. Moreover, let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}')$ *for some algorithm* Tally′ *such that* Γ *is an election scheme with zero-knowledge tallying proofs. Suppose* Π *is perfectly correct and satisfies* IND-PA0 *and well-definedness. Moreover, suppose* Σ *is perfectly complete and* $\mathsf{FS}(\Sigma, \mathcal{H})$ *satisfies zero-knowledge. Further suppose* Γ *satisfies* Tally-Soundness. *We have* Γ *satisfies* Ballot-Secrecy.

Proof. Let $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$ and $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Moreover, let asymmetric encryption scheme $\Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec})$ such that

- $\mathsf{Gen}'(\kappa)$ selects coins s uniformly at random, computes $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa; s); \rho \leftarrow \mathsf{ProveKey}((\kappa, pk, \mathfrak{m}), (sk, s), \kappa); pk' \leftarrow (pk, \mathfrak{m}, \rho)$, and outputs $(pk', sk, \mathfrak{m})$.
- $\mathsf{Enc}'(pk, v)$ parses pk' as a vector $(pk, \mathfrak{m}, \rho)$, outputting $\perp$ if parsing fails or $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) \neq 1$, computes ciphertext $c \leftarrow \mathsf{Enc}(pk, v)$, and outputs c.

Since Π is perfectly correct and Σ is perfectly complete, we have Π' is perfectly correct. Moreover, since Π satisfies well-definedness, we have Π' does too. Furthermore, since $\mathsf{FS}(\Sigma, \mathcal{H})$ satisfies zero-knowledge and Π satisfies IND-PA0, we have Π' satisfies IND-PA0. It follows that $\mathsf{Enc2Vote}(\Pi')$ satisfies Tally-Soundness and IND-CVA (Corollary 11 & Lemma 13).

We have $\mathsf{Enc2Vote}(\Pi') = (\mathsf{Setup}', \mathsf{Vote}', \mathsf{Tally})$ such that Setup′ is Setup except Setup outputs public key pk' as a vector $(pk, \mathfrak{m}, \rho)$, whereas Setup′ outputs public key $(pk, \mathfrak{m})$. Moreover, Vote′ is Vote except Vote inputs public key $(pk, \mathfrak{m})$ whereas Vote′ inputs public key $(pk, \mathfrak{m}, \rho)$. (This blight motivated the inclusion of this appendix.) Hence, it is straightforward to see that $\mathsf{Enc2Vote}^+(\Pi, \Sigma, \mathcal{H})$ satisfies Tally-Soundness and IND-CVA, because $\mathsf{Enc2Vote}(\Pi')$ does. Thus, Γ satisfies IND-CVA (Proposition 9) and Ballot-Secrecy (Theorem 4 & Lemma 8). □

[41] Smyth considers instantiating $\mathsf{Enc2Vote}^+$ with a broad class of asymmetric encryption schemes that produce distinct ciphertexts with overwhelming probability [80], whereas we consider a strictly narrower class of schemes satisfying IND-CPA. This avoids having to recall Smyth's notion of distinct ciphertexts.

Appendix G. Helios Mixnet

We recall a generic construction for election schemes similar to Helios Mixnet (Definition 34). The construction is parameterised on the choice of homomorphic encryption scheme and sigma protocols for the relations introduced in Definition 22 and the following definition.

Definition 33 (from [30]). *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a homomorphic asymmetric encryption scheme and* Σ *be a sigma protocol for a binary relation* R*. Suppose that* $(pk, sk) = \mathsf{Gen}(\kappa; s)$*, for some security parameter* κ *and coins* s*, and* $\mathfrak{m}$ *is the encryption scheme's plaintext space.*

- Σ proves plaintext knowledge *if* $((pk, c), (m, r)) \in R \Leftrightarrow c = \mathsf{Enc}(pk, m; r) \wedge m \in \mathfrak{m}$.
- Σ proves mixing *if* $((pk, \mathbf{c}, \mathbf{c}'), (\mathbf{r}, \chi)) \in R \Leftrightarrow \bigwedge_{1 \leqslant i \leqslant |\mathbf{c}|} \mathbf{c}'[i] = \mathbf{c}[\chi(i)] \otimes \mathsf{Enc}(pk, \mathfrak{e}; \mathbf{r}[i]) \wedge |\mathbf{c}| = |\mathbf{c}'| = |\mathbf{r}|$*, where* $\mathbf{c}$ *and* $\mathbf{c}'$ *are both vectors of ciphertexts encrypted under* pk*,* $\mathbf{r}$ *is a vector of coins,* χ *is a permutation on* $\{1, \ldots, |\mathbf{c}|\}$*, and* $\mathfrak{e}$ *is an identity element of the encryption scheme's message space with respect to* $\odot$*.*

Definition 34 (HeliosM [61, 144]). *Suppose* $\Pi_0 = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a homomorphic asymmetric encryption algorithm,* Σ_1 *is a sigma protocol that proves correct key construction,* Σ_2 *is a sigma protocol that proves plaintext knowledge, and* $\mathcal{H}$ *is a hash function. Let* $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$ *and* $\mathsf{FS}(\Sigma_2, \mathcal{H}) = (\mathsf{ProveCiph}, \mathsf{VerCiph})$*. Moreover, let* $\pi(\Pi, \Sigma_2, \mathcal{H}) = (\mathsf{Gen}, \mathsf{Enc}', \mathsf{Dec}')$ *be an asymmetric encryption scheme such that:*

