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

Ben Smyth

Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg

January 17, 2018

**Abstract**

We propose a definition of ballot secrecy as an indistinguishability game in the computational model of cryptography. Our definition builds upon and strengthens earlier definitions to ensure ballot secrecy is preserved in the presence of an adversary that controls ballot collection. We also propose definitions of ballot independence as adaptations of non-malleability and indistinguishability games 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 voting 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 could not have been detected by earlier definitions, 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, provable security, secrecy, voting.

## 1 Introduction

An election is a decision-making procedure to choose representatives [LG84, Saa95, Gum05, AH10]. Choices should be made by voters with equal influence, and this must be ensured by voting systems, as prescribed by the United Nations [UN48], the Organisation for Security & Cooperation in Europe [OSC90], and the Organization of American States [OAS69]. 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" [Lep08]. 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 [UN48, OSC90, OAS69]. 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" [Mil30].

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" [BBP07]. 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 and collection 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, anyone can check that ballots are only distributed to voters and at most one ballot is deposited by each voter, furthermore, anyone 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 and the ability to transfer that assurance is limited to the observer's "good word or sworn testimony" [NA03].

Many electronic voting systems – including systems that have been used in large-scale, binding elections – rely on art, rather than science, to ensure that votes are freely made, with equal influence. Such systems build upon creativity and skill, rather than scientific foundations. These systems are typically broken in ways that violate free-choice, e.g., [KSRW04, GH07, Bow07, WWH$^+$10, WWIH12, SFD$^+$14], or permit non-voters to unduly influence the selection of representatives, e.g., [KSRW04, UK07, Bow07, Ger09, JS12]. Breaks can be avoided by carefully formulating rigorous and precise security definitions that capture notions of voters voting freely with equal influence, and proving that systems satisfy these definitions. *Universal verifiability* formalises a notion of checking whether voters voted with equal influence, and we focus on a definition that captures voters voting freely. Our definition is presented in the

[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 [Bre06].

computational model of cryptography as a game, 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, assuming ballots are constructed and tallied in the prescribed manner.[2]

- 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 $v_\beta$, in the manner prescribed by the voting system, where $\beta$ 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 $\beta$. 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 distribution of 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 distribution of votes cast (which the adversary controls) remains constant regardless of whether $\beta = 0$ or $\beta = 1$. If the adversary wins, then a voter's vote can be revealed, otherwise, it cannot, i.e., the voting system provides ballot secrecy. Our game builds upon and strengthens games by Bernhard *et al.* [BCP$^+$11,BPW12b,SB13,SB14,BCG$^+$15b] 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.

We introduce two voting systems to demonstrate how ballot secrecy and universal verifiability can be achieved. The first (Nonce) instructs each voter to pair their vote with a nonce and instructs the tallier to publish the distribution of votes. The second (Enc2Vote) instructs each voter to encrypt their vote using an asymmetric encryption scheme and instructs the tallier to decrypt the encrypted votes and publish the distribution of votes. Universal verifiability is ensured by the former voting system. But, ballot secrecy is not, because all 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. More advanced voting systems must satisfy both secrecy and verifiability, and we will consider the Helios voting system.

[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.

Helios is an open-source, web-based electronic voting system,[3] 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 [BVQ10, HBH10],[4] the ACM used Helios for their 2014 general election [Sta14], the Catholic University of Louvain used Helios to elect their university president in 2009 [AMPQ09], and Princeton University has used Helios since 2009 to elect student governments.[5,6] Helios is intended to satisfy universal verifiability whilst maintaining ballot secrecy. For ballot secrecy, each voter is instructed to encrypt their vote using a homomorphic encryption scheme. Encrypted votes are homomorphically combined and the homomorphic combination is decrypted to reveal the outcome [AMPQ09]. Alternatively, a mixnet is applied to the encrypted votes and the mixed encrypted votes are decrypted to reveal the outcome [Adi08, BGP11]. We refer to the former voting system as *Helios* and the latter as *Helios Mixnet*. For universal verifiability, the encryption step is accompanied by a non-interactive zero-knowledge proof demonstrating correct computation. This prevents an adversarial voter encrypting a message that could be combined with legitimate ballots 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.** 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. Section 4 shows how our definition of ballot secrecy can be used to identify a known vulnerability in Helios; discovers that its patched successors do not defend against that vulnerability in the presence of an adversary that controls ballot collection; explains why earlier definitions of ballot secrecy by Bernhard *et al.* could not detect that vulnerability; identifies a new exploit that enables an adversary to determine if a voter did not vote for the adversary's preferred candidate; discusses non-malleable ballots as a fix; and uses our sufficient condition to prove that secrecy is satisfied when the fix is applied. Section 5 proves 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-

[3] `https://vote.heliosvoting.org`, accessed 21 Sep 2017.

[4] `https://www.iacr.org/elections/`, accessed 21 Sep 2017.

[5] `http://heliosvoting.wordpress.com/2009/10/13/helios-deployed-at-princeton/`, accessed 21 Sep 2017.

[6] `https://princeton.heliosvoting.org/`, accessed 21 Sep 2017.

verifiable, encryption-based voting systems. 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), related work (§7), and a brief conclusion (§8); Figure 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 Figure 1, and some readers might like to study the related work before our definition of ballot secrecy.)

# 2 Election scheme syntax

We recall *election scheme* syntax (Definition 1) from Smyth, Frink & Clarkson [SFC17]. Election schemes capture 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 recorded on a bulletin board. Finally the tallier tallies the recorded ballots and announces an outcome, i.e., a distribution of votes.[7]

**Definition 1** (Election scheme [SFC17]). *An* election scheme *is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *such that:*[8]

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

$\mathsf{Vote}$*, 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* $\kappa$*. Vote* $v$ *should be selected from a sequence* $1, \ldots, nc$ *of candidates. The algorithm outputs a ballot* $b$ *or error symbol* $\perp$*.*

$\mathsf{Tally}$*, 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* $\kappa$*, where* $\mathfrak{bb}$ *is a set. And outputs an election outcome* $\mathfrak{v}$ *and a non-interactive tallying proof* $pf$ *demonstrating that the outcome corresponds to votes expressed in ballots on the bulletin board. The election outcome* $\mathfrak{v}$ *should be a vector of length* $nc$ *such that* $\mathfrak{v}[v]$ *indicates the number of votes for candidate* $v$*.*

[7] Smyth, Frink & Clarkson use the syntax to model first-past-the-post voting systems [SFC17] and Smyth shows ranked-choice voting systems can be modelled too [Smy17]. Both works consider a single tallier and we discuss distributing the tallier's role in Section 7.

[8] The syntax bounds the number of ballots $mb$, respectively candidates $mc$, to broaden the correctness definition's scope (indeed, Helios requires $mb$ and $mc$ to be less than or equal to the size of the underlying encryption scheme's message space); represents votes as integers, rather than alphanumeric strings, for brevity; and omits algorithm Verify, because we focus on ballot secrecy, not verifiability.

**Figure 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., what the adversary should not be able to achieve. 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 random 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. 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 $S$. Hence, we can formulate a game $\mathsf{Exp}(H, S, \mathcal{A})$ that 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 adversaryin a game. In particular, adversaries can access earlier assignments. For instance, the adversary's second instantiation in game $\mathsf{Exp}$ 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, $\mathsf{Succ}(\mathsf{Exp}(\cdot)) = \Pr[x \leftarrow \mathsf{Exp}(\cdot) : x = \top]$. We generally require that the adversary's success is negligible for reachability tasks and negligibly better than guessing for indistinguishability tasks.

Game $\mathsf{Exp}$ captures a single interaction between the challenger and adversary. We extend games with oracles to capture arbitrarily many interactions. Hence, we can formulate a strengthening of $\mathsf{Exp}$ 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 $\beta$.

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 $x \leftarrow_R S; x' \leftarrow_R S$.

*Election schemes must satisfy* correctness*: there exists a negligible function* $\mathsf{negl}$*, such that for all security parameters* $\kappa$*, 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 $1 \leq i \leq nb$ do
        $b_i \leftarrow \mathsf{Vote}(pk, v_i, nc, \kappa);$
        $\mathfrak{v}[v_i] \leftarrow \mathfrak{v}[v_i] + 1;$
    $(\mathfrak{v}', pf) \leftarrow \mathsf{Tally}(sk, \{b_1, \ldots, b_{nb}\}, nc, \kappa) :$
    $nb \leq mb \wedge nc \leq mc \Rightarrow \mathfrak{v} = \mathfrak{v}'] > 1 - \mathsf{negl}(\kappa).$
```

The syntax provides a language to express voting systems and their properties.

# 3 Privacy

Some scenarios inevitably reveal voters' votes: Unanimous election outcomes reveal how everyone voted and, more generally, election outcomes can be coupled with partial knowledge on the distribution 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.[9] 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 on the distribution of 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 $v_\beta$ and a second ballot $b_2$ for $v_{1-\beta}$, where $\beta$ is a bit chosen uniformly at random. Those ballots are given to the adversary. Thirdly, the adversary constructs ballots $b_3, \ldots, b_n$. Fourthly, the challenger tallies all the ballots (i.e., $b_1, \ldots, b_n$) to the determine the election outcome, which the adversary is given. Finally, the adversary is tasked with determining bit $\beta$. This game challenges the adversary to determine if the first ballot is for $v_0$ and the second is for $v_1$, or

[9] Voting systems that announce chosen representatives (e.g., [BY86, HK02, HK04, DK05]), rather than distributions of votes, could offer stronger notions of privacy.

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 [CS13, CS11]. 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 [CS11, SC11, Smy12, CS13, SB13, BCG$^+$15b, BCG$^+$15a]. Variants of Helios with ballot weeding seem secure with respect to this game. Unfortunately, ballot weeding mechanisms can be subverted by intercepting ballots or by re-ordering ballots. For instance, Smyth, Frink & Clarkson show how re-ordering ballots can subvert weeding mechanisms in a manner that violates universal verifiability [SFC17], and we will see that ballot secrecy can be violated too (§4.3). Given that current definitions cannot detect such vulnerabilities (§7), 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.[10] 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 as the indistinguishability game described in Section 1.

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

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

*Predicate* $balanced(\mathfrak{bb}, nc, L)$ *holds when: for all votes* $v \in \{1, \ldots, nc\}$ *we have* $|\{b \mid b \in \mathfrak{bb} \wedge \exists v_1 \,.\, (b, v, v_1) \in L\}| = |\{b \mid b \in \mathfrak{bb} \wedge \exists v_0 \,.\, (b, v_0, v) \in L\}|$*. And oracle* $\mathcal{O}$ *is defined as follows:*

[10] The recorded-as-cast notion was introduced by Adida & Neff [AN06, §2].

- $\mathcal{O}(v_0, v_1)$ *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* $\Gamma$ *satisfies* Ballot-Secrecy*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such that for all security parameters* $\kappa$*, we have* $\mathsf{Succ}(\mathsf{Ballot\text{-}Secrecy}(\Gamma, \mathcal{A}, \kappa)) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$.

Game Ballot-Secrecy captures a setting in which the tallier generates a key pair using the scheme's Setup algorithm, publishes the public key, and only uses the private key to compute the election outcome and tallying proof.

In our game, the adversary has access to a left-right oracle which can construct ballots on the adversary's behalf.[11] The oracle constructs ballots in two ways, corresponding to a bit $\beta$ chosen uniformly at random by the challenger. If $\beta = 0$, then, given a pair of votes $v_0, v_1$, the oracle constructs a ballot for $v_0$ and outputs the ballot to the adversary. Otherwise ($\beta = 1$), the oracle outputs a ballot for $v_1$. The adversary computes a bulletin board, which may include ballots constructed by the oracle. Thus, the game captures a setting where the bulletin board is computed by an adversary that casts ballots on behalf of some voters and controls the distribution of votes cast by the remaining voters. The challenger tallies the adversary's bulletin board to derive an election outcome and tallying proof. The adversary is given the outcome and proof, and wins by determining whether $\beta = 0$ or $\beta = 1$. Intuitively, if the adversary wins, then there exists a strategy to distinguish ballots, otherwise, the adversary is unable to distinguish between a ballot for vote $v_0$ and a ballot for vote $v_1$, therefore, voters' votes cannot be revealed.

Our notion of ballot secrecy considers election schemes which reveal the number of votes for each candidate (i.e., the election outcome). Hence, to avoid trivial distinctions in our ballot secrecy game, we require that the game is *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 $(v_{1,0}, v_{1,1}), \dots, (v_{n,0}, v_{n,1})$ and the corresponding outputs are $b_1, \dots, b_n$, further suppose the bulletin board is $\{b_1, \dots, b_\ell\}$ such that $\ell \leq n$. That game is balanced if the "left" inputs $v_{1,0}, \dots, v_{\ell,0}$ are a permutation of the "right" inputs $v_{1,1}, \dots, v_{\ell,1}$. The balanced condition prevents trivial distinctions.[12] 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 bulletin board.

Proving ballot secrecy is time consuming. Indeed, Quaglia & Smyth's ballot-secrecy proof for our simple Enc2Vote scheme consumes over six and a half

[11] Bellare *et al.* introduced left-right oracles in the context of symmetric encryption [BDJR97] and Bellare & Rogaway provide a tutorial on their use [BR05].

[12] 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."

pages [QS17a, Appendix C.6]. Thus, sufficient conditions for ballot secrecy should be sought, and we will see that ballot independence suffices.

## 3.2 Ballot independence

Ballot independence [Gen95, CS13, CGMA85] 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.

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 using non-malleability: We formalise non-malleability for election schemes as a straightforward adaptation of the non-malleability definition for asymmetric encryption by Bellare & Sahai [BS99].[13] Such a formalisation captures an intuitive notion of ballot independence, but the definition 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 game [BS99]. In a similar direction, we derive a simpler, equivalent definition of ballot independence as a straightforward adaptation of that indistinguishability game.

### 3.2.1 Non-malleability game

We formalise ballot independence as a non-malleability game, called comparison based non-malleability under chosen vote attack (CNM-CVA).

**Definition 3** (CNM-CVA)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* $\kappa$ *be a security parameter, and* $\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{A}, \kappa)$ *and* $\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{A}, \kappa)$ *be the following games.*

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

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

[13] Non-malleability was first formalised by Dolev, Dwork & Naor in the context of asymmetric encryption [DDN91, DDN00]; the definition by Bellare & Sahai builds upon their results and results by Bellare *et al.* [BDPR98].

*In the above games, we require that relation $R$ is computable in polynomial time. We say $\Gamma$ 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 $\kappa$, we have* $\mathsf{Succ}(\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{A}, \kappa)) - \mathsf{Succ}(\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{A}, \kappa)) \leq \mathsf{negl}(\kappa)$.

Similarly to game Ballot-Secrecy, games cnm-cva and cnm-cva-$ capture: key generation using algorithm Setup, publication of the public key, and only using the private key to compute the election outcome and tallying proof.

CNM-CVA is satisfied if no adversary can distinguish between games cnm-cva and cnm-cva-$. That is, for all adversaries, the adversary wins cnm-cva iff the adversary loses cnm-cva-$, 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 loses 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 [CS13, CS11] 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.** The main distinction between non-malleability for asymmetric encryption (CNM-CPA) and non-malleability for election schemes (CNM-CVA) is as follows: CNM-CPA performs a parallel decryption, whereas CNM-CVA performs a single tally. Hence, non-malleability for encryption reveals plaintexts corresponding to ciphertexts, whereas non-malleability for elections reveals the number of votes for each candidate.

### 3.2.2 Indistinguishability game

We formalise an alternative definition of ballot independence as an indistinguishability game, called indistinguishability under chosen vote attack (IND-CVA).

**Definition 4** (IND-CVA). *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* $\kappa$ *be the security parameter, and* $\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

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

*We say* $\Gamma$ *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* $\kappa$*, we have* $\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{A}, \kappa) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$.

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.** Unsurprisingly, the distinction between indistinguishability for asymmetric encryption (IND-PA0) and indistinguishability for election schemes (IND-CVA) is similar to the distinction between non-malleability for asymmetric encryption and non-malleability for election schemes (§3.2.1), namely, IND-PA0 performs a parallel decryption, whereas IND-CVA performs a single tally.

### 3.2.3 Equivalence between games

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 [BS99] 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 1** ($\mathsf{CNM\text{-}CVA} = \mathsf{IND\text{-}CVA}$)**.** *Given an election scheme $\Gamma$, we have $\Gamma$ satisfies* $\mathsf{CNM\text{-}CVA}$ *iff $\Gamma$ satisfies* $\mathsf{IND\text{-}CVA}$*.*

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

## 3.3 Secrecy and independence coincide

The main distinctions between our ballot secrecy ($\mathsf{Ballot\text{-}Secrecy}$) and ballot independence ($\mathsf{IND\text{-}CVA}$) games are as follows.

1. The challenger produces one challenge ballot for the adversary in our ballot independence game, whereas the left-right oracle produces arbitrarily many challenge ballots for the adversary in our ballot secrecy game.

2. The adversary in our ballot secrecy game has access to a tallying proof, but the adversary in our ballot independence game does not.

3. The winning condition in our ballot secrecy game requires the bulletin board to be balanced, whereas the bulletin board must not contain the challenge ballot in our ballot independence game.

The second point distinguishes our games and shows ballot secrecy is at least as strong as ballot independence. Hence, non-malleable ballots are necessary in election schemes satisfying ballot secrecy.

**Theorem 2** ($\mathsf{Ballot\text{-}Secrecy} \Rightarrow \mathsf{IND\text{-}CVA}$)**.** *Given an election scheme $\Gamma$ satisfying* $\mathsf{Ballot\text{-}Secrecy}$*, we have $\Gamma$ satisfies* $\mathsf{IND\text{-}CVA}$*.*

Tallying proofs may reveal voters' votes. For example, a variant of $\mathsf{Enc2Vote}$ might define tallying proofs that map ballots to votes. Since proofs are available to the adversary in our ballot secrecy game, but not in our ballot independence game, it follows that ballot secrecy is strictly stronger than ballot independence.

**Proposition 3** ($\mathsf{IND\text{-}CVA} \not\Rightarrow \mathsf{Ballot\text{-}Secrecy}$)**.** *There exists an election scheme $\Gamma$ such that $\Gamma$ satisfies* $\mathsf{IND\text{-}CVA}$*, but not* $\mathsf{Ballot\text{-}Secrecy}$*.*

Proposition 3 follows from our informal reasoning and we omit a formal proof.

Secrecy game $\mathsf{Ballot\text{-}Secrecy}$ is generally stronger than independence game $\mathsf{IND\text{-}CVA}$. Nonetheless, we show that our definitions of ballot independence and ballot secrecy coincide for election schemes without tallying proofs (Definition 5), assuming a soundness condition (Definition 6), which asserts that adding a ballot for vote $v$ to a bulletin board (computed by an adversary), effects the election outcome by exactly vote $v$, hence, honestly constructed ballots are tallied correctly.

**Definition 5.** *An election scheme $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ is* without tallying proofs*, if there exists a constant symbol $\epsilon$ such that for all multisets $\mathfrak{bb}$ we have:* $\Pr[(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa); (\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa) : pf = \epsilon] = 1$.

**Definition 6** (HB-Tally-Soundness)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* $\kappa$ *be a security parameter, and* HB-Tally-Soundness$(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

