Secrecy and independence for election schemes

Ben Smyth
Mathematical and Algorithmic Sciences Lab, Huawei Technologies Co. Ltd., France

July 13, 2016

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.

Keywords. Elections, Helios, independence, non-malleability, privacy, provable security, secrecy, voting.

1 Introduction
An election is a decision-making procedure to choose representatives. Choices should be made freely, and this has started a movement towards voting as a secret act. The 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\footnote{Ballot secrecy and privacy occasionally appear as synonyms in the literature. We favour ballot secrecy because it avoids confusion with other privacy notions, such as receipt-freeness and coercion resistance, for example.} as a \textit{de facto} standard requirement of voting systems.

- \textbf{Ballot secrecy.} A voter’s vote is not revealed to anyone.

Many voting systems – including systems that have been deployed in real-world, large-scale public 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, Bov07, WWH\textsuperscript{+}10, WWIH12, SFD\textsuperscript{+}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 \textit{et al.} [BCP\textsuperscript{+}11, BPW12b, SB13, SB14, BCG\textsuperscript{+}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 \textit{et al.} only consider trusted bulletin boards and channels.

Ballot independence [Gen95, CS13, CGMA85] is seemingly related to ballot secrecy.

- \textbf{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.\footnote{Smyth \& Bernhard [SB13, §5.1] critique the results by Desmedt \& Chaidos [DC12] and argue that their security results do not support their claims.} 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. And our second definition 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], a web-based voting system that has been deployed in the real-world. The original Helios scheme is vulnerable to attacks against ballot secrecy [CS13, CS11, SC11]. And the current version of Helios is intended to mitigate against those attacks. Bernhard [Ber14] and Bernhard et al. [BCG+15a, BCG+15b] prove that variants of the current version satisfy 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.

**Contribution.** This paper contributes to the security of voting systems by: proposing definitions of ballot secrecy (§3) and ballot independence (§4) 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 (§5); and identifying a new attack against Helios, proposing a fix, and proving that ballot secrecy is satisfied when the fix is incorporated (§6).

## 2 Election schemes

We recall syntax for election schemes from Smyth, Frink & Clarkson [SFC16].

**Definition 1** (Election scheme [SFC16]). An election scheme is a tuple of probabilistic polynomial-time algorithms (Setup, Vote, Tally) such that:

\[\text{Setup, denoted } (pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa), \text{ is run by the tallier}.\]

\[\text{Setup takes a security parameter } \kappa \text{ as input and outputs a key pair } pk, sk, \text{ a maximum number of ballots } mb, \text{ and a maximum number of candidates } mc.\]

3Election schemes capture an interesting class of voting systems, which includes Helios.

4We omit algorithm Verify from our syntax and we omit the condition that election schemes must satisfy notions of completeness and injectivity, because we do not focus on verifiability in the main body.

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

6Some 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.
Vote, denoted \( b \leftarrow \text{Vote}(pk, v, nc, \kappa) \), 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 \( \kappa \). A voter’s vote should be selected from a sequence \( 1, \ldots, nc \) of distinct candidates. Vote outputs a ballot \( b \) or error symbol \( \perp \).

Tally, denoted \((v, pf) \leftarrow \text{Tally}(sk, bb, nc, \kappa)\), is run by the tallier. Tally takes as input a private key \( sk \), a bulletin board \( bb \), some number of candidates \( nc \), and a security parameter \( \kappa \), where \( bb \) is a set. It outputs an election outcome \( v \) and a non-interactive tallying proof \( pf \) (i.e., a proof that the outcome is correct). An election outcome is a vector \( v \) of length \( nc \) such that \( v[v] \) indicates the number of votes for candidate \( v \).

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

\[
\Pr[(pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa); \text{for } 1 \leq i \leq nb \text{ do } \begin{align*}
    b_i &\leftarrow \text{Vote}(pk, v_i, nc, \kappa); \\
    v[v_i] &\leftarrow v[v_i] + 1; \\
    (v', pf) &\leftarrow \text{Tally}(sk, \{b_1, \ldots, b_{nb}\}, nc, \kappa) : \\
    nb \leq mb \land nc \leq mc \Rightarrow v = v' > 1 - \text{negl}(\kappa).
\end{align*}
\]

3 Ballot Secrecy

Our informal definition of ballot secrecy (§1) 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.

\[\text{Let } v[v] \text{ denote component } v \text{ of vector } v.\]
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, ballot secrecy must concede that election outcomes reveal partial information about voters’ votes, 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.

### 3.1 Indistinguishability game

We formalise ballot secrecy as an indistinguishability game between an adversary and a challenger.\(^9\)

**Definition 2 (Ballot-Secrecy).** Let \( \Gamma = (\text{Setup}, \text{Vote}, \text{Tally}) \) be an election scheme, \( A \) be an adversary, \( \kappa \) be a security parameter, and \( \text{Ballot-Secrecy}(\Gamma, A, \kappa) \) be the following game.\(^{10}\)

\[
\text{Ballot-Secrecy}(\Gamma, A, \kappa) = (pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa);
nc \leftarrow A(pk, \kappa);
\beta \leftarrow \mathbb{R}\{0, 1\};
L \leftarrow \emptyset;
bb \leftarrow A^O();
(v, pf) \leftarrow \text{Tally}(sk, bb, nc, \kappa);
g \leftarrow A(v, pf);
\text{return } g = \beta \land \text{balanced}(bb, nc, L) \land 1 \leq nc \leq mc \land |bb| \leq mb;
\]

Predicate \( \text{balanced}(bb, nc, L) \) holds when: for all votes \( v \in \{1, \ldots, nc\} \) we have \( |\{b \mid b \in bb \land \exists v_1 . (b, v, v_1) \in L\}| = |\{b \mid b \in bb \land \exists v_0 . (b, v_0, v) \in L\}|. \) And oracle \( O \) is defined as follows.\(^{11}\)

---

\(^8\)We acknowledge that 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.

\(^9\)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 \( \text{Succ}(\text{Exp}()) \) in a game \( \text{Exp}() \) as the probability that the adversary wins, that is, \( \text{Succ}(\text{Exp}()) = \Pr[b \leftarrow \text{Exp}() : b = \top] \). 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.

\(^{10}\)Let \( x \leftarrow \mathbb{R} S \) denote assignment to \( x \) of an element chosen uniformly at random from set \( S \). And let \( |v| \) denote the length of vector \( v \).

\(^{11}\)Oracles may access game parameters, e.g., \( pk \).
3 BALLOT SECRECY

- $O(v_0, v_1)$ computes $b \leftarrow \text{Vote}(pk, v_0, v_1)$; $L \leftarrow L \cup \{(b, v_0, v_1)\}$ and outputs $b$, where $v_0, v_1 \in \{1, \ldots, nc\}$.

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

The game captures a setting where 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 [BDJR97, BR05] which can compute ballots on the adversary’s behalf. Ballots can be computed by the left-right oracle in two ways, corresponding to a bit $\beta$ chosen uniformly at random by the challenger. If $\beta = 0$, then, given a pair of votes $v_0, v_1$, the oracle 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 is satisfiable by election schemes which reveal the number of votes for each candidate (i.e., the election outcome). Hence, to avoid trivial distinctions, we insist the game is balanced: “left” and “right” inputs to the left-right oracle are equivalent, when the corresponding left-right oracle’s 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. 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.

3.2 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
Definition 3 (Enc2Vote [QS16]). Given an asymmetric encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$, we define $\text{Enc2Vote}(\Pi)$ as follows.

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

Algorithm **Setup** requires poly to be a polynomial function and $m = \{1, \ldots, |m|\}$. Algorithm **Tally** requires $\epsilon$ to be a constant symbol.

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

The proof of Lemma 1 appears in [QS16, §C.4].

Intuitively, given a non-malleable asymmetric encryption scheme $\Pi$, the construction $\text{Enc2Vote}(\Pi)$ derives ballot secrecy from $\Pi$ 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 $\text{Enc2Vote}(\Pi)$ satisfies a stronger notion of ballot secrecy [QS16, Proposition 5 & 16], hence, $\text{Enc2Vote}(\Pi)$ satisfies our notion of ballot secrecy too.

Corollary 2. Let $\Pi$ be an encryption scheme with perfect correctness. If $\Pi$ satisfies IND-PA0, then election scheme $\text{Enc2Vote}(\Pi)$ satisfies Ballot-Secrecy.

The reverse implication of Corollary 2 does not hold.

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

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

---

12 The construction by Quaglia & Smyth builds upon constructions by Bernhard et al. [SB14, SB13, BPW12b, BCP+11].
13 We define asymmetric encryption and an associated security definition (namely, IND-PA0) in Appendix A.1.
14 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 $\text{Enc2Vote}(\Pi)$ satisfies injectivity. We adopt the same assumption to capitalise upon their results.
4 Ballot independence

Our informal definition of ballot independence (§1) 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.

4.1 Non-malleability game

We formalise ballot independence as a non-malleability game.

**Definition 4 (CNM-CVA).** Let $\Gamma = (\text{Setup}, \text{Vote}, \text{Tally})$ be an election scheme, $A$ be an adversary, $\kappa$ be a security parameter, and $\text{cnn-cva}(\Gamma, A, \kappa)$ and $\text{cmm-cva-$(} \Gamma, A, \kappa)$ be the following games.\(^\text{15}\)

- $\text{cnn-cva}(\Gamma, A, \kappa) = (pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa); (V, nc) \leftarrow A(pk, \kappa); v \leftarrow RV; b \leftarrow \text{Vote}(pk, v, nc, \kappa); (R, bb) \leftarrow A(b); (v, pf) \leftarrow \text{Tally}(sk, bb, nc, \kappa); \text{return } R(v, v) \land b \notin bb \land V \subseteq \{1, \ldots, nc\} \land 1 \leq nc \leq mc \land |bb| \leq mb;$

- $\text{cmm-cva-$(}(\Gamma, A, \kappa) = (pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa); (V, nc) \leftarrow A(pk, \kappa); v, v' \leftarrow RV; b \leftarrow \text{Vote}(pk, v', nc, \kappa); (R, bb) \leftarrow A(b); (v, pf) \leftarrow \text{Tally}(sk, bb, nc, \kappa); \text{return } R(v, v) \land b \notin bb \land V \subseteq \{1, \ldots, nc\} \land 1 \leq nc \leq mc \land |bb| \leq mb;$

In the above games, we insist that relation $R$ is computable in polynomial time. We say $\Gamma$ satisfies comparison based non-malleability under chosen vote attack (CNM-CVA), if for all probabilistic polynomial-time adversaries $A$, there exists a negligible function $\text{negl}$, such that for all security parameters $\kappa$, we have $\text{Succ}(\text{cnn-cva}(\Gamma, A, \kappa)) - \text{Succ}(\text{cmm-cva-$(}(\Gamma, A, \kappa)) \leq \text{negl}(\kappa)$.\(^\text{15}\)

\(^{15}\)We abbreviate $x \leftarrow R S; x' \leftarrow R S$ as $x, x' \leftarrow R S$. 
Similarly to game Ballot-Secrecy, games \texttt{cnm-cva} and \texttt{cnm-cva-$\dollar$} capture: key generation using algorithm \texttt{Setup}, publication of the public key, and only using the private key to compute the election outcome and tallying proof.