- $\mathsf{Enc}'(pk, v)$ *selects coins* r *uniformly at random, computes* $c \leftarrow \mathsf{Enc}(pk, v; r); \sigma \leftarrow \mathsf{ProveCiph}((pk, c), (v, r), \kappa)$*, and outputs* (c, σ)*.*
- $\mathsf{Dec}'(sk, c')$ *parses* c' *as* (c, σ)*, outputting* $\perp$ *if parsing fails or* $\mathsf{VerCiph}((pk, c), \sigma, \kappa) \neq 1$*, computes* $v \leftarrow \mathsf{Dec}(sk, c)$*, and outputs* v*.*

Let $\mathsf{Enc2Vote}^{+}(\pi(\Pi, \Sigma_2, \mathcal{H}), \Sigma_1, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}')$*. Suppose* Σ_3 *is a sigma protocol that proves correct decryption and* Σ_4 *is a sigma protocol that proves mixing. Let* $\mathsf{FS}(\Sigma_3, \mathcal{H}) = (\mathsf{ProveDec}, \mathsf{VerDec})$ *and* $\mathsf{FS}(\Sigma_4, \mathcal{H}) = (\mathsf{ProveMix}, \mathsf{VerMix})$*. We define* $\mathsf{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$*, where algorithm* Tally *is defined below.*[42]

$\mathsf{Tally}(sk', nc, \mathfrak{bb}, \kappa)$ *initialises* $\mathfrak{v}$ *as a zero-filled vector of length* nc*; parses* sk' *as a pair* (pk, sk)*, outputting* $(\mathfrak{v}, \perp)$ *if parsing fails; and proceeds as follows:*

(1) Remove invalid ballots. *Let* $\{b_1, \ldots, b_\ell\}$ *be the largest subset of* $\mathfrak{bb}$ *such that for all* $1 \leqslant i \leqslant \ell$ *we have* b_i *is a pair and* $\mathsf{VerCiph}((pk, b_i[1]), b_i[2], \kappa) = 1$*. If* $\{b_1, \ldots, b_\ell\} = \emptyset$*, then output* $(\mathfrak{v}, \perp)$*.*

(2) Mix. *Select a permutation* χ *on* $\{1, \ldots, \ell\}$ *uniformly at random, initialise* $\mathbf{bb}$ *and* $\mathbf{r}$ *as a vector of length* ℓ*, fill* $\mathbf{r}$ *with coins chosen uniformly at random, and compute*

```
for $1 \leqslant i \leqslant \ell$ do
    $\mathbf{bb}[i] \leftarrow b_{\chi(i)}[1] \otimes \mathsf{Enc}(pk, \mathfrak{e}; \mathbf{r}[i])$;
$pf_1 \leftarrow \mathsf{ProveMix}((pk, (b_1[1], \ldots, b_\ell[1]), \mathbf{bb}), (\mathbf{r}, \chi), \kappa)$;
```

where $\mathfrak{e}$ *is an identity element of* Π*'s message space with respect to* $\odot$*.*

[42] We omit algorithm Verify for brevity.

(3) Decrypt. *Initialise* $\mathbf{W}$ *and* pf_2 *as vectors of length* ℓ *and compute:*

```
for $1 \leqslant i \leqslant \ell$ do
    $\mathbf{W}[i] \leftarrow \mathsf{Dec}(sk, \mathbf{bb}[i])$;
    $pf_2[i] \leftarrow \mathsf{ProveDec}((pk, \mathbf{bb}[i], \mathbf{W}[i]), sk, \kappa)$;
    if $1 \leqslant \mathbf{W}[i] \leqslant nc$ then
        $\mathfrak{v}[\mathbf{W}[i]] \leftarrow \mathfrak{v}[\mathbf{W}[i]] + 1$;
```

Output $(\mathfrak{v}, (\mathbf{bb}, pf_1, \mathbf{W}, pf_2))$.

Definition 35 (HeliosM'17)**.** HeliosM'17 *is the set of election schemes that includes every* $\mathsf{HeliosM}(\Pi_0, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ *such that* Π_0*,* Σ_1*,* Σ_2*,* Σ_3*,* Σ_4 *and* $\mathcal{H}$ *satisfy the preconditions of Definition 34, moreover,* Π_0 *is perfectly correct and* Σ_1 *and* Σ_2 *are perfectly complete, furthermore,* Π_0 *satisfies* IND-CPA*,* Σ_1*,* Σ_2*,* Σ_3 *and* Σ_4 *satisfy special soundness and special honest verifier zero-knowledge,* $\mathcal{H}$ *is a random oracle, and* $\mathsf{HeliosM}(\Pi_0, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ *satisfies* UV.

Smyth has shown that there exists an election scheme in HeliosM'17 that satisfies UV [144]. Hence, set HeliosM'17 is not empty.

G.1. Proof of Theorem 15

Let election scheme $\Gamma = \mathsf{HeliosM}(\Pi_0, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ and asymmetric encryption scheme $\Pi = \pi(\Pi_0, \Sigma_2, \mathcal{H})$. It follows that election scheme $\mathsf{Enc2Vote}^{+}(\Pi, \Sigma_1, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}')$. Moreover, since Σ_1 satisfies special soundness and special honest verifier zero-knowledge, we have $\mathsf{FS}(\Sigma_1, \mathcal{H})$ satisfies zero-knowledge (Theorem 16). We use Theorem 30 to prove that $\Gamma \in HeliosM'17$ satisfies Ballot-Secrecy.

