Secrecy and independence for election schemes

Ben Smyth

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

September 12, 2017

***Abstract*—We study ballot secrecy and ballot independence for election schemes. First, 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 that ballot secrecy is preserved in the presence of an adversary that controls the bulletin board and communication channel. Secondly, we propose a definition of ballot independence as an adaptation of a non-malleability definition for asymmetric encryption. We also provide a simpler, equivalent definition as an indistinguishability game. Thirdly, we prove relations between our definitions. In particular, we prove that ballot independence is necessary in election schemes satisfying ballot secrecy. And that ballot independence is sufficient for ballot secrecy in election schemes with zero-knowledge tallying proofs. Fourthly, we demonstrate the applicability of our results by analysing Helios. Our analysis identifies a new attack against Helios, which enables an adversary to determine if a voter did not vote for a candidate chosen by the adversary. The attack requires the adversary to control the bulletin board or communication channel, thus, it could not have been detected by earlier definitions of ballot secrecy. Finally, we prove that ballot secrecy is satisfied by a variant of Helios that uses non-malleable ballots. *Index Terms*—Elections, Helios, independence, non-malleability, privacy, provable security, secrecy, voting.**

I. Introduction

An election is a decision-making procedure to choose representatives [Gum05], [AH10]. Choices should be made freely, and this has started a movement towards voting as a secret act. This movement is championed by the United Nations [UN48, Article 21], the Organization for Security and Cooperation in Europe [OSC90, Paragraph 7.4], and the Organization of American States [OAS69, Article 23]. And has led to the emergence of ballot secrecy[1] as a *de facto* standard requirement of voting systems.

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

Many voting systems – including systems that have been used in large-scale, binding elections – attempt to satisfy ballot secrecy by placing extensive trust in software and hardware. Unfortunately, many systems are not trustworthy, and are vulnerable to attacks that could compromise ballot secrecy [GH07], [Bow07], [WWH+10], [WWIH12], [SFD+14]. Such vulnerabilities can be avoided by formulating ballot secrecy as a rigorous and precise security definition, and proving that systems satisfy this definition. We propose such a definition in the computational model of cryptography. Our definition builds upon and strengthens earlier definitions of ballot secrecy by Bernhard *et al.* [BCP+11], [BPW12b], [SB13a], [SB14], [BCG+15b] to ensure that ballot secrecy is preserved in the presence of an adversary that controls the bulletin board and the communication channel, whereas definitions by Bernhard *et al.* only consider trusted bulletin boards and channels.

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, i.e., ballots are non-malleable.

Cortier & Smyth [CS13], [CS11], [SC11] attribute a class of ballot secrecy attacks to the absence of ballot independence. Their attribution caused some debate. In particular, Bulens, Giry & Pereira [BGP11, §3.2] highlight the investigation of systems which allow the submission of related votes, whilst preserving ballot secrecy, as an interesting research problem. And Desmedt & Chaidos [DC12] claim to provide a solution.[2] We facilitate the study of ballot independence by proposing two definitions of independence in the computational model. Our first definition is a straightforward adaptation of a non-malleability definition for asymmetric encryption. Our second is a straightforward adaption of an indistinguishability game for asymmetric encryption. The former definition naturally captures ballot independence, but it is complex and proofs of non-malleability are relatively difficult. The latter definition is equivalent, yet simpler, and proofs of indistinguishability are easier.

We demonstrate relations between our definitions of secrecy and independence. In particular, we prove that ballot secrecy implies ballot independence, hence, ballot independence is necessary, assuming ballot secrecy is required. We also prove the inverse implication for a class of voting systems with zero-knowledge tallying proofs. And show that the inverse implication does not hold in general, hence, ballot secrecy is strictly stronger than ballot independence.

We employ our ballot secrecy definition to analyse Helios [AMPQ09], [Per16], a web-based voting system that has been used in binding elections. This scheme is vulnerable to attacks against ballot secrecy [CS13], [CS11]. The next

1. *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.

2. Smyth & Bernhard [SB13a, §5.1] critique the work by Desmedt & Chaidos [DC12] and argue that the security results do not support their claims.

Helios release [Adi14], henceforth *Helios'12*, is intended to mitigate against those attacks. 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, assuming the bulletin board and communication channel are secure, despite the use of malleable ballots. Nevertheless, it follows from our results that ballot secrecy is not satisfied when this assumption is dropped. And this leads to the discovery of a new attack against Helios, whereby an adversary can determine if a voter did not vote for a candidate chosen by the adversary. Violations of ballot secrecy can be overcome using a variant of Helios that uses non-malleable ballots, and we formally prove that our definition of ballot secrecy is satisfied by that variant.

a) Contribution and structure: This paper contributes to the security of voting systems by: proposing definitions of ballot secrecy (§III) and ballot independence (§IV) in the computational model; proving that ballot secrecy is strictly stronger than ballot independence in general, and that secrecy and independence coincide for elections schemes with zero-knowledge tallying proofs (§V); and identifying a new attack against Helios, proposing a fix, and proving that ballot secrecy is satisfied when the fix is applied (§VI). The remaining sections present election scheme syntax (§II), related work (§VII), and a brief conclusion (§VIII), some readers might like to study the related work before definitions of secrecy and independence. The appendices introduce cryptographic primitives and associated security definitions, present proofs, and provide the details of Helios.

II. Election schemes

We recall syntax for *election schemes* from Smyth, Frink & Clarkson [SFC17].[3] Election schemes capture an interesting class of 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. Finally, the tallier tallies the cast ballots and announces an outcome.[4]

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

- Setup*, denoted*[5] $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$*, is run by the tallier*[6]*.* Setup *takes a security parameter* κ *as input and outputs a key pair* pk, sk*, a maximum number of ballots* mb*, and a maximum number of candidates* mc*.*
- Vote*, denoted* $b \leftarrow \mathsf{Vote}(pk, v, nc, \kappa)$*, is run by voters.* Vote *takes as input a public key* pk*, a voter's vote* v*, some number of candidates* nc*, and a security parameter* κ*. The vote should be selected from a sequence* $1, \dots, nc$ *of candidates.* Vote *outputs a ballot* b *or error symbol* $\perp$*.*
- Tally*, denoted* $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$*, is run by the tallier.* Tally *takes as input a private key* sk*, a bulletin board* $\mathfrak{bb}$*, some number of candidates* nc*, and a security parameter* κ*, where* $\mathfrak{bb}$ *is a set.*[7] *It outputs an election outcome* $\mathfrak{v}$ *and a non-interactive tallying proof* pf *(i.e., a proof that the outcome is correct). An election outcome is a vector* $\mathfrak{v}$ *of length* nc *such that* $\mathfrak{v}[v]$ *indicates*[8] *the number of votes for candidate* v*.*

Election schemes must satisfy correctness*: there exists a negligible function* negl*, such that for all security parameters* κ*, integers* nb *and* nc*, and votes* $v_1, \dots, v_{nb} \in \{1, \dots, nc\}$*, it holds that: if* $\mathfrak{v}$ *is a zero-filled vector of length* nc*, then*

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

III. Ballot Secrecy

Our informal definition of ballot secrecy (§I) could be formulated as an indistinguishability game, similar to indistinguishability games for asymmetric encryption (e.g., IND-CPA): we could challenge the adversary to determine whether a ballot is for one of two possible votes. This formalisation is too weak, because election schemes also output the election outcome and a tallying proof, which needs to be incorporated into the game. Unfortunately, it is insufficient to simply grant the adversary access to an oracle that provides an election outcome and tallying proof corresponding to some ballots, because such a game is unsatisfiable. In particular, the adversary can use the oracle to reveal the vote encapsulated inside the challenge ballot. This reveals some limitations in our informal definition of ballot secrecy.

For simplicity, our informal definition of ballot secrecy deliberately omits some side-conditions, which are necessary for satisfiability. In particular, we did not stress that a voter's vote may be revealed in the following scenarios: unanimous election outcomes reveal how everyone voted and, more generally, election outcomes can be coupled with partial knowledge about the distribution of voters' votes to deduce voters' votes. For example, suppose Alice, Bob and Mallory vote in a referendum and the outcome is two "yes" votes and one "no" vote. 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. Accordingly,

3. We omit algorithm Verify from our syntax, because we focus on ballot secrecy, rather than verifiability, in this paper. (Verifiability is studied elsewhere, e.g., [SFC17].)

4. Smyth, Frink & Clarkson use the syntax to model first-past-the-post voting systems and Smyth shows the syntax is sufficiently versatile to capture ranked-choice voting systems too [Smy17].

5. Let $A(x_1, \dots, x_n; r)$ denote the output of probabilistic algorithm A on inputs $x_1, \dots, x_n$ and random coins r. Let $A(x_1, \dots, x_n)$ denote $A(x_1, \dots, x_n; r)$, where r is chosen uniformly at random. And let $\leftarrow$ denote assignment.

6. Some election schemes (e.g., Helios) permit the tallier's role to be distributed amongst several talliers. For simplicity, we consider only a single tallier in this paper. Generalising syntax and security definitions to multiple talliers is a possible direction for future work.

7. Bulletin boards are modelled as sets to avoid the class of attacks against ballot secrecy that arise when duplicate ballots appear on bulletin boards [CS11], [CS13].

8. Let $\mathfrak{v}[v]$ denote component v of vector $\mathfrak{v}$.

ballot secrecy must concede that election outcomes reveal partial information about voters' votes,[9] hence, we refine our informal definition of ballot secrecy 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.

A. Indistinguishability game

We formalise ballot secrecy as an indistinguishability game between an adversary and a challenger.[10]

Definition 2 (Ballot-Secrecy)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{Ballot\text{-}Secrecy}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*[11]

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

Predicate $\mathit{balanced}(\mathfrak{bb}, nc, L)$ *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:*[12]

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

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

The game 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.