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

\texttt{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 \( v \), respectively \( v' \), in \texttt{cnm-cva}, respectively \texttt{cnm-cva-$\dollar$}, this adversary could output a bulletin board containing only the challenge ballot and a relation \( R \) such that \( R(v, v) \) holds if \( v[v] = 1 \), hence, the relation evaluated in \texttt{cnm-cva} holds, whereas the relation evaluated in \texttt{cnm-cva-$\dollar$} 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.

\textbf{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.

\subsection{Indistinguishability game}

We formalise an alternative definition of ballot independence as an indistinguishability game.
Definition 5 (IND-CVA). Let \( \Gamma = (\text{Setup}, \text{Vote}, \text{Tally}) \) be an election scheme, \( A \) be an adversary, \( \kappa \) be the security parameter, and IND-CVA(\( \Gamma, A, \kappa \)) be the following game.

\[
\text{IND-CVA}(\Gamma, A, \kappa) =
\begin{align*}
& (pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa); \\
& (v_0, v_1, nc) \leftarrow A(pk, \kappa); \\
& \beta \leftarrow R \{0, 1\}; \\
& b \leftarrow \text{Vote}(pk, v_\beta, nc, \kappa); \\
& bb \leftarrow A(b); \\
& (v, pf) \leftarrow \text{Tally}(sk, bb, nc, \kappa); \\
& g \leftarrow A(p); \\
\text{return } g = \beta \land b \notin bb \land 1 \leq v_0, v_1 \leq nc \leq mc \land |bb| \leq mb;
\end{align*}
\]

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

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 \( b' \).

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

4.3 Equivalence between games

Our ballot independence games are adaptations of standard security definitions for asymmetric encryption: CNM-CVA is based on non-malleability for encryption and IND-CVA is based on indistinguishability for encryption. Bellare & Sahai [BS99] have shown that non-malleability is equivalent to indistinguishability for encryption and their proof can be adapted to show that CNM-CVA and IND-CVA are equivalent.

Theorem 4 (CNM-CVA = IND-CVA). Given an election scheme \( \Gamma \), we have \( \Gamma \) satisfies CNM-CVA iff \( \Gamma \) satisfies IND-CVA.
4.4 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 our construction Enc2Vote using encryption schemes satisfying CNM-CPA.\(^\text{16}\)

**Corollary 5.** Let \(\Pi\) be an encryption scheme with perfect correctness. If \(\Pi\) satisfies CNM-CPA, then election scheme Enc2Vote(\(\Pi\)) satisfies CNM-CVA.

The 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 \(\Pi\) such that election scheme Enc2Vote(\(\Pi\)) satisfies CNM-CVA, but \(\Pi\) does not satisfy CNM-CPA.

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

5 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.\(^\text{17}\) Hence, non-malleable ballots are necessary in election schemes satisfying ballot secrecy.

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

\(^{16}\)Bellare & Sahai [BS99, §5] show that IND-PA0 coincides with CNM-CPA, thus it suffices to consider IND-PA0 in Corollaries 5 & 6.

\(^{17}\)Smyth & Bernhard explain that alternative formalisations of election schemes might permit different results [SB13, §5.2].
Proposition 8 (IND-CVA $\not\Rightarrow$ Ballot-Secrecy). There exists an election scheme $\Gamma$ such that $\Gamma$ satisfies IND-CVA, but not Ballot-Secrecy.

The 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. We show the condition is implied by universal verifiability in Appendix C.)

Definition 6. An election scheme $\Gamma = (\text{Setup}, \text{Vote}, \text{Tally})$ is without tallying proofs, if there exists a constant symbol $\epsilon$ such that for all multisets $bb$ we have:

$$\Pr[(pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa) ; (v, pf) \leftarrow \text{Tally}(sk, bb, nc, \kappa) : pf = \epsilon] = 1.$$ 

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

$$\text{Soundness}(\Gamma, A, \kappa) =$$

$$(pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa); (v, nc, bb_0) \leftarrow A(pk, \kappa); b \leftarrow \text{Vote}(pk, v, nc, \kappa);$$

$$(v_0, pf_0) \leftarrow \text{Tally}(sk, bb_0, nc, \kappa); (v_1, pf_1) \leftarrow \text{Tally}(sk, bb_0 \cup \{b\}, nc, \kappa);$$

$$v^* \leftarrow (v_0[1], \ldots, v_0[v-1], v_0[v]+1, v_0[v+1], \ldots, v_0[|v_0|]);$$

return $v^* \neq v_1 \land b \notin bb_0 \land 1 \leq v \leq nc \leq mc \land |bb_0 \cup \{b\}| \leq mb$;

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

Proposition 9 (Ballot-Secrecy $=$ IND-CVA, without tallying proofs). Let $\Gamma$ be an election scheme without tallying proofs. Suppose $\Gamma$ satisfies Soundness. We have $\Gamma$ satisfies Ballot-Secrecy iff $\Gamma$ 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 $\Gamma = (\text{Setup}, \text{Vote}, \text{Tally})$ be an election scheme. We say $\Gamma$ has zero-knowledge tallying proofs, if there exists a zero-knowledge non-interactive proof system $(\text{Prove}, \text{Verify})$ and for all probabilistic polynomial-time adversaries $\mathcal{A}$ and security parameters $\kappa$, we have:
\[
\Pr[(pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa);
\]
\[
(nc, bb) \leftarrow \mathcal{A}(pk, \kappa);
\]
\[
(v, pf) \leftarrow \text{Tally}(sk, bb, nc, \kappa);
\]
\[
|bb| \leq mb \land nc \leq mc \Rightarrow \exists w, r. \text{Prove}((pk, bb, nc, v), w, \kappa; r) = pf = 1.
\]

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

## 6 Case Study: Helios

Helios is an open-source, web-based electronic voting system,\(^{18}\) which has been used in the real-world: the International Association of Cryptologic Research (IACR) has used Helios annually since 2010 to elect board members [BVQ10, HBH10],\(^{19}\) the Catholic University of Louvain used Helios to elect their university president in 2009 [AMPQ09], and Princeton University has used Helios since 2009 to elect student governments [Adi09].\(^{20}\)

Informally, Helios can be modelled as an election scheme \((\text{Setup}, \text{Vote}, \text{Tally})\) 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,\(^{21}\) 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.**\(^{22}\)

The original Helios scheme [AMPQ09] is vulnerable to attacks against ballot secrecy [CS13, CS11, SC11]. The current version of Helios is intended to mitigate against these attacks.\(^{23}\) In particular, it incorporates Smyth’s recommendation

---


\(^{19}\)https://www.iacr.org/elections/, accessed 1 Sep 2015.


\(^{21}\)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].


to reject ballots containing zero-knowledge proofs that have been previously observed [Smy12, §4]. For clarity, we write Helios 3.1.4 for the current version of Helios. Bernhard [Ber14, §6.11] and Bernhard et al. [BCG+15a, §D.3] show that variants of Helios 3.1.4 using the Fiat-Shamir heuristic (rather than the weak Fiat–Shamir transformation, which does not include statements in hashes) satisfy notions of ballot secrecy.24 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 3.1.4 uses malleable ballots (as do the variants studied by Bernhard [Ber14] and Bernhard et al. [BCG+15a], and as does the variant defined by the specification for the next Helios release [Adi14]), which are incompatible with ballot secrecy (§5).

Theorem 11. Helios 3.1.4 does not satisfy Ballot-Secrecy.

Proof sketch. Suppose an adversary queries the left-right oracle with inputs $v_0$ and $v_1$ to derive a ballot for $v_β$, 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\}.25 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 $v$: if $v[v_0] = 1$, then output 0, otherwise, output 1. Since $b$ is a ballot for $v_β$, it follows by correctness that $v[v_0] = 1$ iff $β = 0$, and $v[v_1] = 1$ iff $β = 1$, hence, the adversary wins the game.

Our informal proof of Theorem 11 is straightforward. A formal proof would require a formal description of Helios 3.1.4. Such a formal description can be derived by adapting the formalisation of the original Helios scheme by Smyth, Frink & Clarkson [SFC16] to reject ballots containing zero-knowledge proofs that have been previously observed. These details provide little value, so we do not pursue them further.

The proof sketch of Theorem 11 does not immediately give way to an attack against Helios. Nevertheless, we can derive an 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 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 $v$. (Thus, we exclude 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

24Bernhard, Pereira & Warinschi [BPW12a] consider whether Helios satisfies a notion of ballot secrecy in two candidate elections.
25The recorded-as-cast assumption is violated because the ballot output by the left-right oracle does not appear on the bulletin board.
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 $\{b, b_2, b_3\}$. If the outcome does not contain any votes for $v'$, then Mallory is convinced that Alice did not vote for $v$. This attack also works against the variants of Helios 3.1.4 studied by Bernhard and Bernhard et al.. Neither Bernhard [Ber14, §6.11] nor Bernhard et al. [BCG+15a, §D.3] were able to detect the attack,\footnote{Nor could the attack have been detected by Bernhard, Pereira & Warinschi [BPW12a].} because interception is not possible when ballots are recorded-as-cast.\footnote{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.}

We have seen that non-malleable ballots are necessary for ballot secrecy (§5), hence, future Helios releases should adopt non-malleable ballots. Smyth, Frink & Clarkson [SFC16] 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 can 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 [SFC16], it is also satisfies 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) underlying Helios’16. \(\Box\)

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 [SFC16], 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.