```
HB-Tally-Soundness$(\Gamma, \mathcal{A}, \kappa)$ =
    $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
    $(v, nc, \mathfrak{bb}_0) \leftarrow \mathcal{A}(pk, \kappa)$;
    $b \leftarrow \mathsf{Vote}(pk, v, nc, \kappa)$;
    $(\mathfrak{v}_0, pf_0) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}_0, nc, \kappa)$;
    $(\mathfrak{v}_1, pf_1) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}_0 \cup \{b\}, nc, \kappa)$;
    $\mathfrak{v}^* \leftarrow (\mathfrak{v}_0[1], \dots, \mathfrak{v}_0[v-1], \mathfrak{v}_0[v]+1, \mathfrak{v}_0[v+1], \dots, \mathfrak{v}_0[|\mathfrak{v}_0|])$;
    return $\mathfrak{v}^* \neq \mathfrak{v}_1 \wedge b \notin \mathfrak{bb}_0 \wedge 1 \leq v \leq nc \leq mc \wedge |\mathfrak{bb}_0 \cup \{b\}| \leq mb$;
```

*We say* $\Gamma$ *satisfies* honest-ballot tally soundness (HB-Tally-Soundness)*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such that for all security parameters* $\kappa$*, we have* $\mathsf{Succ}(\text{HB-Tally-Soundness}(\Gamma, \mathcal{A}, \kappa)) \leq \mathsf{negl}(\kappa)$*.*

**Proposition 4** (Ballot-Secrecy = IND-CVA, without proofs)**.** *Let* $\Gamma$ *be an election scheme without tallying proofs. Suppose* $\Gamma$ *satisfies* HB-Tally-Soundness*. We have* $\Gamma$ *satisfies* Ballot-Secrecy *iff* $\Gamma$ *satisfies* IND-CVA*.*

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

**Definition 7** (Zero-knowledge tallying proofs)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *be an election scheme. We say* $\Gamma$ *has* zero-knowledge tallying proofs*, if there exists a non-interactive zero-knowledge proof system* (Prove, Verify)*, such that for all security parameters* $\kappa$*, 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* $pf = \mathsf{Prove}((pk, \mathfrak{bb}, nc, \mathfrak{v}), sk, \kappa; r)$*, such that coins* $r$ *are chosen uniformly at random by* Tally*.*

**Theorem 5** (Ballot-Secrecy = IND-CVA, with ZK proofs)**.** *Let* $\Gamma$ *be an election scheme with zero-knowledge tallying proofs. Suppose* $\Gamma$ *satisfies* HB-Tally-Soundness*. We have* $\Gamma$ *satisfies* Ballot-Secrecy *iff* $\Gamma$ *satisfies* IND-CVA*.*

Honest-ballot tally soundness is implied by universal verifiability (Lemmata 10 & 29). Thus, a special case of Theorem 5 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:

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 in zero-knowledge that the ciphertext is correctly constructed and that the vote is selected from the sequence of candidates, and outputs the ciphertext coupled with the proof.

Tally selects ballots from a 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*,[14] the current release is *Helios 3.1.4*,[15] and a new release is planned.[16] Henceforth, we'll refer to the planned release as *Helios'12*.

## 4.1 Helios 2.0

Cortier & Smyth show that Helios 2.0 does not satisfy ballot secrecy (§3). Thus, we would not expect Ballot-Secrecy to hold. Indeed, we adopt a formal description of Helios 2.0 by Smyth, Frink & Clarkson [SFC17] (Appendix C) and use that description to prove that secrecy does not hold.

**Theorem 6.** *Helios 2.0 does not satisfy* Ballot-Secrecy*.*

*Proof sketch.* Suppose an adversary queries the left-right oracle with inputs $v_0$ and $v_1$ to derive a ballot for $v_\beta$, where bit $\beta$ is chosen by the challenger. Further suppose the adversary abuses malleability to derive a related ballot $b$ for $v_\beta$ 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 $\mathfrak{v}[v_0] = 1$, then output 0, otherwise, output 1. Since $b$ is a ballot for $v_\beta$, it follows by correctness that $\mathfrak{v}[v_0] = 1$ iff $\beta = 0$, and $\mathfrak{v}[v_1] = 1$ iff $\beta = 1$, hence, the adversary wins the game. □

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. 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 $\mathfrak{v}[v_0] = 2$, then output 0, otherwise, output 1. Hence, the adversary wins the game. Moreover, we will see that the vulnerability is not eliminated by ballot weeding (§4.3).

[14] `https://github.com/benadida/helios/releases/tag/2.0`, released 25 Jul 2009, accessed 21 Sep 2017.

[15] `https://github.com/benadida/helios-server/releases/tag/v3.1.4`, released 10 Mar 2011, last patched 27 Oct 2017, accessed 17 Jan 2018.

[16] `http://documentation.heliosvoting.org/verification-specs/helios-v4`, published c. 2012, accessed 21 Sep 2017.

Cortier & Smyth attribute the vulnerability to tallying meaningfully related ballots. Indeed, Helios 2.0 uses malleable ballots: A vote $v$ selected from candidates $1, \dots, nc$ is enciphered to a tuple of ciphertexts $c_1, \dots, c_{nc-1}$ such that if $v < nc$, then ciphertext $c_v$ contains plaintext 1 and the remaining ciphertexts contain plaintext 0, otherwise, all ciphertexts contain plaintext 0. Moreover, correct ciphertext construction is shown using proofs $\sigma_1, \dots, \sigma_{nc}$ such that proof $\sigma_j$ demonstrates ciphertext $c_j$ contains 0 or 1, where $1 \leq j \leq nc - 1$, and proof $\sigma_{nc}$ demonstrates that the homomorphic combination of ciphertexts $c_1 \otimes \cdots \otimes c_{nc-1}$ contains 0 or 1. Hence, given a ballot $c_1, \dots, c_{nc-1}, \sigma_1, \dots, \sigma_{nc}$, we have $c_{\chi(1)}, \dots, c_{\chi(nc-1)}, \sigma_{\chi(1)}, \dots, \sigma_{\chi(nc-1)}, \sigma_{nc}$ is a ballot for all permutations $\chi$ on $\{1, \dots, nc - 1\}$. Thus, ballots are malleable, which is incompatible with ballot secrecy (§3.3).

## 4.2 Helios 3.1.4

Chang-Fong & Essex [CE16] showed that Helios 3.1.4 did not satisfy ballot secrecy (prior to patching). They attributed the vulnerability to not checking the suitability of cryptographic parameters nor checking that ballots are constructed from such parameters. The vulnerability was mitigated against by patching Helios 3.1.4 to perform the necessary checks.[17] Nonetheless, ballots remain malleable, hence, Helios 3.1.4 does not satisfy ballot secrecy, and we prove that Ballot-Secrecy is not satisfied, using a formal description of Helios 3.1.4 that we derive from our description of Helios 2.0 (Appendix C).

**Corollary 7.** *Helios 3.1.4 does not satisfy* Ballot-Secrecy*.*

A proof of Corollary 7 follows from Theorem 6, because Helios 3.1.4 does not address issues arising from related ballots.

## 4.3 Helios'12

Bernhard, Pereira & Warinschi [BPW12a] show that Helios 3.1.4 does not satisfy universal verifiability.[18] They attribute vulnerabilities to application of the Fiat–Shamir transformation without inclusion of statements in hashes (i.e., weak Fiat–Shamir), and including statements in hashes (i.e., applying the Fiat–Shamir transformation) is postulated as a defence. Helios'12 is intended to mitigate against vulnerabilities. In particular, the specification incorporates the Fiat–Shamir transformation (rather than weak Fiat–Shamir), and there are plans to incorporate ballot weeding.[19]

Bernhard, Pereira & Warinschi [BPW12a], Bernhard [Ber14, §6.11] and Bernhard *et al.* [BCG$^+$15a, §D.3] show that Helios'12 satisfies various notions of ballot secrecy. These notions all assume ballots are recorded-as-cast. Unfortunately, ballot secrecy is not satisfied without this assumption, because Helios

[17] `https://github.com/benadida/helios-server/pull/133`, accessed 21 Sep 2017.

[18] Beyond secrecy and verifiability, eligibility is known not to be satisfied [SP13,SP15,MS17].

[19] Cf. `https://github.com/benadida/helios-server/issues/8` and `https://github.com/benadida/helios-server/issues/35`, accessed 21 Sep 2017.

2.0, Helios 3.1.4 and Helios'12 all use malleable ballots in elections with more than two candidates.[20]

**Remark 8.** *Helios'12 does 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 6).[21] Hence, we conclude by Corollary 7. □

A formal proof of Remark 8 would require a formal description of Helios'12. Such a description can be derived as a straightforward variant of Helios 3.1.4 that uses ballot weeding and applies the Fiat–Shamir transformation (rather than the weak Fiat–Shamir transformation). These details provide little value, so we do not pursue them.

The proof sketch of Remark 8 shows that Helios'12 does not defend against a known Helios 2.0 vulnerability, in the presence of an adversary that controls ballot collection. We also derive a new exploit (as the following example demonstrates) by extrapolating from the proof sketch of Theorem 6 and Cortier & Smyth's *permutation attack*, which asserts: given a ballot $b$ for vote $v$, we can abuse malleability to derive a ballot $b'$ for vote $v'$ [CS13, §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 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 $v$. Notions of ballot secrecy used by Bernhard, Pereira & Warinschi [BPW12a], Bernhard [Ber14, §6.11] and Bernhard *et al.* [BCG$^+$15a, §D.3] cannot detect this new exploit nor 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.[22]

[20] Proofs by Bernhard, Pereira & Warinschi and Bernhard *et al.* are limited to two candidate elections, for which Helios'12 uses non-malleable ballots.

[21] The proof sketch of Theorem 6 violates the recorded-as-cast assumption, since the ballot output by the left-right oracle does not appear on the bulletin board.

[22] 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.

The exploit is reliant on a particular candidate not receiving any votes. This is trivial to capture in the context of our ballot secrecy game, because the bulletin board is computed by an adversary that casts ballots on behalf of some voters and controls the distribution of 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 practicality is probably limited to elections in which voters vote in constituencies and each polling station announces its own outcome (cf. Cortier & Smyth [CS13, §3.3]).

**Ballot weeding.** Ballot weeding mechanisms have been proposed, e.g., [CS11, SC11,Smy12,CS13,SB13,BW14,BCG$^+$15b,BCG$^+$15a], but the specification for Helios'12 does not yet define a particular mechanism. One candidate mechanism omits any ballot containing a previously observed hash from the tallying procedure. Another – already in use by the IACR – omits any ballot containing a previously observed hash from the bulletin board.[23] (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 8). Moreover, similarly to our extended proof sketch of Theorem 6 (§4.1), we can extend our proof sketch of Remark 8 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).

## 4.4 Helios'16

We have seen that non-malleable ballots are necessary for ballot secrecy (§3.3), hence, future Helios releases should adopt non-malleable ballots. Smyth, Frink & Clarkson make progress in this direction by proposing Helios'16 [SFC17], a variant of Helios which satisfies verifiability and is intended, but not proven, to use non-malleable ballots. We recall their formal description in Appendix C, and using that formalisation we prove that Helios'16 satisfies secrecy.

**Theorem 9.** *Helios'16 satisfies* Ballot-Secrecy*.*

*Proof sketch.* We prove that Helios'16 has zero-knowledge tallying proofs and, since universal verifiability is satisfied [SFC17], we have HB-Tally-Soundness too (Lemmata 10 & Lemma 29). Hence, by Theorem 5, it suffices to show that Helios'16 satisfies IND-CVA, which we prove by reduction to the security of the underlying encryption scheme (namely, IND-CPA of El Gamal). □

A formal proof of Theorem 9 appears in Appendix C. The proof assumes the random oracle model [BR93]. This proof, coupled with the proof of verifiability by Smyth, Frink & Clarkson [SFC17], provides strong motivation for future

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

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.[24]

# 5 Simplifying ballot-secrecy proofs

We have seen that our definitions of ballot secrecy and ballot independence coincide when tallying proofs are zero-knowledge and honestly constructed ballots are tallied correctly (Theorem 5). Building upon this result and Proposition 11, we show that tallying cannot harm secrecy when all ballots are tallied correctly. 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 $\mathsf{Tally}$ and $\mathsf{Tally}'$ both tally ballots correctly.[25]

Smyth, Frink & Clarkson [SFC17] capture the notion of tallying ballots correctly using function *correct-outcome*. That function uses a *counting quantifier* [Sch05]: A predicate $(\exists^{=\ell} x : P(x))$ that holds exactly when there are $\ell$ distinct values for $x$ such that $P(x)$ is satisfied. (Variable $x$ is bound by the quantifier and integer $\ell$ is free.) Using the counting quantifier, function *correct-outcome* is defined such that $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)[v] = \ell$ iff $\exists^{=\ell} b \in \mathfrak{bb} \backslash \{\bot\} : \exists r : b = \mathsf{Vote}(pk, v, nc, \kappa; r)$, where $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)$ is a vector of length $nc$ and $1 \leq v \leq nc$. Hence, component $v$ of vector $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)$ equals $\ell$ iff there exist $\ell$ 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 8** ($\mathsf{HK\text{-}Injectivity}$)**.** *An election scheme* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *satisfies* honest-key injectivity ($\mathsf{HK\text{-}Injectivity}$)*, if for all probabilistic polynomial-time adversaries $\mathcal{A}$ and security parameters $\kappa$, we have* $\Pr[(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa); (nc, v, v') \leftarrow \mathcal{A}(pk, \kappa); b \leftarrow \mathsf{Vote}(pk, nc, v, \kappa); b' \leftarrow \mathsf{Vote}(pk, nc, v', \kappa) : v \neq v' \wedge b \neq \bot \wedge b' \neq \bot \Rightarrow b \neq b'] = 1$.

Equipped with notions of injectivity and of tallying ballots correctly, we formalise a soundness condition asserting that an election scheme tallies ballots correctly (Definition 9), which allows us to formally state that tallying cannot harm ballot independence when all ballots are tallied correctly (Proposition 11).

**Definition 9** ($\mathsf{Tally\text{-}Soundness}$)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *be an election scheme, $\mathcal{A}$ be an adversary, $\kappa$ be a security parameter, and* $\mathsf{Tally\text{-}Soundness}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

[24] 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. [KTV12b] does not detect vulnerabilities identified by Bernhard et al. [BPW12a] and Chang-Fong & Essex [CE16], possibly because their analysis "does not formalize all the cryptographic primitives used by Helios" [SFC17, §9]).

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

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

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

**Lemma 10.** Tally-Soundness *implies* HB-Tally-Soundness*.*

**Proposition 11.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *and* $\Gamma' = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}')$ *be election schemes. Suppose* $\Gamma$ *and* $\Gamma'$ *satisfy* Tally-Soundness*. We have* $\Gamma$ *satisfies* IND-CVA *iff* $\Gamma'$ *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 11 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,[26] thereby showcasing the applicability of Proposition 11 for a class of encryption-based election schemes.

**Definition 10** (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 outputs* $(pk', sk', p(\kappa), mc)$*, where* $p$ *is a polynomial function.*
- $\mathsf{Vote}(pk', v, nc, \kappa)$ *parses* $pk'$ *as vector* $(pk, \mathfrak{m})$*, outputting* $\perp$ *if parsing fails or* $v \notin \{1, \dots, nc\} \vee \{1, \dots, nc\} \not\subseteq \mathfrak{m}$*, computes* $b \leftarrow \mathsf{Enc}(pk, v)$*, and outputs* $b$*.*
- $\mathsf{Tally}(sk', \mathfrak{bb}, nc, \kappa)$ *parses* $sk'$ *as vector* $(pk, sk)$*, outputting* $\perp$ *if parsing fails, initialises* $\mathfrak{v}$ *as a zero-filled vector of length* $nc$*, computes* **for** $b \in \mathfrak{bb}$ **do** $v \leftarrow \mathsf{Dec}(sk, b)$; **if** $1 \leq v \leq nc$ **then** $\mathfrak{v}[v] \leftarrow \mathfrak{v}[v] + 1$*, and outputs* $(\mathfrak{v}, \epsilon)$*, where* $\epsilon$ *is a constant symbol.*

[26] Our presentation of Enc2Vote extends the presentation by Quaglia & Smyth [QS17a, 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.* [SB14, SB13, BPW12b, BCP$^{+}$11].)

**Lemma 12.** *Given an asymmetric encryption scheme* $\Pi$*, we have* $\mathsf{Enc2Vote}(\Pi)$ *is an election scheme. Moreover, if* $\Pi$ *has perfect correctness, then* $\mathsf{Enc2Vote}(\Pi)$ *satisfies* $\mathsf{HK\text{-}Injectivity}$*.*

A proof of Lemma 12 follows from [QS17a, Lemma 2].[27]

Intuitively, given a non-malleable asymmetric encryption scheme $\Pi$, election scheme $\mathsf{Enc2Vote}(\Pi)$ derives ballot secrecy from $\Pi$ until tallying and tallying maintains ballot secrecy by returning only the number of votes for each candidate. A formal proof of ballot secrecy follows from Quaglia & Smyth, in particular, they show that a stronger notion of ballot secrecy is satisfied [QS17a, Proposition 6], hence, our notion of ballot secrecy is satisfied too, as is ballot independence.

**Corollary 13.** *Given an asymmetric encryption scheme* $\Pi$ *satisfying* $\mathsf{IND\text{-}PA0}$*, we have* $\mathsf{Enc2Vote}(\Pi)$ *satisfies* $\mathsf{IND\text{-}CVA}$*.*

The reverse implication of Corollary 13 does not hold. Indeed, we have the following (stronger) result.

**Proposition 14.** *There exists an asymmetric encryption scheme* $\Pi$ *such that election scheme* $\mathsf{Enc2Vote}(\Pi)$ *satisfies* $\mathsf{Ballot\text{-}Secrecy}$*, but* $\Pi$ *does not satisfy* $\mathsf{IND\text{-}PA0}$*.*

To capitalise on Proposition 11, we demonstrate that $\mathsf{Enc2Vote}$ produces election schemes satisfying tallying soundness (Lemma 15), assuming "ill-formed" ciphertexts are distinguishable from "well-formed" ciphertexts, and combine our results to derive Theorem 16.

**Definition 11.** *Given an asymmetric encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, we say* $\Pi$ *satisfies* well-definedness*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}$*, such that for all security parameters* $\kappa$*, we have* $\Pr[(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa); c \leftarrow \mathcal{A}(pk, \mathfrak{m}, \kappa) : \mathsf{Dec}(sk, c) \neq \perp \Rightarrow \exists m, r \,.\, m \in \mathfrak{m} \wedge c = \mathsf{Enc}(pk, m; r) \wedge c \neq \perp] > 1 - \mathsf{negl}(\kappa)$*.*

**Lemma 15.** *Given a perfectly-correct asymmetric encryption scheme* $\Pi$ *satisfying well-definedness, we have* $\mathsf{Enc2Vote}(\Pi)$ *satisfies* $\mathsf{Tally\text{-}Soundness}$*.*

**Theorem 16.** *Let* $\Pi$ *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* $\mathsf{Tally}'$ *such that* $\Gamma$ *is an election scheme with zero-knowledge tallying proofs. Suppose* $\Pi$ *is perfectly correct and satisfies* $\mathsf{IND\text{-}PA0}$ *and well-definedness. Further suppose* $\Gamma$ *satisfies* $\mathsf{Tally\text{-}Soundness}$*. We have* $\Gamma$ *satisfies* $\mathsf{Ballot\text{-}Secrecy}$*.*

*Proof.* Election scheme $\mathsf{Enc2Vote}(\Pi)$ satisfies $\mathsf{Tally\text{-}Soundness}$ (Lemma 15) and $\mathsf{IND\text{-}CVA}$ (Corollary 13). Thus, $\Gamma$ satisfies $\mathsf{IND\text{-}CVA}$ (Proposition 11) and $\mathsf{Ballot\text{-}Secrecy}$ (Theorem 5 & Lemma 10). □

[27] Quaglia & Smyth only consider asymmetric encryption schemes with perfect correctness, because they require election schemes to satisfy a slightly stronger notion of $\mathsf{HK\text{-}Injectivity}$, and perfect correctness is used to show that $\mathsf{Enc2Vote}(\Pi)$ satisfies that notion. Nonetheless, perfect correctness is not required to ensure the construction produces election schemes. Indeed, the proof by Quaglia & Smyth [QS17a] can trivially be adapted to prove Lemma 12.

We show that tally soundness is implied by universal verifiability in Appendix D. Thus, a special case of Theorem 16 requires universal verifiability rather than tally soundness. It follows that ballot 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 suffice for ballot secrecy.

# 6 Case study II: Helios Mixnet

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

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

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

$\mathsf{Tally}$ selects ballots from a 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 distribution of votes), and announces that outcome, along with zero-knowledge proofs demonstrating correct decryption.

Neither Adida [Adi08] nor Bulens, Giry & Pereira [BGP11] have released an implementation of Helios Mixnet.[28] Tsoukalas *et al.* [TPLT13] released *Zeus* as a fork of Helios spliced with mixnet code to derive an implementation,[29] and Yingtong Li released *helios-server-mixnet* as an extension of Zeus with threshold asymmetric encryption and some other minor changes.[30] We discussed the problem of malleable ballots (§4) with the developers of Zeus and helios-server-mixnet, and they explained that their systems use non-malleable ballots.[31]Email communication, Oct & Dec 2017.