The adversary has access to a left-right oracle which can compute ballots on the adversary's behalf.[13] Ballots can be computed by the left-right oracle in two ways, corresponding to a bit β chosen uniformly at random by the challenger. If $\beta = 0$, then, given a pair of votes v_0, v_1, the oracle computes 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 constructs a bulletin board, which may include ballots computed by the oracle. Thus, the game captures a setting where the bulletin board is constructed by an adversary that casts ballots on behalf of a subset of 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. On the other hand, if the adversary loses, then 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 insist 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 $(v_{1,0}, v_{1,1}), \ldots, (v_{n,0}, v_{n,1})$ and the corresponding outputs are $b_1, \ldots, b_n$, further suppose the bulletin board is $\{b_1, \ldots, b_\ell\}$ such that $\ell \leq n$; that game is balanced if the "left" inputs $v_{1,0}, \ldots, v_{\ell,0}$ are a permutation of the "right" inputs $v_{1,1}, \ldots, v_{\ell,1}$. The balanced condition prevents trivial distinctions.[14] For instance, an adversary that constructs 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.

B. Non-malleable encryption is sufficient for secrecy

To demonstrate the applicability of our definition, we recall a construction by Quaglia & Smyth [QS16] for election schemes from asymmetric encryption schemes.[15]

Definition 3 (Enc2Vote [QS16])**.** *Given an asymmetric encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, we define* $\mathsf{Enc2Vote}(\Pi)$ *as follows.*

- $\mathsf{Setup}(\kappa)$ *computes* $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa)$ *and outputs* $(pk, sk, \mathit{poly}(\kappa), |\mathfrak{m}|)$.

9. Alternative formalisations of election schemes might permit different results. For instance, voting systems which only announce the winning candidate [BY86], [HK02], [HK04], [DK05], rather than the number of votes for each candidate (i.e., the election outcome, in our terminology), could offer stronger notions of ballot secrecy.

10. Games are probabilistic algorithms that output booleans. An adversary *wins* a game by causing it to output true ($\top$). We denote an adversary's *success* $\mathsf{Succ}(\mathsf{Exp}(\cdot))$ in a game $\mathsf{Exp}(\cdot)$ as the probability that the adversary wins, that is, $\mathsf{Succ}(\mathsf{Exp}(\cdot)) = \Pr[g \leftarrow \mathsf{Exp}(\cdot) : g = \top]$. Adversaries are assumed to be *stateful*, that is, information persists across invocations of the adversary in a single game, in particular, the adversary can access earlier assignments.

11. Let $x \leftarrow_R S$ denote assignment to x of an element chosen uniformly at random from set S. And let $|\mathbf{v}|$ denote the length of vector $\mathbf{v}$.

12. Oracles may access game parameters, e.g., pk.

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

14. 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."

15. The construction by Quaglia & Smyth builds upon constructions by Bernhard *et al.* [SB14], [SB13a], [BPW12b], [BCP+11].

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

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

Lemma 1. *Given an asymmetric encryption scheme* Π, *we have* $\mathsf{Enc2Vote}(\Pi)$ *is an election scheme (i.e.,* $\mathsf{Enc2Vote}(\Pi)$ *satisfies correctness).*

A proof of Lemma 1 follows from [QS16, §C.2].[16]

Intuitively, given a non-malleable asymmetric encryption scheme Π, election scheme $\mathsf{Enc2Vote}(\Pi)$ derives ballot secrecy from Π until tallying and algorithm Tally 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, Quaglia & Smyth show that $\mathsf{Enc2Vote}(\Pi)$ satisfies a stronger notion of ballot secrecy [QS16, Proposition 5 & 16], hence, $\mathsf{Enc2Vote}(\Pi)$ satisfies our notion of ballot secrecy too.

Corollary 2. *Given an asymmetric encryption scheme* Π *satisfying* IND-PA0, *we have election scheme* $\mathsf{Enc2Vote}(\Pi)$ *satisfies* Ballot-Secrecy.

The reverse implication of Corollary 2 does not hold.

Proposition 3. *There exists an asymmetric encryption scheme* Π *such that election scheme* $\mathsf{Enc2Vote}(\Pi)$ *satisfies* Ballot-Secrecy, *but* Π *does not satisfy* IND-PA0.

A proof of Proposition 3 and all further proofs, except where otherwise stated, appear in Appendix B.

IV. Ballot independence

Our informal definition of ballot independence (§I) essentially states that an adversary is unable to construct a ballot meaningfully related to a non-adversarial ballot. That is, ballots are non-malleable. Hence, we formulate ballot independence using non-malleability. The first formalisation of non-malleability is due to Dolev, Dwork & Naor [DDN91], [DDN00], in the context of asymmetric encryption. Bellare & Sahai [BS99] build upon their results, and results by Bellare *et al.* [BDPR98], to introduce an alternative non-malleability definition for asymmetric encryption. We formalise non-malleability for election schemes as a straightforward adaptation of that definition.

Our formalisation of non-malleability for election schemes captures an intuitive notion of ballot independence, but the definition is complex and proofs of non-malleability are relatively difficult. Bellare & Sahai [BS99] observe similar complexities of non-malleability for encryption and show that their non-malleability definition for encryption is equivalent to a simpler, indistinguishability game for encryption. In a similar direction, we derive a simpler, equivalent definition of ballot independence as a straightforward adaptation of that indistinguishability game.

A. Non-malleability game

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

Definition 4 (CNM-CVA)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{A}, \kappa)$ *and* $\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{A}, \kappa)$ *be the following games.*[17]

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

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

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

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, we have with negligible probability that the adversary wins cnm-cva iff the adversary loses cnm-cva-\$. 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 $\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{A}, \kappa)$, respectively game $\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{A}, \kappa)$, proceeds as follows: the challenger constructs a challenge ballot b for v, respectively v'; the adversary is given ballot b and must compute a

16. Quaglia & Smyth only consider asymmetric encryption schemes with perfect correctness, because they require election schemes to satisfy injectivity, and perfect correctness is required to show that $\mathsf{Enc2Vote}(\Pi)$ satisfies injectivity. Nonetheless, perfect correctness is not required to ensure the construction produces election schemes. Indeed, the proof by Quaglia & Smyth [QS16, §C.2] can trivially be adapted to prove Lemma 1.

17. We abbreviate $x \leftarrow_R S; x' \leftarrow_R S$ as $x, x' \leftarrow_R S$.

relation R and bulletin board $\mathfrak{bb}$; the challenger tallies $\mathfrak{bb}$ and outputs the election outcome $\mathfrak{v}$; and the challenger evaluates whether $R(v, \mathfrak{v})$ holds. Hence, CNM-CVA is satisfied if there is no advantage of the relation constructed by an adversary given a challenge ballot for v, over the relation constructed by an 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] demonstrate the following attack: an adversary observes a voter's ballot, casts a meaningfully related ballot, and exploits the relation to recover the voter's vote from the election outcome.

b) 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: CNM-CPA performs a parallel decryption, whereas, CNM-CVA performs a single tally. It follows that non-malleability for encryption reveals plaintexts corresponding to ciphertexts, whereas, non-malleability for elections reveals the number of ballots for each candidate.

B. Indistinguishability game

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

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

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

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

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 constructed 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'.

c) 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 (§IV-A), namely, IND-PA0 performs a parallel decryption, whereas, IND-CVA performs a single tally.

C. 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 4 (CNM-CVA = IND-CVA)**.** *Given an election scheme* Γ*, we have* Γ *satisfies* CNM-CVA *iff* Γ *satisfies* IND-CVA.

D. Non-malleable encryption is sufficient for independence

It follows naturally from our definitions that non-malleable ciphertexts are sufficient for ballot independence. Indeed, we can derive non-malleable ballots in election schemes produced by construction Enc2Vote on input of encryption schemes satisfying CNM-CPA.[18]

Corollary 5. *Given an asymmetric encryption scheme* Π *satisfying* CNM-CPA*, we have election scheme* $\mathsf{Enc2Vote}(\Pi)$ *satisfies* CNM-CVA.

A proof of Corollary 5 follows from Corollary 2 and Theorems 4 & 7. The reverse implication of Corollary 5 does not hold.

Corollary 6. *There exists an asymmetric encryption scheme* Π *such that election scheme* $\mathsf{Enc2Vote}(\Pi)$ *satisfies* CNM-CVA*, but* Π *does not satisfy* CNM-CPA.

A proof of Corollary 6 follows from Proposition 3 and Theorems 4 & 7.

18. Bellare & Sahai [BS99, §5] show that IND-PA0 coincides with CNM-CPA, thus it suffices to consider IND-PA0 in Corollaries 5 & 6.

V. Relations between secrecy and independence

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

1) The challenger produces one challenge ballot for the adversary in 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 two games and shows that ballot secrecy is stronger than ballot independence.[19] Hence, non-malleable ballots are necessary in election schemes satisfying ballot secrecy.

Theorem 7 (Ballot-Secrecy $\Rightarrow$ IND-CVA)**.** *Given an election scheme Γ satisfying* Ballot-Secrecy*, we have Γ satisfies* IND-CVA.

Moreover, since tallying proofs can reveal voters' votes (e.g., a variant of Enc2Vote could define tallying proofs that map ballots to votes) and since these 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 8 (IND-CVA $\not\Rightarrow$ Ballot-Secrecy)**.** *There exists an election scheme Γ such that Γ satisfies* IND-CVA*, but not* Ballot-Secrecy.

A proof of Proposition 8 follows immediately from our informal reasoning and we omit a formal proof.

Although ballot secrecy is generally stronger than ballot independence, we show that ballot independence is sufficient for ballot secrecy in the class of election schemes without tallying proofs (Definition 6), assuming a soundness condition (Definition 7), which asserts that adding a ballot for v to the bulletin board effects the election outcome by exactly vote v. (This condition is required to hold in the presence of an adversary, whereas correctness is not.)