### 7 Related work

Discussion of ballot secrecy originates from Chaum [Cha81] and the earliest definitions of ballot secrecy are due to Benaloh et al. [BY86, BT94, Ben96].\footnote{Bernhard et al. [BCG+15b, BCG+15a] survey ballot secrecy definitions.} 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, SB13, SB14, BCG+15b]. The definition of ballot secrecy by Bernhard, Pereira & Warinschi computes tallying proofs using algorithm Tally or a simulator [BPW12a], but the resulting definition is too weak [BCG+15b, §3.4] and some strengthening is required [BCG+15b, §4]. (Cortier et al. [CGGI13a, CGGI13b] propose a variant...
of the ballot secrecy definition by Bernhard, Pereira & Warinschi. That vari-
ant is also too weak [BCG+15b].) By comparison, the definition by Smyth 
& Bernhard computes tallying proofs using only algorithm Tally [SB13], but 
the resulting definition is too strong [BCG+15b, §3.5] and a weakening is re-
quired [SB14]. The relative merits of ballot secrecy definitions due to Smyth & 
Bernhard [SB14, Definition 5] and Bernhard et al. [BCG+15b, Definition 7] 
are unknown, in particular, it is unknown whether one definition is stronger 
than the other.

Discussion of ballot independence originates from Gennaro [Gen95]. And the 
apparent relationship between ballot secrecy and ballot independence has been 
considered. 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 gener-
ally, Sako & Kilian [SK95, §2.4], Michels & Horster [MH96, §3] and Cortier & 
Smyth [CS13, CS11] discuss how malleable ballots can be exploited to compro-
mise ballot secrecy. The first definition of ballot independence seems to be due to 
Smyth & Bernhard [SB13, SB14]. Moreover, Smyth & Bernhard formally prove 
relations between their definitions of ballot secrecy and ballot independence.

All of the ballot secrecy definitions by Bernhard et al. [BCP+11, BPW12b, 
BPW12a, SB13, SB14, BCG+15b] and the ballot independence definition by 
Smyth & Bernhard [SB13, SB14] focus on detecting 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 [SB13, §4.3], and Bern-
hard [Ber14] and Bernhard et al. [BCG+15a, BCG+15b] prove that variants of 
Helios 3.1.4 satisfy notions of ballot secrecy, despite the use of malleable ballots. 
By comparison, we prove that non-malleable ballots are necessary for ballot se-
crecy without this trust assumption. Hence, the aforementioned variants of 
Helios 3.1.4 do 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 defi-
nitions by Bernhard et al. were identified by Smyth [Smy14]. And Definition 2 
is based upon the definition of ballot secrecy proposed by Smyth [Smy14, Def-
nition 3]. The main distinction between Definition 2 and the definition by 
Smyth is syntax: this paper adopts syntax for election schemes from Smyth, 
Frink & Clarkson [SFC16], whereas, Smyth adopts syntax by Smyth & Bern-
hard [SB14, SB13]. The change in syntax is motivated by the superiority of 
syntax by Smyth, Frink & Clarkson. Unfortunately, the change has a drawback: 
we cannot immediately prove that the definition of ballot secrecy proposed in 
this paper is strictly stronger than the definition proposed by Smyth & Bern-
hard [SB14, SB13]. By comparison, the technical reports contain such proofs. Nevertheless, the advantages of the syntax change outweigh the disadvantages. Moreover, we can capitalise upon results by Smyth, Frink & Clarkson [SFC16] and Quaglia & Smyth [QS16].

8 Conclusion

This work was initiated by a desire to eliminate the trust assumptions placed upon the bulletin board and the communication channel in definitions of ballot secrecy by Bernhard et al. and the definition of ballot independence by Smyth & Bernhard. This necessitated the introduction of new security definitions.

The definition of ballot secrecy was largely constructed from intuition, with inspiration from indistinguishability games for asymmetric encryption and existing definitions of ballot secrecy. Moreover, the definition was guided by the desire to strengthen existing definitions of ballot secrecy.

The definition of ballot independence was inspired by the realisation that independence requires non-malleable ballots. This enabled definitions of ballot independence to be constructed as straightforward adaptations of non-malleability and indistinguishability definitions for asymmetric encryption; the former adaptation being a more natural formulation of ballot independence and the latter being simpler.

Relationships between security definitions aid our understanding and offer insights that facilitate the construction of secure election schemes. This prompted the study of relations between 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 their definitions.

We exploit our results to prove that a variant of Helios satisfies ballot secrecy. This proof is particularly significant due to the use of Helios in the real-world. 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 3) 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. This work was performed in part at INRIA, with support from the European Research Council under the European Union’s Seventh Framework Programme (FP7/2007-2013) / ERC project CRYSP (259639).

A Cryptographic primitives

A.1 Asymmetric encryption

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

- Gen, denoted (pk, sk, m) ← Gen(κ), inputs a security parameter κ and outputs a key pair (pk, sk) and message space m.
- Enc, denoted c ← Enc(pk, m), inputs a public key pk and message m ∈ m, and outputs a ciphertext c.
- Dec, denoted m ← 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, m) ← Gen(κ); c ← Enc(pk, m) : m ∈ m ⇒ Dec(sk, c) = m] > 1 − negl(κ). A scheme has perfect correctness if the probability is 1.

Definition 10 (Homomorphic encryption [SFC16]). An asymmetric encryption scheme Γ = (Gen, Enc, Dec) is homomorphic, with respect to ternary operators ⊙, ⊕, and ⊗,30 if there exists a negligible function negl, such that for all security parameters κ, we have the following.31 First, for all messages m1 and m2 we have Pr[(pk, sk, m) ← Gen(κ); c1 ← Enc(pk, m1); c2 ← Enc(pk, m2) : m1, m2 ∈ m ⇒ Dec(sk, c1 ⊕ pk Dec(sk, c2)) > 1 − negl(κ)). Secondly, for all messages m1 and m2, and all coins r1 and r2, we have Pr[(pk, sk, m) ← Gen(κ) : m1, m2 ∈ m ⇒ Enc(pk, m1; r1) ⊕ pk Enc(pk, m2; r2) = Enc(pk, m1 ⊕ sk, m2)}]

29Our definition differs from Katz and Lindell’s original definition [KL07, Definition 10.1] in that we formally state the plaintext space.

30Henceforth, 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 ⊙, ⊕, and ⊗.

31We write X ⊕ pk Y for the application of ternary operator ⊕ to inputs X, Y, and pk. We occasionally abbreviate X ⊕ pk Y as X ⊕ Y, when pk is clear from the context.
A CRYPTOGRAPHIC PRIMITIVES

Let \( r_1 \oplus_{pk} r_2 \) be a random variable independent of \( m \). We say \( \Gamma \) is additively homomorphic, if for all security parameters \( \kappa \), key pairs \( pk, sk \), and message spaces \( m \), such that there exists coins \( r \) and \((pk, sk, m) = \text{Gen}(\kappa; r)\), we have \( \odot_{pk} \) is the addition operator in group \((m, \odot_{pk})\).

**Definition 11 (IND-CPA [BDPR98]).** Let \( \Pi = (\text{Gen, Enc, Dec}) \) be an asymmetric encryption scheme, \( A \) be an adversary, \( \kappa \) be the security parameter, and IND-CPA(\( \Pi, A, \kappa \)) be the following game.\(^{32}\)

\[
\Pi = (pk, sk, m) \leftarrow \text{Gen}(\kappa);
\beta \leftarrow R \{0, 1\};
c \leftarrow \text{Enc}(pk, m_\beta);
g \leftarrow A(c);
\]

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

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

\[
\Pi = (pk, sk, m) \leftarrow \text{Gen}(\kappa);
\beta \leftarrow R \{0, 1\};
c \leftarrow \text{Enc}(pk, m_\beta);
m \leftarrow (\text{Dec}(sk, c[1]), \ldots, \text{Dec}(sk, c[|c|]));
g \leftarrow A(m);
\]

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

### A.2 Proof systems

**Definition 13 (Sigma protocol [SFC16, Dam10, HL10]).** A sigma protocol for a relation \( R \) is a tuple \((\text{Comm, Chal, Resp, Verify})\) of probabilistic polynomial-time algorithms such that:

\(^{32}\)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.
• **Comm**, denoted $\text{Comm}(s,w,\kappa) \leftarrow \text{Comm}(s,w,\kappa)$, is executed by a prover. Comm takes a statement $s$, witness $w$ and security parameter $\kappa$ as input, and outputs a commitment $\text{comm}$ and some state information $t$.

• **Chal**, denoted $\text{chal} \leftarrow \text{Chal}(s,\text{comm},\kappa)$, is executed by a verifier. Chal takes a statement $s$, a commitment $\text{comm}$ and a security parameter $\kappa$ as input, and outputs a string $\text{chal}$.

• **Resp**, denoted $\text{resp} \leftarrow \text{Resp}(\text{chal},t,\kappa)$, is executed by a prover. Resp takes a challenge $\text{chal}$, state information $t$ and security parameter $\kappa$ as input, and outputs a response $\text{resp}$.

• **Verify**, denoted $v \leftarrow \text{Verify}(s,(\text{comm},\text{chal},\text{resp}),\kappa)$ is executed by a verifier. Verify takes a statement $s$, a transcript $(\text{comm},\text{chal},\text{resp})$ and a security parameter $\kappa$ 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 $\text{negl}$, such that for all statements and witnesses $(s,w) \in R$ and security parameters $\kappa$, we have $\Pr[(\text{comm},t) \leftarrow \text{Comm}(s,w,\kappa); \text{chal} \leftarrow \text{Chal}(s,\text{comm},\kappa); \text{resp} \leftarrow \text{Resp}(\text{chal},t,\kappa) : \text{Verify}(s,(\text{comm},\text{chal},\text{resp}),\kappa) = 1] > 1 - \text{negl}(\kappa)$.