We can treat Helios Mixnet as $\mathsf{Enc2Vote}$ instantiated with $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where algorithm $\mathsf{Gen}$ proves correct key generation and algorithm $\mathsf{Enc}$ 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. To avoid this

[28] The planned implementation of Helios Mixnet (`http://documentation.heliosvoting.org/verification-specs/mixnet-support`, published c. 2010, accessed 19 Dec 2017, & `https://web.archive.org/web/20110119223848/http://documentation.heliosvoting.org/verification-specs/helios-v3-1`, published Dec 2010, accessed 15 Sep 2017) has not been released.

[29] `https://github.com/grnet/zeus`, accessed 15 Sep 2017.

[30] `https://github.com/RunasSudo/helios-server-mixnet`, accessed 15 Sep 2017.

blight, we extend Enc2Vote with such proofs and show that results in Section 5 still hold (Appendix E). This leads us to treat our extension 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:

- $\mathsf{Setup}(\kappa)$ selects coins $s$ uniformly at random, computes $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa; s)$ and a proof $\rho$ of correct key generation using $sk$ and $s$ as the witness, derives $mc$ as the largest integer such that $\{0, \ldots, mc\} \subseteq \{0\} \cup \mathfrak{m}$, 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, $\rho$ does not verify, $v \notin \{1, \ldots, nc\}$, or $\{0, \ldots, nc\} \nsubseteq \mathfrak{m}$, computes $b \leftarrow \mathsf{Enc}(pk, v)$, and outputs $b$.

where

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

It follows that our results can be applied: it is known that $\Pi$ is a non-malleable encryption scheme [BPW12a, Theorem 2], assuming the proof system used by algorithm Enc satisfies simulation sound extractability and $\Pi_0$ satisfies IND-CPA. Moreover, we have $\Pi$ satisfies well-definedness, by the former assumption. Furthermore, Smyth has shown that universal verifiability is satisfied [Smy18], 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 construction for election schemes similar to Helios Mixnet, define a set HeliosM′17 of election schemes using that construction, and prove ballot secrecy for every scheme in that set.

**Theorem 17.** *Each election scheme in* HeliosM′17 *satisfies* Ballot-Secrecy.

A proof of Theorem 17 along with a definition of HeliosM′17 appear in Appendix F.

# 7 Related work

Discussion of ballot secrecy originates from Chaum [Cha81] and the earliest definitions of ballot secrecy are due to Benaloh *et al.* [BY86, BT94, Ben96].[32] More recently, Bernhard *et al.* propose a series of ballot secrecy definitions:

[32] Quaglia & Smyth present a tutorial-style introduction to modelling ballot secrecy [QS17b], and Bernhard *et al.* survey ballot secrecy definitions [BCG+15b, BCG+15a].

They consider election schemes without tallying proofs [BCP$^+$11,BPW12b] and, subsequently, schemes with tallying proofs [BPW12a, SB13, SB14, BCG$^+$15b]. The definition of ballot secrecy by Bernhard, Pereira & Warinschi computes tallying proofs using algorithm Tally or a simulator [BPW12a], but the resulting definition is too weak [BCG$^+$15b, §3.4] and some strengthening is required [BCG$^+$15b, §4]. (Cortier *et al.* [CGGI13a, CGGI13b] propose a variant of the ballot secrecy definition by Bernhard, Pereira & Warinschi. That variant is also too weak [BCG$^+$15b].) By comparison, the definition by Smyth & Bernhard computes tallying proofs using only algorithm Tally [SB13], but the resulting definition is too strong [BCG$^+$15b, §3.5] and a weakening is required [SB14]. The relative merits of ballot secrecy definitions due to Smyth & Bernhard [SB14, Definition 5] and Bernhard *et al.* [BCG$^+$15b, Definition 7] are unknown, in particular, it is unknown whether one definition is stronger than the other.

Discussion of ballot independence originates from Gennaro [Gen95] and the apparent relationship between ballot secrecy and ballot independence has been considered. In particular, Benaloh [Ben96, §2.9] 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. More generally, Sako & Kilian [SK95, §2.4], Michels & Horster [MH96, §3], Wikström [Wik06, Wik08, Wik16] and Cortier & Smyth [CS13, CS11] discuss how malleable ballots can be abused to compromise ballot secrecy. The first definition of ballot independence seems to be due to Smyth & Bernhard [SB13,SB14]. Moreover, Smyth & Bernhard formally prove relations between their definitions of ballot secrecy and ballot independence. Independence has also been studied beyond elections, e.g., [CGMA85], and the possibility of compromising security in the absence of independence has been considered, e.g., [CR87, PP89, Pfi94, DDN91, DDN00, Gen00].

All of the ballot secrecy definitions by Bernhard *et al.* [BCP$^+$11, BPW12b, BPW12a, SB13, SB14, BCG$^+$15b] and the ballot independence definition by Smyth & Bernhard [SB13,SB14] focus on detecting vulnerabilities exploitable by adversaries that control some voters. Vulnerabilities that require control of the bulletin board or the communication channel are not detected, i.e., the bulletin board is implicitly assumed to operate in accordance with the election scheme's rules and the communication channel is implicitly assumed to be secure. This introduces a trust assumption. Under this assumption, Smyth & Bernhard prove that non-malleable ballots are not necessary for ballot secrecy [SB13, §4.3], and Bernhard, Pereira & Warinschi [BPW12a], Bernhard [Ber14] and Bernhard *et al.* [BCG$^+$15a,BCG$^+$15b] prove that Helios'12 satisfies various notions of ballot secrecy. By comparison, we prove that non-malleable ballots are necessary for ballot secrecy without this trust assumption. Hence, Helios'12 does not satisfy our definition of ballot secrecy. 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]" [BCG$^+$15b, §1],[33] 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, developing an appropriate ideal functionality is non-trivial. Moreover, voting systems must be careful cast into a real functionality that appropriately captures the adversary, which is also non-trivial.[34] We leave further exploration of simulation-based definitions of ballot secrecy as an extension for future work.

Bulens, Giry & Pereira pose the investigation of voting systems which allow submission of meaningfully related ballots, whilst preserving ballot secrecy, as an interesting research question [BGP11, §3.2]. Desmedt & Chaidos claim to provide such a system [DC12]. Smyth & Bernhard critique their work and argue that the security results do not support their claims [SB13, §5.1]. We have shown that meaningfully related ballots and ballot secrecy are incompatible, providing negative results for the question posed by Bulens, Giry & Pereira.

Some of the ideas presented in this paper previously appeared in a technical report by Smyth [Smy14] and an extended version of that technical report by Bernhard & Smyth [BS15]. In particular, the limitations of ballot secrecy definitions by Bernhard *et al.* were identified by Smyth [Smy14]. And Definition 2 is based upon the definition of ballot secrecy proposed by Smyth [Smy14, Definition 3]. The main distinction between Definition 2 and the earlier definition is syntax: this paper adopts syntax for election schemes from Smyth, Frink & Clarkson [SFC17], whereas the earlier definition adopts syntax by Smyth & Bernhard [SB14, SB13]. The change in syntax is motivated by the superiority of syntax by Smyth, Frink & Clarkson. Unfortunately, the change has a drawback: we cannot immediately prove that the definition of ballot secrecy proposed in this paper is strictly stronger than the definition proposed by Smyth & Bernhard [SB14, SB13]. By comparison, the technical reports contain such proofs. Nevertheless, the advantages of the syntax change outweigh the disadvantages. Moreover, we can capitalise upon results by Smyth, Frink & Clarkson [SFC17] and Quaglia & Smyth [QS17a].

Following the initial release of these results [Smy15, Smy16], Cortier *et al.* [CSD$^+$17] presented a machine-checked proof that variants of Helios satisfy the notion of ballot secrecy by Bernhard *et al.* [BCG$^+$15b]. As discussed above, that

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

[34] 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.

notion is too weak. In particular, vulnerabilities that require control of ballot collection are not detected. 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.

Beyond the computational model of security, Delaune, Kremer & Ryan formulate a definition of ballot secrecy in the applied pi calculus [DKR09] and Smyth *et al.* show that this definition is amenable to automated reasoning [DRS08, Smy11, BS16, BS17]. An alternative definition is proposed by Cremers & Hirschi, along with sufficient conditions which are also amenable to automated reasoning [CH17]. Albeit, the scope of automated reasoning is limited by analysis tools (e.g., ProVerif [BSCS16]), because the function symbols and equational theory used to model cryptographic primitives might not be suitable for automated analysis (cf. [DKRS11, PB12, ABR12]).

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 distribution of votes cast by the remaining voters. We have seen that Helios'16 satisfies our definition, but ballot secrecy does not ensure free-choice when adversaries are able to communicate with voters nor when voters deviate from the prescribed voting procedure to follow instructions provided by adversaries. 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 [MN06, KZZ15, CCFG16] and coercion resistance [JCJ05, GGR09, UM10, KTV12a], are needed in the presence of such adversaries.

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. Alternatively, 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. Extending our results in this direction is an opportunity for future work. Ultimately, we would prefer not to trust talliers. Unfortunately, this is only known to be possible for decentralised voting systems, e.g., [Sch99,KY02,Gro04,HRZ10,KSRH12], which are designed such that ballots cannot be individually tallied, but are unsuitable for large-scale elections.

McCarthy, Smyth & Quaglia [MSQ14] have shown that auction schemes can be constructed from election schemes, and Quaglia & Smyth [QS17a] 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 [SFC17] to a definition of verifiability for auctions, and prove that their construction produces schemes satisfying verifiability.) Thus, this research has applications beyond voting.

# 8 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. This enabled definitions of ballot independence 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 election 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 ballot secrecy. We also proved non-malleable ballots are sufficient for 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.

In light of the revelation that non-malleable ballots are necessary for ballot secrecy, and in the knowledge that ballots are malleable in Helios, it was discovered that Helios 3.1.4 does not defend against the vulnerability identified by Cortier & Smyth. Moreover, it was discovered that ballot weeding does not offer a defence. Consequently, Helios'12 is also vulnerable. We also revealed a new exploit against Helios'12 that allows an adversary to determine if a voter did not vote for the adversary's preferred candidate. This naturally led to the consideration of whether definitions of ballot secrecy by Bernhard *et al.* could have detected these vulnerabilities and to the conclusion that they could not, because the vulnerabilities require the adversary to control ballot collection, which is prohibited by those definitions.

We have considered vulnerabilities that are only exploitable by an adversary that controls ballot collection. Hence, the vulnerability in Helios'12 can be vacuously eliminated by trusting the tallier to collect ballots. However, Smyth, Frink & Clarkson have shown that Helios'12 does not satisfy universal verifiability, which requires election outcomes to represent votes expressed in collected ballots, without trusting the tallier. Thus, even if we are willing to accept additional trust assumptions to ensure ballot secrecy, we cannot accept such trust assumptions to ensure universal verifiability, because they defy the purpose of verifiability. An alternative solution is necessary and non-malleable ballots are proposed. Moreover, we prove 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 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 ballot secrecy if ballots are tallied correctly (a condition implied by verifiability), and this trivialised our ballot-secrecy proof for Helios Mixnet. Thereby demonstrating the possibility of simple, inexpensive proofs.

This paper 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 paper will simplify future proofs of ballot secrecy and, ultimately, aid democracy-builders in deploying their systems securely.

# Acknowledgements

I am grateful to David Bernhard and to Elizabeth Quaglia for extensive discussions that helped improve this paper and, more generally, my knowledge of cryptography. In addition, I am grateful to Constantin Cătălin Drăgan for explaining subtleties of his work, to Maxime Meyer for his careful proofreading, and to the anonymous reviewers who provided constructive criticism. This work was performed in part at INRIA, with support from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC project *CRYSP* (259639).

# A Cryptographic primitives

## A.1 Asymmetric encryption

**Definition 12** (Asymmetric encryption scheme [KL07])**.** *An* asymmetric encryption scheme *is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, such that:*[35]

- **Gen***, denoted* $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa)$*, inputs a security parameter* $\kappa$ *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* $c$*.*

[35] Our definition differs from Katz and Lindell's original definition [KL07, Definition 10.1] in that we formally state the plaintext space.

- **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* $\kappa$ *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 13** (Homomorphic encryption [SFC17])**.** *An asymmetric encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* homomorphic*, with respect to ternary operators* $\odot$*,* $\oplus$*, and* $\otimes$*,*[36] *if there exists a negligible function* negl*, such that for all security parameters* $\kappa$*, we have the following.*[37] *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* $m_2$*, 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* $\Pi$ *is* additively homomorphic*, if for all security parameters* $\kappa$*, 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 14** (IND-CPA [BDPR98])**.** *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an asymmetric encryption scheme,* $\mathcal{A}$ *be an adversary,* $\kappa$ *be the security parameter, and* $\mathsf{IND\text{-}CPA}(\Pi, \mathcal{A}, \kappa)$ *be the following game.*[38]

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

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

**Definition 15** (IND-PA0 [BS99])**.** *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an asymmetric encryption scheme,* $\mathcal{A}$ *be an adversary,* $\kappa$ *be the security parameter, and* $\mathsf{IND\text{-}PA0}(\Pi, \mathcal{A}, \kappa)$ *be the following game.*

[36] Henceforth, we implicitly bind ternary operators, i.e., we write $\Pi$ *is a homomorphic asymmetric encryption scheme* as opposed to the more verbose $\Pi$ *is a homomorphic asymmetric encryption scheme, with respect to ternary operators* $\odot$*,* $\oplus$*, and* $\otimes$*.*

[37] 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.

[38] Our definition of an asymmetric encryption scheme explicitly defines the plaintext space, whereas Bellare *et al.* [BDPR98] 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.

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

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

## A.2 Proof systems

**Definition 16** (Non-interactive proof system [SFC17])**.** *A* non-interactive proof system *for a relation $R$ is a tuple of algorithms* $(\mathsf{Prove}, \mathsf{Verify})$*, such that:*

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

*Moreover, the system must be* complete*: there exists a negligible function* $\mathsf{negl}$*, such that for all statement and witnesses $(s, w) \in R$ and security parameters $\kappa$, 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 17** (Fiat-Shamir transformation [FS87])**.** *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:*

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

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

*A string $m$ can be included in the hashes computed by algorithms* $\mathsf{Prove}$ *and* $\mathsf{Verify}$*. 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* $\mathsf{Prove}$ *and* $\mathsf{Verify}$ *which include string $m$.*

**Definition 18** (Zero-knowledge [QS17a]). *Let* $\Delta = (\mathsf{Prove}, \mathsf{Verify})$ *be a non-interactive proof system for a relation* $R$*, derived by application of the Fiat-Shamir transformation [FS87] to a random oracle* $\mathcal{H}$ *and a sigma protocol. Moreover, let* $\mathcal{S}$ *be an algorithm,* $\mathcal{A}$ *be an adversary,* $\kappa$ *be a security parameter, and* $\mathit{ZK}(\Delta, \mathcal{A}, \mathcal{H}, \mathcal{S}, \kappa)$ *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* $\sigma$*.*

*And algorithm* $\mathcal{S}$ *can patch random oracle* $\mathcal{H}$*.*[39] *We say* $\Delta$ *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* $\mathsf{negl}$*, and for all security parameters* $\kappa$*, we have* $\mathsf{Succ}(\mathit{ZK}(\Delta, \mathcal{A}, \mathcal{H}, \mathcal{S}, \kappa)) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$*. An algorithm* $\mathcal{S}$ *for which zero-knowledge holds is called a* simulator *for* $(\mathsf{Prove}, \mathsf{Verify})$*.*

**Definition 19** (Simulation sound extractability [SFC17,BPW12a,Gro06]). *Suppose* $\Sigma$ *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* $\mathsf{negl}$*, such that for all security parameters* $\kappa$*, we have:*[40]

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

*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* $\sigma$*.*

[39] 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. [BPW12a] for a formalisation.

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

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

**Theorem 18** (from [BPW12a]). *Let $\Sigma$ be a sigma protocol for relation $R$, and let $\mathcal{H}$ be a random oracle. Suppose $\Sigma$ 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 $\mathsf{Prove}$ and $\mathsf{Verify}$. 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 18 can be extended to this generalisation.

# B Proofs

## B.1 Proof of Theorem 1

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

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

## B.2 Proof of Theorem 2

Suppose $\Gamma$ does not satisfy ballot independence, hence, there exists a probabilistic polynomial-time adversary $\mathcal{A}$, such that for all negligible functions $\mathsf{negl}$, there exists a security parameter $\kappa$ 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. □

## B.3 Proof of Proposition 4

In essence, the proof follows from Theorem 5. 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. □

## B.4 Proof of Theorem 5

Let BS-0, respectively BS-1, be the game derived from Ballot-Secrecy by replacing $\beta \leftarrow_R \{0,1\}$ with $\beta \leftarrow 0$, respectively $\beta \leftarrow 1$. These games are trivially related to Ballot-Secrecy, namely, $\mathsf{Succ}(\mathsf{Ballot\text{-}Secrecy}(\Gamma, \mathcal{A}, \kappa)) = \frac{1}{2} \cdot \mathsf{Succ}(\mathsf{BS\text{-}0}(\Gamma, \mathcal{A}, \kappa)) + \frac{1}{2} \cdot \mathsf{Succ}(\mathsf{BS\text{-}1}(\Gamma, \mathcal{A}, \kappa))$. Moreover, let BS-1:0 be the game derived from BS-1 by replacing $g = \beta$ with $g = 0$. We relate game BS-1:0 to BS-1, and games BS-0 and BS-1:0 to the hybrid games $\mathsf{G}_0, \mathsf{G}_1, \ldots$ introduced in Definition 20. We prove Theorem 5 using these relations.

**Lemma 19.** *Given an adversary $\mathcal{A}$ that wins game* Ballot-Secrecy *against election scheme $\Gamma$, we have* $\mathsf{Succ}(\mathsf{BS\text{-}1}(\Gamma, \mathcal{A}, \kappa)) = 1 - \mathsf{Succ}(\mathsf{BS\text{-}1{:}0}(\Gamma, \mathcal{A}, \kappa))$ *for all security parameters $\kappa$.*

**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 $\kappa$ 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 introduce games $\mathsf{G}_0, \mathsf{G}_1, \ldots$, defined as follows.*

