Secret, verifiable auctions from elections

Elizabeth A. Quaglia\textsuperscript{1} and Ben Smyth\textsuperscript{2}

\textsuperscript{1}Information Security Group, Royal Holloway, University of London
\textsuperscript{2}Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg

March 28, 2018

Abstract

Auctions and elections are seemingly disjoint. Nevertheless, similar cryptographic primitives are used in both domains. For instance, mixnets, homomorphic encryption and trapdoor bit-commitments have been used by state-of-the-art schemes in both domains. These developments have appeared independently. For example, the adoption of mixnets in elections preceded a similar adoption in auctions by over two decades. In this paper, we demonstrate a relation between auctions and elections: we present a generic construction for auctions from election schemes. Moreover, we show that the construction guarantees secrecy and verifiability, assuming the underlying election scheme satisfies analogous security properties. We demonstrate the applicability of our work by deriving auction schemes from the Helios family of election schemes. Our results advance the unification of auctions and elections, thereby facilitating the progression of both domains.

Keywords. Auctions, elections, privacy, secrecy, verifiability.

1 Introduction

We present a construction for auction schemes from election schemes, and prove the construction guarantees security, assuming the underlying election scheme is secure.

Auctions. An auction is a process for the trade of goods and services from sellers to bidders [Kri00, MM87], with the support of an auctioneer. We study first-price sealed-bid auctions [Bra10], whereby bidders create bids which encapsulate the price they are willing to pay, and the auctioneer opens the bids to determine the winning price (namely, the highest price bid) and winning bidder.
Elections. An election is a decision-making procedure used by voters to choose a representative from some candidates [Gum05, AH10], with the support of a tallier. We study first-past-the-post secret ballot elections [LG84, Saa95], which are defined as follows. First, each voter creates a ballot which encapsulates the voter’s chosen candidate (i.e., the voter’s vote). Secondly, all ballots are tallied by the tallier to derive the distribution of votes. Finally, the representative – namely, the candidate with the most votes – is announced.

Bidders and voters should express their choice freely in auctions and elections; this can be achieved by participating in private [OSC90, US90, UN48, OAS69], which has led to the emergence of the following requirements.

- **Bid secrecy**: A losing bidder cannot be linked to a price.
- **Ballot secrecy**: A voter cannot be linked to a vote.

Ballot secrecy aims to protect the privacy of all voters, whereas bid secrecy is only intended to protect the privacy of losing bidders. This intuitive weakening is necessary, because the auctioneer reveals the winning price and winning bidder, hence, a winning bidder can be linked to the winning price.

Bidders and voters should be able to check that auctions and elections are run correctly [JCJ02, Dag07, CRS05, Adi06, Adi08, DJL13]; this is known as verifiability. Sometimes we write auction verifiability and election verifiability to distinguish verifiability in each field. Kremer, Ryan & Smyth [KRS10] decompose verifiability into the following properties.

- **Individual verifiability**: bidders/voters can check whether their bid/ballot is included.
- **Universal verifiability**: anyone can check whether the result is computed properly.

Conceptually, individual and universal verifiability do not differ between auctions and elections.

### 1.1 Constructing auctions from elections

Our construction for auction schemes from election schemes works as follows.

1. We represent prices as candidates, and instruct bidders to create bids by “voting” for the candidate that represents the price they are willing to pay.

2. Bids are “tallied” to derive the distribution of prices and the winning price is determined from this distribution.

The relation between auctions and elections is so far straightforward. The challenge is to establish the winning bidder. This is non-trivial, because election
schemes satisfying ballot secrecy ensure voters cannot be linked to votes, hence, the bidder mentioned above cannot be linked to the price they are willing to pay. We overcome this by extending the tallier’s role to additionally reveal the set of ballots for a specific vote,\(^1\) and exploit this extension to complete the final step.

3. The tallier determines the winning bids and a winning bidder can be selected from these bids.\(^2\)

Extending the tallier’s role is central to our construction.

1.2 Motivation and related work

There is an abundance of rich research on elections which can be capitalised upon to advance auctions. This statement can be justified with hindsight: Chaum [Cha81] exploited mixnets in elections twenty-three years before Peng et al. [PBDV04] made similar advances in auctions (Jakobsson & Juels [JJ00] use mixnets in a distinct way from Chaum and Peng et al.), Benaloh & Fischer [CF85] proposed using homomorphic encryption seventeen years before Abe & Suzuki [AS02a], and Okamoto [oka96] demonstrated the use of trapdoor bit-commitments six years before Abe & Suzuki [AS02b].

Magkos, Alexandris & Chrissikopoulos [MAC02] and Her, Imamot & Sakurai [HIS05] have studied the relation between auctions and elections. Magkos, Alexandris & Chrissikopoulos remark that auctions and elections have a similar structure and share similar security properties. And Her, Imamot & Sakurai contrast privacy properties of auctions and elections, and compare the use of homomorphic encryption and mixnets between fields. More concretely, McCarthy, Smyth & Quaglia [MSQ14] derive auction schemes from the Helios and Civitas election schemes. And Lipmaa, Asokan & Niemi study the converse [LAN02, §9].

1.3 Contribution

We formally demonstrate a relation between auctions and elections: we present a generic construction for auction schemes from election schemes, moreover, we prove that auction schemes produced by our construction satisfy bid secrecy and auction verifiability, assuming the underlying election scheme satisfies ballot secrecy and election verifiability. To achieve this, we formalise syntax and security definitions for auction schemes, since these are prerequisites to rigorous, formal results.

---

\(^1\)Ballot secrecy does not prohibit such behaviour, because ballot secrecy assumes the tallier is trusted.

\(^2\)Handling tie-breaks (i.e., selecting a winning bid from a set of winning bids) is straightforward. For instance, the set could be sampled uniformly at random. So this paper does not address this problem.
Summary of contributions and paper structure. We summarise our contributions as follows.