### A.3 Non-interactive proof systems

**Definition 14** (Non-interactive proof system [SFC16]). A non-interactive proof system for a relation $R$ is a tuple of algorithms $(\text{Prove},\text{Verify})$, such that:

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

• **Verify**, denoted $v \leftarrow \text{Verify}(s,\sigma,\kappa)$, 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 $\text{negl}$, such that for all statements and witnesses $(s,w) \in R$ and security parameters $\kappa$, we have $\Pr[\sigma \leftarrow \text{Prove}(s,w,\kappa) : \text{Verify}(s,\sigma,\kappa) = 1] > 1 - \text{negl}(\kappa)$.

**Definition 15** (Fiat-Shamir transformation [FS87]). Given a sigma protocol $\Sigma = (\text{Comm},\text{Chal},\text{Resp},\text{Verify}_\Sigma)$ for relation $R$ and a hash function $\mathcal{H}$, the Fiat-Shamir transformation, denoted $\text{FS}(\Sigma,\mathcal{H})$, is the tuple $(\text{Prove},\text{Verify}_\Sigma)$ of algorithms, defined as follows:

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

\[
\text{Verify}(s,(\text{comm},\text{resp}),\kappa) = \\
\text{chal} \leftarrow \mathcal{H}(\text{comm},s); \\
\text{return } \text{Verify}_\Sigma(s,(\text{comm},\text{chal},\text{resp}),\kappa);
\]
Definition 16 (Zero-knowledge [QS16]). Let $\Delta = \langle \text{Prove}, \text{Verify} \rangle$ 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 the sigma protocol. Moreover, let $\mathcal{S}$ be an algorithm, $A$ be an adversary, $\kappa$ be a security parameter, and $\text{ZK}(\Delta, A, \mathcal{H}, S, \kappa)$ be the following game.

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

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

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

And algorithm $\mathcal{S}$ can patch random oracle $\mathcal{H}$.

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

Definition 17 (Simulation sound extractability [SFC16,BPW12a,Gro06]). Suppose $\Sigma$ is a sigma protocol for relation $R$, $\mathcal{H}$ is a random oracle, and $(\text{Prove}, \text{Verify})$ is a non-interactive proof system, such that $\text{FS}(\Sigma, \mathcal{H}) = (\text{Prove}, \text{Verify})$. Further suppose $\mathcal{S}$ is a simulator for $(\text{Prove}, \text{Verify})$ and $\mathcal{H}$ can be patched by $\mathcal{S}$. Proof system $(\text{Prove}, \text{Verify})$ satisfies simulation sound extractability if there exists a probabilistic polynomial-time algorithm $\mathcal{K}$, such that for all probabilistic polynomial-time adversaries $A$ and coins $r$, there exists a negligible function $\text{negl}$, such that for all security parameters $\kappa$, we have:

\[
\text{Pr}[\mathcal{P} \leftarrow () ; Q \leftarrow A^{H, \mathcal{P}}(\leftarrow ; r) ; W \leftarrow \mathcal{K}^A(\mathcal{H}, \mathcal{P}, Q) : \\
|Q| \neq |W| \lor \exists j \in \{1, \ldots, |Q|\} . (Q[j][1], W[j]) \in R \land \\
\forall(s, \sigma) \in Q, (t, \tau) \in \mathcal{P} . \text{Verify}(s, \sigma, \kappa) = 1 \land \sigma \neq \tau] \leq \text{negl}(\kappa)
\]

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

- $A'()$. Computes $Q' \leftarrow A(\leftarrow ; r)$, forwarding any of $A$’s oracle queries to $\mathcal{K}$, and outputs $Q'$. By running $A(\leftarrow ; r)$, $\mathcal{K}$ is rewinding the adversary.

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

---

$^{33}$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.

$^{34}$We extend set membership notation to vectors: we write $x \in \mathbf{x}$ if $x$ is an element of the set $\{\mathbf{x}[i] : 1 \leq i \leq |\mathbf{x}|\}$. 

---

A CRYPTOGRAPHIC PRIMITIVES 21
Algorithm $K$ is an extractor for $(\text{Prove}, \text{Verify})$.

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

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

## B Proofs

### B.1 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 $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ and a constant symbol $\omega$, let $\text{Leak}(\Pi, \omega) = (\text{Gen}, \text{Enc}, \text{Dec}')$, such that $\text{Dec}'(sk, c)$ proceeds as follows: if $c = \omega$, then output $sk$, otherwise, compute $m \leftarrow \text{Dec}(sk, c)$ and output $m$.

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

**Proof sketch.** The proof follows immediately from correctness of the underlying encryption scheme, because constant symbol $\omega$ does not appear in the scheme’s ciphertext space.

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

**Proof sketch.** The proof is trivial: an adversary can output two distinct messages and a vector containing constant symbol $\omega$ during the first two adversary calls, learn the private key from the parallel decryption, and use the key to recover the plaintext from the challenge ciphertext, which allows the adversary to win the game.
Lemma 16. Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be an asymmetric encryption scheme and $\omega$ be a constant symbol. Suppose $\Pi$’s ciphertext space does not contain $\omega$ and $\Pi$’s message space is smaller than the private key. Further suppose $\text{Enc2Vote}(\Pi)$ satisfies Ballot-Secrecy. We have $\text{Enc2Vote}(\text{Leak}(\Pi, \omega))$ satisfies Ballot-Secrecy.

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

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

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

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

B.2 Proof of Theorem 4

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

- $B(pk, \kappa)$ computes $(V, nc) \leftarrow A(pk, \kappa); v, v' \leftarrow_R V$ and outputs $(v, v', nc)$.
- $B(b)$ computes $(R, bb) \leftarrow A(b)$ and outputs $bb$.
- $B(v)$ outputs 0 if $R(v, v)$ holds and 1 otherwise.
If the challenger selects \( \beta = 0 \) in \( \text{IND-CVA}(\Gamma, B, \kappa) \), then adversary \( B \) simulates \( A \)'s challenger to \( A \) in \( \text{cnm-cva}(\Gamma, A, \kappa) \) and \( B \)'s success (which requires \( R(v, v) \) to hold) is \( \text{Succ}\{\text{cnm-cva}(\Gamma, A, \kappa)\} \). Otherwise (\( \beta = 1 \)), adversary \( B \) simulates \( A \)'s challenger to \( A \) in \( \text{cnm-cva}(\Gamma, A, \kappa) \) and, since \( B \) will evaluate \( R(v, v) \), \( B \)'s success (which requires \( R(v, v) \) not to hold) is \( 1 - \text{Succ}\{\text{cnm-cva}(\Gamma, A, \kappa)\} \).

It follows that \( \text{Succ}(\text{IND-CVA}(\Gamma, A, \kappa)) = 1/2 \cdot (\text{Succ}(\text{cnm-cva}(\Gamma, A, \kappa)) + 1 - \text{Succ}(\text{cnm-cva}(\Gamma, A, \kappa))) \) and, therefore, \( 2 \cdot \text{Succ}(\text{IND-CVA}(\Gamma, A, \kappa)) - 1 = \text{Succ}(\text{cnm-cva}(\Gamma, A, \kappa)) - \text{Succ}(\text{cnm-cva}(\Gamma, A, \kappa)) \). Since \( \Gamma \) does not satisfy \( \text{CNM-CVA} \) and a function that doubles the output of a negligible function is a negligible function, we have \( \text{Succ}(\text{cnm-cva}(\Gamma, A, \kappa)) = \text{Succ}(\text{cnm-cva}(\Gamma, A, \kappa)) = 2 \cdot \text{negl}(\kappa) \).

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

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

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

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

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

### B.3 Proof of Theorem 7

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

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

Adversary $B$ 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 $\text{Succ}(\text{IND-CVA}(\Gamma, \mathcal{A}, \kappa)) = \text{Succ}(\text{Ballot-Secrecy}(\Gamma, B, \kappa))$, hence, $\text{Succ}(\text{Ballot-Secrecy}(\Gamma, B, \kappa)) > 1/2 + \text{negl}(\kappa)$, concluding our proof.

**B.4 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.

**B.5 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, $\text{Succ}(\text{Ballot-Secrecy}(\Gamma, A, \kappa)) = \frac{1}{2} \cdot \text{Succ}(\text{BS-0}(\Gamma, A, \kappa)) + \frac{1}{2} \cdot \text{Succ}(\text{BS-1}(\Gamma, A, \kappa))$. 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 $G_0, G_1, \ldots$ 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 $\Gamma$, we have $\text{Succ}(\text{BS-1}(\Gamma, \mathcal{A}, \kappa)) = 1 - \text{Succ}(\text{BS-1:0}(\Gamma, \mathcal{A}, \kappa))$ for all security parameters $\kappa$.

**Definition 19.** Let $\Gamma = (\text{Setup}, \text{Vote}, \text{Tally})$ be an election scheme with zero-knowledge tallying proofs, $\mathcal{A}$ be an adversary, and $\kappa$ be a security parameter. Moreover, let $\mathcal{S}$ be the simulator for the zero-knowledge non-interactive proof system using by algorithm Tally to construct tallying proofs. We introduce games $G_0, G_1, \ldots$, defined as follows.
B PROOFS

\( G_j(\Gamma, A, \kappa) = \)

\[
(pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa);
\]

\( nc \leftarrow A(pk, \kappa); \)

\( L \leftarrow \emptyset; \)

\( \mathcal{O} \leftarrow A^O(); \)

\( (v, \mathcal{O}) \leftarrow \text{Tally}(sk, \mathcal{O}, \{ b \mid (b, v_0, v_1) \in L \}, nc, \kappa); \)

\( \text{for } b \in \mathcal{O} \cup (b, v_0, v_1) \in L \text{ do} \)

\( \mathcal{O}[v_0] \leftarrow \mathcal{O}[v_0] + 1; \)

\( \mathcal{O} \leftarrow S((pk, nc, \mathcal{O}, v); \kappa); \)

\( g \leftarrow A(v, \mathcal{O}); \)

\( \text{return } g = 0 \land \text{balanced}(\mathcal{O}, nc, L) \land 1 \leq nc \leq mc \land |\mathcal{O}| \leq mb; \)

\( \text{Oracle } \mathcal{O} \text{ is defined such that } \mathcal{O}(v_0, v_1) \text{ computes, on inputs } v_0, v_1 \in \{1, \ldots, nc\}; \)

\( \text{the following:} \)

\( \text{if } |L| < j \text{ then} \)

\( b \leftarrow \text{Vote}(pk, v_1, nc, \kappa); \)

\( \text{else} \)

\( b \leftarrow \text{Vote}(pk, v_0, nc, \kappa); \)

\( L \leftarrow L \cup \{(b, v_0, v_1)\}; \)

\( \text{return } b; \)

Games \( G_0, G_1, \ldots \) are distinguished from games BS-0 and BS-1:0 by their left-right oracle 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 \( \Gamma \) be an election scheme, \( A \) be an adversary, and \( \kappa \) be a security parameter. If \( \Gamma \) satisfies Soundness, then \( \text{Succ}(\text{BS}-0(\Gamma, A, \kappa)) = \text{Succ}(G_0(\Gamma, A, \kappa)) \) and \( \text{Succ}(\text{BS}-1:0(\Gamma, A, \kappa)) = \text{Succ}(G_q(\Gamma, A, \kappa)) \), where \( q \) is an upper-bound on \( A \)'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 \( O(v_0, v_1) \) 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 \( O(v_0, v_1) \) 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 \( 1 \leq nc \leq mc \land |bb| \leq mb \) in BS-0 iff \( 1 \leq nc \leq mc \land |bb| \leq mb \) in \( G_0 \), and similarly for BS-1-0 and \( G_q \). Moreover, predicate \textit{balanced} is satisfied in BS-0 iff it is satisfied in \( G_0 \), and similarly for BS-1-0 and \( G_q \).