```
$\mathsf{G}_j(\Gamma, \mathcal{A}, \kappa) =$
    $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
    $nc \leftarrow \mathcal{A}(pk, \kappa)$;
    $L \leftarrow \emptyset$;
    $\mathfrak{bb} \leftarrow \mathcal{A}^{\mathcal{O}}()$;
    $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\}, nc, \kappa)$;
    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 $g = 0 \wedge balanced(\mathfrak{bb}, nc, L) \wedge 1 \leq nc \leq mc \wedge |\mathfrak{bb}| \leq mb$;
```

*Oracle $\mathcal{O}$ is defined 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 and tallying procedures. In particular, the first $j$ left-right oracle queries in $\mathsf{G}_j$ construct ballots for the oracle's "left" input and any remaining queries construct ballots for the oracle's "right" input, whereas the left-right oracle in BS-0, respectively BS-1:0, always constructs ballots for the oracle's "left," respectively "right," input. Moreover, the tallying procedure in

$\mathsf{G}_j$ computes the outcome by tallying the ballots on the bulletin board that were constructed by the adversary and by simulating the tallying of any remaining ballots (i.e., ballots constructed by the oracle). And the tallying proof is simulated in $\mathsf{G}_j$. By comparison, the outcome and tallying proof are computing by tallying all the ballots on the bulletin board in both $\mathsf{BS}$-0 and $\mathsf{BS}$-1:0.

**Lemma 20.** *Let $\Gamma$ be an election scheme, $\mathcal{A}$ be an adversary, and $\kappa$ be a security parameter. If $\Gamma$ satisfies* $\mathsf{HB\text{-}Tally\text{-}Soundness}$*, then* $\mathsf{Succ}(\mathsf{BS}\text{-}0(\Gamma, \mathcal{A}, \kappa)) = \mathsf{Succ}(\mathsf{G}_0(\Gamma, \mathcal{A}, \kappa))$ *and* $\mathsf{Succ}(\mathsf{BS}\text{-}1{:}0(\Gamma, \mathcal{A}, \kappa)) = \mathsf{Succ}(\mathsf{G}_q(\Gamma, \mathcal{A}, \kappa))$*, where $q$ is an upper-bound on $\mathcal{A}$'s left-right oracle queries.*

*Proof.* The challengers in games $\mathsf{BS}$-0 and $\mathsf{G}_0$, respectively $\mathsf{BS}$-1:0 and $\mathsf{G}_q$, 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.

Left-right oracles queries $\mathcal{O}(v_0, v_1)$ in games $\mathsf{BS}$-0 and $\mathsf{G}_0$ output ballots for vote $v_0$, hence, the bulletin boards are equivalent in both games. The bulletin boards in $\mathsf{BS}$-1:0 and $\mathsf{G}_q$ 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 $\mathsf{G}_q$. Thus, the bulletin board output by the second adversary call is equivalent in $\mathsf{BS}$-0 and $\mathsf{G}_0$, respectively $\mathsf{BS}$-1:0 and $\mathsf{G}_q$.

It follows that $1 \le nc \le mc \wedge |\mathfrak{bb}| \le mb$ in $\mathsf{BS}$-0 iff $1 \le nc \le mc \wedge |\mathfrak{bb}| \le mb$ in $\mathsf{G}_0$, and similarly for $\mathsf{BS}$-1:0 and $\mathsf{G}_q$. Moreover, predicate *balanced* is satisfied in $\mathsf{BS}$-0 iff it is satisfied in $\mathsf{G}_0$, and similarly for $\mathsf{BS}$-1:0 and $\mathsf{G}_q$. Hence, if $1 \le nc \le mc \wedge |\mathfrak{bb}| \le mb$ is not satisfied or predicate *balanced* is not satisfied, then $\mathsf{Succ}(\mathsf{BS}\text{-}0(\Gamma, \mathcal{A}, \kappa)) = \mathsf{Succ}(\mathsf{G}_0(\Gamma, \mathcal{A}, \kappa))$ and $\mathsf{Succ}(\mathsf{BS}\text{-}1{:}0(\Gamma, \mathcal{A}, \kappa)) = \mathsf{Succ}(\mathsf{G}_q(\Gamma, \mathcal{A}, \kappa))$, concluding our proof. Otherwise, it suffices to show that the outcome and tallying proof are equivalently computed in $\mathsf{BS}$-0 and $\mathsf{G}_0$, respectively $\mathsf{BS}$-1:0 and $\mathsf{G}_q$, since this ensures the inputs to the third adversary call are equivalent, thus the corresponding outputs are equivalent too, which suffices to conclude.

In $\mathsf{BS}$-0, respectively $\mathsf{BS}$-1:0, the outcome is computed by tallying the bulletin board. By comparison, in $\mathsf{G}_0$, respectively $\mathsf{G}_q$, 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 such that $nc \le mc$. Since $\Gamma$ satisfies $\mathsf{HB\text{-}Tally\text{-}Soundness}$, 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