- We propose auction scheme syntax, and the first computational security definitions of bid secrecy and verifiability for auctions (§2).
- We present a construction for auction schemes from election schemes (§3).
- We prove that our construction guarantees bid secrecy (§4) and verifiability (§5), assuming the underlying election scheme satisfies analogous security properties.
- We show that our construction is applicable to a large class of election schemes and justify our choice of security definitions (§6).
- We use our construction to derive auction schemes from the Helios family of election schemes (§7).
- We define cryptographic primitives and relevant security definitions in Appendix A, and we present further supplementary material in the remaining appendices.

It follows from our results that secure auction schemes can be constructed from election schemes, allowing advances in election schemes to be capitalised upon to advance auction schemes.

1.4 Comparison with McCarthy, Smyth & Quaglia

The idea underlying our construction was introduced by McCarthy, Smyth & Quaglia [MSQ14]. Our contributions improve upon their work by providing a strong theoretical foundation to their idea. In particular, we provide a generic construction for auction schemes from election schemes, they consider the derivation of only two auction schemes from Helios and Civitas. We prove that auction schemes produced by our construction satisfy bid secrecy and verifiability, they do not provide security proofs. Thus, the auction schemes we construct from Helios satisfy bid secrecy and verifiability, whereas the auction schemes they derive have no such proofs. Moreover, we are the first to introduce computational security definitions of bid secrecy and auction verifiability.

2 Auction schemes

2.1 Syntax

We formulate syntax for auction schemes, which capture the class of auctions that consist of the following four steps. First, an auctioneer generates a key pair. Secondly, each bidder constructs and casts a bid for their price. These bids are recorded on a bulletin board. Thirdly, the auctioneer opens the recorded bids and announces the winning price and the winning bids, i.e., bids that contain
the winning price. Finally, bidders and other interested parties check that the result corresponds to the recorded bids.

**Definition 1** (Auction scheme). An auction scheme is a tuple of efficient algorithms \((\text{Setup}, \text{Bid}, \text{Open}, \text{Verify})\) such that:

- **Setup**, denoted\(^3\) \((pk, sk, mb, mp) \leftarrow \text{Setup}(\kappa)\), is run by the auctioneer. Setup takes a security parameter \(\kappa\) as input and outputs a key pair \(pk, sk\), a maximum number of bids \(mb\), and a maximum price \(mp\).

- **Bid**, denoted \(b \leftarrow \text{Bid}(pk, np, p, \kappa)\), is run by bidders. Bid takes as input a public key \(pk\), an upper-bound \(np\) on the range of biddable prices,\(^4\) a bidder's chosen price \(p\), and a security parameter \(\kappa\). A bidder's price should be selected from the range \(1, \ldots, np\) of prices. Bid outputs a bid \(b\) or error symbol \(\bot\).

- **Open**, denoted \((p, b, pf) \leftarrow \text{Open}(sk, np, bb, \kappa)\), is run by the auctioneer. Open takes as input a private key \(sk\), an upper-bound \(np\) on the range of biddable prices, a bulletin board \(bb\), and a security parameter \(\kappa\), where \(bb\) is a set. It outputs a (winning) price \(p\), a set of (winning) bids \(b\), and a non-interactive proof \(pf\) of correct opening.

- **Verify**, denoted \(s \leftarrow \text{Verify}(pk, np, bb, p, b, pf, \kappa)\), is run to audit an auction. It takes as input a public key \(pk\), an upper-bound \(np\) on the range of biddable prices, a bulletin board \(bb\), a price \(p\), a set of bids \(b\), a proof \(pf\), and a security parameter \(\kappa\). It outputs a bit \(s\), which is 1 if the auction verifies successfully and 0 otherwise.

Auction schemes must satisfy correctness, completeness, and injectivity, which we define in Appendix B.1.

Our proposed syntax is based upon syntax by McCarthy, Smyth & Quaglia [MSQ14]. Moreover, our correctness, completeness and injectivity properties are based upon similar properties of election schemes. (Cf. Section 3.1.)

### 2.1.1 Example: Enc2Bid

We demonstrate the applicability of our syntax with a construction \((\text{Enc2Bid})\) for auction schemes from asymmetric encryption schemes.

**Definition 2** (Enc2Bid). Given an asymmetric encryption scheme \(\Pi = (\text{Gen}, \text{Enc}, \text{Dec})\), we define \(\text{Enc2Bid}(\Pi)\) as follows.

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

\(^3\)Let \(A(x_1, \ldots, x_n; r)\) denote the output of probabilistic algorithm \(A\) on inputs \(x_1, \ldots, x_n\) and coins \(r\). 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.

\(^4\)An upper-bound on the range of biddable prices is sometimes useful for efficiency reasons, as can be observed from our case studies (§7).
Bid\((pk, np, p, \kappa)\) computes \(b \leftarrow \text{Enc}(pk, p)\) and outputs \(b\) if \(1 \leq p \leq np \leq |m|\) and \(\bot\) otherwise.

Open\((sk, np, bb, \kappa)\) proceeds as follows. Computes \(d \leftarrow \{(b, \text{Dec}(sk, b)) \mid b \in bb\}\). Finds the largest integer \(p\) such that \((b, p) \in d \land 1 \leq p \leq np\), outputting \((0, \emptyset, \epsilon)\) if no such integer exists. Computes \(b' \leftarrow \{b \mid (b, p') \in d \land p' = p\}\). Outputs \((p, b, \epsilon)\).

Verify\((pk, np, bb, p, b, pf, \kappa)\) outputs \(1\).

Algorithm Setup requires \(\text{poly}\) to be a polynomial function, algorithms Setup and Bid require \(m = \{1, \ldots, |m|\}\) to be the encryption scheme’s plaintext space, and algorithm Open requires \(\epsilon\) to be a constant symbol.

To ensure \(\text{Enc2Bid}''(\Pi)\) is an auction scheme, we require asymmetric encryption scheme \(\Pi\) to produce distinct ciphertexts with overwhelming probability. Hence, we must restrict the class of asymmetric encryption schemes used to instantiate \(\text{Enc2Bid}''\). We could consider a broad class of schemes that produce distinct ciphertexts with overwhelming probability, but we favour the narrower class of schemes satisfying IND-CPA, because we require IND-PA0 (which implies IND-CPA) for bid secrecy.