Since Π_0 is perfectly correct and Σ_2 is perfectly complete, we have Π is a perfectly correct. Moreover, since Σ_2 satisfies special soundness and special honest verifier zero-knowledge, we have $\mathsf{FS}(\Sigma_2, \mathcal{H})$ satisfies simulation sound extractability (Theorem 16), hence, Π satisfies CNM-CPA [71, Theorem 2] and, equivalently, IND-PA0 [65].

To prove $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ satisfies well-definedness, suppose $\mathcal{A}$ is a probabilistic polynomial-time adversary, κ is a security parameter, $(pk, sk, \mathfrak{m})$ is an output of $\mathsf{Gen}(\kappa)$, and c is an output of $\mathcal{A}(pk, \mathfrak{m}, \kappa)$ such that $\mathsf{Dec}(sk, c) \neq \perp$. By definition of Dec, we have c is a pair (hence, $c \neq \perp$) such that $\mathsf{FS}(\Sigma_2, \mathcal{H})$ can verify $c[2]$ with respect to pk and $c[1]$. Since $\mathsf{FS}(\Sigma_2, \mathcal{H})$ satisfies simulation sound extractability, we have $c[2]$ is a proof computed using $\mathsf{FS}(\Sigma_2, \mathcal{H})$ and there exists plaintext $m \in \mathfrak{m}$ and coins r such that $c[1] = \mathsf{Enc}(pk, m; r)$, with overwhelming probability. Thus, Π satisfies well-definedness.

Since Σ_3 and Σ_4 satisfy special soundness and special honest verifier zero-knowledge, we have $\mathsf{FS}(\Sigma_3, \mathcal{H})$ and $\mathsf{FS}(\Sigma_4, \mathcal{H})$ satisfy zero-knowledge (Theorem 16), therefore, Γ has zero-knowledge tallying proofs by reasoning similar to that given in the proof sketch of Lemma 24. Moreover, since Γ satisfies universal verifiability, we have Γ satisfies Tally-Soundness (Lemma 28).

We conclude by Theorem 30. □

Appendix H. Ballot-Secrecy is strictly stronger than IND-SEC

Smyth & Bernhard propose definitions of ballot secrecy that consider an adversary that cannot control the bulletin board nor the communication channel [27, 28]. As discussed in Section 8, their original definition [27] is too strong [29, §3.5] and they propose a revision [28]. We recall their syntax (Definition 36) and revised definition (Definition 37), define a transformation from their syntax to ours (Definition 38), and prove Ballot-Secrecy is strictly stronger than their definition (Theorem 32).

Definition 36 (Election scheme with a trusted bulletin board)**.** *An* election scheme with a trusted bulletin board *is a tuple of efficient algorithms* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{BB}, \mathsf{Tally})$ *such that:*

- Setup*, denoted* $(pk, sk, \mathfrak{m}, \mathfrak{bb}) \leftarrow \mathsf{Setup}(\kappa)$*, takes a security parameter* κ *as input and outputs a key pair* pk, sk*, a vote space* $\mathfrak{m}$*, and a bulletin board* $\mathfrak{bb}$*, where* $\mathfrak{m}$ *and* $\mathfrak{bb}$ *are both sets.*
- Vote*, denoted* $b \leftarrow \mathsf{Vote}(pk, v)$*, takes a public key* pk *and vote* v *as input and outputs a ballot* b*. Vote* v *should be selected from the vote space* $\mathfrak{m}$*.*
- BB*, denoted* $\mathfrak{bb}' \leftarrow \mathsf{BB}(\mathfrak{bb}, b)$*, takes a bulletin board* $\mathfrak{bb}$ *and ballot* b *as input and outputs an updated bulletin board* $\mathfrak{bb}'$*.*
- Tally*, denoted* $(\mathfrak{o}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb})$*, takes a private key* sk *and bulletin board* $\mathfrak{bb}$ *as input and outputs an election outcome* $\mathfrak{o}$ *and a tallying proof* pf*, where* $\mathfrak{o}$ *is a multiset of votes.*

Election schemes with a trusted bulletin board must satisfy correctness*, that is, for all security parameters* κ*, votes* v *and multisets* $\mathfrak{bb}$*, we have:* $\Pr[(pk, sk, \mathfrak{m}, \mathfrak{bb}_0) \leftarrow \mathsf{Setup}(\kappa); b \leftarrow \mathsf{Vote}(pk, v); \mathfrak{bb}' \leftarrow \mathsf{BB}(\mathfrak{bb}, b); (\mathfrak{o}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}); (\mathfrak{o}', pf') \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}') : \mathfrak{bb}' = \mathfrak{bb} \cup \{b\} \wedge (\mathfrak{o} \neq \emptyset \Rightarrow \mathfrak{o}' = \mathfrak{o} \cup \{v\} \wedge |\mathfrak{v}| = |\mathfrak{bb}|) \wedge (\mathfrak{o} = \emptyset \Rightarrow \mathfrak{o}' = \emptyset)] > 1 - \mathsf{negl}(\kappa)$.