Definition 6. *An election scheme $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ is* without tallying proofs*, if there exists a constant symbol ϵ such that for all 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 7 (HB-Tally-Soundness)**.** *Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ be an election scheme, $\mathcal{A}$ be an adversary, κ be a security parameter, and $\mathsf{HB\text{-}Tally\text{-}Soundness}(\Gamma, \mathcal{A}, \kappa)$ be the following game.*

$\mathsf{HB\text{-}Tally\text{-}Soundness}(\Gamma, \mathcal{A}, \kappa) =$
$(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
$(v, nc, \mathfrak{bb}_0) \leftarrow \mathcal{A}(pk, \kappa)$;
$b \leftarrow \mathsf{Vote}(pk, v, nc, \kappa)$;
$(\mathfrak{v}_0, pf_0) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}_0, nc, \kappa)$;
$(\mathfrak{v}_1, pf_1) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}_0 \cup \{b\}, nc, \kappa)$;
$\mathfrak{v}^* \leftarrow (\mathfrak{v}_0[1], \ldots, \mathfrak{v}_0[v-1], \mathfrak{v}_0[v]+1, \mathfrak{v}_0[v+1], \ldots, \mathfrak{v}_0[|\mathfrak{v}_0|])$;
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 Γ 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 κ, we have* $\mathsf{Succ}(\mathsf{HB\text{-}Tally\text{-}Soundness}(\Gamma, \mathcal{A}, \kappa)) \leq \mathsf{negl}(\kappa)$.

Proposition 9 (Ballot-Secrecy = IND-CVA, without proofs)**.** *Let Γ be an election scheme without tallying proofs. Suppose Γ satisfies* HB-Tally-Soundness. *We have Γ satisfies* Ballot-Secrecy *iff Γ satisfies* IND-CVA.

Our equivalence result generalises to the class of election schemes with zero-knowledge tallying proofs, that is, election schemes that construct tallying proofs using zero-knowledge non-interactive proof systems.

Definition 8 (Zero-knowledge tallying proofs)**.** *Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ be an election scheme. We say Γ has* zero-knowledge tallying proofs*, if there exists a zero-knowledge non-interactive proof system* $(\mathsf{Prove}, \mathsf{Verify})$*, such that for all security parameters κ, integers nc, bulletin boards $\mathfrak{bb}$, outputs (pk, sk, mb, mc) of $\mathsf{Setup}(\kappa)$, and outputs $(\mathfrak{v}, pf)$ of $\mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$, we have $pf = \mathsf{Prove}((pk, \mathfrak{bb}, nc, \mathfrak{v}), sk, \kappa; r)$, such that coins r are chosen uniformly at random by* Tally.

Theorem 10 (Ballot-Secrecy = IND-CVA, with ZK proofs)**.** *Let Γ be an election scheme with zero-knowledge tallying proofs. Suppose Γ satisfies* HB-Tally-Soundness. *We have Γ satisfies* Ballot-Secrecy *iff Γ satisfies* IND-CVA.

We show that honest-ballot tally soundness is implied by universal verifiability in Appendix C. Hence, a special case of Theorem 10 requires Γ to satisfy universal verifiability. Thus, applications of Theorem 10 are simplified for verifiable election schemes.

VI. Case Study: Helios

Helios is an open-source, web-based electronic voting system,[20] 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],[21] the ACM used Helios for their 2014 general election [Sta14], the Catholic University of Louvain used Helios to elect their university

19. Smyth & Bernhard explain that alternative formalisations of election schemes might permit different results [SB13a, §5.2].

20. https://vote.heliosvoting.org, accessed 19 Jan 2017.

21. https://www.iacr.org/elections/, accessed 19 Jan 2017.

president in 2009 [AMPQ09], and Princeton University has used Helios since 2009 to elect student governments.[22,23] Informally, Helios can be modelled as an election scheme $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ such that:

- Setup generates a key pair for an asymmetric homomorphic encryption scheme, proves correct key generation in zero-knowledge, and outputs the public key coupled with the proof.
- Vote enciphers the vote to a ciphertext, proves correct ciphertext construction in zero-knowledge, and outputs the ciphertext coupled with the proof.
- Tally proceeds as follows. First, any ballots on the bulletin board for which proofs do not hold are discarded. Secondly, the ciphertexts in the remaining ballots are homomorphically combined,[24] the homomorphic combination is decrypted to reveal the election outcome, and correctness of decryption is proved in zero-knowledge. Finally, the election outcome and proof of correct decryption are output.

Helios was first implemented as *Helios 2.0*.[25]

Helios 2.0 is vulnerable to attacks against ballot secrecy [CS13], [CS11], [SC11], [BPW12a].[26] And the next Helios release (Helios'12) is intended to mitigate against those attacks. In particular, the specification [Adi14] incorporates the Fiat-Shamir heuristic (rather than the weak Fiat–Shamir heuristic [BPW12a], which does not include statements in hashes), and there are plans to omit meaningfully related ballots before tallying.[27,28] 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.[29] These notions assume ballots are *recorded-as-cast*, i.e., cast ballots are preserved with integrity through the ballot collection process [AN06, §2]. Unfortunately, ballot secrecy is not satisfied without this assumption, because Helios'12 uses malleable ballots in elections with more than two candidates.[30] Indeed, 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 ciphtertexts contain plaintext 0, otherwise, all ciphertexts contain plaintext 0. Moreover, correct ciphertext construction is shown using proofs $\sigma_1, \dots,$ σ_{nc} such that proof σ_j demonstrates ciphertext c_j contains 0 or 1 for all $j \in \{1, \dots, nc-1\}$, and proof σ_{nc} demonstrates that the homomorphic combination of ciphertexts $c_1 \otimes \cdots \otimes c_{nc-1}$ contains 0 or 1. Hence, given a ballot $c_1, \dots, c_{nc-1}, \sigma_1, \dots,$ σ_{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 χ on $\{1, \dots, nc-1\}$. Thus, ballots are malleable, which is incompatible with ballot secrecy (§V).

Theorem 11. *Helios'12 does not satisfy* $\mathsf{Ballot\text{-}Secrecy}$.

Proof sketch. Suppose an adversary queries the left-right oracle with inputs v_0 and v_1 to derive a ballot for v_β, where β is the bit chosen by the challenger. Further suppose the adversary exploits malleability to derive a related ballot b for v_β and outputs bulletin board $\{b\}$.[31] The board is balanced, because it does not contain the ballot output by the left-right oracle. Suppose the adversary performs the following computation on input of the election outcome $\mathfrak{v}$: if $\mathfrak{v}[v_0] = 1$, then output 0, otherwise, output 1. Since b is a ballot for v_β, 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. $\square$

Our informal proof of Theorem 11 is straightforward. A formal proof would require a formal description of Helios'12. Such a formal description can be derived by adapting the formalisation of Helios 3.1.4 by Smyth, Frink & Clarkson [SFC17] to omit meaningfully related ballots during tallying. These details provide little value, so we do not pursue them further.

The proof sketch of Theorem 11 gives way to the attacks described by Cortier & Smyth [CS13], [CS11], whereby an adversary casts a ballot meaningfully related to a voter's ballot and exploits the relation to recover the voter's vote from the election outcome. We can also derive a new attack (as the following example demonstrates) by extrapolating from the proof sketch and Cortier & Smyth's *permutation attack*, which asserts: given a ballot b for vote v, we can exploit 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 voted for v. (Thereby excluding election outcomes without any votes for candidate v, which would permit Mallory to trivially 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, exploits 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 prevent the attack, 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

22. http://heliosvoting.wordpress.com/2009/10/13/helios-deployed-at-princeton/, accessed 19 Jan 2017.

23. https://princeton.heliosvoting.org/, accessed 19 Jan 2017.

24. The homomorphic combination of ciphertexts is straightforward for two-candidate elections [CF85], [BY86], [SK94], [Ben96], [HS00], since votes (e.g., "yes" or "no") can be encoded as 1 or 0. Multi-candidate elections are also possible [BY86], [Hir10], [DJN10].

25. https://github.com/benadida/helios/releases/tag/2.0, released 25 Jul 2009, accessed 19 Jan 2017.

26. Beyond ballot secrecy, attacks against universal verifiability [BPW12a], [SFC17], [CE16] and eligibility [SP13], [SP15], [MS16] are known.

27. Cf. http://documentation.heliosvoting.org/attacks-and-defenses, https://github.com/benadida/helios-server/issues/8, and https://github.com/benadida/helios-server/issues/35, accessed 19 Jan 2017.

28. Mechanisms to omit ballots have been proposed, e.g., [CS11], [SC11], [Smy12], [CS13], [SB13b], [BCG$^+$15b], [BCG$^+$15a], but the specification for Helios'12 does not yet define a particular mechanism.

29. Proofs by Bernhard, Pereira & Warinschi and Bernhard *et al.* are limited to two candidate elections.

30. Ballots are non-malleable for two candidate elections. (Bernhard, Pereira & Warinschi and Bernhard *et al.* are reliant on non-malleability for their proofs.)

31. The recorded-as-cast assumption is violated because the ballot output by the left-right oracle does not appear on the bulletin board.

Bernhard, Pereira & Warinschi [BPW12a], Bernhard [Ber14, §6.11] and Bernhard *et al.* [BCG$^+$15a, §D.3] would not detect the attack, because interception is not possible when ballots are recorded-as-cast.[32]

The attack 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 constructed by an adversary that casts ballots on behalf of a subset of 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 practicality of an attack 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]).

We have seen that non-malleable ballots are necessary for ballot secrecy (§V), hence, future Helios releases should adopt non-malleable ballots. Smyth, Frink & Clarkson [SFC17] make progress in this direction by proposing Helios'16, a variant of Helios which satisfies verifiability and is intended, but not proven, to use non-malleable ballots (cf. [SHM15]). We recall their formal description in Appendix D. And, using that formalisation, we prove that Helios'16 satisfies ballot secrecy.

Theorem 12. *Helios'16 satisfies* Ballot-Secrecy.