Lemma 1. Suppose \(\Pi\) is an asymmetric encryption scheme with perfect correctness that satisfies IND-CPA. We have \(\text{Enc2Bid}''(\Pi)\) is an auction scheme (i.e., correctness, completeness and injectivity are satisfied).

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

2.2 Bid secrecy

Our informal definition of bid secrecy (§1) omits a side condition which is necessary for satisfiability. In particular, we did not stress that a losing bidder can be linked to a price when a link can be deduced from the winning bidder and knowledge about the distribution of prices bidders are willing to pay. For example, suppose Alice, Bob and Charlie participate in an auction and Alice wins. Bob and Charlie can both deduce the price the other is willing to pay. Accordingly, our definitions must concede that auction results reveal partial information about prices bidders are willing to pay, hence, we refine our informal definition of bid secrecy as follows:

A losing bidder cannot be linked to a price, except when a link can be deduced from auction results and any partial knowledge about the distribution of prices bidders are willing to pay.

This refinement ensures that the aforementioned example does not constitute a violation of bid secrecy. And we formalise the refined notion of bid secrecy as an
indistinguishability game between an adversary and a challenger.\(^5\) Our game captures a setting where the challenger generates a key pair using the Setup algorithm, publishes the public key, and only uses the private key for opening.

The adversary has access to a left-right oracle which can compute bids on the adversary’s behalf.\(^6\) Bids can be computed by the left-right oracle in two ways, corresponding to a randomly chosen bit \(\beta\). If \(\beta = 0\), then, given a pair of prices \(p_0, p_1\), the oracle outputs a bid for \(p_0\). Otherwise (\(\beta = 1\)), the oracle outputs a bid for \(p_1\). The left-right oracle essentially allows the adversary to control the distribution of prices in bids, but bids computed by the oracle are always computed using the prescribed Bid algorithm. The adversary outputs a bulletin board (this may contain bids output by the oracle and bids generated by the adversary), which is opened by the challenger to reveal price \(p\), set of winning bids \(b\), and non-interactive proof \(pf\) of correct opening. Using these values, the adversary must determine whether \(\beta = 0\) or \(\beta = 1\).

To avoid trivial distinctions, we insist that a bid for price \(p\) was not output by the left-right oracle, assuming \(p\) is the winning price. This assumption is required to capture attacks that exploit poorly designed Open algorithms, in particular, we cannot assume that Open outputs the winning price, because algorithm Open might have been designed maliciously or might contain a design flaw. We ensure winning bids were not output by the left-right oracle using a logical proposition. The proposition uses predicate \(\text{correct-price}(pk, np, bb, p, \kappa)\), which holds when: \((p = 0 \lor (\exists r . \text{Bid}(pk, np, p, \kappa; r) \in bb \setminus \{\bot\} \land 1 \leq p \leq np)) \land (\neg \exists p', r' . \text{Bid}(pk, np, p', \kappa; r') \in bb \setminus \{\bot\} \land p < p' \leq np)\). Intuitively, the predicate holds when price \(p\) has been correctly computed: when there exists a bid for price \(p\) on the bulletin board and there is no bid for a higher price, i.e., when \(p\) is the winning price. Moreover, injectivity ensures that the bid was created for that price.\(^7\)

By design, our notion of bid secrecy is satisfiable by auction schemes which reveal losing prices,\(^8\) assuming that these prices cannot be linked to bidders, except in the aforementioned cases. And our construction will produce auction schemes of this type. Hence, to avoid trivial distinctions, we insist, for each price \(p\), that the number of bids on the bulletin board produced by the left-right oracle with left input \(p\), is equal to the number of bids produced by the left-right oracle with right input \(p\). This can be formalised using predicate \(\text{balanced}(bb, np, L)\), which holds when: for all prices \(p \in \{1, \ldots, np\}\) we have \(|\{b \mid b \in bb \land (b, p, p_1) \in L\}| = |\{b \mid b \in bb \land (b, p_0, p) \in L\}|\), where \(L\) is the set

---

\(^5\) Games are algorithms that output 0 or 1. An adversary wins a game by causing it to output 1. We denote an adversary’s success \(\text{Succ}([\text{Exp}()])\) in a game \(\text{Exp}()\) as the probability that the adversary wins, i.e., \(\text{Succ}([\text{Exp}()]) = \Pr[g \leftarrow \text{Exp}() : g = 1]\). Adversaries are assumed to be stateful, i.e., information persists across invocations of the adversary in a single game, in particular, the adversary can access earlier assignments.

\(^6\) Bellare et al. introduced left-right oracles in the context of symmetric encryption [BDJR97]. And Bellare & Rogaway provide a tutorial on their use [BR05].

\(^7\) The existential quantifiers in \(\text{correct-price}\) demonstrate the importance of defining injectivity perfectly rather than computationally. In particular, \(\text{correct-price}\) cannot interpret a bid for more than one price.
of oracle call inputs and outputs.

Intuitively, if the adversary loses the game, then the adversary is unable to
distinguish between bids for different prices, assuming that a bid is not a winning
bid; it follows that losing prices cannot be linked to bidders. By comparison, if
the adversary wins the game, then there exists a strategy to distinguish honestly
cast bids.

**Definition 3 (Bid secrecy).** Let $\Sigma = (\text{Setup}, \text{Bid}, \text{Open}, \text{Verify})$ be an auction
scheme, $A$ be an adversary, $\kappa$ be a security parameter, and $\text{Bid-Secrecy}(\Sigma, A, \kappa)$
be the following game.\(^9\)

$$\text{Bid-Secrecy}(\Sigma, A, \kappa) = (pk, sk, mb, mp) \leftarrow \text{Setup}(\kappa);$$
$$\beta \leftarrow \{0, 1\}; L \leftarrow \emptyset;$$
$$np \leftarrow A(pk, \kappa); bb \leftarrow A^a(\emptyset);$$
$$(p, b, pf) \leftarrow \text{Open}(sk, np, bb, \kappa);$$
$$g \leftarrow A(p, b, pf);$$

if $g = \beta \land \text{balanced}(bb, np, L) \land |bb| \leq mb \land np \leq mp$
and $(\text{correct-price}(pk, np, bb, p, \kappa) \Rightarrow \forall b \in bb . (b, p, p_1) \not\in L \land (b, p_0, p) \not\in L)$
then
1. return 1
else
L. return 0

Oracle $O$ is defined as follows.\(^10\)

- $O(p_0, p_1)$ computes $b \leftarrow \text{Bid}(pk, np, p_\beta, \kappa); L \leftarrow L \cup \{(b, p_0, p_1)\}$ and outputs $b$, where $p_0, p_1 \in \{1, \ldots, np\}$.