```
$(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\}, nc, \kappa)$;
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 $\mathsf{BS\text{-}0}$ and $\mathsf{G}_0$, respectively $\mathsf{BS\text{-}1{:}0}$ and $\mathsf{G}_q$, it suffices to prove that the simulations are valid, i.e., computing the above is equivalent to computing

```
$(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\}, nc, \kappa)$;
for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ do
    $\mathfrak{v}[v_0] \leftarrow \mathfrak{v}[v_0] + 1$;
```

In $\mathsf{G}_0$, respectively $\mathsf{G}_q$, we have for all $(b, v_0, v_1) \in L$ that $b$ is an output of $\mathsf{Vote}(pk, v_0, nc, \kappa)$, respectively $\mathsf{Vote}(pk, v_1, nc, \kappa)$, such that $v_0, v_1 \in \{1, ..., nc\}$. Moreover, by correctness of $\Gamma$, we have $\mathsf{Tally}(sk, \{b\}, nc, \kappa)$ outputs $(\mathfrak{v}', pf')$ such that $\mathfrak{v}'$ is a zero-filled vector, except for index $v_0$, respectively $v_1$, which contains one. Hence, the simulation is valid in $\mathsf{G}_0$. Furthermore, since predicate *balanced* holds in $\mathsf{G}_q$, 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 $\mathsf{G}_q$, computing

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

is equivalent to computing

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

Thus, the simulation is valid in $\mathsf{G}_q$ too.

In $\mathsf{BS\text{-}0}$, respectively $\mathsf{BS\text{-}1{:}0}$, the tallying proof is computed by tallying the bulletin board. By comparison, in $\mathsf{G}_0$, respectively $\mathsf{G}_q$, the tallying proof is computed by simulator $\mathcal{S}$. Since $\Gamma$ 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 $pf = \mathsf{Prove}((pk, \mathfrak{bb}, nc, \mathfrak{v}), sk, \kappa; r)$, such that coins $r$ are chosen uniformly at random by $\mathsf{Tally}$. Moreover, since $\mathcal{S}$ is a simulator for $(\mathsf{Prove}, \mathsf{Verify})$, proofs output by $\mathsf{Prove}((pk, nc, \mathfrak{bb}, \mathfrak{v}), w, \kappa)$ are indistinguishable from outputs of $\mathcal{S}((pk, nc, \mathfrak{bb}, \mathfrak{v}), \kappa)$. Thus, tallying proofs are equivalently computed in $\mathsf{BS\text{-}0}$ and $\mathsf{G}_0$, respectively $\mathsf{BS\text{-}1{:}0}$ and $\mathsf{G}_q$, thereby concluding our proof. □

*Proof of Theorem 5.* By Theorem 2, it suffices to prove that ballot independence implies ballot secrecy. Suppose $\Gamma$ does not satisfy ballot secrecy, hence, there exists a probabilistic polynomial-time adversary $\mathcal{A}$, such that for all negligible functions $\mathsf{negl}$, there exists a security parameter $\kappa$ and

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

By definition of $\mathsf{BS\text{-}0}$ and $\mathsf{BS\text{-}1}$, we have

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

And, by Lemma 19, we have

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

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

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

which can be rewritten as the telescoping series

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

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

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

Thus,

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

From $\mathcal{A}$, we construct an adversary $\mathcal{B}$ against IND-CVA whose success is at least $\frac{1}{2} + \frac{1}{2} \cdot (\mathsf{Succ}(\mathsf{G}_{j-1}(\Gamma, \mathcal{A}, \kappa)) - \mathsf{Succ}(\mathsf{G}_j(\Gamma, \mathcal{A}, \kappa)))$.

Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$. Since $\Gamma$ 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. We define $\mathcal{B}$ as follows.

- $\mathcal{B}(pk, \kappa)$ 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** $\mathfrak{v}[v_0] \leftarrow \mathfrak{v}[v_0] + 1$, and $pf \leftarrow \mathcal{S}((pk, nc, \mathfrak{bb}, \mathfrak{v}), \kappa); g \leftarrow \mathcal{A}(\mathfrak{v}, pf)$, and outputs $g$.

We prove that $\mathcal{B}$ wins $\mathsf{IND\text{-}CVA}$.

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 $\mathsf{G}_{j-1}$ and $\mathsf{G}_j$ 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 $\mathsf{G}_{j-1}$ and $\mathsf{G}_j$ are equivalent to adversaries that make fewer than $j$ left-right oracle queries, adversary $\mathcal{A}$ must make at least $j$ queries to ensure $\mathsf{Succ}(\mathsf{G}_{j-1}(\Gamma, \mathcal{A}, \kappa)) - \mathsf{Succ}(\mathsf{G}_j(\Gamma, \mathcal{A}, \kappa))$ 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 $j$th oracle query by $\mathcal{A}$. Further suppose $b$ is an output of $\mathsf{Vote}(pk, v_\beta, nc, \kappa)$, where $\beta$ is a bit. If $\beta = 0$, then $\mathcal{B}(b)$ simulates the oracle in $\mathsf{G}_{j-1}$ to $\mathcal{A}$, otherwise, $\mathcal{B}(b)$ simulates the oracle in $\mathsf{G}_j$ to $\mathcal{A}$. In particular, $\mathcal{B}(b)$ responds to the $j$th oracle query with ballot $b$ for $v_\beta$, thus simulating the challenger in $\mathsf{G}_{j-1}$ when $\beta = 0$, respectively $\mathsf{G}_j$ 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 $\mathsf{G}_{j-1}$ to $\mathcal{A}$, thus, $g = \beta$ with at least the probability that $\mathcal{A}$ wins $\mathsf{G}_{j-1}$; or
2. $\beta = 1$ and $\mathcal{B}$ simulates $\mathsf{G}_j$ to $\mathcal{A}$, thus, $g \neq 0$ with at least the probability that $\mathcal{B}$ looses $\mathsf{G}_j$ and, since $\mathcal{A}$ wins game $\mathsf{Ballot\text{-}Secrecy}$, we have $g$ is a bit, hence, $g = \beta$.

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

## B.5 Proof of Lemma 10

**Lemma 21.** *Given an election scheme* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$*, there exists a negligible function* $\mathsf{negl}$*, such that for all security parameters* $\kappa$*, integers* $nc$*, and votes* $v \in \{1, \dots, nc\}$*, we have* $\Pr[(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa); b \leftarrow \mathsf{Vote}(pk, v, nc, \kappa) : 1 \leq mb \wedge nc \leq mc \Rightarrow b \neq \perp] > 1 - \mathsf{negl}(\kappa)$.

*Proof.* Suppose $\kappa$ is a security parameter and $nc$ and $v$ are integers, such that $v \in \{1, \dots, nc\}$. Further suppose $(pk, sk, mb, mc)$ is an output of $\mathsf{Setup}(\kappa)$, $b$ is an output of $\mathsf{Vote}(pk, v, nc, \kappa)$, and $(\mathfrak{v}, pf)$ is an output of $\mathsf{Tally}(sk, \{b\}, nc, \kappa)$, such that $1 \leq mb \wedge nc \leq mc$. By correctness, we have $\mathfrak{v}$ is a zero-filled vector of length $nc$, except for index $v$ which contains integer 1, with overwhelming probability. Given that $\mathsf{Tally}(sk, \{b\}, nc, \kappa)$ and $\mathsf{Tally}(sk, \{b, b\}, nc, \kappa)$ input the same set $\{b\}$, correctness ensures the probability of $\mathsf{Vote}(pk, v, nc, \kappa)$ outputting two identical ballots is upper-bounded by a negligible function. It follows that the probability of $\mathsf{Vote}(pk, v, nc, \kappa)$ outputting error symbol $\perp$ twice is upper-bounded by a negligible function too. Moreover, the probability of $\mathsf{Vote}(pk, v, nc, \kappa)$ outputting error symbol $\perp$ is also upper-bounded by a negligible function, thereby concluding our proof. □

*Proof of Lemma 10.* Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$. Suppose $\Gamma$ does not satisfy $\mathsf{HB\text{-}Tally\text{-}Soundness}$, hence, there exists a probabilistic polynomial-time adversary $\mathcal{A}$, such that for all negligible functions $\mathsf{negl}$, there exists a security parameter $\kappa$ and $\mathsf{negl}(\kappa) < \mathsf{Succ}(\mathsf{HB\text{-}Tally\text{-}Soundness}(\Gamma, \mathcal{A}, \kappa))$. We construct an adversary $\mathcal{B}$ against $\mathsf{Tally\text{-}Soundness}$ from $\mathcal{A}$. We define $\mathcal{B}$ as follows.

```
$\mathcal{B}(pk, \kappa) =$
    $(v, nc, \mathfrak{bb}_0) \leftarrow \mathcal{A}(pk, \kappa)$;
    $(\mathfrak{v}_0, pf_0) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}_0, nc, \kappa)$;
    $\beta \leftarrow_R \{0, 1\}$;
    if $\beta = 1$ then
        $b \leftarrow \mathsf{Vote}(pk, v, nc, \kappa)$;
        $\mathfrak{bb}_1 \leftarrow \mathfrak{bb} \cup \{b\}$;
        $(\mathfrak{v}_1, pf_1) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}_1, nc, \kappa)$;
    return $(nc, \mathfrak{bb}_\beta)$;
```

We prove that $\mathcal{B}$ wins $\mathsf{Tally\text{-}Soundness}$.

Suppose $(pk, sk, mb, mc)$ is an output of $\mathsf{Setup}(\kappa)$, $(v, nc, \mathfrak{bb}_0)$ is an output of $\mathcal{A}(pk, \kappa)$, $b$ is an output of $\mathsf{Vote}(pk, v, nc, \kappa)$, $(\mathfrak{v}_0, pf_0)$ is an output of $\mathsf{Tally}(sk, \mathfrak{bb}_0, nc, \kappa)$, and $(\mathfrak{v}_1, pf_1)$ is an output of $\mathsf{Tally}(sk, \mathfrak{bb}_1, nc, \kappa)$, where $\mathfrak{bb}_1 = \mathfrak{bb}_0 \cup \{b\}$. Let $\mathfrak{v}^* = (\mathfrak{v}_0[1], \ldots, \mathfrak{v}_0[v-1], \mathfrak{v}_0[v] + 1, \mathfrak{v}_0[v+1], \ldots, \mathfrak{v}_0[|\mathfrak{v}_0|])$. Since $\mathcal{A}$ is a winning adversary, we have $\mathfrak{v}^* \neq \mathfrak{v}_1 \wedge b \notin \mathfrak{bb}_0 \wedge 1 \leq v \leq nc \leq mc \wedge |\mathfrak{bb}_0 \cup \{b\}| \leq mb$, with probability greater than $\mathsf{negl}(\kappa)$.

Suppose $\beta$ is a bit chosen uniformly at random. It suffices to prove that $\mathfrak{v}_\beta \neq \textit{correct-outcome}(pk, nc, \mathfrak{bb}_\beta, \kappa)$, with non-negligible probability. Let $\delta_0$, respectively $\delta_1$, be the probability that $\mathfrak{v}_0 \neq \textit{correct-outcome}(pk, nc, \mathfrak{bb}_0, \kappa)$, respectively $\mathfrak{v}_1 \neq \textit{correct-outcome}(pk, nc, \mathfrak{bb}_1, \kappa)$. It follows that $\mathsf{Succ}(\mathsf{Tally\text{-}Soundness}(\Gamma, \mathcal{B}, \kappa)) = \frac{1}{2} \cdot \delta_0 + \frac{1}{2} \cdot \delta_1$ and it remains to show that $\frac{1}{2} \cdot \delta_0 + \frac{1}{2} \cdot \delta_1$ is non-negligible. It suffices to prove that $\mathfrak{v}_0 = \textit{correct-outcome}(pk, nc, \mathfrak{bb}_0, \kappa) \wedge \mathfrak{v}_1 = \textit{correct-outcome}(pk, nc, \mathfrak{bb}_1, \kappa)$ is false with overwhelming probability.