Hence, if \( 1 \leq nc \leq mc \land |bb| \leq mb \) is not satisfied or predicate \textit{balanced} is not satisfied, then \( \text{Succ}(\text{BS-0}(\Gamma, A, \kappa)) = \text{Succ}(G_0(\Gamma, A, \kappa)) \) and \( \text{Succ}(\text{BS-1-0}(\Gamma, A, \kappa)) = \text{Succ}(G_q(\Gamma, A, \kappa)) \), 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 \( bb \setminus \{ b \mid (b, v_0, v_1) \in L \} \), where \( bb \) 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 \( bb \cap \{ b \mid (b, v_0, v_1) \in L \} \)). Suppose \((pk, sk, mb, mc)\) is an output of \textit{Setup}(\( \kappa \)) and \( nc \) is an integer such that \( nc \leq mc \). Since \( \Gamma \) satisfies \textit{Soundness}, computing \( v \) as

\[
(v, pf) \leftarrow \text{Tally}(sk, bb, nc, \kappa);
\]

is equivalent to computing \( v \) as

\[
(v, pf) \leftarrow \text{Tally}(sk, bb \setminus \{ b \mid (b, v_0, v_1) \in L \}, nc, \kappa);
\]

\[
(v', pf') \leftarrow \text{Tally}(sk, bb \cap \{ b \mid (b, v_0, v_1) \in L \}, nc, \kappa);
\]

\[
v \leftarrow v + v';
\]

and as

\[
(v, pf) \leftarrow \text{Tally}(sk, bb \setminus \{ b \mid (b, v_0, v_1) \in L \}, nc, \kappa);
\]

\[
\text{for } b \in bb \land (b, v_0, v_1) \in L \text{ do}
\]

\[
(v', pf') \leftarrow \text{Tally}(sk, \{ b \}, nc, \kappa);
\]

\[
v \leftarrow v + v';
\]

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

\[
(v, pf) \leftarrow \text{Tally}(sk, bb \setminus \{ b \mid (b, v_0, v_1) \in L \}, nc, \kappa);
\]

\[
\text{for } b \in bb \land (b, v_0, v_1) \in L \text{ do}
\]

\[
[v[v_0]] \leftarrow v[v_0] + 1;
\]

In \( G_0 \), respectively \( G_q \), we have for all \( (b, v_0, v_1) \in L \) that \( b \) is an output of \textit{Vote}(\( pk, v_0, v_1, \kappa \)), respectively \textit{Vote}(\( pk, v_1, \kappa \)), such that \( v_0, v_1 \in \{1, ..., nc \} \). Moreover, by correctness of \( \Gamma \), we have \textit{Tally}(\( sk, \{ b \}, nc, \kappa \)) outputs \((v', pf')\) such that \( v' \) 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 \textit{balanced} holds in \( G_q \), we have for all \( v \in \{1, \ldots, nc \} \) that \( |\{ b \mid b \in bb \land \exists v_1 . (b, v, v_1) \in L \}| = |\{ b \mid b \in bb \land \exists v_0 . (b, v, v) \in L \}| \). Hence, in \( G_q \), computing
for $b \in \mathbb{b} \land (b, v_0, v_1) \in L$ do $v[v_0] \leftarrow v[v_0] + 1$;
is equivalent to computing
for $b \in \mathbb{b} \land (b, v_0, v_1) \in L$ do $v[v_1] \leftarrow v[v_1] + 1$;
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 $S$. Since $\Gamma$ has zero-knowledge tallying proofs, there exists
a non-interactive proof system $(\text{Prove}, \text{Verify})$ such that for all $(v, pf)$ output by
$\text{Tally}(sk, \mathbb{b}, nc, \kappa)$ there exists $w$ and $r$ such that $\text{Prove}((pk, nc, \mathbb{b}, v), w, \kappa; r) = pf$. Moreover, since $S$ is a simulator for $(\text{Prove}, \text{Verify})$, proofs output by
$\text{Prove}((pk, nc, \mathbb{b}, v), w, \kappa)$ are indistinguishable from outputs of $S((pk, nc, \mathbb{b}, v), \kappa)$. 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 indepen-
dence implies ballot secrecy. Suppose $\Gamma$ does not satisfy ballot secrecy, hence,
there exists a probabilistic polynomial-time adversary $A$, such that for all neg-
ligible functions $\text{negl}$, there exists a security parameter $\kappa$ and
$$\frac{1}{2} + \text{negl}(\kappa) < \text{Succ}($$
Ballot-Secrecy($\Gamma, A, \kappa$))
By definition of BS-0 and BS-1, we have
$$= \frac{1}{2} \cdot (\text{Succ}(\text{BS-0}($$
$\Gamma, A, \kappa$)) + \text{Succ}(\text{BS-1}($$
$\Gamma, A, \kappa$))))
And, by Lemma 17, we have
$$= \frac{1}{2} \cdot (\text{Succ}(\text{BS-0}($$
$\Gamma, A, \kappa$)) + 1 - \text{Succ}(\text{BS-1:0}($$
$\Gamma, A, \kappa$))))
$$= \frac{1}{2} + \frac{1}{2} \cdot (\text{Succ}(\text{BS-0}($$
$\Gamma, A, \kappa$)) - \text{Succ}(\text{BS-1:0}($$
$\Gamma, A, \kappa$))))
Let $q$ be an upper-bound on $A$’s left-right oracle queries. Hence, by Lemma 18,
we have
$$= \frac{1}{2} + \frac{1}{2} \cdot (\text{Succ}(\text{G}_0($$
$\Gamma, A, \kappa$)) - \text{Succ}(\text{G}_q($$
$\Gamma, A, \kappa$))))
which can be rewritten as the telescoping series
$$= \frac{1}{2} + \frac{1}{2} \cdot \sum_{1 \leq j \leq q} \text{Succ}(\text{G}_{j-1}($$
$\Gamma, A, \kappa$)) - \text{Succ}(\text{G}_j($$
$\Gamma, A, \kappa$))))
Let $j \in \{1, \ldots, q\}$ be such that $\text{Succ}(\text{G}_{j-1}($$
$\Gamma, A, \kappa$)) - \text{Succ}(\text{G}_j($$
$\Gamma, A, \kappa$))$ is the
largest term in that series. Hence,
$$\leq \frac{1}{2} + \frac{1}{2} \cdot q \cdot (\text{Succ}(\text{G}_{j-1}($$
$\Gamma, A, \kappa$)) - \text{Succ}(\text{G}_j($$
$\Gamma, A, \kappa$)))))
Thus,

$$\frac{1}{2} + \frac{1}{q} \cdot \text{negl}(\kappa) \leq \frac{1}{2} + \frac{1}{2} \cdot (\text{Succ}(G_{j-1}(\Gamma, A, \kappa)) - \text{Succ}(G_j(\Gamma, A, \kappa)))$$

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

Let $\Gamma = (\text{Setup}, \text{Vote}, \text{Tally})$. Since $\Gamma$ has zero-knowledge tallying proofs, tallying proofs output by Tally are constructed by a zero-knowledge non-interactive proof system. Let algorithm $S$ be the simulator for that proof system. We define $B$ as follows.

- $B(pk, \kappa)$ computes $nc \leftarrow A(pk, \kappa)$; $L \leftarrow \emptyset$ and runs $A^O()$, handling $A$’s oracle queries $O(v_0, v_1)$ as follows: if $|L| < j$, then compute $b \leftarrow \text{Vote}(pk, v_1, nc, \kappa)$; $L \leftarrow L \cup \{b, v_0, v_1\}$ and return $b$ to $A$, otherwise, assign $v_0^j \leftarrow v_0; v_1^j \leftarrow v_1$, and output $(v_0, v_1, nc)$.
- $B(b)$ assigns $L \leftarrow L \cup \{(b, v_0^j, v_1^j)\}$; returns $b$ to $A$ and handles any further oracle queries $O(v_0, v_1)$ as follows, namely, compute $b \leftarrow \text{Vote}(pk, v_0, nc, \kappa)$; $L \leftarrow L \cup \{(b, v_0, v_1)\}$ and return $b$ to $A$; assigns $A$’s output to $bb$; and outputs $bb \setminus \{b \mid (b, v_0, v_1) \in L\}$.
- $B(v)$ computes for $b \in bb \land (b, v_0, v_1) \in L$ do $v[v_0] \leftarrow v[v_0] + 1$, and $pf \leftarrow S((pk, nc, bb, v, \kappa)); g \leftarrow A(v, pf)$, and outputs $g$.

We prove that $B$ wins IND-CVA.

Suppose $(pk, sk, mb, nc)$ is an output of $\text{Setup}(\kappa)$. Further suppose we run $B(pk, \kappa)$. It is straightforward to see that $B$ simulates the challenger and oracle in both $G_{j-1}$ and $G_j$ to $A$. In particular, $B$ simulates query $O(v_0, v_1)$ by computing $b \leftarrow \text{Vote}(pk, v_1, nc, \kappa)$ 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 $A$ must make at least $j$ queries to ensure $\text{Succ}(G_{j-1}(\Gamma, A, \kappa)) - \text{Succ}(G_j(\Gamma, A, \kappa))$ is non-negligible. Hence, $B(pk, \kappa)$ terminates with non-negligible probability.