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

Roughly speaking, our definition of bid secrecy corresponds to a symbolic bid secrecy definition by Dreier, Lafourcade & Lakhnech [DLL13, Definition 15].

### 2.3 Auction verifiability

We formalise individual and universal verifiability as games between an adversary and a challenger. Our definitions are based upon analogous definitions for election schemes by Smyth, Frink & Clarkson [SFC16] (cf. Section 5.1).\(^11\)

---

\(^8\)The class of auctions that reveal losing prices seem to have been first considered by Franklin & Reiter [FR96]. Auctions that do not reveal losing prices have also been considered, e.g., [HTK98].

\(^9\)We write $x \leftarrow R S$ for the assignment to $x$ of an element chosen uniformly at random from set $S$.

\(^10\)The oracle may access game parameters, e.g., $pk$. Henceforth, we allow oracles to access game parameters without an explicit mention.

\(^11\)We discuss our motivation to base the definitions on the notions of verifiability by Smyth, Frink & Clarkson in Section 6.3.
2.3.1 Individual verifiability

Individual verifiability challenges the adversary to generate a collision from algorithm Bid. If the adversary cannot win, then bidders can uniquely identify their bids, hence, bidders can check whether their bid is included.

**Definition 4** (Individual verifiability). Let $\Sigma = (\text{Setup}, \text{Bid}, \text{Open}, \text{Verify})$ be an auction scheme, $A$ be an adversary, $\kappa$ be a security parameter, and $\text{Exp-IV}(\Sigma, A, \kappa)$ be the following game.

\[
\text{Exp-IV}(\Sigma, A, \kappa) =
\begin{align*}
(pk, np, p, p') &\leftarrow A(\kappa); \\
b &\leftarrow \text{Bid}(pk, np, p, \kappa); \\
b' &\leftarrow \text{Bid}(pk, np, p', \kappa); \\
\text{if } b = b' \land b \neq \bot \land b' \neq \bot &\text{ then return 1} \\
\text{else return 0}
\end{align*}
\]

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

Individual verifiability resembles injectivity, but game $\text{Exp-IV}$ allows an adversary to choose the public key and prices, whereas there is no adversary in the definition of injectivity (the public key is an output of algorithm $\text{Setup}$ and prices are universally quantified, under the restriction that prices are distinct).

2.3.2 Universal verifiability

Universal verifiability challenges the adversary to concoct a scenario in which Verify accepts, but the winning price or the set of winning bids is not correct.\(^\text{12}\) Formally, we check the validity of the winning price using predicate $\text{correct-price}$. And we check the validity of the set of winning bids using predicate $\text{correct-bids}(pk, np, bb, p, b, \kappa)$, which holds when $b = bb \cap \{ b \mid b = \text{Bid}(pk, np, p, b, \kappa) \}$, i.e., it holds when $b$ is the intersection of the bulletin board and the set of all bids for the winning price. Since function $\text{correct-price}$ will be parameterised with a public key constructed by the adversary, rather than one constructed by algorithm $\text{Setup}$ (§2.2), we must adopt a stronger definition of injectivity which holds for adversarial keys. We define strong injectivity in Appendix B.1.

**Definition 5** (Universal verifiability). Let $\Sigma = (\text{Setup}, \text{Bid}, \text{Open}, \text{Verify})$ be an auction scheme satisfying strong injectivity, $A$ be an adversary, $\kappa$ be a security parameter, and $\text{Exp-UV}(\Sigma, A, \kappa)$ be the following game.

\(^{12}\)Universal verifiability captures a notion of non-repudiation, i.e., anyone can check whether the winning bids encapsulate the winning price.
\[ \text{Exp-UV}(\Sigma, A, \kappa) = \]
\[
(pk, np, bb, p, b, pf) \leftarrow A(\kappa);
\]
\[
\text{if } (\neg \text{correct-price}(pk, np, bb, p, \kappa) \lor \neg \text{correct-bids}(pk, np, bb, p, b, \kappa)) \land
\]
\[
\text{Verify}(pk, np, bb, p, b, pf, \kappa) = 1 \text{ then return 1}
\]
\[
\text{else return 0}
\]
We say \( \Sigma \) satisfies \textit{universal verifiability} (Exp-UV), if for all probabilistic polynomial-time adversaries \( A \), there exists a negligible function \( \text{negl} \), such that for all security parameters \( \kappa \), we have \( \text{Succ}(\text{Exp-UV}(\Sigma, A, \kappa)) \leq \text{negl}(\kappa) \).

3 Auctions from elections

3.1 Election scheme syntax

We recall syntax for \textit{election schemes} [SFC16], which capture the class of elections that consist of the following four steps. First, a tallier generates a key pair. Secondly, each voter constructs and casts a ballot for their choice. These ballots are recorded on a bulletin board. Thirdly, the tallier tallies the recorded ballots and announces an outcome, i.e., a distribution of choices. This distribution is used to select a representative. For example, in first-past-the-post elections the representative corresponds to the choice with highest frequency. Finally, voters and other interested parties check that the outcome corresponds to votes expressed in recorded ballots.\(^{13}\) This class of elections includes state-of-the-art schemes such as the Helios family, but notably excludes schemes reliant on paper, e.g., Prêt à Voter [CRS05], Scantegrity II [CCC+08], and Remoteegrity [ZCC+13].