Suppose $\mathfrak{v}_0 = \textit{correct-outcome}(pk, nc, \mathfrak{bb}_0, \kappa)$. By definition of function *correct-outcome*, we have $\exists^{=\mathfrak{v}_0[v]} b' \in \mathfrak{bb}_0 \setminus \{\perp\} : \exists r : b' = \mathsf{Vote}(pk, v, nc, \kappa; r)$. Since $1 \leq |\mathfrak{bb}_0 \cup \{b\}| \leq mb$, we have $b \neq \perp$ by Lemma 21, with overwhelming probability. Given that $b$ is an output of $\mathsf{Vote}(pk, v, nc, \kappa)$, $b \notin \mathfrak{bb}_0$, and $\mathfrak{v}^*[v] = \mathfrak{v}_0[v]+1$, it follows that $\exists^{=\mathfrak{v}^*[v]} b' \in \mathfrak{bb}_0 \cup \{b\} \setminus \{\perp\} : \exists r : b' = \mathsf{Vote}(pk, v, nc, \kappa; r)$. Moreover, by $\mathsf{HK\text{-}Injectivity}$, $b$ is not an output of $\mathsf{Vote}(pk, v', nc, \kappa)$ for all $v' \in \{1, \ldots, |\mathfrak{v}^*|\} \setminus \{v\}$. Thus, for all $v' \in \{1, \ldots, |\mathfrak{v}^*|\} \setminus \{v\}$ we have $\exists^{=\mathfrak{v}^*[v']} b' \in \mathfrak{bb}_0 \cup \{b\} \setminus \{\perp\} : \exists r : b' = \mathsf{Vote}(pk, v', nc, \kappa; r)$. Given that $\mathfrak{bb}_1 = \mathfrak{bb}_0 \cup \{b\}$, we have $\mathfrak{v}^* = \textit{correct-outcome}(pk, nc, \mathfrak{bb}_1, \kappa)$. Moreover, given that $\mathfrak{v}^* \neq \mathfrak{v}_1$, we have $\mathfrak{v}_1 \neq \textit{correct-outcome}(pk, nc, \mathfrak{bb}_1, \kappa)$ with overwhelming probability, which suffices to conclude our proof. □

## B.6 Proof of Proposition 14

We present a construction (Definition 21) for encryption schemes (Lemma 22) which are clearly not secure (Lemma 23). Nevertheless, the construction produces encryption schemes that are sufficient for ballot secrecy (Lemma 24). The proof of Proposition 14 follows from Lemmata 22–24.

**Definition 21.** *Given an asymmetric encryption scheme* $\Pi = (\mathsf{Gen}_\Pi, \mathsf{Enc}_\Pi, \mathsf{Dec}_\Pi)$ *and a constant symbol* $\omega$*, 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 22.** *Given an asymmetric encryption scheme* $\Pi$ *and a constant symbol* $\omega$ *such that* $\Pi$*'s ciphertext space does not contain* $\omega$*, 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 $\omega$ does not appear in the scheme's ciphertext space. □

**Lemma 23.** *Given an asymmetric encryption scheme* $\Pi$ *and a constant symbol* $\omega$ *such that* $\Pi$*'s ciphertext space does not contain* $\omega$ *and* $\Pi$*'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 $\omega$ 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 24.** *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an asymmetric encryption scheme and* $\omega$ *be a constant symbol. Suppose* $\Pi$*'s ciphertext space does not contain* $\omega$ *and* $\Pi$*'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 $\mathsf{Leak}$, we have $\mathsf{Setup} = \mathsf{Setup}'$ and $\mathsf{Vote} = \mathsf{Vote}'$. Suppose $\mathfrak{m}$ is $\Pi$'s message space. By definition of $\mathsf{Leak}$, 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 $\kappa$, bulletin boards $\mathfrak{bb}$, and number of candidates $nc$, that $nc \leq |\mathfrak{m}|$ 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 $\omega$ (witness that decrypting $\omega$ outputs $sk$ such that $sk > |\mathfrak{m}| \geq nc$) and $pf$ is a constant symbol. It follows for all adversaries $\mathcal{A}$ and security parameters $\kappa$ that games Ballot-Secrecy$(\mathsf{Enc2Vote}(\Pi), \mathcal{A}, \kappa)$ and Ballot-Secrecy$(\mathsf{Enc2Vote}(\mathsf{Leak}(\Pi, \omega)), \mathcal{A}, \kappa)$ are equivalent, hence, we have $\mathsf{Succ}($Ballot-Secrecy$(\mathsf{Enc2Vote}(\Pi), \mathcal{A}, \kappa)) = \mathsf{Succ}($ Ballot-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 14.* Let $\Pi$ be an asymmetric encryption scheme and $\omega$ be a constant symbol. Suppose $\Pi$'s ciphertext space does not contain $\omega$. Further suppose $\Pi$'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 22) such that $\mathsf{Enc2Vote}(\mathsf{Leak}(\Pi, \omega))$ satisfies Ballot-Secrecy (Lemma 24), but $\mathsf{Leak}(\Pi, \omega)$ does not satisfy IND-PA0 (Lemma 23), concluding our proof. □

## B.7 Proof of Lemma 15

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 12). 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 $\kappa$ and $\mathsf{negl}(\kappa) < \mathsf{Succ}(\mathsf{Tally\text{-}Soundness}(\mathsf{Enc2Vote}(\Pi), \mathcal{A}, \kappa))$. 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 vector $(pk, \mathfrak{m})$ 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 $nc \leq mc$. 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** $1 \leq v \leq nc$ **then** $\mathfrak{v}[v] \leftarrow \mathfrak{v}[v] + 1$. Since $\Pi$ 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 $1 \leq v \leq nc$ 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 $\Pi$ 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 $1 \leq v \leq nc$ 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 $\mathfrak{v} = \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)$, with overwhelming probability, which concludes our proof. □

# C Helios

Smyth, Frink & Clarkson [SFC17] 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 [SFC17])**.** *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a homomorphic asymmetric encryption scheme and* $\Sigma$ *be a sigma protocol for a binary relation* $R$.[41]

- $\Sigma$ 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* $\kappa$ *and coins* $s$.

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

**Definition 23** (Generalised Helios [SFC17])**.** *Suppose* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is an additively homomorphic asymmetric encryption scheme,* $\Sigma_1$ *is a sigma protocol that proves correct key generation,* $\Sigma_2$ *is a sigma protocol that proves plaintext knowledge in a subspace,* $\Sigma_3$ *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.*[42]

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

[41] 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 $k$.

[42] We omit algorithm $\mathsf{Verify}$ for brevity.

*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** $1 \leq j \leq nc$ **do** $\mathfrak{v}[j] \leftarrow 0$*. Parse* $sk'$ *as a vector* $(pk, sk)$*. 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* $1 \leq i \leq \ell$ *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, compute:*

```
for $1 \leq j \leq nc - 1$ 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 [BPW12a])**.** *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 [SFC17])**.** *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* $\Pi$ *is additively homomorphic El Gamal [CGS97, §2],* $\Sigma_1$ *is the sigma protocol for proving knowledge of discrete logarithms by Chaum et al. [CEGP87, Protocol 2],* $\Sigma_2$ *is the sigma protocol for proving knowledge of disjunctive equality between discrete logarithms by Cramer et al. [CFSY96, Figure 1],* $\Sigma_3$ *is the sigma protocol for proving knowledge of equality between discrete logarithms by Chaum and Pedersen [CP93, §3.2], and* $\mathcal{H}$ *is SHA-256 [NIS12].*

**Definition 26** (Helios 3.1.4 [SFC17])**.** *Election scheme* Helios 3.1.4 *is* Helios 2.0 *after modifying the sigma protocols to perform the checks proposed by Chang-Fong & Essex [CE16, §4].*

**Definition 27** (Helios'16 [SFC17])**.** *Election scheme* Helios'16 *is* $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$*, where* $\Pi$ *is additively homomorphic El Gamal [CGS97, §2],* $\Sigma_1$ *is the sigma protocol for proving knowledge of discrete logarithms by Chaum et al. [CEGP87, Protocol 2],* $\Sigma_2$ *is the sigma protocol for proving knowledge of* *disjunctive equality between discrete logarithms by Cramer et al. [CFSY96, Figure 1], $\Sigma_3$ is the sigma protocol for proving knowledge of equality between discrete logarithms by Chaum & Pedersen [CP93, §3.2], and $\mathcal{H}$ is a random oracle.*

Although Helios actually uses SHA-256 [NIS12], we assume that $\mathcal{H}$ is a random oracle to prove Theorem 9. Moreover, we assume the sigma protocols used by Helios'16 satisfy the preconditions of generalised Helios, that is, [CEGP87, Protocol 2] is a sigma protocol for proving correct key generation, [CFSY96, Figure 1] is a sigma protocol for proving plaintext knowledge in a subspace, and [CP93, §3.2] is a sigma protocol for proving decryption. We leave formally proving this assumption as future work.

## C.1 Proof of Theorem 9

The construction for Helios-like schemes produces election schemes with zero-knowledge tallying proofs (Lemma 25) that satisfy universal verifiability [SFC17] and, thus, honest-ballot tally soundness (Lemma 29). They also satisfy ballot independence (Proposition 26). Hence, they satisfy ballot secrecy too (Theorem 5). We show that Helios'16 satisfies ballot secrecy.

Henceforth, we assume $\Pi$, $\Sigma_1$, $\Sigma_2$ and $\Sigma_3$ 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 25.** *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 $\kappa$ 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 $|\mathfrak{bb}| \leq mb \wedge nc \leq mc$. By inspection of algorithm $\mathsf{Tally}$, 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 26.** *Suppose* $\Pi$ *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 18, 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 $1 \leq i \leq |\mathbf{m_0}|$ 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 $c_1 \leftarrow$

$\mathsf{Enc}(pk, \mathbf{m}_\beta[1]); \dots; c_{|\mathbf{m}_\beta|} \leftarrow \mathsf{Enc}(pk, \mathbf{m}_\beta[|\mathbf{m}_\beta|])$ and inputs $c_1, \dots, c_{|\mathbf{m}_\beta|}$ to the adversary. We have $\Pi$ satisfies $\mathsf{IND\text{-}CPA}^*$ [KL07, §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 $\mathsf{negl}$, there exists a security parameter $\kappa$ 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 $\mathsf{Tally}$. 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:

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

$$\mathbf{W} \leftarrow \mathsf{ExtProveCiph}(\mathbf{H}, \mathbf{P}, \mathbf{Q});$$

$$\mathfrak{v} \leftarrow (\Sigma_{i=1}^{\ell}\mathbf{W}[i][1],\ \dots,\ \Sigma_{i=\ell\cdot(nc-2)+1}^{\ell\cdot(nc-1)}\mathbf{W}[i][1],\ \ell - \Sigma_{j=1}^{nc-1}\mathfrak{v}[j]);$$

$$g \leftarrow \mathcal{A}(\mathfrak{v});$$

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 $\rho$ 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 $\mathsf{Gen}$, 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 27.** *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 27.* 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 $v_0, v_1 \in \{1, \dots, nc\}$, 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 $r_1, \dots, r_{nc-1}$ uniformly at random and compute:

```
for $1 \leq j \leq nc-1$ 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 $v_0 < v_1 \leq nc$, 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 $\beta$. 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 28.** *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 28.* Let $\{b_1, \dots, b_\ell\}$ be the largest subset of $\mathfrak{bb}$ satisfying the conditions of algorithm $\mathsf{Tally}$. 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 $1 \leq i \leq \ell$ and $1 \leq j \leq nc - 1$, 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 $1 \leq i \leq \ell$ and $1 \leq j \leq nc - 1$ 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 $1 \leq j \leq nc - 1$ 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 $\Pi$ is a homomorphic encryption scheme, we have for all $1 \leq j \leq nc-1$ 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 $\Pi$ 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 $\ell \leq mb$. Moreover, since $m_{1,j}, \dots, m_{\ell,j} \in \{0,1\}$ for all $1 \leq j \leq nc - 1$ 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 $\Pi$ is additively homomorphic El Gamal [CGS97, §2]. Moreover, $(\mathsf{ProveKey}, \mathsf{VerKey})$, respectively $(\mathsf{ProveCiph}, \mathsf{VerCiph})$ and $(\mathsf{ProveDec}, \mathsf{VerDec})$, is the non-interactive proof system derived by application of the Fiat-Shamir transformation [FS87] to a random oracle $\mathcal{H}$ and the sigma protocol for proving knowledge of discrete logarithms by Chaum *et al.* [CEGP87, Protocol 2], respectively the sigma protocol for proving knowledge of disjunctive equality between discrete logarithms by Cramer *et al.* [CFSY96, Figure 1] and the sigma protocol for proving knowledge of equality between discrete logarithms by Chaum & Pedersen [CP93, §3.2].

Bernhard, Pereira & Warinschi [BPW12a, §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 18 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 18] still hold." We adopt the same position and assume that Theorem 18 is applicable.

*Proof of Theorem 9.* Helios'16 has zero-knowledge tallying proofs (Lemma 25), subject to the applicability of Theorem 18 to the sigma protocol underlying $(\mathsf{ProveDec}, \mathsf{VerDec})$. Moreover, since Helios'16 satisfies $\mathsf{UV}$ [SFC17], we have Helios'16 satisfies $\mathsf{HB\text{-}Tally\text{-}Soundness}(\Gamma, \mathcal{A}, \kappa)$ (Lemma 29). Furthermore, since El Gamal satisfies $\mathsf{IND\text{-}CPA}$ [TY98,KL07] 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 $\mathsf{IND\text{-}CVA}$ (Proposition 26). Hence, Helios'16 satisfies $\mathsf{Ballot\text{-}Secrecy}$ too (Theorem 5). □

# D Universal verifiability implies tally soundness

We recall the definition of universal verifiability by Smyth, Frink & Clarkson [SFC17] and show that verifiable election schemes satisfy $\mathsf{Tally\text{-}Soundness}$ (Lemma 29). This is useful to simplify applications of Theorems 5, 16, & 31. 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 $\mathsf{Verify}$:

- $\mathsf{Verify}$, 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 $\kappa$. It outputs a bit $s$, where 1 signifies success and 0 failure.

We previously omitted algorithm $\mathsf{Verify}$, 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 $\mathsf{Verify}$ 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 $\mathsf{Tally}$ 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 [SFC17]). *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* $\kappa$*, we have* $\Pr[(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa); (\mathfrak{bb}, nc) \leftarrow \mathcal{A}(pk, \kappa); (\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa) : |\mathfrak{bb}| \leq mb \wedge nc \leq mc \Rightarrow \mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa) = 1] > 1 - \mathsf{negl}(\kappa)$.