Proof sketch. We prove that Helios'16 has zero-knowledge tallying proofs. And, since Helios'16 satisfies universal verifiability [SFC17], it is also satisfies HB-Tally-Soundness (§C). Hence, by Theorem 10, it suffices to prove that Helios'16 satisfies IND-CVA. And we show that satisfying IND-CVA reduces to the security of the encryption scheme (namely, IND-CPA of El Gamal) underlying Helios'16. □

A formal proof of Theorem 12 appears in Appendix E. 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 Helios releases being based upon Helios'16, since it is the only variant of Helios which is known to be secure.

VII. 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].[33] More recently, Bernhard *et al.* propose a series of ballot secrecy definitions: they consider election schemes without tallying proofs [BCP$^+$11], [BPW12b] and, subsequently, schemes with tallying proofs [BPW12a], [SB13a], [SB14], [BCG$^+$15b]. The definition of ballot secrecy by Bernhard, Pereira & Warinschi computes tallying proofs using algorithm Tally or a simulator [BPW12a], but the resulting definition is too weak [BCG$^+$15b, §3.4] and some strengthening is required [BCG$^+$15b, §4]. (Cortier *et al.* [CGGI13a], [CGGI13b] propose a variant of the ballot secrecy definition by Bernhard, Pereira & Warinschi. That variant is also too weak [BCG$^+$15b].) By comparison, the definition by Smyth & Bernhard computes tallying proofs using only algorithm Tally [SB13a], but the resulting definition is too strong [BCG$^+$15b, §3.5] and a weakening is required [SB14]. The relative merits of ballot secrecy definitions due to Smyth & Bernhard [SB14, Definition 5] and Bernhard *et al.* [BCG$^+$15b, Definition 7] are unknown, in particular, it is unknown whether one definition is stronger than the other.

In the context of elections, discussion of ballot independence originates from Gennaro [Gen95]. And the apparent relationship between ballot secrecy and ballot independence has been considered. 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. And, 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 exploited to compromise ballot secrecy. The first definition of ballot independence seems to be due to Smyth & Bernhard [SB13a], [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 properties due to the lack of independence has been considered, e.g., [CR87], [PP89], [Pfi94], [DDN91], [DDN00], [Gen00].

All of the ballot secrecy definitions by Bernhard *et al.* [BCP$^+$11], [BPW12b], [BPW12a], [SB13a], [SB14], [BCG$^+$15b] and the ballot independence definition by Smyth & Bernhard [SB13a], [SB14] focus on detecting attacks by adversaries that control some voters. Attacks by adversaries that control the bulletin board or communication channel are not detected, i.e., the bulletin board is implicitly assumed to operate in accordance with the election scheme's rules and the communication channel is implicitly assumed to be secure. This introduces a trust assumption. Under this assumption, Smyth & Bernhard prove that non-malleable ballots are not necessary for ballot secrecy [SB13a, §4.3], and Bernhard, Pereira & Warinschi [BPW12a], Bernhard [Ber14] and Bernhard *et al.* [BCG$^+$15a], [BCG$^+$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 due to Bernhard *et al.* by detecting more attacks.

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].

32. 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.

33. Quaglia & Smyth present a tutorial-style introduction to modelling ballot secrecy [QS17], and Bernhard *et al.* survey ballot secrecy definitions [BCG$^+$15b], [BCG$^+$15a].

The main distinction between Definition 2 and the definition by Smyth is syntax: this paper adopts syntax for election schemes from Smyth, Frink & Clarkson [SFC17], whereas, Smyth adopts syntax by Smyth & Bernhard [SB14], [SB13a]. 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], [SB13a]. 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 [QS16].

Following the initial release of these results [Smy15], [Smy16], Cortier *et al.* [CSD$^+$17] presented a machined-checked proof that variants of Helios satisfy the notion of ballot secrecy by Bernhard *et al.* [BCG$^+$15b]. As discussed above, that notion is too weak. In particular, attacks by adversaries that control the bulletin board or communication channel are not detected. Thus, the proof presented here is more appropriate. Nonetheless, their proof builds upon ideas similar to those presented here. 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]. Smyth *et al.* show that this definition is amenable to automated reasoning [DRS08], [Smy11], [BS16]. 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], [SAR13]).

Ballot secrecy formalises a notion of privacy assuming the adversary's capabilities are limited to controlling the set of recorded ballots and assuming ballots are constructed in the prescribed manner. We have seen that Helios'16 satisfies ballot secrecy, but ballot secrecy does not ensure privacy 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 privacy, such as receipt-freeness [MN06], [KZZ15], [CCFG16] and coercion resistance [JCJ05], [GGR09], [UMQ10], [KTV12], are needed in the presence of such adversaries.

Ballot secrecy also assumes that ballots are tallied in the prescribed manner. Hence, the tallier must be trusted. Alternatively, we can design election schemes that distribute the tallier's role amongst several talliers and ensure privacy assuming at least one tallier tallies ballots in the prescribed manner. Extending our results in this direction is an oportunity for future work. Ultimately, we would prefer not to trust talliers. But, this is only known to be possible for decentralised voting systems, e.g., [Sch99], [KY02], [Gro04], [HRZ10], [KSRH12], which are unsuitable for large-scale elections.

McCarthy, Smyth & Quaglia [MSQ14] have shown that auction schemes can be constructed from election schemes, and Quaglia & Smyth [QS16] 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 election verifiability by Smyth, Frink & Clarkson [SFC17] to a definition of auction verifiability, and prove that their construction produces schemes satisfying auction verifiability.) Thus, this research has applications beyond voting.

VIII. 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 ballot secrecy and ballot independence, resulting in a proof that non-malleable ballots are necessary for ballot secrecy. And, moreover, a proof that non-malleable ballots are sufficient for ballot secrecy in election schemes with zero-knowledge tallying proofs. Furthermore, a separation result demonstrates that ballot secrecy is strictly stronger than ballot independence.

In light of the revelation that non-malleable ballots are necessary for ballot secrecy, and in the knowledge that Helios ballots are malleable, it was discovered that Helios does not satisfy ballot secrecy. Although the proof sketch of this result did not immediately uncover an attack against Helios, an extrapolation from that proof sketch revealed an attack that allows an adversary to determine if a voter did not vote for a candidate chosen by the adversary. This naturally led to the consideration of whether definitions of ballot secrecy by Bernhard *et al.* could have detected this attack and the conclusion that they could not, because the attack requires the adversary to control the bulletin board or communication channel, which is prohibited by those definitions.

We exploited our results to prove that a variant of Helios satisfies ballot secrecy. This proof is particularly significant due to the use of Helios in binding elections. And we encourage Helios developers to base future releases on this variant, since it is the only variant of Helios which is known to be secure.

Acknowledgements

Some of the prose (in particular, the two opening paragraphs of Section III) were prepared in collaboration with David Bernhard and I am very grateful for David's contribution. I am also grateful to David for extensive discussions that helped improve this paper and, more generally, my knowledge of cryptography. In addition, I am grateful to Elizabeth Quaglia for her valuable feedback that also helped improve this paper and to Constantin Cătălin Drăgan for explaining subtleties of his work. 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).

Appendix A
Cryptographic primitives

A. Asymmetric encryption

Definition 9 (Asymmetric encryption scheme [KL07])**.** *An* asymmetric encryption scheme *is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, such that:*[34]

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

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

Definition 10 (Homomorphic encryption [SFC17])**.** *An asymmetric encryption scheme* $\Gamma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* homomorphic*, with respect to ternary operators* $\odot$*,* $\oplus$*, and* $\otimes$*,*[35] *if there exists a negligible function* negl*, such that for all security parameters* κ*, we have the following.*[36] *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* Γ *is* additively homomorphic*, if for all security parameters* κ*, key pairs* pk, sk*, and message spaces* $\mathfrak{m}$*, such that there exists coins* r *and* $(pk, sk, \mathfrak{m}) = \mathsf{Gen}(\kappa; r)$*, we have* $\odot_{pk}$ *is the addition operator in group* $(\mathfrak{m}, \odot_{pk})$*.*

Definition 11 (IND-CPA [BDPR98])**.** *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an asymmetric encryption scheme,* $\mathcal{A}$ *be an adversary,* κ *be the security parameter, and* $\mathsf{IND\text{-}CPA}(\Pi, \mathcal{A}, \kappa)$ *be the following game.*[37]

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

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

Definition 12 (IND-PA0 [BS99])**.** *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an asymmetric encryption scheme,* $\mathcal{A}$ *be an adversary,* κ *be the security parameter, and* $\mathsf{IND\text{-}PA0}(\Pi, \mathcal{A}, \kappa)$ *be the following game.*

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

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

B. Proof systems

Definition 13 (Sigma protocol [SFC17], [Dam10], [HL10])**.** *A* sigma protocol *for a relation* R *is a tuple* $(\mathsf{Comm}, \mathsf{Chal}, \mathsf{Resp}, \mathsf{Verify})$ *of probabilistic polynomial-time algorithms such that:*

- **Comm***, denoted* $(\mathsf{comm}, t) \leftarrow \mathsf{Comm}(s, w, \kappa)$*, is executed by a* prover*.* Comm *takes a statement* s*, witness* w *and security parameter* k *as input, and outputs a commitment* comm *and some state information* t*.*
- **Chal***, denoted* $\mathsf{chal} \leftarrow \mathsf{Chal}(s, \mathsf{comm}, \kappa)$*, is executed by a* verifier*.* Chal *takes a statement* s*, a commitment* comm

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

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

36. 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.

37. 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.

and a security parameter k as input, and outputs a string chal.