**Definition 6** (Election scheme [SFC16]). An election scheme is a tuple of efficient algorithms \((\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})\) such that:

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

\[
\text{Vote, denoted } b \leftarrow \text{Vote}(pk, nc, v, \kappa), \text{ is run by voters. Vote takes as input a public key } pk, \text{ some number of candidates } nc, \text{ a voter's vote } v, \text{ and a security parameter } \kappa. \text{ A voter's vote should be selected from a sequence } 1, \ldots, nc \text{ of candidates.}^{14} \text{ Vote outputs a ballot } b \text{ or error symbol } \perp.
\]

\[
\text{Tally, denoted } (v, pf) \leftarrow \text{Tally}(sk, nc, bb, \kappa), \text{ is run by the tallier. Tally takes as input a private key } sk, \text{ some number of candidates } nc, \text{ a bulletin board}
\]

\(^{13}\)Smyth, Frink & Clarkson use the syntax to model first-past-the-post voting systems and Smyth shows the syntax is sufficiently versatile to capture ranked-choice voting systems too [Smy17].

\(^{14}\)Votes are (abstractly) modelled as integers, rather than alphanumeric strings (such as representatives' names), for brevity.
and a security parameter $\kappa$, where $\mathbb{b}$ is a set. It outputs an election outcome $v$ and a non-interactive proof $pf$ 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 $v$.

$\text{Verify}$, denoted $s \leftarrow \text{Verify}(pk, nc, \mathbb{b}, v, pf, \kappa)$, is run to audit an election. It takes as input a public key $pk$, some number of candidates $nc$, a bulletin board $\mathbb{b}$, 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 and 0 otherwise.

Election schemes must satisfy correctness, completeness, and injectivity, which we define in Appendix B.2.

3.1.1 Example: $\text{Enc2Vote}$

We demonstrate the applicability of our syntax using a construction ($\text{Enc2Vote}$) for election schemes from asymmetric encryption schemes.\[16\]

**Definition 7 ($\text{Enc2Vote}$).** Given an asymmetric encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$, we define $\text{Enc2Vote}(\Pi)$ as follows.

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

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

To ensure $\text{Enc2Vote}(\Pi)$ is an election scheme, we require asymmetric encryption scheme $\Pi$ to produce distinct ciphertexts with overwhelming probability (cf. §2.1.1). Hence, we restrict the class of asymmetric encryption schemes used to instantiate $\text{Enc2Vote}$.

**Lemma 2.** Suppose $\Pi$ is an asymmetric encryption scheme with perfect correctness that satisfies $\text{IND-CPA}$. We have $\text{Enc2Vote}(\Pi)$ is an election scheme.

\[15\text{Let } v[v] \text{ denote component } v \text{ of vector } v.\]

\[16\text{The construction was originally presented by Bernhard et al. [SB14, SB13, BPW12b, BCP+11] in a slightly different setting.}\]
3.1.2 Comparing auction and election schemes

Auction schemes are distinguished from election schemes as follows: auction schemes open the bulletin board to recover the winning price and winning bids, whereas, election schemes tally the bulletin board to recover the distribution of votes. Our goal is to bridge this gulf; we introduce reveal algorithms to do so.

3.2 Reveal algorithm

To achieve the functionality required to construct auction schemes from election schemes, we define reveal algorithms which link a vote to a set of ballots for that vote, given the tallier’s private key. We stress that ballot secrecy does not prohibit the existence of such algorithms, because ballot secrecy asserts that the tallier’s private key cannot be derived by the adversary.

Definition 8 (Reveal algorithm). A reveal algorithm is an efficient algorithm Reveal defined as follows:

\[
\text{Reveal}, \text{ denoted } b \leftarrow \text{Reveal}(sk, nc, bb, v, \kappa), \text{ is run by the tallier. Reveal takes as input a private key } sk, \text{ some number of candidates } nc, \text{ a bulletin board } bb, \text{ a vote } v, \text{ and a security parameter } \kappa. \text{ It outputs a set of ballots } b.
\]

Let \( \Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}) \) be an election scheme. A reveal algorithm is correct with respect to \( \Gamma \), if there exists a negligible function \( \text{negl} \), such that for all security parameters \( \kappa \), integers \( nb \) and \( nc \), and votes \( v, v_1, \ldots, v_nb \in \{1, \ldots, nc\} \), it holds that:

\[
\text{Pr}\left[ (pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa); \right.
\text{ for } 1 \leq i \leq nb \text{ do }
\text{ } b_i \leftarrow \text{Vote}(pk, nc, v_i, \kappa);
\text{ } b \leftarrow \text{Reveal}(sk, nc, \{b_1, \ldots, b_nb\}, v, \kappa)
\text{ } : nb \leq mb \land nc \leq mc \Rightarrow b = \{b_i \mid v_i = v \land 1 \leq i \leq nb\} > 1 - \text{negl}(\kappa).
\]

Reveal algorithms are run by talliers to disclose sets of ballots for a specific vote. Hence, we extend the tallier’s role to include the execution of a reveal algorithm (cf. Section 1.1), thereby bridging the gap between elections and auctions. It is natural to consider whether this extension is meaningful, i.e., given an arbitrary election scheme, does there exist a reveal algorithm that is correct with respect to that election scheme? We answer this question positively in Appendix D.

3.3 Construction

We show how to construct auction schemes from election schemes. We first describe a construction which can produce auction schemes satisfying bid secrecy (§3.3.1). Building upon this result, we present our second construction which can produce auction schemes satisfying bid secrecy and auction verifiability (§3.3.2).
3.3.1 Non-verifiable auction schemes

Our first construction follows intuitively from our informal description (§3.3.1). Algorithm Bid is derived from Vote, simply by representing prices as candidates. Algorithm Open uses algorithm Tally to derive the distribution of prices and the winning price is determined from this distribution. Moreover, we exploit a reveal algorithm Reveal to disclose the set of winning bids.