Suppose adversary $B$ terminates by outputting $(v_0, v_1, nc)$, where $v_0, v_1$ correspond to the inputs of the $j$th oracle query by $A$. Further suppose $b$ is an output of $\text{Vote}(pk, v_0, v_1, nc, \kappa)$, where $\beta$ is a bit. If $\beta = 0$, then $B(b)$ simulates the oracle in $G_{j-1}$ to $A$, otherwise, $B(b)$ simulates the oracle in $G_j$ to $A$. In particular, $B(b)$ responds to the $j$th oracle query with ballot $b$ for $v_3$, thus simulating the challenger in $G_{j-1}$ when $\beta = 0$, respectively $G_j$ when $\beta = 1$. And $B(b)$ responds to any further oracle queries $O(v_0, v_1)$ with ballots for $v_0$. Suppose $bb$ is an output of $A$, thus $B(b)$ outputs $bb \setminus \{b \mid (b, v_0, v_1) \in L\}$. Further suppose $(v, pf)$ is an output of $\text{Tally}(sk, bb \setminus \{b \mid (b, v_0, v_1) \in L\}, nc, \kappa)$ and $g$ is an output of $B(v)$. It is trivial to see that $B(v)$ simulates $A$’s challenger. Thus, either

1. $\beta = 0$ and $B$ simulates $G_{j-1}$ to $A$, thus, $g = \beta$ with at least the probability that $A$ wins $G_{j-1}$; or

2. $\beta = 1$ and $B$ simulates $G_j$ to $A$, thus, $g \neq 0$ with at least the probability that $B$ loses $G_i$ and, since $A$ wins game Ballot-Secrecy, we have $g$ is a bit, hence, $g = \beta$. 


It follows that the success of adversary $B$ is at least $\frac{1}{2} \cdot \text{Succ}(G_{j-1}(\Gamma, A, \kappa)) + \frac{1}{2} \cdot (1 - \text{Succ}(G_j(\Gamma, A, \kappa)))$, thus we conclude our proof.

\section{Universal verifiability implies soundness}

We recall the full syntax for election schemes in Definition 20. (The syntax for election schemes presented in Section 2 omitted algorithm \textit{Verify} and the condition that election schemes must satisfy notions of completeness and injectivity, because we did not consider verifiability in the main body.)

\begin{definition}[Election scheme \cite{SFC16}] An election scheme is a tuple of probabilistic polynomial-time algorithms $(\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$ such that $(\text{Setup}, \text{Vote}, \text{Tally})$ is an election scheme and

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

Election schemes must satisfy Completeness: there exists a negligible function $\text{negl}$, such that for all security parameters $\kappa$, bulletin boards $bb$, and integers $nc$, we have

$$\Pr[(pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa); (v, pf) \leftarrow \text{Tally}(sk, bb, nc, \kappa) : |bb| \leq mb \land nc \leq mc \Rightarrow \text{Verify}(pk, bb, nc, v, pf, \kappa) = 1] > 1 - \text{negl}(\kappa).$$

Election schemes must also satisfy Injectivity: for all security parameters $\kappa$, integers $nc$, and votes $v$ and $v'$, such that $v \neq v'$, we have

$$\Pr[(pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa); b \leftarrow \text{Vote}(pk, v, nc, \kappa); b' \leftarrow \text{Vote}(pk, v', nc, \kappa) : b \neq \bot \land b' \neq \bot \Rightarrow b \neq b'] = 1.$$

Universal verifiability (Definition 21) challenges the adversary to concoct a scenario in which \textit{Verify} accepts, but the election outcome is not correct. Formally, we capture the correct outcome using function $\text{correct-outcome}$,\footnote{Function \text{correct-outcome} uses a counting quantifier \cite{Sch05} denoted $\exists^*$. Predicate $(\exists^* x : P(x))$ holds exactly when there are $\ell$ distinct values for $x$ such that $P(x)$ is satisfied. Variable $x$ is bound by the quantifier, whereas $\ell$ is free.} defined such that for all $pk$, $nc$, $bb$, $\kappa$, and $v \in \{1, \ldots, nc\}$, we have:

$$\text{correct-outcome}(pk, nc, bb, \kappa)[v] = \ell \iff \exists^\ell b \in bb \setminus \{\bot\} : \exists r : b = \text{Vote}(pk, v, nc, \kappa; r)$$

That is, component $v$ of vector $\text{correct-outcome}(pk, bb, nc, \kappa)$ equals $\ell$ iff there exist $\ell$ ballots on the bulletin board that are votes for candidate $v$. The vector produced by $\text{correct-outcome}$ must be of length $nc$.\footnote{Function \text{correct-outcome} uses a counting quantifier \cite{Sch05} denoted $\exists^*$. Predicate $(\exists^* x : P(x))$ holds exactly when there are $\ell$ distinct values for $x$ such that $P(x)$ is satisfied. Variable $x$ is bound by the quantifier, whereas $\ell$ is free.}
Definition 21 (Exp-UV-Ext [SFC16]). Let $\Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$ be an election scheme, $A$ be an adversary, $\kappa$ be a security parameter, and $\text{Exp-UV-Ext}(\Gamma, A, \kappa)$ be the following game.

$\text{Exp-UV-Ext}(\Gamma, A, \kappa) = (pk, nc, bb, v, pf) \leftarrow A(\kappa);
\text{return } v \neq \text{correct-outcome}(pk, nc, bb, \kappa) \land \text{Verify}(pk, bb, nc, v, pf, \kappa) = 1;$

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

We show that universally verifiable election schemes satisfy Soundness (Proposition 20).

Lemma 19. Given an election scheme $(\text{Setup}, \text{Vote}, \text{Tally})$, there exists a negligible function $\text{negl}$, such that for all security parameters $\kappa$, integers $nc$, and votes $v \in \{1, \ldots, nc\}$, we have

$$\Pr[(pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa); b \leftarrow \text{Vote}(pk, v, nc, \kappa) : 1 \leq mb \land nc \leq mc \Rightarrow b \neq \perp] > 1 - \text{negl}(\kappa).$$

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

Proposition 20 (Exp-UV-Ext $\Rightarrow$ Soundness). If election scheme $\Gamma$ satisfies Exp-UV-Ext, then $\Gamma$ satisfies Soundness.

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

$B(\kappa) =$
\[(pk, sk, mb, nc) \leftarrow \text{Setup}(\kappa)\; ;
\]
\[(v, nc, bb_0) \leftarrow \mathcal{A}(pk, \kappa)\; ;
\]
\[(v_0, pf_0) \leftarrow \text{Tally}(sk, bb_0, nc, \kappa)\; ;
\]
\[\beta \leftarrow \{0, 1\};
\]
\[\text{if } \beta = 1 \text{ then}
\]
\[b \leftarrow \text{Vote}(pk, v, nc, \kappa)\; ;
\]
\[bb_1 \leftarrow bb_0 \cup \{b\}\; ;
\]
\[(v_1, pf_1) \leftarrow \text{Tally}(sk, bb_1, nc, \kappa)\; ;
\]
\[\text{return } (pk, nc, bb_0, bb_1, v, pf_0, pf_1)\; ;
\]

We prove that \(B\) wins \(\text{Exp-UV-Ext}\) with non-negligible probability.

Suppose \((pk, sk, mb, nc)\) is an output of \(\text{Setup}(\kappa)\), \(\{v, nc, bb_0\}\) is an output of \(\mathcal{A}(pk, \kappa)\), \(v_0, pf_0\) is an output of \(\text{Tally}(sk, bb_0, nc, \kappa)\), and \((v_1, pf_1)\) is an output of \(\text{Tally}(sk, bb_1, nc, \kappa)\), where \(bb_1 = bb_0 \cup \{b\}\). Let \(v^* \leftarrow (v_0[1], \ldots, v_0[v - 1], v_0[v] + 1, v_0[v + 1], \ldots, v_0[|v_0|])\).

Since \(\mathcal{A}\) is a winning adversary, we have \(v^* \neq v_1 \land b \notin bb_0 \land 1 \leq v \leq nc \leq mc \land |bb_0 \cup \{b\}| \leq mb\), with probability greater than negl(\(\kappa\)).

Suppose \(\beta\) is a bit chosen uniformly at random. By Completeness, we have \(\text{Verify}(pk, bb_0, \nu, v_0, pf_0, \kappa) = 1\), with overwhelming probability. Hence, it suffices to prove that \(v_0 \neq \text{correct-outcome}(pk, nc, bb_0, \kappa)\), with non-negligible probability. Let \(\delta_0\), respectively \(\delta_1\), be the probability that \(v_0 \neq \text{correct-outcome}(pk, nc, bb_0, \kappa)\), respectively \(v_1 \neq \text{correct-outcome}(pk, nc, bb_0, \kappa)\), and \(\text{Succ}(\text{Exp-UV-Ext}(\Gamma, B, \kappa)) = \frac{1}{2} \cdot \delta_0 + \frac{1}{2} \cdot \delta_1\). It follows that \(\delta_0 + \delta_1\) is non-negligible. It suffices to prove that \(v_0 = \text{correct-outcome}(pk, nc, bb_0, \kappa)\) is false with overwhelming probability.

Suppose \(v_0 = \text{correct-outcome}(pk, nc, bb_0, \kappa)\). By definition of function \(\text{correct-outcome}\), we have \(\exists v^* \in bb_0 \setminus \{\bot\} : \exists r : b^* = \text{Vote}(pk, v, nc, \kappa, r)\). Since \(1 \leq |bb_0 \cup \{b\}| \leq mb\), we have \(b \neq \bot\) by Lemma 19, with overwhelming probability. Given that \(b\) is an output of \(\text{Vote}(pk, v, nc, \kappa)\), \(b \notin bb_0\), and \(v^* = v_0[v] + 1\), it follows that \(\exists v^* \in bb_0 \cup \{\bot\} \setminus \{v\} : \exists r : b^* = \text{Vote}(pk, v^*, \kappa, r)\).