**Definition 29** (Injectivity [Smy18, SFC17]). *An election scheme* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *satisfies* injectivity*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, security parameters* $\kappa$ *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 [SFC17]). *An election scheme* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *satisfies* soundness*, if* $\Gamma$ *satisfies injectivity and for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such that for all security parameters* $\kappa$*, we have* $\Pr[(pk, nc, \mathfrak{bb}, \mathfrak{v}, pf) \leftarrow \mathcal{A}(\kappa) : \mathfrak{v} \neq \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa) \wedge \mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa) = 1] \leq \mathsf{negl}(\kappa)$*.*

**Definition 31** (UV [Smy18, SFC17]). *An election scheme* $\Gamma$ *satisfies* universal verifiability (UV)*, if completeness, injectivity and soundness are satisfied.*

**Lemma 29.** *If election scheme* $\Gamma$ *satisfies completeness and soundness, then* $\Gamma$ *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 $\Gamma$. We construct an adversary $\mathcal{B}$ against Exp-UV-Ext from $\mathcal{A}$. We define $\mathcal{B}$ such that $\mathcal{B}(\kappa) = (pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa); (nc, \mathfrak{bb}) \leftarrow \mathcal{A}(pk, \kappa); (\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa);$ $\mathbf{return}(pk, nc, \mathfrak{bb}, \mathfrak{v}, pf)$. 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 $\mathfrak{v} \neq \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa) \wedge |\mathfrak{bb}| \leq mb \wedge nc \leq mc$, 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 29 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.

# E Encryption-based voting systems

We have seen that election scheme Enc2Vote($\Pi$) satisfies HK-Injectivity, if $\Pi$ is perfectly correct (Lemma 12). 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 subclass of schemes generated by $\mathsf{Enc2Vote}$ prove correct key generation. Indeed, we can consider schemes $\mathsf{Enc2Vote}(\Pi)$ such that $\mathsf{Gen}$ proves correct key generation and $\mathsf{Enc}$ verifies such proofs, where $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Alternatively, we can couple $\mathsf{Enc2Vote}$ with proofs of correct key generation:

**Definition 32** ($\mathsf{Enc2Vote}^+$ [Smy18])**.** *Suppose* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is an asymmetric encryption scheme,* $\Sigma$ *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 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* $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) \neq 1 \vee v \notin \{1, \ldots, nc\} \vee \{1, \ldots, nc\} \nsubseteq \mathfrak{m}$*, computes* $b \leftarrow \mathsf{Enc}(pk, v)$*, and outputs* $b$*.*

- $\mathsf{Tally}(sk', \mathfrak{bb}, nc, \kappa)$ *parses* $sk'$ *as vector* $(pk, sk)$*, outputting* $\perp$ *if parsing fails, initialises* $\mathfrak{v}$ *as a zero-filled vector of length* $nc$*, computes* **for** $b \in \mathfrak{bb}$ **do** $v \leftarrow \mathsf{Dec}(sk, b)$; **if** $1 \leq v \leq nc$ **then** $\mathfrak{v}[v] \leftarrow \mathfrak{v}[v] + 1$*, and outputs* $(\mathfrak{v}, \epsilon)$*, where* $\epsilon$ *is a constant symbol.*

**Lemma 30.** *Given an asymmetric encryption scheme* $\Pi$*, a sigma protocol* $\Sigma$ *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 30 appears in [Smy18].

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

**Theorem 31.** *Let* $\mathsf{Enc2Vote}^+(\Pi, \Sigma, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$*, where* $\Pi$ *is an asymmetric encryption scheme,* $\Sigma$ *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* $\mathsf{Tally}'$ *such that* $\Gamma$ *is an election scheme with zero-knowledge tallying proofs. Suppose* $\Pi$ *is perfectly correct and satisfies* $\mathsf{IND\text{-}PA0}$ *and well-definedness. Moreover, suppose* $\Sigma$ *is perfectly complete and* $\mathsf{FS}(\Sigma, \mathcal{H})$ *satisfies zero-knowledge. Further suppose* $\Gamma$ *satisfies* $\mathsf{Tally\text{-}Soundness}$*. We have* $\Gamma$ *satisfies* $\mathsf{Ballot\text{-}Secrecy}$*.*

*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 $\Pi$ is perfectly correct and $\Sigma$ is perfectly complete, we have $\Pi'$ is perfectly correct. Moreover, since $\Pi$ satisfies well-definedness, we have $\Pi'$ does too. Furthermore, since $\mathsf{FS}(\Sigma, \mathcal{H})$ satisfies zero-knowledge and $\Pi$ satisfies $\mathsf{IND\text{-}PA0}$, we have $\Pi'$ satisfies $\mathsf{IND\text{-}PA0}$. It follows that $\mathsf{Enc2Vote}(\Pi')$ satisfies $\mathsf{Tally\text{-}Soundness}$ and $\mathsf{IND\text{-}CVA}$ (Corollary 13 & Lemma 15).

We have $\mathsf{Enc2Vote}(\Pi') = (\mathsf{Setup}', \mathsf{Vote}', \mathsf{Tally})$ such that $\mathsf{Setup}'$ is $\mathsf{Setup}$ except $\mathsf{Setup}$ outputs public key $pk'$ as a vector $(pk, \mathfrak{m}, \rho)$, whereas $\mathsf{Setup}'$ outputs public key $(pk', \mathfrak{m})$. Moreover, $\mathsf{Vote}'$ is $\mathsf{Vote}$ except $\mathsf{Vote}$ inputs public key $pk'$ whereas $\mathsf{Vote}'$ inputs public key $(pk', \mathfrak{m})$. (This blight motivated the inclusion of this appendix.) Hence, it is straightforward to see that $\mathsf{Enc2Vote}^{+}(\Pi, \Sigma, \mathcal{H})$ satisfies $\mathsf{Tally\text{-}Soundness}$ and $\mathsf{IND\text{-}CVA}$, because $\mathsf{Enc2Vote}(\Pi')$ does. Thus, $\Gamma$ satisfies $\mathsf{IND\text{-}CVA}$ (Proposition 11) and $\mathsf{Ballot\text{-}Secrecy}$ (Theorem 5 & Lemma 10). □

# F 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 [SFC17]). *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a homomorphic asymmetric encryption scheme and* $\Sigma$ *be a sigma protocol for a binary relation* $R$*. Suppose that* $(pk, sk) = \mathsf{Gen}(\kappa; s)$*, for some security parameter* $\kappa$ *and coins* $s$*, and* $\mathfrak{m}$ *is the encryption scheme's plaintext space.*

- $\Sigma$ proves plaintext knowledge *if* $((pk, c), (m, r)) \in R \Leftrightarrow c = \mathsf{Enc}(pk, m; r) \wedge m \in \mathfrak{m}$.
- $\Sigma$ proves mixing *if* $((pk, \mathbf{c}, \mathbf{c}'), (\mathbf{r}, \chi)) \in R \Leftrightarrow \bigwedge_{1 \leq i \leq |\mathbf{c}|} \mathbf{c}'[i] = \mathbf{c}[\chi(i)] \otimes \mathsf{Enc}(pk, \mathfrak{e}; \mathbf{r}[i]) \wedge |\mathbf{c}| = |\mathbf{c}'| = |\mathbf{r}|$*, where* $\mathbf{c}$ *and* $\mathbf{c}'$ *are both vectors of ciphertexts encrypted under* $pk$*,* $\mathbf{r}$ *is a vector of coins,* $\chi$ *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** ($\mathsf{HeliosM}$ [Smy18,QS17a]). *Suppose* $\Pi_0 = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a homomorphic asymmetric encryption algorithm,* $\Sigma_1$ *is a sigma protocol that proves correct key construction,* $\Sigma_2$ *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* $(c, \sigma)$.
- $\mathsf{Dec}'(sk, c')$ *parses* $c'$ *as* $(c, \sigma)$, *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* $\Sigma_3$ *is a sigma protocol that proves correct decryption and* $\Sigma_4$ *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.*[43]