**Definition 9.** Given an election scheme \( \Gamma = (\text{Setup}_\Gamma, \text{Vote}, \text{Tally}, \text{Verify}_\Gamma) \) and a reveal algorithm Reveal, we define \( \Lambda(\Gamma, \text{Reveal}) = (\text{Setup}_\Lambda, \text{Bid}, \text{Open}, \text{Verify}_\Lambda) \) as follows.

\[
\text{Setup}_\Lambda(\kappa) \text{ computes } (pk, sk, mb, mc) \leftarrow \text{Setup}_\Gamma(\kappa) \text{ and outputs } (pk, sk, mb, mc).
\]

\[
\text{Bid}(pk, np, p, \kappa) \text{ computes } b \leftarrow \text{Vote}(pk, np, p, \kappa) \text{ and outputs } b.
\]

\[
\text{Open}(sk, np, bb, \kappa) \text{ proceeds as follows. Computes } (v, pf) \leftarrow \text{Tally}(sk, np, bb, \kappa).
\]

Finds the largest integer \( p \) such that \( v[p] > 0 \land 1 \leq p \leq np \), outputting \((0, \emptyset, \epsilon)\) if no such integer exists. Computes \( b \leftarrow \text{Reveal}(sk, np, bb, p, \kappa) \) and \( pf' \leftarrow \text{Prove}((pk, np, bb, p, b, \kappa), sk, \kappa) \), and outputs \((p, b, (v, pf, pf'))\).

\[
\text{Verify}_\Lambda(pk, np, bb, p, b, \sigma, \kappa) \text{ proceeds as follows. Parses } \sigma \text{ as } (v, pf, pf'), \text{ outputting } 0 \text{ if parsing fails. The algorithm performs the following checks:}
\]

Algorithm Open requires \( \epsilon \) to be a constant symbol.

**Lemma 3.** Let \( \Gamma \) be an election scheme and Reveal be a reveal algorithm. Suppose Reveal is correct with respect to \( \Gamma \). We have \( \Lambda(\Gamma, \text{Reveal}) \) is an auction scheme.

3.3.2 Verifiable auction schemes

Our second construction extends our first construction to ensure verifiability, in particular, algorithm Open is extended to include a proof of correct tallying and a proof of correct revealing. Moreover, algorithm Verify is used to check proofs.

**Definition 10.** Given an election scheme \( \Gamma = (\text{Setup}_\Gamma, \text{Vote}, \text{Tally}, \text{Verify}_\Gamma) \), a reveal algorithm Reveal, and a non-interactive proof system \( \Delta = (\text{Prove}, \text{Verify}) \), we define \( \Lambda(\Gamma, \text{Reveal}, \Delta) = (\text{Setup}_\Lambda, \text{Bid}, \text{Open}, \text{Verify}_\Lambda) \) as follows.

\[
\text{Setup}_\Lambda(\kappa) \text{ computes } (pk, sk, mb, mc) \leftarrow \text{Setup}_\Gamma(\kappa) \text{ and outputs } (pk, sk, mb, mc).
\]

\[
\text{Bid}(pk, np, p, \kappa) \text{ computes } b \leftarrow \text{Vote}(pk, np, p, \kappa) \text{ and outputs } b.
\]

\[
\text{Open}(sk, np, bb, \kappa) \text{ proceeds as follows. Computes } (v, pf) \leftarrow \text{Tally}(sk, np, bb, \kappa).
\]

Finds the largest integer \( p \) such that \( v[p] > 0 \land 1 \leq p \leq np \), outputting \((0, \emptyset, \epsilon)\) if no such integer exists. Computes \( b \leftarrow \text{Reveal}(sk, np, bb, p, \kappa) \) and \( pf' \leftarrow \text{Prove}((pk, np, bb, p, b, \kappa), sk, \kappa) \), and outputs \((p, b, (v, pf, pf'))\).

\[
\text{Verify}_\Lambda(pk, np, bb, p, b, \sigma, \kappa) \text{ proceeds as follows. Parses } \sigma \text{ as } (v, pf, pf'), \text{ outputting } 0 \text{ if parsing fails. The algorithm performs the following checks:}
\]
1. Checks that $\text{Verify}_\Gamma((pk, np, bb, v, pf, \kappa)) = 1$.

2. Checks that $p$ is the largest integer such that $v[p] > 0 \land 1 \leq p \leq np$ or there is no such integer and $(p, b, pf') = (0, \emptyset, \epsilon)$.

3. Checks that $\text{Verify}((pk, np, bb, p, b, \kappa), pf', \kappa) = 1$.

Outputs 1 if all of the above checks hold, and outputs 0 otherwise.

Algorithms Tally and Verify require $\epsilon$ to be a constant symbol.

To ensure that our construction produces auction schemes, the non-interactive proof system must be defined for a suitable relation. We define it as follows.

**Definition 11.** Given an election scheme $\Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$ and a reveal algorithm $\text{Reveal}$, we define binary relation $R(\Gamma, \text{Reveal})$ over vectors of length 6 and bitstrings such that $((pk, nc, bb, v, b, \kappa), sk) \in R(\Gamma, \text{Reveal})$ ⇔ $\exists mb, mc, r, r' . b = \text{Reveal}(sk, nc, bb, v, b, \kappa; r) \land (pk, sk, mb, mc) = \text{Setup}(\kappa; r') \land 1 \leq v \leq nc \leq mc \land |bb| \leq mb$.

**Lemma 4.** Let $\Gamma$ be an election scheme, $\text{Reveal}$ be a reveal algorithm, and $\Delta$ be a non-interactive proof system for relation $R(\Gamma, \text{Reveal})$. Suppose $\text{Reveal}$ is correct with respect to $\Gamma$. We have $\Lambda(\Gamma, \text{Reveal}, \Delta)$ is an auction scheme.

Next, we study the security of auction schemes produced by our constructions, in particular, we present conditions under which our constructions produce auction schemes satisfying bid secrecy and verifiability.

### 4 Privacy results

We introduce a definition of ballot secrecy which is sufficient to ensure that our construction produces auction schemes satisfying bid secrecy (assuming some soundness conditions on the underlying election scheme and reveal algorithm).\(^{17}\)

#### 4.1 Ballot secrecy

Our definition of ballot secrecy strengthens an earlier definition by Smyth [Smy16].\(^{18,19}\)

**Definition 12** (Ballot secrecy). Let $\Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$ be an election scheme, $A$ be an adversary, $\kappa$ be a security parameter, and $\text{Ballot-Secrecy}(\Gamma, A, \kappa)$ be the following game.

---

\(^{17}\)Our privacy results could be extended to other definitions of bid secrecy and ballot secrecy, by modifying our proofs.

\(^{18}\)We discuss the suitability of Smyth’s definition in Section 6.3.