Moreover, by Injectivity, \(b\) is not an output of \(\text{Vote}(pk, v^*, \kappa)\) for all \(v^* \in \{1, \ldots, |v^*|\} \setminus \{v\}\). Thus, for all \(v^* \in \{1, \ldots, |v^*|\} \setminus \{v\}\) we have \(\exists v^* \in bb_0 \cup \{b\} \setminus \{v\} : \exists r : b^* = \text{Vote}(pk, v^*, nc, \kappa, r)\). Given that \(bb_1 = bb_0 \cup \{b\}\), we have \(v^* = \text{correct-outcome}(pk, nc, bb_1, \kappa)\). Moreover, given that \(v^* \neq v_1\), we have \(v_1 \neq \text{correct-outcome}(pk, nc, bb_1, \kappa)\) with overwhelming probability, which suffices to conclude our proof.

\(\square\)

D Helios

Smyth, Frink & Clarkson [SFC16] formalise a generic construction for Helios-like election schemes (Definition 23), which is parameterized on the choice of homomorphic encryption scheme and sigma protocols for the relations introduced in the following definition.
Definition 22 (from [SFC16]). Let \((\text{Gen}, \text{Enc}, \text{Dec})\) be a homomorphic asymmetric encryption scheme and \(\Sigma\) be a sigma protocol for a binary relation \(R\).\(^{36}\)

- \(\Sigma\) proves correct key construction if a \(((\kappa, pk, m), (sk, r))\) \(\in R \Leftrightarrow (pk, sk, m) = \text{Gen}(\kappa; r)\).

Further, suppose that \((pk, sk, m)\) is the output of \(\text{Gen}(\kappa; r)\), for some security parameter \(\kappa\) and coins \(r\).

- \(\Sigma\) proves plaintext knowledge in a subspace if \(((pk, c, m'), (m, r)) \in R \Leftrightarrow c = \text{Enc}(pk, m; r) \land m \in m' \land m' \subseteq m\).

- \(\Sigma\) proves correct decryption if \(((pk, c, m), sk) \in R \Leftrightarrow m = \text{Dec}(sk, c)\).

Definition 23 (Generalized Helios [SFC16]). Suppose \(\Gamma = (\text{Gen}, \text{Enc}, \text{Dec})\) is an additively homomorphic asymmetric encryption scheme with a message space that, for sufficiently large security parameters, includes \(\{0, 1\}\), \(\Sigma_1\) proves correct key construction, \(\Sigma_2\) proves plaintext knowledge in a subspace, \(\Sigma_3\) proves correct decryption, and \(H\) is a hash function. Let \(\text{FS}(\Sigma_1, H) = (\text{ProveKey}, \text{VerKey})\), \(\text{FS}(\Sigma_2, H) = (\text{ProveCiph}, \text{VerCiph})\), and \(\text{FS}(\Sigma_3, H) = (\text{ProveDec}, \text{VerDec})\). We define election scheme generalised Helios, denoted Helios(\(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, H\)) = (\(\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}\)), as follows.

**Setup(\(\kappa\)).** Select coins \(s\), compute \((pk, sk, m) \leftarrow \text{Gen}(\kappa; s); r \leftarrow \text{ProveKey}((\kappa, pk, m), (sk, s), \kappa); PK_T \leftarrow (pk, m, \rho); SK_T \leftarrow (pk, sk), \) let \(m\) be the largest integer such that \(\{0, \ldots, m\} \subseteq m\), and output \((PK_T, SK_T, m, m)\).

**Vote(\(PK_T, v, nc, \kappa\)).** Parse \(PK_T\) as a vector \((pk, m, \rho)\). Output \(\perp\) if parsing fails or \(\text{VerKey}((\kappa, pk, m), \rho, \kappa) \neq 1 \lor v \notin \{1, \ldots, nc\}\). Select coins \(r_1, \ldots, r_{nc-1}\) and compute:

\[
\text{for } 1 \leq j \leq nc - 1 \text{ do } \begin{align*}
\text{if } j = v & \text{ then } m_j \leftarrow 1; \text{ else } m_j \leftarrow 0; \\
& c_j \leftarrow \text{Enc}(pk, m_j; r_j); \\
& \sigma_j \leftarrow \text{ProveCiph}((pk, c_j, \{0, 1\}, (m_j, r_j), j, \kappa); \\
& c \leftarrow c_1 \otimes \cdots \otimes c_{nc-1}; \\
& m \leftarrow m_1 \otimes \cdots \otimes m_{nc-1}; \\
& r \leftarrow r_1 \otimes \cdots \otimes r_{nc-1}; \\
& \sigma_{nc} \leftarrow \text{ProveCiph}((pk, c, \{0, 1\}, (m, r), nc, \kappa); \\
\end{align*}
\]

Output ballot \((c_1, \ldots, c_{nc-1}, \sigma_1, \ldots, \sigma_{nc})\).

\(^{36}\)Given a binary relation \(R\), we write \(((s_1, \ldots, s_l), (w_1, \ldots, w_k)) \in R \Leftrightarrow P(s_1, \ldots, s_l, w_1, \ldots, w_k)\) for \((s, w) \in R \Leftrightarrow P(s_1, \ldots, s_l, w_1, \ldots, w_k) \land s = (s_1, \ldots, s_l) \land w = (w_1, \ldots, w_k), \) hence, \(R\) is only defined over pairs of vectors of lengths \(l\) and \(k\).
Tally($SK_τ, bb, nc, κ$). Initialise vectors $v$ of length $nc$ and $pf$ of length $nc - 1$. Compute for $1 \leq j \leq nc$ do $v[j] ← 0$. Parse $SK_τ$ as a vector $(pk, sk)$. Output $(v, pf)$ if parsing fails. Let $\{b_1, \ldots, b_{\ell}\}$ be the largest subset of $bb$ 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 $\bigwedge_{j=1}^{nc-1} \text{VerCiph}((pk, b_1[j], (0, 1)), b_j[j + nc - 1], j, κ) = 1 \wedge \text{VerCiph}((pk, b_1[1] \otimes \cdots \otimes b_j[nc - 1], (0, 1)), b_j[2 \cdot nc - 1], nc, κ) = 1$. If $\{b_1, \ldots, b_{\ell}\} = \emptyset$, then output $(v, pf)$, otherwise, compute:

- for $1 \leq j \leq nc - 1$ do:
  - $c ← b_1[j] \otimes \cdots \otimes b_{\ell}[j]$;
  - $v[j] ← \text{Dec}(sk, c)$;
  - $pf[j] ← \text{ProveDec}((pk, c, v[j]), sk, κ)$;
  - $v[nc] ← \ell - \sum_{j=1}^{nc-1} v[j]$;

Output $(v, pf)$.

Verify($PK_τ, bb, nc, v, pf, κ$). Parse $v$ as a vector of length $nc$, parse $pf$ as a vector of length $nc - 1$, parse $PK_τ$ as a vector $(pk, m, ρ)$. Output 0 if parsing fails or $\text{VerKey}((κ, pk, m, ρ, κ) \neq 1$. Let $\{b_1, \ldots, b_{\ell}\}$ be the largest subset of $bb$ satisfying the conditions given by the tally algorithm and let $mb$ be the largest integer such that $\{0, \ldots, mb\} \subseteq m$. If $\{b_1, \ldots, b_{\ell}\} = \emptyset \wedge \bigwedge_{j=1}^{nc-1} v[j] = 0$ or $\bigwedge_{j=1}^{nc-1} \text{VerDec}((pk, b_1[j] \otimes \cdots \otimes b_{\ell}[j], v[j]), pf[j], κ) = 1 \wedge v[nc] = \ell - \sum_{j=1}^{nc-1} v[j] \wedge 1 \leq \ell \leq mb$, 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.

The scheme works as follows [SFC16].

- **Setup** generates the taller’s key pair. The public key includes a non-interactive proof demonstrating that the key pair is correctly constructed.

- **Vote** takes a vote $v \in \{1, \ldots, nc\}$ and outputs ciphertexts $c_1, \ldots, c_{nc-1}$ 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 $σ_1, \ldots, σ_{nc}$ 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 $v \in \{1, \ldots, nc - 1\}$ is simply the homomorphic combination of ciphertexts representing votes for that 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 24** (Helios'16 [SFC16]). *Election scheme Helios'16 is Helios*(\(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}\)), where \(\Gamma\) is additively homomorphic El Gamal \([CGS97, \S2]\), \(\Sigma_1\) is the sigma protocol for proving knowledge of discrete logarithms by Chaum et al. \([CEGP87, \text{Protocol 2}]\), \(\Sigma_2\) is the sigma protocol for proving knowledge of disjointive equality between discrete logarithms by Cramer et al. \([CFSY96, \text{Figure 1}]\), \(\Sigma_3\) is the sigma protocol for proving knowledge of equality between discrete logarithms by Chaum \& Pedersen \([CP93, \S3.2]\), 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, \([CEGP87, \text{Protocol 2}]\) is a sigma protocol for proving correct key construction, \([CFSY96, \text{Figure 1}]\) is a sigma protocol for proving plaintext knowledge in a subspace, and \([CP93, \S3.2]\) is a sigma protocol for proving decryption. We leave formally proving this assumption as future work.

**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 \([SFC16]\) and, thus, soundness (Proposition 20). They also satisfy ballot independence (Proposition 22). Hence, they satisfy ballot secrecy too (Theorem 10). Moreover, Helios'16 satisfies ballot secrecy.

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

**Lemma 21.** If \((\text{ProveDec}, \text{VerDec})\) is zero-knowledge, then Helios\((\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})\) has zero-knowledge tallying proofs.

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

**Proposition 22.** Given an election scheme \(\Gamma\) satisfying IND-CPA and non-interactive proof systems \((\text{ProveKey}, \text{VerKey})\) and \((\text{ProveCiph}, \text{VerCiph})\) satisfying
functions

exists a probabilistic polynomial-time adversary $A$ such that for all $1 \leq i \leq |m_0|$ we have $|m_0[i]| = |m_1[i]|$ and $m_0[i]$ and $m_1[i]$ are from the encryption scheme’s message space, and 2) the challenger computes $c_1 \leftarrow \text{Enc}(pk, m_1[1]); \ldots; c_{|m_0|} \leftarrow \text{Enc}(pk, m_1[|m_0|])$ and inputs $c_1, \ldots, c_{|m_0|}$ to the adversary. We have $\Gamma$ satisfies IND-CPA$^*$ [KL07, §10.2.2].