- **Resp**, *denoted* $\mathsf{resp} \leftarrow \mathsf{Resp}(\mathsf{chal}, t, \kappa)$, *is executed by a prover.* Resp *takes a challenge* chal, *state information* t *and security parameter* k *as input, and outputs a response* resp.
- **Verify**, *denoted* $v \leftarrow \mathsf{Verify}(s, (\mathsf{comm}, \mathsf{chal}, \mathsf{resp}), \kappa)$ *is executed by a verifier.* Verify *takes a statement* s, *a transcript* $(\mathsf{comm}, \mathsf{chal}, \mathsf{resp})$ *and a security parameter* k *as input, and outputs a bit* v, *which is* 1 *if the transcript successfully verifies and* 0 *otherwise. We assume* Verify *is deterministic.*

Moreover, the sigma protocol must be complete*: there exists a negligible function* negl, *such that for all statements and witnesses* $(s, w) \in R$ *and security parameters* k, *we have* $\Pr[(\mathsf{comm}, t) \leftarrow \mathsf{Comm}(s, w, \kappa); \mathsf{chal} \leftarrow \mathsf{Chal}(s, \mathsf{comm}, \kappa); \mathsf{resp} \leftarrow \mathsf{Resp}(\mathsf{chal}, t, \kappa) : \mathsf{Verify}(s, (\mathsf{comm}, \mathsf{chal}, \mathsf{resp}), \kappa) = 1] > 1 - \mathsf{negl}(\kappa)$.

Definition 14 (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* Verify *is deterministic.*

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

Definition 15 (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 tuple* $(\mathsf{Prove}, \mathsf{Verify})$ *of algorithms, defined as follows:*

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

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

Definition 16 (Zero-knowledge [QS16])**.** *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,* κ *be a security parameter, and* $\mathsf{ZK}(\Delta, \mathcal{A}, \mathcal{H}, \mathcal{S}, \kappa)$ *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* σ.

And algorithm $\mathcal{S}$ *can patch random oracle* $\mathcal{H}$.[38] *We say* Δ *satisfies* zero-knowledge, *if there exists a probabilistic polynomial-time algorithm* $\mathcal{S}$, *such that for all probabilistic polynomial-time algorithm adversaries* $\mathcal{A}$, *there exists a negligible function* negl, *and for all security parameters* κ, *we have* $\mathsf{Succ}(\mathsf{ZK}(\Delta, \mathcal{A}, \mathcal{H}, \mathcal{S}, \kappa)) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$. *An algorithm* $\mathcal{S}$ *for which zero-knowledge holds is called a* simulator *for* $(\mathsf{Prove}, \mathsf{Verify})$.

Definition 17 (Simulation sound extractability [SFC17], [BPW12a], [Gro06])**.** *Suppose* Σ *is a sigma protocol for relation* R, $\mathcal{H}$ *is a random oracle, and* $(\mathsf{Prove}, \mathsf{Verify})$ *is a non-interactive proof system, such that* $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{Prove}, \mathsf{Verify})$. *Further suppose* $\mathcal{S}$ *is a simulator for* $(\mathsf{Prove}, \mathsf{Verify})$ *and* $\mathcal{H}$ *can be patched by* $\mathcal{S}$. *Proof system* $(\mathsf{Prove}, \mathsf{Verify})$ *satisfies* simulation sound extractability *if there exists a probabilistic polynomial-time algorithm* $\mathcal{K}$, *such that for all probabilistic polynomial-time adversaries* $\mathcal{A}$ *and coins* r, *there exists a negligible function* negl, *such that for all security parameters* κ, *we have:*[39]

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

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* $\sigma \leftarrow \mathcal{S}(s); \mathbf{P} \leftarrow (\mathbf{P}[1], \ldots, \mathbf{P}[|\mathbf{P}|], (s, \sigma))$ *and outputs* σ.

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

Theorem 13 (from [BPW12a])**.** *Let* Σ *be a sigma protocol for relation* R, *and let* $\mathcal{H}$ *be a random oracle. Suppose* Σ *satisfies special soundness and special honest verifier zero-knowledge. Non-interactive proof system* $\mathsf{FS}(\Sigma, \mathcal{H})$ *satisfies zero-knowledge and simulation sound extractability.*

The Fiat-Shamir transformation can be generalised to include an optional string in the hashes produced by functions Prove and Verify. Simulators can be generalised to include an optional string m too. We write $\mathcal{S}(s, m, \kappa)$ for invocations of simulator $\mathcal{S}$ which include an optional string. Theorem 13 can be extended to this generalisation.

The Fiat-Shamir transformation can be generalised to include an optional string m in the hashes produced by functions Prove and Verify. We write $\mathsf{Prove}(s, w, m, \kappa)$ and $\mathsf{Verify}(s,$

38. 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.

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

$(\mathsf{comm}, \mathsf{resp}), m, k)$ for invocations of Prove and Verify which include an optional string. When m is provided, it is included in the hashes in both algorithms. That is, given $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{Prove}, \mathsf{Verify})$, the hashes are computed as follows in both algorithms: $\mathsf{chal} \leftarrow \mathcal{H}(\mathsf{comm}, s, m)$. Simulators can be generalised to include an optional string m too. We write $\mathcal{S}(s, m, \kappa)$ for invocations of simulator $\mathcal{S}$ which include an optional string. Theorem 13 can be extended to this generalisation.

Appendix B
Proofs

A. Proof of Proposition 3

We present a construction (Definition 18) for encryption schemes (Lemma 14) which are clearly not secure (Lemma 15). Nevertheless, the construction produces encryption schemes that are sufficient for ballot secrecy (Lemma 16). The proof of Proposition 3 follows from Lemmata 14–16.

Definition 18. *Given an asymmetric encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *and a constant symbol* ω*, let* $\mathsf{Leak}(\Pi, \omega) = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}')$*, such that* $\mathsf{Dec}'(sk, c)$ *proceeds as follows: if* $c = \omega$*, then output* sk*, otherwise, compute* $m \leftarrow \mathsf{Dec}(sk, c)$ *and output* m*.*

Lemma 14. *Given an asymmetric encryption scheme* Π *and a constant symbol* ω*, such that* Π*'s ciphertext space does not contain* ω*, we have* $\mathsf{Leak}(\Pi, \omega)$ *is an asymmetric encryption scheme.*

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

Lemma 15. *Given an asymmetric encryption scheme* Π *and a constant symbol* ω*, such that* Π*'s ciphertext space does not contain* ω *and* Π*'s message space is larger than one for some security parameter, we have* $\mathsf{Leak}(\Pi, \omega)$ *does not satisfy* IND-PA0.

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

Lemma 16. *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an asymmetric encryption scheme and* ω *be a constant symbol. Suppose* Π*'s ciphertext space does not contain* ω *and* Π*'s message space is smaller than the private key. Further suppose* $\mathsf{Enc2Vote}(\Pi)$ *satisfies* Ballot-Secrecy*. We have* $\mathsf{Enc2Vote}(\mathsf{Leak}(\Pi, \omega))$ *satisfies* Ballot-Secrecy.

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

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

because Enc2Vote ensures that $\overline{\mathfrak{v}}$ is not influenced by decrypting ω (witness that decrypting ω 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 κ that games $\mathsf{Ballot\text{-}Secrecy}(\mathsf{Enc2Vote}(\Pi), \mathcal{A}, \kappa)$ and $\mathsf{Ballot\text{-}Secrecy}(\mathsf{Enc2Vote}(\mathsf{Leak}(\Pi, \omega)), \mathcal{A}, \kappa)$ are equivalent, hence, we have $\mathsf{Succ}(\mathsf{Ballot\text{-}Secrecy}(\mathsf{Enc2Vote}(\Pi), \mathcal{A}, \kappa)) = \mathsf{Succ}(\mathsf{Ballot\text{-}Secrecy}(\mathsf{Enc2Vote}(\mathsf{Leak}(\Pi, \omega)), \mathcal{A}, \kappa))$. Moreover, since $\mathsf{Enc2Vote}(\Pi)$ satisfies Ballot-Secrecy, it follows that $\mathsf{Enc2Vote}(\mathsf{Leak}(\Pi, \omega))$ satisfies Ballot-Secrecy too. □

Proof of Proposition 3. Let Π be an asymmetric encryption scheme and ω be a constant symbol. Suppose Π's ciphertext space does not contain ω. Further suppose Π's message space is larger than one for some security parameter, but smaller than the private key. We have $\mathsf{Enc2Vote}(\mathsf{Leak}(\Pi, \omega))$ is an asymmetric encryption scheme (Lemma 14) such that $\mathsf{Enc2Vote}(\mathsf{Leak}(\Pi, \omega))$ satisfies Ballot-Secrecy (Lemma 16), but $\mathsf{Leak}(\Pi, \omega)$ does not satisfy IND-PA0 (Lemma 15), concluding our proof. □

B. Proof of Theorem 4

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

- $\mathcal{B}(pk, \kappa)$ 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 $\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{B}, \kappa)$, then adversary $\mathcal{B}$ simulates $\mathcal{A}$'s challenger to $\mathcal{A}$ in $\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{A}, \kappa)$ 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 $\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{A}, \kappa)$ 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 $\mathsf{Succ}(\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{A}, \kappa)) = 1/2 \cdot (\mathsf{Succ}(\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{A}, \kappa)) + 1 - \mathsf{Succ}(\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{A}, \kappa)))$ and, therefore, $2 \cdot \mathsf{Succ}(\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{A}, \kappa)) - 1 = \mathsf{Succ}(\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{A}, \kappa)) - \mathsf{Succ}(\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{A}, \kappa))$. Since Γ does not satisfy CNM-CVA and a function that doubles the output of a negligible function is a negligible function, we have $\mathsf{Succ}(\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{A}, \kappa)) - \mathsf{Succ}(\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{A}, \kappa)) > 2 \cdot \mathsf{negl}(\kappa)$. It follows that $2 \cdot \mathsf{Succ}(\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{A}, \kappa)) - 1 > 2 \cdot \mathsf{negl}(\kappa)$, hence, $\mathsf{Succ}(\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{A}, \kappa)) > 1/2 + \mathsf{negl}(\kappa)$, concluding our proof.

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

- $\mathcal{B}(pk, \kappa)$ 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 $\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{A}, \kappa)$. 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 one time in the game. Let use consider adversary $\mathcal{B}$'s success in $\mathsf{cnm\text{-}cva}(\Gamma, \mathcal{B}, \kappa)$ and $\mathsf{cnm\text{-}cva\text{-}\$}(\Gamma, \mathcal{B}, \kappa)$.