\(^{19}\)Quaglia & Smyth present a tutorial-style introduction to modelling ballot secrecy [QS17], and Smyth provides a technical introduction [Smy18a].
Ballot-Secrecy(\(\Gamma, \mathcal{A}, \kappa\)) =
\[
\begin{align*}
(pk, sk, mb, mc) & \leftarrow \text{Setup}(\kappa); \\
\beta & \leftarrow_R \{0, 1\}; L \leftarrow \emptyset; W \leftarrow \emptyset; \\
nc & \leftarrow \mathcal{A}(pk, \kappa); \mathbb{b} & \leftarrow \mathcal{A}^O(); \\
(v, pf) & \leftarrow \text{Tally}(sk, nc, \mathbb{b}, \kappa); \\
& \text{for } b \in \mathbb{b} \land (b, v_0, v_1) \notin L \text{ do} \\
& \quad (v', pf') \leftarrow \text{Tally}(sk, nc, \{b\}, \kappa); \\
& \quad W \leftarrow W \cup \{(b, v')\}; \\
g & \leftarrow \mathcal{A}(v, pf, W); \\
& \text{if } g = \beta \land \text{balanced}(\mathbb{b}, nc, L) \land |\mathbb{b}| \leq mb \land nc \leq mc \text{ then} \\
& \quad \text{return } 1 \\
& \quad \text{else} \\
& \quad \text{return } 0
\end{align*}
\]

Oracle \(\mathcal{O}\) is defined as follows:
- \(\mathcal{O}(v_0, v_1)\) computes \(b \leftarrow \text{Vote}(pk, nc, v_\beta, \kappa); L \leftarrow L \cup \{(b, v_0, v_1)\}\) 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 \(\mathcal{A}\), there exists a negligible function \(\text{negl}\), such that for all security parameters \(\kappa\), we have \(\text{Succ}(\text{Ballot-Secrecy}(\Gamma, \mathcal{A}, \kappa)) \leq \frac{1}{2} + \text{negl}(\kappa)\).

Our formalisation of ballot secrecy challenges an adversary to determine whether the left-right oracle produces ballots for “left” or “right” inputs. In addition to the oracle’s outputs, the adversary is given the election outcome and tallying proof derived by tallying the board (intuitively, this 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 adversary is also given a mapping, denoted \(W\) in Definition 12, from ballots to votes, for all ballots on the bulletin board which were not output by the oracle. To avoid trivial distinctions, we insist that oracle queries are balanced, i.e., predicate balanced must hold. Intuitively, if the adversary does not succeed, then ballots for different votes cannot be distinguished, hence, a voter cannot be linked to a vote, i.e., ballot secrecy is preserved. By comparison, if the adversary succeeds, then ballots can be distinguished and ballot secrecy is not preserved.

Comparing notions of ballot secrecy. The adversary is given the outcome corresponding to tallying any ballot that was not computed by the oracle in Ballot-Secrecy, whereas the adversary does not have this capability in the definition by Smyth [Smy16]. Formally, the definition of ballot secrecy by Smyth, henceforth Smy-Ballot-Secrecy, can be derived from Ballot-Secrecy by removing the for-loop and replacing \(\mathcal{A}(v, pf, W)\) with \(\mathcal{A}(v, pf)\). It is trivial to see that Ballot-Secrecy strengthens Smy-Ballot-Secrecy, because any adversary against Smy-Ballot-Secrecy (without access to \(W\)) is also an adversary against
Ballot-Secrecy (with access to \(W\)). Moreover, Ballot-Secrecy is strictly stronger (Proposition 5), because the outcome might leak information.\(^{20}\)

**Proposition 5.** Ballot-Secrecy is strictly stronger than Smy-Ballot-Secrecy.

Nonetheless, we present conditions under which the two notions coincide (§6.2).

### 4.1.1 Example: Enc2Vote satisfies ballot secrecy

Intuitively, given an encryption scheme \(\Pi\) satisfying non-malleability, the election scheme \(\text{Enc2Vote}(\Pi)\) derives ballot secrecy from the encryption scheme until tallying and tallying maintains ballot secrecy by only disclosing the number of votes for each candidate. Formally, the following holds.\(^{21}\)

**Proposition 6.** Suppose \(\Pi\) is an asymmetric encryption scheme with perfect correctness. If \(\Pi\) satisfies IND-PA0, then \(\text{Enc2Vote}(\Pi)\) satisfies Ballot-Secrecy.

### 4.2 Relations between ballot and bid secrecy

The main distinctions between our formalisations of privacy for elections and auctions are as follows.

1. Our ballot secrecy game tallies the bulletin board, whereas our bid secrecy game opens the bulletin board.
2. Our ballot secrecy game is intended to protect the privacy of all voters, whereas our bid secrecy game is only intended to protect the privacy of losing bidders.
3. Our ballot secrecy game provides the adversary with the vote corresponding to any ballot that was not computed by the oracle, whereas the adversary is not given a similar mapping in our bid secrecy game.

These distinctions support our intuition: we can construct auction schemes satisfying bid secrecy from election schemes satisfying ballot secrecy. Yet, interestingly, ballot secrecy alone is insufficient to ensure that our construction produces auction schemes satisfying bid secrecy. This is because our construction is reliant upon the underlying tally algorithm, and a poorly designed tally algorithm could lead to the construction of auction schemes that do not satisfy bid secrecy. In particular, a tally algorithm that outputs an incorrect winning price (in the presence of an adversary) can cause the set of bids for this price to be disclosed, thereby enabling losing bidders that bid at this price to be identified, which violates bid secrecy. (E.g., suppose Alice, Bob and Charlie bid for 1, 2 and 3, respectively. And suppose 2 is incorrectly announced as the winning price.)

\(^{20}\)Parallel decryption leaks information similarly [BPW12b, Appendix A].