Definition 37. *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{BB}, \mathsf{Tally})$ *be an election scheme with a trusted bulletin board,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* IND-SEC *be the following game.*

```
$\mathsf{IND\text{-}SEC}(\Gamma, \mathcal{A}, \kappa) =$
    $(pk, sk, \mathfrak{m}, \mathfrak{bb}_0) \leftarrow \mathsf{Setup}(\kappa);$
    $\mathfrak{bb}_1 \leftarrow \mathfrak{bb}_0; \beta \leftarrow_R \{0,1\};$
    $L_0 \leftarrow \emptyset; L_1 \leftarrow \emptyset;$
    $x \leftarrow \mathcal{A}^{\mathcal{O}}(pk, \mathfrak{m});$
    if $L_0 = L_1$ then
        $(\mathfrak{o}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}_\beta);$
    else
        $(\mathfrak{o}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}_0);$
        $pf \leftarrow \perp;$
    $g \leftarrow \mathcal{A}(\mathfrak{o}, pf);$
    return $g = \beta;$
```

In the above game, L_0 *and* L_1 *are multisets and oracle* $\mathcal{O}$ *is defined as follows:*

- $\mathcal{O}(v_0, v_1)$ *computes* $L_0 \leftarrow L_0 \cup \{v_0\}; L_1 \leftarrow L_1 \cup \{v_1\}; b_0 \leftarrow \mathsf{Vote}(pk, v_0); \mathfrak{bb}_0 \leftarrow \mathsf{BB}(\mathfrak{bb}_0, b_0); b_1 \leftarrow \mathsf{Vote}(pk, v_1); \mathfrak{bb}_1 \leftarrow \mathsf{BB}(\mathfrak{bb}_1, b_1)$*, where* $v_0, v_1 \in \mathfrak{m}$*.*
- $\mathcal{O}(b)$ *computes* $\mathfrak{bb}' \leftarrow \mathfrak{bb}_\beta; \mathfrak{bb}_\beta \leftarrow \mathsf{BB}(\mathfrak{bb}_\beta, b);$ **if** $\mathfrak{bb}_\beta \neq \mathfrak{bb}'$ **then** $\mathfrak{bb}_{1-\beta} \leftarrow \mathsf{BB}(\mathfrak{bb}_{1-\beta}, b)$*.*
- $\mathcal{O}()$ *outputs* $\mathfrak{bb}_\beta$*.*

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

Observe that game $\mathsf{IND\text{-}SEC}$ uses bulletin boards $\mathfrak{bb}_\beta$ and $\mathfrak{bb}_0$, hence, $\mathfrak{bb}_1$ is unused when $\beta = 0$. It follows that game $\mathsf{IND\text{-}SEC}$ is equivalent to a variant that redefines the following oracle calls:

- $\mathcal{O}(v_0, v_1)$ computes $L_0 \leftarrow L_0 \cup \{v_0\}; L_1 \leftarrow L_1 \cup \{v_1\}; b_\beta \leftarrow \mathsf{Vote}(pk, v_\beta); \mathfrak{bb}_\beta \leftarrow \mathsf{BB}(\mathfrak{bb}_\beta, b_\beta);$ **if** $\beta = 1$ **then** $b_0 \leftarrow \mathsf{Vote}(pk, v_0); \mathfrak{bb}_0 \leftarrow \mathsf{BB}(\mathfrak{bb}_0, b_0)$, where $v_0, v_1 \in \mathfrak{m}$.
- $\mathcal{O}(b)$ computes $\mathfrak{bb}' \leftarrow \mathfrak{bb}_\beta; \mathfrak{bb}_\beta \leftarrow \mathsf{BB}(\mathfrak{bb}_\beta, b);$ **if** $\mathfrak{bb}_\beta \neq \mathfrak{bb}' \wedge \beta = 1$ **then** $\mathfrak{bb}_0 \leftarrow \mathsf{BB}(\mathfrak{bb}_0, b)$.

Let that variant be $\mathsf{IND\text{-}SEC}^*$.

Lemma 31. *An election scheme with a trusted bulletin board* Γ *satisfies* $\mathsf{IND\text{-}SEC}$ *iff* Γ *satisfies* $\mathsf{IND\text{-}SEC}^*$.

Lemma 31 follows from our informal reasoning and we omit a formal proof.

Definition 38. *Given an election scheme with a trusted bulletin board* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{BB}, \mathsf{Tally})$ *such that for all security parameters* κ *and computations* $(pk, sk, \mathfrak{m}, \mathfrak{bb}) \leftarrow \mathsf{Setup}(\kappa)$ *we have* $\mathfrak{m} = \{1, \ldots, mc\}$ *for some integer mc, we define* $\gamma(\Gamma) = (\mathsf{Setup}', \mathsf{Vote}', \mathsf{Tally}')$ *such that*

- $\mathsf{Setup}'(\kappa)$ *computes* $(pk, sk, \mathfrak{m}, \mathfrak{bb}) \leftarrow \mathsf{Setup}(\kappa); mc \leftarrow |\mathfrak{m}|; pk' \leftarrow (pk, mc)$ *and outputs* $(pk', sk, mc, p(\kappa))$, *where p is a polynomial function.*
- $\mathsf{Vote}'(pk', v, nc, \kappa)$ *parses* pk' *as* (pk, mc), *aborting if parsing fails; computes* $b \leftarrow \mathsf{Vote}(pk, v)$; *and outputs b.*
- $\mathsf{Tally}'(sk, \mathfrak{bb}, nc, \kappa)$ *computes* $(\mathfrak{o}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb})$, *outputting an empty vector if* $\mathfrak{o}$ *is empty; assigns the largest integer in* $\mathfrak{o}$ *to nc; initialises* $\mathfrak{v}$ *as a zero-filled vector of length nc; computes* **while** $v \in \mathfrak{o}$ **do** $\mathfrak{v}[v] \leftarrow \mathfrak{v}[v] + 1; \mathfrak{o} \leftarrow \mathfrak{o} \setminus \{v\}$; *and outputs* $(\mathfrak{v}, pf)$.