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

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

C. Proof of Theorem 7

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

- $\mathcal{B}(pk, \kappa)$ 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, $\mathsf{Succ}(\mathsf{Ballot\text{-}Secrecy}(\Gamma, \mathcal{B}, \kappa)) > 1/2 + \mathsf{negl}(\kappa)$, concluding our proof. □

D. Proof of Proposition 9

In essence, the proof follows from Theorem 10. 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 constructed by a zero-knowledge non-interactive proof system, but an election scheme without tallying proofs need not construct 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. □

E. Proof of Theorem 10

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 19. We prove Theorem 10 using these relations.

Lemma 17. *Given an adversary $\mathcal{A}$ that wins game* Ballot-Secrecy *against election scheme Γ, we have $\mathsf{Succ}(\mathsf{BS\text{-}1}(\Gamma, \mathcal{A}, \kappa)) = 1 - \mathsf{Succ}(\mathsf{BS\text{-}1\text{:}0}(\Gamma, \mathcal{A}, \kappa))$ for all security parameters κ.*

Definition 19. *Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$ be an election scheme with zero-knowledge tallying proofs, $\mathcal{A}$ be an adversary, and κ be a security parameter. Moreover, let $\mathcal{S}$ be the simulator for the zero-knowledge non-interactive proof system used by algorithm* Tally *to construct 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 $g = 0 \wedge \mathit{balanced}(\mathfrak{bb}, nc, L) \wedge 1 \le nc \le mc \wedge |\mathfrak{bb}| \le mb$;

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 G_j compute ballots for the oracle's "left" input and any remaining queries compute ballots for the oracle's "right" input, whereas the left-right oracle in BS-0, respectively BS-1:0, always computes ballots for the oracle's "left," respectively "right," input. Moreover, the tallying procedure in 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 G_j. By comparison, the outcome and tallying proof are computing by tallying all the ballots on the bulletin board in both BS-0 and BS-1:0.

Lemma 18. *Let Γ be an election scheme, $\mathcal{A}$ be an adversary, and κ be a security parameter. If Γ satisfies* HB-Tally-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 BS-0 and G_0, respectively BS-1:0 and G_q, both construct 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 BS-0 and G_0 output ballots for vote v_0, hence, the bulletin boards are equivalent in both games. The bulletin boards in BS-1:0 and G_q are similarly equivalent, in particular, left-right oracles queries $\mathcal{O}(v_0, v_1)$ in both games output ballots for vote v_1, because q is an upper-bound on the left-right oracle queries, therefore, $|L| < q$ in G_q. Thus, the bulletin board output by the second adversary call is equivalent in BS-0 and G_0, respectively BS-1:0 and G_q.

It follows that $1 \leq nc \leq mc \wedge |\mathfrak{bb}| \leq mb$ in BS-0 iff $1 \leq nc \leq mc \wedge |\mathfrak{bb}| \leq mb$ in G_0, and similarly for BS-1:0 and G_q. Moreover, predicate *balanced* is satisfied in BS-0 iff it is satisfied in G_0, and similarly for BS-1:0 and G_q. Hence, if $1 \leq nc \leq mc \wedge |\mathfrak{bb}| \leq mb$ 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 BS-0 and G_0, respectively BS-1:0 and 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 BS-0, respectively BS-1:0, the outcome is computed by tallying the bulletin board. By comparison, in G_0, respectively 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 $nc \leq mc$. Since Γ satisfies HB-Tally-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 BS-0 and G_0, respectively BS-1:0 and 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 G_0, respectively 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 Γ, we have $\mathsf{Tally}(sk, \{b\}, nc, \kappa)$ outputs $(\mathfrak{v}', pf')$ such that $\mathfrak{v}'$ is a zero-filled vector, except for index v_0, respectively v_1, which contains one. Hence, the simulation is valid in G_0. Furthermore, since predicate *balanced* holds in 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 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 G_q too.

In BS-0, respectively BS-1:0, the tallying proof is computed by tallying the bulletin board. By comparison, in G_0, respectively G_q, the tallying proof is computed by simulator $\mathcal{S}$. Since Γ has zero-knowledge tallying proofs, there exists a non-interactive proof system $(\mathsf{Prove}, \mathsf{Verify})$ such that for all $(\mathfrak{v}, pf)$ output by $\mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$, we have $pf = \mathsf{Prove}((pk, \mathfrak{bb}, nc, \mathfrak{v}), sk, \kappa; r)$, such that coins r are chosen uniformly at random by 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 BS-0 and G_0, respectively BS-1:0 and G_q, thereby concluding our proof. □

Proof of Theorem 10. By Theorem 7, it suffices to prove that ballot independence implies ballot secrecy. Suppose Γ does not satisfy ballot secrecy, hence, there exists a probabilistic polynomial-time adversary $\mathcal{A}$, such that for all negligible functions negl, there exists a security parameter κ and

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

By definition of BS-0 and BS-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 17, 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 18, 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

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

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,

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

Thus,

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

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

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

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

Appendix C
Universal verifiability implies HB-Tally-Soundness

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

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

We previously omitted algorithm Verify, because we did not consider 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. And the formal definition of universal verifiability by Smyth, Frink & Clarkson [SFC17] requires algorithm Verify to accept if and only if the election outcome is correct.

The notion of a correct election outcome is captured using function *correct-outcome*, which is defined such that for all pk, nc, $\mathfrak{bb}$, κ, ℓ, and $v \in \{1, \ldots, nc\}$, we have $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)[v] = \ell$ iff $\exists^{=\ell} b \in \mathfrak{bb} \setminus \{\perp\} : \exists r : b = \mathsf{Vote}(pk, v, nc, \kappa; r)$,[40] and the vector produced by *correct-outcome* is of length nc. Hence, component v of vector $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)$ equals ℓ iff there exist ℓ ballots on the bulletin board that are votes for candidate v.

40. Function *correct-outcome* uses a *counting quantifier* [Sch05] denoted $\exists^{=}$. Predicate $(\exists^{=\ell} x : P(x))$ holds exactly when there are ℓ distinct values for x such that $P(x)$ is satisfied. Variable x is bound by the quantifier, whereas ℓ is free.

The function requires ballots to be interpreted for only one candidate, which can be ensured by injectivity.

Definition 20 (Injectivity [SFC17]). *An election scheme* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *satisfies* injectivity*, if for all security parameters* κ*, integers* nc*, and votes* v *and* v'*, such that* $v \neq v'$*, we have* $\Pr[(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa); b \leftarrow \mathsf{Vote}(pk, v, nc, \kappa); b' \leftarrow \mathsf{Vote}(pk, v', nc, \kappa) : b \neq \perp \wedge b' \neq \perp \Rightarrow b \neq b'] = 1$.

The *if* requirement of universal verifiability is captured by completeness (Definition 21), which stipulates that election outcomes produced by algorithm Tally will actually be accepted by algorithm Verify. And the *only if* requirement is captured by soundness (Definition 22), which challenges an adversary to concoct a scenario in which algorithm Verify accepts, but the election outcome is not correct. We take these definitions together to formulate universal verifiability (Definition 23).

Definition 21 (Completeness [SFC17]). *An election scheme* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *satisfies* completeness*, if there exists a negligible function* negl*, such that for all security parameters* κ*, bulletin boards* $\mathfrak{bb}$*, and integers* nc*, we have* $\Pr[(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa); (\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa) : |\mathfrak{bb}| \leq mb \wedge nc \leq mc \Rightarrow \mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa) = 1] > 1 - \mathsf{negl}(\kappa)$.

Definition 22 (Soundness [SFC17]). *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme satisfying injectivity,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and game* $\mathsf{Exp\text{-}UV\text{-}Ext}(\Gamma, \mathcal{A}, \kappa) = (pk, nc, \mathfrak{bb}, \mathfrak{v}, pf) \leftarrow \mathcal{A}(\kappa)$; **return** $\mathfrak{v} \neq \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa) \wedge \mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa) = 1$. *We say* Γ *satisfies soundness, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such that for all security parameters* κ*, we have* $\mathsf{Succ}(\mathsf{Exp\text{-}UV\text{-}Ext}(\Gamma, \mathcal{A}, \kappa)) \leq \mathsf{negl}(\kappa)$.

Definition 23 (UV [SFC17]). *An election scheme* Γ *satisfies* universal verifiability (UV)*, if completeness, injectivity and soundness are satisfied.*

We show that universally verifiable election schemes satisfy HB-Tally-Soundness (Proposition 20). This is useful to simplify applications of Theorem 10. Indeed, our proof that Helios'16 satisfies Ballot-Secrecy makes use of this result.

Lemma 19. *Given an election scheme* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally})$*, there exists a negligible function* negl*, such that for all security parameters* κ*, 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 κ 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. $\square$

Proposition 20 (UV $\Rightarrow$ HB-Tally-Soundness). *If election scheme* Γ *satisfies* UV*, then* Γ *satisfies* HB-Tally-Soundness.

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

$\mathcal{B}(\kappa) =$
$(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
$(v, nc, \mathfrak{bb}_0) \leftarrow \mathcal{A}(pk, \kappa)$;
$(\mathfrak{v}_0, pf_0) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}_0, nc, \kappa)$;
$\beta \leftarrow_R \{0, 1\}$;
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 $(pk, nc, \mathfrak{bb}_\beta, \mathfrak{v}_\beta, pf_\beta)$;

We prove that $\mathcal{B}$ wins UV with non-negligible probability.