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

- $B(pk, m, \kappa)$ outputs $((1, 0), (0, 1))$.
- $B(c)$ proceeds as follows. First, compute:
  
  $\rho \leftarrow \text{SimProveKey}((\kappa, pk, m), \kappa)$;
  
  $PK_T \leftarrow (pk, m, \rho)$;
  
  $(v_0, v_1, nc) \leftarrow A(PK_T, \kappa)$;

  Secondly, select coins $r_1, \ldots, r_{nc-1}$ and compute:

  for $j \in \{1, \ldots, nc - 1\} \setminus \{v_0, v_1\}$ do
  
  $c_j \leftarrow \text{Enc}(pk, 0; r_j)$;
  
  $\sigma_j \leftarrow \text{ProveCiph}((pk, c_j, \{0, 1\}), (0, r_j), j, \kappa)$;

  $c_{v_0} \leftarrow c[1]$;
  
  $\sigma_{v_0} \leftarrow \text{SimProveCiph}((pk, c_{v_0}, \{0, 1\}), v_0, \kappa)$;

  if $v_1 \neq nc$ then
  
  $c_{v_1} \leftarrow c[2]$;
  
  $\sigma_{v_1} \leftarrow \text{SimProveCiph}((pk, c_{v_1}, \{0, 1\}), v_1, \kappa)$;

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

  $b \leftarrow (c_1, \ldots, c_{nc-1}, \sigma_1, \ldots, \sigma_{nc})$;

  $bb \leftarrow A(b)$;

  Thirdly, compute $\{b_1, \ldots, b_3\}$ as the largest subset of $bb$ satisfying the conditions of algorithm $\text{Tally}$. Fourthly, initialise $H$ as a transcript of the
random oracle’s input and output, $P$ as a transcript of simulated proofs, and $v$ as a zero-filled vector of length $nc$. Fifthly, compute:

\[
\text{for } 1 \leq i \leq \ell \text{ do} \\
\quad \text{for } 1 \leq j \leq nc - 1 \text{ do} \\
\quad \quad Q \leftarrow (\{pk, b_i[j], \{0,1\}\}, b_i[j + nc - 1]); \\
\quad \quad W \leftarrow \text{ExtProveCiph}(H, P, Q); \\
\quad \quad v[j] \leftarrow v[j] + W[1][1]; \\
\quad v[nc] \leftarrow \ell - \sum_{j=1}^{nc-1} v[j]; \\
\quad g \leftarrow A(v); \\
\]

Finally, output $g$.

We prove that $B$ wins IND-CPA.

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

**Claim 23.** Adversary $B$’s computation of $b$ is equivalent to computing $b$ as $b \leftarrow \text{Vote}(PK_T, v_\beta, nc, \kappa)$.

**Proof of Claim 23.** We have $PK_T$ parses as a vector $(pk, m, \rho)$. Moreover, since $(pk, sk, m)$ is an output of $\text{Gen}(\kappa)$, there exist coins $r$ such that $(pk, sk, m) = \text{Gen}(\kappa; r)$. Hence, $(sk, r)$ is a witness for statement $(\kappa,pk,m)$. Furthermore, since $\text{SimProveKey}$ is a simulator for $(\text{ProveKey}, \text{VerKey})$ and proofs output by $\text{ProveKey}$ are indistinguishable from outputs of $\text{SimProveKey}$, we have $\text{VerKey}(\kappa, pk, m, \rho, \kappa) = 1$, with non-negligible probability. In addition, since $B$ is a winning adversary, we have $v_0, v_1 \in \{1, \ldots, nc\}$, with non-negligible probability. It follows that $\text{Vote}(PK_T, v_\beta, nc, \kappa)$ does not output $\perp$, with non-negligible probability. Indeed, computation $b \leftarrow \text{Vote}(PK_T, v_\beta, nc, \kappa)$ is equivalent to the following. Select coins $r_1, \ldots, r_{nc-1}$ and compute:
for $1 \leq j \leq nc - 1$ do
  if $j = v_β$ then $m_j \leftarrow 1$; else $m_j \leftarrow 0$;
  $c_j \leftarrow \text{Enc}(pk, m_j; r_j)$;
  $\sigma_j \leftarrow \text{ProveCiph}((pk, c_j, \{0, 1\}), (m_j, r_j), j, \kappa)$;
  $c \leftarrow c_1 \otimes \cdots \otimes c_{nc-1}$;
  $m \leftarrow m_1 \otimes \cdots \otimes m_{nc-1}$;
  $r \leftarrow r_1 \otimes \cdots \otimes r_{nc-1}$;
  $\sigma_{nc} \leftarrow \text{ProveCiph}((pk, c, \{0, 1\}), (m, r), nc, \kappa)$;
  $b \leftarrow (c_1, \ldots, c_{nc-1}, \sigma_1, \ldots, \sigma_{nc})$;

Since $v_β \in \{v_0, v_1\}$, 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 $c_{v_0}$ only exists if $v_1 < nc$.) Given that $v_0 < v_1 \leq nc$, ciphertext $c_{v_0}$ contains $1 - \beta$, i.e., if $\beta = 0$, then $c_{v_0}$ contains 1, otherwise ($\beta = 1$), $c_{v_0}$ contains 0. If $v_1 < nc$, then ciphertext $c_{v_1}$ contains $\beta$. Moreover, since $\otimes$ is the addition operator in group $(m, \otimes)$ 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, $m = 1 - \beta \cdot \beta = 1$. Hence, the above algorithm is equivalent to selecting coins $r_1, \ldots, r_{nc-1}$ and computing:

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

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

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

can be simulated by
c ← c₁ ⊕ ⋯ ⊕ c_nc−1;
σ_nc ← SimProveCiph((pk, c, {0, 1}), nc, κ);

Hence, we conclude the proof of this claim.

Claim 24. Adversary B’s computation of v is equivalent to computing v as
(v, pf) ← Tally(SKₜ, bb, nc, κ), where SKₜ = (pk, sk).

Proof of Claim 24. Let \{b₁, \ldots, bₗ\} be the largest subset of bb satisfying the
conditions of algorithm Tally. It is trivial to see that the claim holds when
\{b₁, \ldots, bₗ\} = ∅, because v is computed as a zero-filled vector of length nc in
both cases. We prove the claim also holds when \{b₁, \ldots, bₗ\} ≠ ∅.

By simulation sound extractability, for all 1 ≤ i ≤ ℓ and 1 ≤ j ≤ 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:

\[ b_i[j] = \text{Enc}(pk, m_{i,j}; r_{i,j}) \]
\[ b_i[j + nc - 1] = \text{ProveCiph}((pk, b_i[j], \{0, 1\}), (m_{i,j}, r_{i,j}), j; r_{i,j+nc-1}) \]

with overwhelming probability. It follows for all 1 ≤ i ≤ ℓ and 1 ≤ j ≤ nc − 1
that computation \(Q ← ((pk, b_i[j], \{0, 1\}), b_i[j + nc - 1]); W ← \text{ExtProveCiph}(H, P_{(Q)})\) is such that
\(W[1]\) is a witness for \((pk, b_i[j], \{0, 1\})\), i.e., \((m_{i,j}, r_{i,j})\),
and \(W[1][1] = m_{i,j}\). Hence, adversary B’s computation of v is equivalent to
initialising v as a zero-filled vector of length nc and computing:

for 1 ≤ i ≤ ℓ do
    for 1 ≤ j ≤ nc − 1 do
        v[j] ← v[j] + m_{i,j};
    v[nc] ← ℓ = \(\sum_{j=1}^{nc-1} v[j]\);
and, therefore,

for 1 ≤ j ≤ nc − 1 do
    v[j] ← \(\sum_{i=1}^{\ell} m_{i,j}\);
    v[nc] ← ℓ = \(\sum_{j=1}^{nc-1} v[j]\);

Moreover, computing v as (v, pf) ← Tally(SKₜ, bb, nc, κ) is equivalent to
initialising v as a zero-filled vector of length nc and computing

for 1 ≤ j ≤ nc − 1 do
    c ← b_i[j] ⊕ ⋯ ⊕ b_i[j];
    v[j] ← \text{Dec}(sk, c);
    v[nc] ← ℓ = \(\sum_{j=1}^{nc-1} v[j]\);
and, therefore,

for 1 ≤ j ≤ nc − 1 do
    v[j] ← m_{1,j} ⊕ ⋯ ⊕ m_{ℓ,j};
    v[nc] ← ℓ = \(\sum_{j=1}^{nc-1} v[j]\);

Let mb be the largest integer such that \{0, \ldots, mb\} ⊆ m. Since A is a winning
adversary, we have ℓ ≤ mb. Moreover, since \(m_{1,j}, \ldots, m_{\ell,j} \in \{0, 1\}\) for all
REFERENCES

1 \leq j \leq nc - 1 and \odot is the addition operator in group \((m, \odot)\), 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 \(\Gamma\) is additively homomorphic El Gamal [CGS97, §2]. Moreover, \((\text{ProveKey}, \text{VerKey})\), respectively \((\text{ProveCiph}, \text{VerCiph})\) and \((\text{ProveDec}, \text{VerDec})\), is the non-interactive proof system derived by application of the Fiat-Shamir transformation [FS87] to a random oracle \(H\) 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 \((\text{ProveKey}, \text{VerKey})\) and \((\text{ProveCiph}, \text{VerCiph})\) 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 \((\text{ProveDec}, \text{VerDec})\) 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 \((\text{ProveDec}, \text{VerDec})\). Moreover, since Helios’16 satisfies Exp-UV-Ext [SFC16], we have Helios’16 satisfies Soundness (Proposition 20). Furthermore, since El Gamal satisfies IND-CPA [TY98, KL07] and non-interactive proof systems \((\text{ProveKey}, \text{VerKey})\) and \((\text{ProveCiph}, \text{VerCiph})\) 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


REFERENCES


elections/eVoting/finalReportHelios_2010-09-27.html,
Sept 2010.


REFERENCES


REFERENCES


REFERENCES


[SK95] Kazue Sako and Joe Kilian. Receipt-Free Mix-Type Voting Scheme: A practical solution to the implementation of a voting booth.
REFERENCES