Theorem 32. *Let* Γ *be an election scheme with a trusted bulletin board. If* $\gamma(\Gamma)$ *is an election scheme satisfying* $\mathsf{Ballot\text{-}Secrecy}$, *then* Γ *satisfies* $\mathsf{IND\text{-}SEC}$.

Proof sketch. Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{BB}, \mathsf{Tally})$ and $\gamma(\Gamma) = (\mathsf{Setup}', \mathsf{Vote}', \mathsf{Tally}')$. Suppose $\gamma(\Gamma)$ is an election scheme satisfying $\mathsf{Ballot\text{-}Secrecy}$. Moreover, suppose Γ does not satisfy $\mathsf{IND\text{-}SEC}$. Hence, there exists a probabilistic polynomial-time adversary $\mathcal{A}$, such that for negligible functions negl, there exists a security parameter κ and $\mathsf{Succ}(\mathsf{IND\text{-}SEC}(\Gamma, \mathcal{A}, \kappa)) > \frac{1}{2} + \mathsf{negl}(\kappa)$. We construct the following adversary $\mathcal{B}$ against $\mathsf{Ballot\text{-}Secrecy}$ from $\mathcal{A}$, where $\mathcal{O}_\mathcal{A}$ denotes $\mathcal{A}$'s oracle and $\mathcal{O}_\mathcal{B}$ denotes $\mathcal{B}$'s oracle:

- $\mathcal{B}(pk', \kappa)$ parses pk' as (pk, mc), aborting if parsing fails, and outputs mc.
- $\mathcal{B}()$ initialises $\mathfrak{bb}$ and $\mathfrak{bb}_0$ as empty sets and L_0 and L_1 as empty multisets; computes $\mathfrak{m} \leftarrow \{1, \ldots, mc\}; x \leftarrow \mathcal{A}(pk, \mathfrak{m})$, handling $\mathcal{A}$'s oracle calls as follows, namely,

  $*$ $\mathcal{O}_\mathcal{A}(v_0, v_1)$ computes $L_0 \leftarrow L_0 \cup \{v_0\}; L_1 \leftarrow L_1 \cup \{v_1\}; b \leftarrow \mathcal{O}_\mathcal{B}(v_0, v_1); \mathfrak{bb} \leftarrow \mathsf{BB}(\mathfrak{bb}, b); b_0 \leftarrow \mathsf{Vote}(pk, v_0); \mathfrak{bb}_0 \leftarrow \mathsf{BB}(\mathfrak{bb}, b_0)$,
  $*$ $\mathcal{O}_\mathcal{A}(b)$ computes $\mathfrak{bb}' \leftarrow \mathfrak{bb}; \mathfrak{bb} \leftarrow \mathsf{BB}(\mathfrak{bb}, b);$ **if** $\mathfrak{bb} \neq \mathfrak{bb}'$ **then** $bb_0 \leftarrow \mathsf{BB}(\mathfrak{bb}_0, b)$, and
  $*$ $\mathcal{O}_\mathcal{A}()$ outputs $\mathfrak{bb}$,

  and outputs $\mathfrak{bb}$ if $L_0 = L_1$ and $\mathfrak{bb}_0$ otherwise.

- $\mathcal{B}(\mathfrak{v}, pf)$ computes $\mathfrak{o} \leftarrow \{v^{\mathfrak{v}[v]} \mid 1 \leqslant v \leqslant mc\};$**if** $L_0 \neq L_1$ **then** $pf \leftarrow \perp; g \leftarrow \mathcal{A}(\mathfrak{o}, pf)$ and outputs g.

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

Suppose (pk', sk, mb, mc) is an output of $\mathsf{Setup}'(\kappa)$. By definition of algorithm Setup', we have pk' is a pair (pk, mc), where $(pk, sk, \mathfrak{m}, \mathfrak{bb})$ is an output of $\mathsf{Setup}(\kappa)$ and $mc = |\mathfrak{m}|$. Hence, $\mathcal{B}(pk', \kappa)$ outputs mc. Let β be a bit and suppose $\mathfrak{bb}$ is an output of $\mathcal{B}()$. It is trivial to see that $\mathcal{B}()$ simulates $\mathcal{A}$'s challenger to $\mathcal{A}$. Moreover, by Lemma 31 it is straightforward to see that $\mathcal{B}$ simulates $\mathcal{A}$'s oracle too. Indeed, adversary $\mathcal{B}$ maintains bulletin board $\mathfrak{bb}$ such that $\mathcal{O}_{\mathcal{A}}(v_0, v_1)$ constructs a ballot b for v_β using $\mathcal{B}$'s oracle, hence, the ballot is constructed by algorithm Vote by way of algorithm Vote$'$, and adds that ballot to the bulletin board using algorithm BB. Suppose $(\mathfrak{v}, pf)$ is an output of $\mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$ and g is an output of $\mathcal{B}(\mathfrak{v}, pf)$. It is straightforward to see that $\mathcal{B}(\mathfrak{v}, pf)$ simulates $\mathcal{A}$'s challenger to $\mathcal{A}$, thus, $g = \beta$, with at least the probability that $\mathcal{A}$ wins IND-SEC, concluding our proof. □

Appendix I. Stronger privacy notions

Ballot secrecy does not ensure free-choice when an adversary is able to communicate with voters nor when voters deviate from the prescribed voting procedure to follow instructions provided by an adversary. Stronger notions of free-choice, such as receipt-freeness [47, 114, 126–128] and coercion resistance [129–133], are needed in the presence of such adversaries. This appendix introduces these notions, proves that our syntax cannot be used to construct (interesting) schemes satisfying them, and discusses variants of our syntax that can.