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 $\mathfrak{v}^* \leftarrow (\mathfrak{v}_0[1], \dots, \mathfrak{v}_0[v-1], \mathfrak{v}_0[v]+1, \mathfrak{v}_0[v+1], \dots, \mathfrak{v}_0[|\mathfrak{v}_0|])$. 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 β is a bit chosen uniformly at random. By completeness, we have $\mathsf{Verify}(pk, \mathfrak{bb}_\beta, nc, \mathfrak{v}_\beta, pf_\beta, \kappa) = 1$, with overwhelming probability. Hence, it suffices to prove that $\mathfrak{v}_\beta \neq \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}_\beta, \kappa)$, with non-negligible probability. Let δ_0, respectively δ_1, be the probability that $\mathfrak{v}_0 \neq \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}_0, \kappa)$, respectively $\mathfrak{v}_1 \neq \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}_1, \kappa)$. It follows that $\mathsf{Succ}(\mathsf{UV}(\Gamma, \mathcal{B}, \kappa)) = \frac{1}{2} \cdot \delta_0 + \frac{1}{2} \cdot \delta_1$ and it remains to show that $\delta_0 + \delta_1$ is non-negligible. It suffices to prove that $\mathfrak{v}_0 = \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}_0, \kappa) \wedge \mathfrak{v}_1 = \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}_1, \kappa)$ is false with overwhelming probability.

Suppose $\mathfrak{v}_0 = \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}_0, \kappa)$. By definition of function $\mathit{correct\text{-}outcome}$, 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 19, 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 injectivity, b is not an output of $\mathsf{Vote}(pk, v', nc, \kappa)$ for all $v' \in \{1, \dots, |\mathfrak{v}^*|\} \setminus \{v\}$. Thus, for all $v' \in \{1, \dots, |\mathfrak{v}^*|\} \setminus \{v\}$ 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 $\mathfrak{v}^* = \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}_1, \kappa)$. Moreover, given that $\mathfrak{v}^* \neq \mathfrak{v}_1$, we have $\mathfrak{v}_1 \neq \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}_1, \kappa)$ with overwhelming probability, which suffices to conclude our proof. $\square$

Appendix D
Helios

Smyth, Frink & Clarkson [SFC17] formalise a generic construction for Helios-like election schemes (Definition 25), which is parameterised on the choice of homomorphic encryption scheme and sigma protocols for the relations introduced in the following definition.

Definition 24 (from [SFC17])**.** *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a homomorphic asymmetric encryption scheme and* Σ *be a sigma protocol for a binary relation* R.[41]

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

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

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

Definition 25 (Generalised Helios [SFC17])**.** *Suppose* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is an additively homomorphic asymmetric encryption scheme with a message space that, for sufficiently large security parameters, includes* $\{0, 1\}$*,* Σ_1 *proves correct key construction,* Σ_2 *proves plaintext knowledge in a subspace,* Σ_3 *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* $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$*, as follows.*[42]

- $\mathsf{Setup}(\kappa)$. *Select coins* s *uniformly at random, compute* $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa; s)$; $\rho \leftarrow \mathsf{ProveKey}((\kappa, pk, \mathfrak{m}), (sk, s), \kappa)$; $PK_{\mathcal{T}} \leftarrow (pk, \mathfrak{m}, \rho)$; $SK_{\mathcal{T}} \leftarrow (pk, sk)$*, let* m *be the largest integer such that* $\{0, \dots, m\} \subseteq \mathfrak{m}$*, and output* $(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m, m)$.
- $\mathsf{Vote}(PK_{\mathcal{T}}, v, nc, \kappa)$. *Parse* $PK_{\mathcal{T}}$ *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** $m_j \leftarrow 0$;
      $c_j \leftarrow \mathsf{Enc}(pk, m_j; r_j)$;
      $\sigma_j \leftarrow \mathsf{ProveCiph}((pk, c_j, \{0, 1\}), (m_j, r_j), j, \kappa)$;
  $c \leftarrow c_1 \otimes \cdots \otimes c_{nc-1}$;
  $m \leftarrow m_1 \odot \cdots \odot m_{nc-1}$;
  $r \leftarrow r_1 \oplus \cdots \oplus r_{nc-1}$;
  $\sigma_{nc} \leftarrow \mathsf{ProveCiph}((pk, c, \{0, 1\}), (m, r), nc, \kappa)$;

  Output ballot $(c_1, \dots, c_{nc-1}, \sigma_1, \dots, \sigma_{nc})$.
- $\mathsf{Tally}(SK_{\mathcal{T}}, \mathfrak{bb}, nc, \kappa)$. *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* $SK_{\mathcal{T}}$ *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 $(\mathfrak{v}, pf)$.
- $\mathsf{Verify}(PK_{\mathcal{T}}, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa)$. *Parse* $\mathfrak{v}$ *as a vector of length* nc*, parse* pf *as a vector of length* $nc - 1$*, parse* $PK_{\mathcal{T}}$ *as a vector* $(pk, \mathfrak{m}, \rho)$*. Output* 0 *if parsing fails or* $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) \neq 1$*. Let* $\{b_1, \dots, b_\ell\}$ *be the largest subset of* $\mathfrak{bb}$ *satisfying the conditions given by the tally algorithm and let* mb *be the largest integer such that* $\{0, \dots, mb\} \subseteq \mathfrak{m}$*. If* $\{b_1, \dots, b_\ell\} = \emptyset \wedge \bigwedge_{j=1}^{nc} \mathfrak{v}[j] = 0$ *or* $\bigwedge_{j=1}^{nc-1} \mathsf{VerDec}((pk, b_1[j] \otimes \cdots \otimes b_\ell[j], \mathfrak{v}[j]), pf[j], \kappa) = 1 \wedge \mathfrak{v}[nc] = \ell - \sum_{j=1}^{nc-1} \mathfrak{v}[j] \wedge 1 \leq \ell \leq mb$*, then output* 1*, otherwise, output* 0.

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

Instantiations of generalised Helios work as follows [SFC17].

- Setup generates the tallier's key pair. The public key includes a non-interactive proof demonstrating that the key pair is correctly constructed.
- Vote takes a vote $v \in \{1, \dots, nc\}$ and outputs 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. Vote also outputs proofs $\sigma_1, ..., \sigma_{nc}$ so that this can be verified. In particular, proof σ_j demonstrates ciphertext c_j contains 0 or 1, for all $1 \leq j \leq nc - 1$. And proof σ_{nc} demonstrates that the homomorphic combination of ciphertexts $c_1 \otimes \cdots \otimes c_{nc-1}$ contains 0 or 1. (It follows that the voter's ballot contains a vote for exactly one candidate.)
- Tally homomorphically combines ciphertexts representing votes for a particular candidate and decrypts the homomorphic combinations. The number of votes for a candidate $v \in \{1, \dots, nc-1\}$ is simply the homomorphic combination of ciphertexts representing votes for that

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 Verify for brevity.

candidate. The number of votes for candidate nc is equal to the number of votes for all other candidates subtracted from the total number of valid ballots on the bulletin board.

- Verify checks that each of the above steps has been performed correctly.

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

Definition 26 (Helios'16 [SFC17]). *Election scheme* Helios'16 *is* $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$, *where* Π *is additively homomorphic El Gamal [CGS97, §2],* Σ_1 *is the sigma protocol for proving knowledge of discrete logarithms by Chaum et al. [CEGP87, Protocol 2],* Σ_2 *is the sigma protocol for proving knowledge of disjunctive equality between discrete logarithms by Cramer et al. [CFSY96, Figure 1],* Σ_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 12. 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 construction, [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.

APPENDIX E
HELIOS SATISFIES BALLOT SECRECY

The construction for Helios-like schemes produces election schemes with zero-knowledge tallying proofs (Lemma 21) that satisfy universal verifiability [SFC17] and, thus, honest-ballot tally soundness (Proposition 20). They also satisfy ballot independence (Proposition 22). Hence, they satisfy ballot secrecy too (Theorem 10). We show that Helios'16 satisfies ballot secrecy.

Henceforth, we assume Π, Σ_1, Σ_2 and Σ_3 satisfy the preconditions of Definition 25, and $\mathcal{H}$ is a random oracle. Let $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ 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 21. *If* $(\mathsf{ProveDec}, \mathsf{VerDec})$ *is zero-knowledge, then* $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ *has zero-knowledge tallying proofs.*

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

Proposition 22. *Suppose* Π *is perfectly correct and satisfies* IND-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* IND-CVA.

Proof. By Theorem 13, the proof systems have extractors and simulators. Let SimProveKey be the simulator for $(\mathsf{ProveKey}, \mathsf{VerKey})$. And let ExtProveCiph be the extractor for $(\mathsf{ProveCiph}, \mathsf{VerCiph})$.

Let IND-CPA* be a variant of IND-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 $1 \le i \le |\mathbf{m_0}|$ 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 Π satisfies IND-CPA* [KL07, §10.2.2].

Suppose $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ does not satisfy IND-CVA. Hence, there exists a probabilistic polynomial-time adversary $\mathcal{A}$, such that for all negligible functions negl, there exists a security parameter κ and $1/2 + \mathsf{negl}(\kappa) < \mathsf{IND\text{-}CVA}(\Gamma, \mathcal{A}, \kappa)$. Since $\mathcal{A}$ is a winning adversary, we have $\mathcal{A}(PK_\mathcal{T}, \kappa)$ 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 IND-CPA* from $\mathcal{A}$:

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

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

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

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

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

$$\mathbf{Q} \leftarrow \Big(\Big(\big(pk, b_1[1], \{0,1\}\big), b_1[nc]\Big), \ldots, \Big(\big(pk, b_\ell[1], \{0,1\}\big), b_\ell[nc]\Big), \ldots, \Big(\big(pk, b_1[nc-1], \{0,1\}\big), b_1[2\cdot(nc-1)]\Big), \ldots, \Big(\big(pk, b_\ell[nc-1], \{0,1\}\big), b_\ell[2\cdot(nc-1)]\Big)\Big);$$
$\mathbf{W} \leftarrow \mathsf{ExtProveCiph}(\mathbf{H}, \mathbf{P}, \mathbf{Q});$
$\mathfrak{v} \leftarrow (\Sigma_{i=1}^{\ell}\mathbf{W}[i][1],\ \ldots,\ \Sigma_{i=\ell\cdot(nc-2)+1}^{\ell\cdot(nc-1)}\mathbf{W}[i][1],\ \ell - \Sigma_{j=1}^{nc-1}\mathfrak{v}[j]);$
$g \leftarrow \mathcal{A}(\mathfrak{v});$

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 ρ is an output of $\mathsf{SimProveKey}((\kappa, pk, \mathfrak{m}), \kappa)$. Let $PK_\mathcal{T} = (pk, \mathfrak{m}, \rho)$. Suppose (v_0, v_1, nc) is an output of $\mathcal{A}(PK_\mathcal{T}, \kappa)$. Since SimProveKey is a simulator for (ProveKey, VerKey), we have $\mathcal{B}$ simulates the challenger in IND-CVA to $\mathcal{A}(PK_\mathcal{T}, \kappa)$. In particular, $PK_\mathcal{T}$ is a triple containing a public key and corresponding message space generated Gen, and a (simulated) proof of correct construction. 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 IND-CVA to $\mathcal{A}(b)$ and $\mathcal{A}(\mathfrak{v})$, hence, $g = \beta$, with at least the probability that $\mathcal{A}$ wins IND-CVA, concluding our proof.