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

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

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

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

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

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

```
for $1 \leq i \leq \ell$ 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 $1 \leq \mathbf{W}[i] \leq nc$ 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* $\Pi_0$, $\Sigma_1$, $\Sigma_2$, $\Sigma_3$, $\Sigma_4$ *and* $\mathcal{H}$ *satisfy the preconditions of Definition 34, moreover,* $\Pi_0$ *is perfectly correct and* $\Sigma_1$ *and* $\Sigma_2$ *are perfectly complete, furthermore,* $\Pi_0$ *satisfies* IND-CPA, $\Sigma_1$, $\Sigma_2$, $\Sigma_3$ *and* $\Sigma_4$ *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 [Smy18]. Hence, set HeliosM′17 is not empty.

[43] We omit algorithm Verify for brevity.

## F.1 Proof of Theorem 17

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 $\Sigma_1$ satisfies special soundness and special honest verifier zero-knowledge, we have $\mathsf{FS}(\Sigma_1, \mathcal{H})$ satisfies zero-knowledge (Theorem 18). We use Theorem 31 to prove that $\Gamma \in \mathsf{HeliosM'17}$ satisfies Ballot-Secrecy.

Since $\Pi_0$ is perfectly correct and $\Sigma_2$ is perfectly complete, we have $\Pi$ is a perfectly correct. Moreover, since $\Sigma_2$ satisfies special soundness and special honest verifier zero-knowledge, we have $\mathsf{FS}(\Sigma_2, \mathcal{H})$ satisfies simulation sound extractability (Theorem 18), hence, $\Pi$ satisfies CNM-CPA [BPW12a, Theorem 2] and, equivalently, IND-PA0 [BS99].

To prove $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ satisfies well-definedness, suppose $\mathcal{A}$ is a probabilistic polynomial-time adversary, $\kappa$ 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, $\Pi$ satisfies well-definedness.

Since $\Sigma_3$ and $\Sigma_4$ 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 18), therefore, $\Gamma$ has zero-knowledge tallying proofs by reasoning similar to that given in the proof sketch of Lemma 25. Moreover, since $\Gamma$ satisfies universal verifiability, we have $\Gamma$ satisfies Tally-Soundness (Lemma 29).

We conclude by Theorem 31. □

# References

[ABR12] Myrto Arapinis, Sergiu Bursuc, and Mark 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*, volume 7215 of *LNCS*, pages 169–188. Springer, 2012.

[Adi08] Ben Adida. Helios: Web-based Open-Audit Voting. In *USENIX Security'08: 17th USENIX Security Symposium*, pages 335–348. USENIX Association, 2008.

[AH10] R. Michael Alvarez and Thad E. Hall. *Electronic Elections: The Perils and Promises of Digital Democracy.* Princeton University Press, 2010.

[AMPQ09] Ben Adida, Olivier de Marneffe, Olivier Pereira, and Jean-Jacques 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.

[AN06] Ben Adida and C. Andrew Neff. Ballot casting assurance. In *EVT'06: Electronic Voting Technology Workshop*. USENIX Association, 2006.

[BBP07] Romain Bertrand, Jean-Louis Briquet, and Peter Pels. Introduction: Towards a Historical Ethnography of Voting. In *The Hidden History of the Secret Ballot*. Indiana University Press, 2007.

[BCG$^+$15a] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. A comprehensive analysis of game-based ballot privacy definitions. Cryptology ePrint Archive, Report 2015/255 (version 20150319:100626), 2015.

[BCG$^+$15b] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. SoK: A comprehensive analysis of game-based ballot privacy definitions. In *S&P'15: 36th Security and Privacy Symposium*, pages 499–516. IEEE Computer Society, 2015.

[BCP$^+$11] David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting Helios for provable ballot privacy. In *ESORICS'11: 16th European Symposium on Research in Computer Security*, volume 6879 of *LNCS*, pages 335–354. Springer, 2011.

[BDJR97] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *FOCS'97: 38th Annual Symposium on Foundations of Computer Science*, pages 394–403. IEEE Computer Society, 1997.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *CRYPTO'98: 18th International Cryptology Conference*, volume 1462 of *LNCS*, pages 26–45. Springer, 1998.

[Ben96] Josh Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Department of Computer Science, Yale University, 1996.

[Ber14] David Bernhard. *Zero-Knowledge Proofs in Theory and Practice*. PhD thesis, Department of Computer Science, University of Bristol, 2014.

[BGP11] Philippe Bulens, Damien Giry, and Olivier Pereira. Running Mixnet-Based Elections with Helios. In *EVT/WOTE'11: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. USENIX Association, 2011.

[Bow07] Debra Bowen. Secretary of State Debra Bowen Moves to Strengthen Voter Confidence in Election Security Following Top-to-Bottom Review of Voting Systems. California Secretary of State, press release DB07:042, August 2007.

[BPW12a] David Bernhard, Olivier Pereira, and Bogdan 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*, volume 7658 of *LNCS*, pages 626–643. Springer, 2012.

[BPW12b] David Bernhard, Olivier Pereira, and Bogdan Warinschi. On Necessary and Sufficient Conditions for Private Ballot Submission. Cryptology ePrint Archive, Report 2012/236 (version 20120430:154117b), 2012.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS'93: 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[BR05] Mihir Bellare and Phillip Rogaway. Symmetric Encryption. In *Introduction to Modern Cryptography*, chapter 4. 2005. `http://cseweb.ucsd.edu/~mihir/cse207/w-se.pdf`.

[Bre06] Peter Brent. The Australian ballot: Not the secret ballot. *Australian Journal of Political Science*, 41(1):39–50, 2006.

[BS99] Mihir Bellare and Amit Sahai. Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In *CRYPTO'99: 19th International Cryptology Conference*, volume 1666 of *LNCS*, pages 519–536. Springer, 1999.

[BS15] David Bernhard and Ben Smyth. Ballot secrecy with malicious bulletin boards. Cryptology ePrint Archive, Report 2014/822 (version 20150413:170300), 2015.

[BS16] Bruno Blanchet and Ben Smyth. Automated reasoning for equivalences in the applied pi calculus with barriers. In *CSF'16: 29th Computer Security Foundations Symposium*, pages 310–324. IEEE Computer Society, 2016.

[BS17] Bruno Blanchet and Ben Smyth. Automated reasoning for equivalences in the applied pi calculus with barriers. *Journal of Computer Security*, 2017. To appear.

[BSCS16] Bruno Blanchet, Ben Smyth, Vincent Cheval, and Marc Sylvestre. *ProVerif 1.96: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial*, 2016.

[BT94] Josh Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections. In *STOC'94: 26th Theory of computing Symposium*, pages 544–553. ACM Press, 1994.

[BVQ10] Josh Benaloh, Serge Vaudenay, and Jean-Jacques Quisquater. Final Report of IACR Electronic Voting Committee. International Association for Cryptologic Research. `http://www.iacr.org/elections/eVoting/finalReportHelios_2010-09-27.html`, Sept 2010.

[BW14] David Bernhard and Bogdan Warinschi. Cryptographic Voting — A Gentle Introduction. In *Foundations of Security Analysis and Design VII*, volume 8604 of *LNCS*, pages 167–211. Springer, 2014.

[BY86] Josh Benaloh and Moti Yung. Distributing the Power of a Government to Enhance the Privacy of Voters. In *PODC'86: 5th Principles of Distributed Computing Symposium*, pages 52–62. ACM Press, 1986.

[CCFG16] Pyrros Chaidos, Véronique Cortier, Georg Fuschbauer, and David Galido. BeleniosRF: A Non-interactive Receipt-Free Electronic Voting Scheme. In *CCS'16: 23rd ACM Conference on Computer and Communications Security*, pages 1614–1625. ACM Press, 2016.

[CE16] Nicholas Chang-Fong and Aleksander 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*, pages 324–335. ACM Press, 2016.

[CEGP87] David Chaum, Jan-Hendrik Evertse, Jeroen van de Graaf, and René Peralta. Demonstrating Possession of a Discrete Logarithm Without Revealing It. In *CRYPTO'86: 6th International Cryptology Conference*, volume 263 of *LNCS*, pages 200–212. Springer, 1987.

[CFSY96] Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-Autority Secret-Ballot Elections with Linear Work. In *EUROCRYPT'96: 15th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 1070 of *LNCS*, pages 72–83. Springer, 1996.

[CGGI13a] Veronique Cortier, David Galindo, Stephane Glondu, and Malika Izabachene. A generic construction for voting correctness at minimum cost - Application to Helios. Cryptology ePrint Archive, Report 2013/177 (version 20130521:145727), 2013.

[CGGI13b] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachene. Distributed elgamal à la pedersen: Application to helios. In *WPES'13: Workshop on Privacy in the Electronic Society*, pages 131–142. ACM Press, 2013.

[CGMA85] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. In *FOCS'85: 26th Foundations of Computer Science Symposium*, pages 383–395. IEEE Computer Society, 1985.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In *EUROCRYPT'97: 16th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 1233 of *LNCS*, pages 103–118. Springer, 1997.

[CH17] Cas Cremers and Lucca Hirschi. Improving Automated Symbolic Analysis for E-voting Protocols: A Method Based on Sufficient Conditions for Ballot Secrecy. arXiv, Report 1709.00194, September 2017.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24:84–90, 1981.

[CP93] David Chaum and Torben P. Pedersen. Wallet Databases with Observers. In *CRYPTO'92: 12th International Cryptology Conference*, volume 740 of *LNCS*, pages 89–105. Springer, 1993.

[CR87] Benny Chor and Michael O. Rabin. Achieving Independence in Logarithmic Number of Rounds. In *PODC'87: 6th Principles of Distributed Computing Symposium*, pages 260–268. ACM Press, 1987.

[CS11] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. In *CSF'11: 24th Computer Security Foundations Symposium*, pages 297–311. IEEE Computer Society, 2011.

[CS13] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. *Journal of Computer Security*, 21(1):89–148, 2013.

[CSD$^+$17] Véronique Cortier, Benedikt Schmidt, Constantin Cătălin Drăgan, Pierre-Yves Strub, Francois Dupressoir, and Bogdan 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.

[DC12] Yvo Desmedt and Pyrros Chaidos. Applying Divertibility to Blind Ballot Copying in the Helios Internet Voting System. In *ESORICS'12: 17th European Symposium on Research in Computer Security*, volume 7459 of *LNCS*, pages 433–450. Springer, 2012.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-Malleable Cryptography. In *STOC'91: 23rd Theory of computing Symposium*, pages 542–552. ACM Press, 1991.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable Cryptography. *Journal on Computing*, 30(2):391–437, 2000.

[DK05] Yvo Desmedt and Kaoru Kurosawa. Electronic Voting: Starting Over? In *ISC'05: International Conference on Information Security*, volume 3650 of *LNCS*, pages 329–343. Springer, 2005.

[DKR09] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.

[DKRS11] Stéphanie Delaune, Steve Kremer, Mark D. Ryan, and Graham Steel. Formal analysis of protocols based on TPM state registers. In *CSF'11: 24th Computer Security Foundations Symposium*, pages 66–80. IEEE Computer Society, 2011.

[DRS08] Stéphanie Delaune, Mark D. Ryan, and Ben 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*, volume 263 of *International Federation for Information Processing (IFIP)*, pages 263–278. Springer, 2008.

[FS87] Amos Fiat and Adi Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO'86: 6th International Cryptology Conference*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.

[Gen95] Rosario Gennaro. Achieving independence efficiently and securely. In *PODC'95: 14th Principles of Distributed Computing Symposium*, pages 130–136. ACM Press, 1995.

[Gen00] Rosario Gennaro. A Protocol to Achieve Independence in Constant Rounds. *IEEE Transactions on Parallel and Distributed Systems*, 11(7):636–647, 2000.

[Ger09] Bundesverfassungsgericht (Germany's Federal Constitutional Court). *Use of voting computers in 2005 Bundestag election unconstitutional*, March 2009. Press release 19/2009.

[GGR09] Ryan W. Gardner, Sujata Garera, and Aviel D. Rubin. Coercion Resistant End-to-end Voting. In *FC'09: 13th International Conference on Financial Cryptography and Data Security*, volume 5628 of *LNCS*, pages 344–361. Springer, 2009.

[GH07] Rop Gonggrijp and Willem-Jan Hengeveld. Studying the Nedap/Groenendaal ES3B Voting Computer: A Computer Security Perspective. In *EVT'07: Electronic Voting Technology Workshop*. USENIX Association, 2007.

[Gro04] Jens Groth. Efficient maximal privacy in boardroom voting and anonymous broadcast. In *FC'04: 8th International Conference on Financial Cryptography*, volume 3110 of *LNCS*, pages 90–104. Springer, 2004.

[Gro06] Jens 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*, volume 4284 of *LNCS*, pages 444–459. Springer, 2006.

[Gum05] Andrew Gumbel. *Steal This Vote: Dirty Elections and the Rotten History of Democracy in America*. Nation Books, 2005.

[HBH10] Stuart Haber, Josh Benaloh, and Shai Halevi. The Helios e-Voting Demo for the IACR. International Association for Cryptologic Research. `http://www.iacr.org/elections/eVoting/heliosDemo.pdf`, May 2010.

[HK02] Alejandro Hevia and Marcos A. Kiwi. Electronic Jury Voting Protocols. In *LATIN'02: Theoretical Informatics*, volume 2286 of *LNCS*, pages 415–429. Springer, 2002.

[HK04] Alejandro Hevia and Marcos A. Kiwi. Electronic jury voting protocols. *Theoretical Computer Science*, 321(1):73–94, 2004.

[HRZ10] Fao Hao, Peter Y. A. Ryan, and Piotr Zieliński. Anonymous voting by two-round public discussion. *Journal of Information Security*, 4(2):62 – 67, 2010.

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. In *WPES'05: 4th Workshop on Privacy in the Electronic Society*, pages 61–70. ACM Press, 2005.

[JS12] Douglas W. Jones and Barbara Simons. *Broken Ballots: Will Your Vote Count?*, volume 204 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford University, 2012.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2007.

[KSRH12] Dalia Khader, Ben Smyth, Peter Y. A. Ryan, and Feng Hao. A Fair and Robust Voting System by Broadcast. In *EVOTE'12: 5th International Conference on Electronic Voting*, volume 205 of *Lecture Notes in Informatics*, pages 285–299. Gesellschaft für Informatik, 2012.

[KSRW04] Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach. Analysis of an Electronic Voting System. In *S&P'04: 25th Security and Privacy Symposium*, pages 27–40. IEEE Computer Society, 2004.

[KTV12a] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. A Game-Based Definition of Coercion-Resistance and its Applications. *Journal of Computer Security*, 20(6):709–764, 2012.

[KTV12b] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Clash Attacks on the Verifiability of E-Voting Systems. In *S&P'12: 33rd IEEE Symposium on Security and Privacy*, pages 395–409. IEEE Computer Society, 2012.

[KY02] Aggelos Kiayias and Moti Yung. Self-tallying elections and perfect ballot secrecy. In *PKC'01: 3rd International Workshop on Practice and Theory in Public Key Cryptography*, volume 2274 of *LNCS*, pages 141–158. Springer, 2002.

[KZZ15] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In *EUROCRYPT'15: 34th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 9057 of *LNCS*, pages 468–498. Springer, 2015.

[Lep08] Jill Lepore. Rock, Paper, Scissors: How we used to vote. *Annals of Democracy, The New Yorker*, October 2008.

[LG84] Arend Lijphart and Bernard Grofman. *Choosing an electoral system: Issues and Alternatives.* Praeger, 1984.

[MH96] Markus Michels and Patrick 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*, volume 1163 of *LNCS*, pages 125–132. Springer, 1996.

[Mil30] James Mill. The Ballot. In *The Westminster Review*, volume 13. Robert Heward, 1830.

[MN06] Tal Moran and Moni Naor. Receipt-Free Universally-Verifiable Voting with Everlasting Privacy. In *CRYPTO'06: 26th International Cryptology Conference*, volume 4117 of *LNCS*, pages 373–392. Springer, 2006.

[MS17] Maxime Meyer and Ben Smyth. An attack against the helios election system that exploits re-voting. arXiv, Report 1612.04099, 2017.

[MSQ14] Adam McCarthy, Ben Smyth, and Elizabeth 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*, volume 8437 of *LNCS*, pages 51–63. Springer, 2014.

[NA03] C. Andrew Neff and Jim Adler. Verifiable e-Voting: Indisputable electronic elections at polling places. Technical report, VoteHere, 2003.

[NIS12] NIST. Secure Hash Standard (SHS). FIPS PUB 180-4, Information Technology Laboratory, National Institute of Standards and Technology, March 2012.

[OAS69] Organization of American States. *American Convention on Human Rights, "Pact of San Jose, Costa Rica"*, 1969.

[OSC90] Organization for Security and Co-operation in Europe. *Document of the Copenhagen Meeting of the Conference on the Human Dimension of the CSCE*, 1990.

[PB12] Miriam Paiola and Bruno Blanchet. Verification of Security Protocols with Lists: From Length One to Unbounded Length. In *POST'12: First Conference on Principles of Security and Trust*, volume 7215 of *LNCS*, pages 69–88. Springer, 2012.

[Pfi94] Birgit Pfitzmann. Breaking Efficient Anonymous Channel. In *EUROCRYPT'94: 11th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 950 of *LNCS*, pages 332–340. Springer, 1994.

[PP89] Birgit Pfitzmann and Andreas Pfitzmann. How to Break the Direct RSA-Implementation of Mixes. In *EUROCRYPT'89: 6th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 434 of *LNCS*, pages 373–381. Springer, 1989.

[QS17a] Elizabeth A. Quaglia and Ben Smyth. Secret, verifiable auctions from elections. Cryptology ePrint Archive, Report 2015/1204 (version 20171030:125012), 2017.

[QS17b] Elizabeth A. Quaglia and Ben Smyth. A short introduction to secrecy and verifiability for elections. arXiv, Report 1702.03168, 2017.

[Saa95] Thomas Saalfeld. On Dogs and Whips: Recorded Votes. In Herbert Döring, editor, *Parliaments and Majority Rule in Western Europe*, chapter 16. St. Martin's Press, 1995.

[SB13] Ben Smyth and David Bernhard. Ballot secrecy and ballot independence coincide. In *ESORICS'13: 18th European Symposium on Research in Computer Security*, volume 8134 of *LNCS*, pages 463–480. Springer, 2013.

[SB14] Ben Smyth and David Bernhard. Ballot secrecy and ballot independence: definitions and relations. Cryptology ePrint Archive, Report 2013/235 (version 20141010:082554), 2014.

[SC11] Ben Smyth and Véronique Cortier. A note on replay attacks that violate privacy in electronic voting schemes. Technical Report RR-7643, INRIA, June 2011.

[Sch99] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *CRYPTO'99: 19th International Cryptology Conference*, volume 1666 of *LNCS*, pages 148–164. Springer, 1999.

[Sch05] Nicole Schweikardt. Arithmetic, first-order logic, and counting quantifiers. *ACM Transactions on Computational Logic*, 6(3):634–671, July 2005.

[SFC17] Ben Smyth, Steven Frink, and Michael R. Clarkson. Election Verifiability: Cryptographic Definitions and an Analysis of Helios and JCJ. Cryptology ePrint Archive, Report 2015/233 (version 20170111:122701), 2017.

[SFD$^+$14] Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J. Alex Halderman. Security Analysis of the Estonian Internet Voting System. In *CCS'14: 21st ACM Conference on Computer and Communications Security*, pages 703–715. ACM Press, 2014.

[SK95] Kazue Sako and Joe 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*, volume 921 of *LNCS*, pages 393–403. Springer, 1995.

[Smy11] Ben Smyth. *Formal verification of cryptographic protocols with automated reasoning.* PhD thesis, School of Computer Science, University of Birmingham, 2011.

[Smy12] Ben Smyth. Replay attacks that violate ballot secrecy in Helios. Cryptology ePrint Archive, Report 2012/185, 2012.

[Smy14] Ben Smyth. Ballot secrecy with malicious bulletin boards. Cryptology ePrint Archive, Report 2014/822 (version 20141012:004943), 2014.

[Smy15] Ben Smyth. Secrecy and independence for election schemes. Cryptology ePrint Archive, Report 2015/942 (version 20150928:195428), 2015.

[Smy16] Ben Smyth. Secrecy and independence for election schemes. Cryptology ePrint Archive, Report 2015/942 (version 20160713:142934), 2016.

[Smy17] Ben Smyth. First-past-the-post suffices for ranked voting. `https://bensmyth.com/publications/2017-FPTP-suffices-for-ranked-voting/`, 2017.

[Smy18] Ben Smyth. Verifiability of helios mixnet. Cryptology ePrint Archive, Report 2018/017, 2018.

[SP13] Ben Smyth and Alfredo 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.).

[SP15] Ben Smyth and Alfredo Pironti. Truncating TLS Connections to Violate Beliefs in Web Applications. Technical Report hal-01102013, INRIA, 2015.

[Sta14] CACM Staff. ACM's 2014 General Election: Please Take This Opportunity to Vote. *Communications of the ACM*, 57(5):9–17, May 2014.

[TPLT13] Georgios Tsoukalas, Kostas Papadimitriou, Panos Louridas, and Panayiotis Tsanakas. From Helios to Zeus. *Journal of Election Technology and Systems*, 1(1), 2013.

[TY98] Yiannis Tsiounis and Moti Yung. On the Security of ElGamal Based Encryption. In *PKC'98: First International Workshop on Practice and Theory in Public Key Cryptography*, volume 1431 of *LNCS*, pages 117–134. Springer, 1998.

[UK07] UK Electoral Commission. *Key issues and conclusions: May 2007 electoral pilot schemes*, May 2007.

[UM10] Dominique Unruh and Jörn Müller-Quade. Universally Composable Incoercibility. In *CRYPTO'10: 30th International Cryptology Conference*, volume 6223 of *LNCS*, pages 411–428. Springer, 2010.

[UN48] United Nations. *Universal Declaration of Human Rights*, 1948.

[Wik06] Douglas Wikström. Simplified Submission of Inputs to Protocols. Cryptology ePrint Archive, Report 2006/259, 2006.

[Wik08] Douglas Wikström. Simplified Submission of Inputs to Protocols. In *SCN'08: 6th International Conference on Security and Cryptography for Networks*, volume 5229 of *LNCS*, pages 293–308. Springer, 2008.

[Wik16] Douglas 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`.

[WWH+10] Scott Wolchok, Eric Wustrow, J. Alex Halderman, Hari K. Prasad, Arun Kankipati, Sai Krishna Sakhamuri, Vasavya Yagati, and Rop Gonggrijp. Security Analysis of India's Electronic Voting Machines. In *CCS'10: 17th ACM Conference on Computer and Communications Security*, pages 1–14. ACM Press, 2010.

[WWIH12] Scott Wolchok, Eric Wustrow, Dawn Isabel, and J. Alex Halderman. Attacking the Washington, D.C. Internet Voting System. In *FC'12: 16th International Conference on Financial Cryptography and Data Security*, volume 7397 of *LNCS*, pages 114–128. Springer, 2012.