Receipt-freeness formalises a notion of free-choice in the presence of an adversary that can communicate with voters.

- Receipt-freeness. A voter cannot collaborate with a conspirator to produce information which can be used to prove how they voted.

Free-choice may be compromised in receipt-free voting systems if voters deviate from the prescribed voting procedure.[43] Coercion-resistance formalises a stronger notion of free-choice assuming that not only can voters deviate, but the adversary can instruct voters how to deviate.

- Coercion resistance. A voter can deviate from a coercer's instructions, to cast their vote, without detection.

The distinction between receipt-freeness and coercion resistance is subtle: "receipt-freeness deals with a coercer who is only concerned with deducing information about how someone voted from receipts and public information, but who does not give detailed instructions on how to cast the vote. Coercion resistance, on the other hand, includes dealing with a coercer who gives details not just on which candidate to vote for but also on how to cast the vote" [145, §1.1]. Both receipt-freeness and coercion resistance retain the assumption that voters' ballots are tallied in the prescribed manner, and receipt-freeness additionally assumes voters' ballots are constructed in the prescribed manner. Moreover, both require side conditions to exclude inevitable revelations (§3).

[43] For receipt-freeness to be an effective notion of free-choice it might be necessary for the voting system to prevent voters deviating from the prescribed voting procedure. This might be achieved by physically securing devices that can be used to cast ballots.

I.1. Receipt-freeness

We cast the definition of receipt-freeness by Delaune, Kremer & Ryan [114, 115] from the symbolic model to the computational model of cryptography, in the context of our election scheme syntax. Moreover, we extend their work to consider arbitrarily many voters, rather than just the two considered by the original definition. The resulting formalisation (Definition 39) is a pair of games.

The first game (Receipt-Freeness-A) is an extension of Ballot-Secrecy that tasks the adversary to: select a list of their preferred votes and a list of voter preferred votes (although it is somewhat unnatural for the adversary to specify voter preferred votes, this is useful to quantify over all possible voter preferences); construct a bulletin board from either (i) ballots for their preferred votes and coins used to construct those ballots, or (ii) ballots for voter preferred votes and simulated coins; and non-trivially determine whether their preferred or voter preferred votes were used, from the resulting election outcome and tallying proof. The game proceeds as per game Ballot-Secrecy, except a new oracle is used (Line 5). That oracle constructs ballots for either the adversary's preferred votes (using algorithm Vote) or for voter preferred votes (using algorithm V). Those ballots are output along with either the coins used by algorithm Vote or coins simulated by algorithm V. Intuitively, algorithm V provides a strategy for voters to cast voter preferred votes whilst convincing the adversary that its preferred votes were cast, hence, information cannot be produced to prove how voters voted.

The second game (Receipt-Freeness-B) tasks the adversary to compute inputs (including a public key) to algorithms Vote and V that cause the algorithms to output ballots that can be distinguished, thereby over-approximating the requirement that algorithm V must produce a ballot for voter preferred votes. (Indeed, the adversary can use algorithm Tally to determine whether a ballot is for the expected vote.) The game proceeds as follows: The adversary computes inputs to algorithms Vote and V (Line 1); the challenger flips a coin (Line 2) and computes a ballot using one of the algorithms (Lines 3–6), where the choice between algorithms is determined by the coin flip; and the adversary is tasked with determining the result of the coin flip from the ballot (Lines 7 & 8).

Definition 39 (Receipt-Freeness)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *be an election scheme,* V *be an algorithm,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* Receipt-Freeness-A *be the following game.*

```
Receipt-Freeness-A$(\Gamma, \mathsf{V}, \mathcal{A}, \kappa)$ =

1 $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
2 $nc \leftarrow \mathcal{A}(pk, \kappa)$;
3 $\beta \leftarrow_R \{0,1\}$;
4 $L \leftarrow \emptyset$;
5 $\mathfrak{bb} \leftarrow \mathcal{A}^{\mathcal{O}}()$;
6 $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$;
7 $g \leftarrow \mathcal{A}(\mathfrak{v}, pf)$;
8 return $g = \beta \wedge balanced(\mathfrak{bb}, nc, L) \wedge 1 \leqslant nc \leqslant mc \wedge |\mathfrak{bb}| \leqslant mb$;
```

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

- $\mathcal{O}(v_0, v_1)$ *chooses coins r uniformly at random from the coin space of algorithm* Vote*, computes* **if** $\beta = 0$ **then** $b \leftarrow \mathsf{Vote}(pk, v_0, nc, \kappa; r)$ **else** $(b, r) \leftarrow \mathsf{V}(pk, v_1, v_0, nc, \kappa)$ *and* $L \leftarrow L \cup \{(b, v_0, v_1)\}$*, and outputs* (b, r)*, where* $v_0, v_1 \in \{1, ..., nc\}$.