Claim 23. *Adversary $\mathcal{B}$'s computation of b is equivalent to computing b as $b \leftarrow \mathsf{Vote}(PK_\mathcal{T}, v_\beta, nc, \kappa)$.*

Proof of Claim 23. We have $PK_\mathcal{T}$ 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 SimProveKey is a simulator for (ProveKey, VerKey) and proofs output by ProveKey are indistinguishable from outputs of SimProveKey, we have $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) = 1$, with non-negligible probability. In addition, since $\mathcal{B}$ is a winning adversary, we have $v_0, v_1 \in \{1, \ldots, nc\}$, with non-negligible probability. It follows that $\mathsf{Vote}(PK_\mathcal{T}, v_\beta, nc, \kappa)$ does not output $\perp$, with non-negligible probability. Indeed, computation $b \leftarrow \mathsf{Vote}(PK_\mathcal{T}, v_\beta, nc, \kappa)$ is equivalent to the following. Select coins $r_1, \ldots, r_{nc-1}$ 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 β. Moreover, since $\odot$ is the addition operator in group $(\mathfrak{m}, \odot)$ and 0 is the identity element in that group, if $v_1 = nc$, then plaintext m computed by the above algorithm is $1-\beta$, otherwise, $m = 1 - \beta \odot \beta = 1$. Hence, the above algorithm is equivalent to selecting coins $r_1, \ldots, r_{nc-1}$ uniformly at random and computing:

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

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

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

can be simulated by

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

Hence, we conclude the proof of this claim.

Claim 24. *Adversary $\mathcal{B}$'s computation of $\mathfrak{v}$ is equivalent to computing $\mathfrak{v}$ as $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(SK_\mathcal{T}, \mathfrak{bb}, nc, \kappa)$, where $SK_\mathcal{T} = (pk, sk)$.*

Proof of Claim 24. Let $\{b_1, \ldots, b_\ell\}$ be the largest subset of $\mathfrak{bb}$ satisfying the conditions of algorithm Tally. It is trivial to see that the claim holds when $\{b_1, \ldots, b_\ell\} = \emptyset$, because $\mathfrak{v}$ is computed as a zero-filled vector of length nc in both cases. We prove the claim also holds when $\{b_1, \ldots, b_\ell\} \neq \emptyset$.

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 $1 \le i \le \ell$ and $1 \le j \le nc-1$ 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:

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

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

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

Since Π is a homomorphic encryption scheme, we have for all $1 \le j \le nc-1$ 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 constructed using coins chosen uniformly at random, we nevertheless have $\mathsf{Dec}(sk, b_1[j] \otimes \cdots \otimes b_\ell[j]) = m_{1,j} \odot \cdots \odot m_{\ell,j}$ with overwhelming probability, because Π is perfectly correct. It follows that $\mathfrak{v} = (m_{1,1} \odot \cdots \odot m_{\ell,1},\ \ldots,\ m_{1,nc-1} \odot \cdots \odot m_{\ell,nc-1},\ \ell - \sum_{j=1}^{nc-1}\mathfrak{v}[j])$, with overwhelming probability. Let mb be the largest integer such that $\{0,\ldots,mb\} \subseteq \mathfrak{m}$. Since $\mathcal{A}$ is a winning adversary, we have $\ell \le mb$. Moreover, since $m_{1,j},\ldots,m_{\ell,j} \in \{0,1\}$ for all $1 \le j \le nc-1$ and $\odot$ is the addition operator in group $(\mathfrak{m}, \odot)$, we have $m_{1,j} \odot \cdots \odot m_{\ell,j} = \sum_{i=1}^{\ell} m_{i,j}$, which suffices to conclude the proof of this claim. □

For Helios'16, encryption scheme Π is additively homomorphic El Gamal [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 13 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 13] still hold." We adopt the same position and assume that Theorem 13 is applicable.

Proof of Theorem 12. Helios'16 has zero-knowledge tallying proofs (Lemma 21), subject to the applicability of Theorem 13 to the sigma protocol underlying $(\mathsf{ProveDec}, \mathsf{VerDec})$. Moreover, since Helios'16 satisfies UV [SFC17], we have Helios'16 satisfies HB-Tally-Soundness (Proposition 20). Furthermore, since El Gamal satisfies IND-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 IND-CVA (Proposition 22). Hence, Helios'16 satisfies Ballot-Secrecy too (Theorem 10). □

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.

[Adi14] Ben Adida. Helios v4 Verification Specs. Helios documentation, http://documentation.heliosvoting.org/verification-specs/helios-v4 (accessed 19 Jan 2017), 2014. A snapshot of the specification on 5 May 2016 is available from https://web.archive.org/web/20160505163104/http://documentation.heliosvoting.org/verification-specs/helios-v4..

[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.

[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 http://admin.cdn.sos.ca.gov/press-releases/prior/2007/DB07_111.pdf (accessed 19 Jan 2017), August 2007. A snapshot of the press release on 6 February 2008 is available from https://web.archive.org/web/20080206210142/http://www.sos.ca.gov/elections/voting_systems/ttbr/db07_042_ttbr_system_decisions_release.pdf..

[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. A snapshot of the chapter on 21 Mar 2015 is available from https://web.archive.org/web/20150321170845/http://cseweb.ucsd.edu/~mihir/cse207/w-se.pdf..

[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.

[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.

[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.

[CF85] Josh Daniel Cohen and Michael J. Fischer. A Robust and Verifiable Cryptographically Secure Election Scheme. In *FOCS'85: 26th Symposium on Foundations of Computer Science*, pages 372–382. IEEE Computer Society, 1985.

[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.

[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. To appear.

[Dam10] Ivan Damgård. On Σ-protocols, 2010. Available from http://www.daimi.au.dk/~ivan/Sigma.pdf.

[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.

[DJN10] Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. A Generalization of Paillier's Public-Key System with Applications to Electronic Voting. *International Journal of Information Security*, 9(6):371–385, 2010.

[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.

[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.

[Hir10] Martin Hirt. Receipt-Free K-out-of-L Voting Based on ElGamal Encryption. In David Chaum, Markus Jakobsson, Ronald L. Rivest, and Peter Y. A. Ryan, editors, *Towards Trustworthy Elections: New Directions in Electronic Voting*, volume 6000 of *LNCS*, pages 64–82. Springer, 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.

[HL10] Carmit Hazay and Yehuda Lindell. Sigma protocols and efficient zero-knowledge. In *Efficient Secure Two-Party Protocols*, Information Security and Cryptography, chapter 6, pages 147–175. Springer Berlin Heidelberg, 2010.

[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.

[HS00] Martin Hirt and Kazue Sako. Efficient Receipt-Free Voting Based on Homomorphic Encryption. In *EUROCRYPT'06: 25th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 1807 of *LNCS*, pages 539–556. Springer, 2000.

[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. See also http://www.rsa.com/rsalabs/node.asp?id=2860.

[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.

[KTV12] 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.

[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.

[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.

[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.

[MS16] Maxime Meyer and Ben Smyth. An attack against the helios election system that violates eligibility. arXiv, Report 1612.04099, 2016.

[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.

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

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

[OSC90] 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.

[Per16] Olivier Pereira. Internet Voting with Helios. In *Real-World Electronic Voting: Design, Analysis and Deployment*, chapter 11. CRC Press, 2016.

[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.

[QS16] Elizabeth A. Quaglia and Ben Smyth. Constructing secret, verifiable auction schemes from election schemes. Cryptology ePrint Archive, Report 2015/1204 (version 20160524:130412), 2016.

[QS17] Elizabeth A. Quaglia and Ben Smyth. A short introduction to secrecy and verifiability for elections. arXiv, Report 1702.03168, 2017.

[SAR13] Ben Smyth, Myrto Arapinis, and Mark D Ryan. Translating between equational theories for automated reasoning. FCS'13: Workshop on Foundations of Computer Security, 2013.

[SB13a] 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.

[SB13b] Ben Smyth and David Bernhard. Ballot secrecy and ballot independence coincide. Cryptology ePrint Archive, Report 2013/235 (version 20130618:102144), 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. http://hal.inria.fr/inria-00599182/.

[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.

[SHM15] Ben Smyth, Yoshikazu Hanatani, and Hirofumi Muratani. NM-CPA secure encryption with proofs of plaintext knowledge. In *IWSEC'15: 10th International Workshop on Security*, volume 9241 of *LNCS*, pages 115–134. Springer, 2015.

[SK94] Kazue Sako and Joe Kilian. Secure Voting Using Partially Compatible Homomorphisms. In *CRYPTO'94: 14th International Cryptology Conference*, volume 839 of *LNCS*, pages 411–424. Springer, 1994.

[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. Technical report, 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.

[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.

[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.

[UMQ10] 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] 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 Standalone 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.