Moreover, let $\mathcal{B}$ be an adversary and Receipt-Freeness-B *be the following game.*

```
Receipt-Freeness-B$(\Gamma, \mathsf{V}, \mathcal{B}, \kappa)$ =
$(pk, v_0, v_1, nc) \leftarrow \mathcal{B}(\kappa)$;
$\beta \leftarrow_R \{0,1\}$;
if $\beta = 0$ then
    $b \leftarrow \mathsf{Vote}(pk, v_0, nc, \kappa)$
else
    $(b, r) \leftarrow \mathsf{V}(pk, v_0, v_1, nc, \kappa)$
$g \leftarrow \mathcal{B}(b)$;
return $g = \beta \wedge 1 \leqslant v_0, v_1 \leqslant nc$;
```

We say Γ satisfies Receipt-Freeness*, if there exists a probabilistic polynomial-time algorithm* V *such that for all probabilistic polynomial-time adversaries $\mathcal{A}$ and $\mathcal{B}$, there exists a negligible function* negl *and for all security parameters κ, we have* $\mathsf{Succ}(\mathsf{Receipt\text{-}Freeness\text{-}A}(\Gamma, \mathsf{V}, \mathcal{A}, \kappa)) \leqslant \frac{1}{2} + \mathsf{negl}(\kappa)$ *and* $\mathsf{Succ}(\mathsf{Receipt\text{-}Freeness\text{-}B}(\Gamma, \mathsf{V}, \mathcal{B}, \kappa)) \leqslant \frac{1}{2} + \mathsf{negl}(\kappa)$.

An election scheme satisfies receipt-freeness when ballot secrecy is preserved even when coins used to construct ballots are revealed.

Similarly to ballot secrecy (§3), receipt-freeness tolerates inevitable revelations, e.g., unanimous election outcomes. It follows that an election scheme for one candidate satisfies our definition of receipt-freeness, because that scheme will always produce unanimous election outcomes.

Proposition 33. *Given an election scheme Γ for one candidate (i.e., for all security parameters the maximum number of candidates is one), we have Γ satisfies* Ballot-Secrecy *and* Receipt-Freeness.

Proof. Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$. Suppose (pk, sk, mb, mc) is an output of $\mathsf{Setup}(\kappa)$ and nc is an output of $\mathcal{A}(pk, \kappa)$ such that $1 \leqslant nc \leqslant mc$, for some security parameter κ and some adversary $\mathcal{A}$ against game Ballot-Secrecy or game Receipt-Freeness-A. Since $mc = 1$ by hypothesis, we have $nc = 1$ too. For game Receipt-Freeness-A, let algorithm V be such that $\mathsf{V}(pk, v_0, v_1, nc, \kappa)$ chooses coins r uniformly at random from the coin space of algorithm Vote, computes $b \leftarrow \mathsf{Vote}(pk, v_0, nc, \kappa)$, and outputs (b, r). Suppose β is a bit chosen uniformly at random and $\mathfrak{bb}$ is an output of $\mathcal{A}()$. By inspection of the oracle definition in each game, we have for every oracle call $\mathcal{O}(v_0, v_1)$ that $v_0 = v_1$, moreover, the ballot output by the oracle is independent of bit β. It follows that the adversary looses each game, hence Γ satisfies Ballot-Secrecy. To show that Receipt-Freeness is satisfied too, we must consider game Receipt-Freeness-B.

Suppose (pk, v_0, v_1, nc) is an output of $\mathcal{B}(\kappa)$ such that $1 \leqslant v_0, v_1 \leqslant nc$, for some adversary $\mathcal{B}$ against game Receipt-Freeness-B. Let β be a bit chosen uniformly at random. If $\beta = 0$, then suppose b is an output of $\mathsf{Vote}(pk, v_0, nc, \kappa)$, otherwise ($\beta = 1$), suppose (b, r) is an output of $\mathsf{V}(pk, v_0, v_1, nc, \kappa)$, hence, b is an output of $\mathsf{Vote}(pk, v_0, nc, \kappa)$ by definition of V. Thus, the ballot computed by the challenger is independent of bit β. It follows that the adversary looses game Receipt-Freeness-B, hence, Γ satisfies Receipt-Freeness, thereby concluding our proof. □

Beyond the uninteresting case (Proposition 33), we prove that Receipt-Freeness cannot be satisfied by election schemes for more than one candidate (Proposition 34), assuming the election scheme is perfectly correct or satisfies tally soundness.

Proposition 34. *Let Γ be an election scheme for more than one candidate (i.e., there exists a security parameter such that the maximum number of candidates is greater than one). Suppose Γ is perfectly correct or Γ satisfies* $\mathsf{Tally\text{-}Soundness}$. *We have Γ does not satisfy* $\mathsf{Receipt\text{-}Freeness}$.

Proof. Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$. Suppose to the contrary that Γ satisfies $\mathsf{Receipt\text{-}Freeness}$, hence, there exists a probabilistic polynomial-time algorithm V such that for all probabilistic polynomial-time adversaries $\mathcal{A}$ and $\mathcal{B}$, there exists a negligible function negl and for all security parameters κ, we have $\mathsf{Succ}(\mathsf{Receipt\text{-}Freeness\text{-}A}(\Gamma, \mathsf{V}, \mathcal{A}, \kappa)) + \mathsf{Succ}(\mathsf{Receipt\text{-}Freeness\text{-}B}(\Gamma, \mathsf{V}, \mathcal{B}, \kappa)) \leqslant \frac{1}{2} + \mathsf{negl}(\kappa)$. Further suppose κ is such that the maximum number of candidates output by algorithm $\mathsf{Setup}(\kappa)$ is greater than one. Moreover, suppose $\mathcal{A}$ is the following adversary.

- $\mathcal{A}(pk, \kappa)$ computes $nc \leftarrow 2$ and outputs nc.
- $\mathcal{A}()$ computes $v_0 \leftarrow 1; v_1 \leftarrow 2; (b, r) \leftarrow \mathcal{O}(v_0, v_1)$ and outputs $\emptyset$.
- $\mathcal{A}(\mathfrak{v}, pf)$ outputs 0 if $v = \mathsf{Vote}(pk, v_0, nc, \kappa; r)$ and 1 otherwise.

Since Γ satisfies $\mathsf{Receipt\text{-}Freeness}$, it follows that $\mathsf{V}(pk, v_1, v_0, nc, \kappa)$ outputs (b, r) such that $b = \mathsf{Vote}(pk, v_0, nc, \kappa; r)$, with overwhelming probability. Suppose $\mathcal{B}$ is the following adversary.

- $\mathcal{B}(\kappa)$ computes $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa); v_0 \leftarrow 1; v_1 \leftarrow 2; nc \leftarrow 2$ and outputs (pk, v_0, v_1, nc).
- $\mathcal{B}(b)$ computes $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \{b\}, nc, \kappa)$ and outputs 0 if $\mathfrak{v} = (1, 0)$ and 1 otherwise.

We prove that $\mathcal{B}$ wins $\mathsf{Receipt\text{-}Freeness\text{-}B}(\Gamma, \mathsf{V}, \mathcal{B}, \kappa)$.

Suppose (pk, v_0, v_1, nc) is an output of $\mathcal{B}(\kappa)$ and β is a bit chosen uniformly at random. If $\beta = 0$, then further suppose b is an output of $\mathsf{Vote}(pk, v_0, nc, \kappa)$, and g is an output of $\mathcal{B}(b)$. Outputs $(\mathfrak{v}, pf)$ of $\mathsf{Tally}(sk, \{b\}, nc, \kappa)$ are such that $\mathfrak{v} = (1, 0)$ by correctness, which ensures $g = \beta$ by definition of $\mathcal{B}$. Otherwise ($\beta = 1$), suppose (b, r) is an output of $\mathsf{V}(pk, v_0, v_1, nc, \kappa)$, hence, $b = \mathsf{Vote}(pk, v_1, nc, \kappa; r)$, and g is an output of $\mathcal{B}(b)$. Outputs $(\mathfrak{v}, pf)$ of $\mathsf{Tally}(sk, \{b\}, nc, \kappa)$ are such that $\mathfrak{v} = (0, 1)$ by perfect correctness and $\mathfrak{v} = \mathit{correct\text{-}outcome}(pk, nc, \{b\}, \kappa) = (0, 1)$ by $\mathsf{Tally\text{-}Soundness}$, which ensures $g = \beta$ by definition of $\mathcal{B}$. (Correctness, rather than perfect correctness, is insufficient, because algorithm V may not have chosen coins r uniformly at random.) Thus, $\mathsf{Succ}(\mathsf{Receipt\text{-}Freeness\text{-}A}(\Gamma, \mathsf{V}, \mathcal{A}, \kappa)) + \mathsf{Succ}(\mathsf{Receipt\text{-}Freeness\text{-}B}(\Gamma, \mathsf{V}, \mathcal{B}, \kappa)) \not\leqslant \frac{1}{2} + \mathsf{negl}(\kappa)$, concluding our proof. □

A special case of our proposition holds for universal verifiability, rather than tally soundness, because universal verifiability is strictly stronger (Appendix E).

It follows from Proposition 34 that our syntax cannot be used to construct (interesting) election schemes satisfying stronger notions of privacy such as $\mathsf{Receipt\text{-}Freeness}$. Nonetheless, algorithm Vote could be distributed between the voter and some other (possibly untrusted) parties to enable stronger privacy notions. Indeed, a variant of Helios that delegates ballot construction to a trusted party would satisfy receipt-freeness, since voters cannot access coins used by the trusted party to construct ballots, hence, cannot communicate such coins to the adversary. More generally, an election scheme satisfying ballot secrecy can delegate ballot construction to achieve receipt-freeness.

We have seen that election schemes for more than one candidate cannot satisfy $\mathsf{Receipt\text{-}Freeness}$ (Proposition 34), assuming the election scheme is perfectly correct or satisfies tally soundness. Intuitively, it follows that coercion resistance cannot be satisfied by such election schemes either, because coercion resistance strengthens receipt-freeness. Nevertheless, receipt-freeness can be satisfied by election schemes with interactive voting algorithms and we now consider whether coercion resistance can hold too.

I.2. Coercion resistance

Coercion resistance requires a mechanism to deviate from a coercer's instructions. Intuitively, no such mechanism exists for election schemes with interactive voting algorithms, because there exists instructions for which deviations are impossible. Indeed, the coercer can instruct the voter to run the algorithms themselves (i.e., without the aid of another party) and furnish the coercer with the coins used, thereby enabling the coercer to check whether voters followed their instructions. It follows that election schemes with interactive voting algorithms cannot satisfy coercion resistance.

Interactive voting algorithms can be extended with private inputs which cannot be simulated by voters, thereby enabling voters to deviate from the coercer's instructions. Alternatively, the variant of our syntax with voter credentials [30, Definition 6] can be used; that variant extends algorithm Vote to input a private credential (essentially resulting in interactive ballot construction). The latter has been used by Smyth, Frink & Clarkson [30, §6] to model the coercion-resistant voting system by Juels, Catalano & Jakobsson [43, 129] and by Smyth as a foundation for his coercion-resistant voting system [41], thus, the syntax with voter credentials is compatible with stronger privacy notions.