\(^{21}\)Bellare & Sahai [BS99, §5] show that their notion of non-malleability (\(\text{CNM-CPA}\)) coincides with a simpler indistinguishability notion (IND-PA0), thus it suffices to consider IND-PA0 in Proposition 6.
price. Hence, Bob is linked to his bid, which violates bid secrecy.) Tallying algorithms that output incorrect winning prices can satisfy correctness, because correctness does not consider an adversary, hence, tallying might produce correct output under ideal conditions and incorrect output in the presence of an adversary. This leads to a separation result (cf. Appendix E). Moreover, our construction is also reliant upon the underlying reveal algorithm, which might output an incorrect set of ballots (in the presence of an adversary), hence, there is a further separation. Nevertheless, we can formulate soundness conditions (that must hold in the presence of adversaries) which capture a class of election schemes for which our intuition holds.

Weak tally soundness. Correctness for election schemes ensures that algorithm Tally produces the expected election outcome under ideal conditions. A similar property, which we call weak tally soundness, can hold in the presence of an adversary. Our formulation of weak tally soundness (Definition 13) challenges the adversary to concoct a scenario in which the election outcome does not include the votes of all ballots on the bulletin board that were produced by Vote.22 We capture correct election outcomes using function correct-outcome,23 which is defined such that for all \( pk, nc, bb, \kappa, \ell \), and \( v \in \{1, \ldots, nc\} \), we have \( \text{correct-outcome}(pk, nc, bb, \kappa)[v] = \ell \iff \exists b \in bb \setminus \{\perp\} : \exists r : b = \text{Vote}(pk, nc, v, \kappa; r) \). 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 \( v \).

**Definition 13 (Weak tally soundness).** Let \( \Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}) \) be an election scheme, \( A \) be an adversary, \( \kappa \) be a security parameter, and \( \text{W-Tally-Soundness}(\Gamma, A, \kappa) \) be the following game.

\[
\text{W-Tally-Soundness}(\Gamma, A, \kappa) = \begin{cases} 
(pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa); \\
(nc, bb) \leftarrow A(pk, \kappa); \\
(v, pf) \leftarrow \text{Tally}(sk, nc, bb, \kappa); \\
\text{if } \exists v \in \{1, \ldots, nc\} \cdot v[v] < \text{correct-outcome}(pk, nc, bb, \kappa)[v] \wedge |bb| \leq mb \wedge nc \leq mc \text{ then} \\
\quad \text{return } 1 \\
\text{else} \\
\quad \text{return } 0 
\end{cases}
\]

22Our construction produces auction schemes satisfying bid secrecy when the underlying tallying algorithm produces election outcomes containing too many votes for some candidates. (E.g., suppose Alice, Bob and Charlie bid for 1, 2 and 3, respectively. And suppose 4 is incorrectly announced as the winning price. In this case no bids are linked to that price, thus bid secrecy is preserved.) And weak tally soundness can be satisfied by such schemes. Thus, there is a further distinction between weak tally soundness and correctness.

23Function \( \text{correct-outcome} \) uses a counting quantifier [Sch05] denoted \( \exists^\ell \). Predicate \( (\exists^\ell x : P(x)) \) 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.
We say $\Gamma$ satisfies weak tally soundness ($W$-Tally-Soundness), if for all probabilistic polynomial-time adversaries $A$, there exists a negligible function $\text{negl}$, such that for all security parameters $\kappa$, we have $\text{Succ}(W\text{-Tally-Soundness}(\Gamma, A, \kappa)) \leq \text{negl}(\kappa)$.

**Reveal soundness.** Correctness for reveal algorithms ensures that algorithm $\text{Reveal}$ produces the set of ballots for a particular vote under ideal conditions. A similar property, which we call reveal soundness, can hold in the presence of an adversary. Our formulation of reveal soundness challenges the adversary to output a vote and bulletin board for which the reveal algorithm produces a set of ballots that does not coincide with the set of ballots on the bulletin board that tally to that vote.

**Definition 14** (Reveal soundness). Let $\Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$ be an election scheme, $\text{Reveal}$ be a reveal algorithm, $A$ be an adversary, $\kappa$ be a security parameter, and $\text{Reveal-Soundness}(\Gamma, A, \kappa)$ be the following game.

$$\text{Reveal-Soundness}(\Gamma, A, \kappa) = \left( \begin{array}{l}
(pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa); \\
(nc, bb, v) \leftarrow A(pk, \kappa);
\end{array} \right)
$$

$$b \leftarrow \text{Reveal}(sk, nc, bb, v, \kappa);$$

W $\leftarrow \emptyset$;

for $b \in bb$

$$\left( \begin{array}{l}
(v, pf) \leftarrow \text{Tally}(sk, nc, \{b\}, \kappa);
\end{array} \right)$$

$$W \leftarrow W \cup \{(b, v)\};$$

if $b \not\in \{b \mid (b, v) \in W \land v[v] = 1\} \land |bb| \leq mb \land 1 \leq v \leq nc \leq mc$ then

| return 1

else

| return 0

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

**Lemma 7.** Let $\Gamma$ be an election scheme and $\text{Reveal}$ be a reveal algorithm. If $\text{Reveal}$ satisfies reveal soundness with respect to $\Gamma$, then $\text{Reveal}$ is correct with respect to $\Gamma$.

### 4.2.1 Bid secrecy for non-verifiable auction schemes

We prove that our construction presented in Section 3.3.1 produces auction schemes satisfying bid secrecy, assuming the underlying election scheme satisfies ballot secrecy and weak tally soundness, and the underlying reveal algorithm satisfies reveal soundness.

**Proposition 8.** Let $\Gamma$ be an election scheme and $\text{Reveal}$ be a reveal algorithm. If $\Gamma$ satisfies Ballot-Secrecy and $W$-Tally-Soundness, and $\text{Reveal}$ satisfies reveal soundness with respect to $\Gamma$, then $\Lambda(\Gamma, \text{Reveal})$ satisfies Bid-Secrecy.
We demonstrate the applicability of our result in the following example.

**Example: Enc2Bid satisfies bid secrecy**

Intuitively, given a non-malleable asymmetric encryption scheme $\Pi$, auction scheme $\text{Enc2Bid}(\Pi)$ derives bid secrecy from the encryption scheme until opening and opening maintains bid secrecy by only disclosing winning bids and the winning price. We can use our construction and accompanying security results to formally prove this result.

**Proposition 9.** Suppose $\Pi$ is an asymmetric encryption scheme with perfect correctness. If $\Pi$ satisfies $\text{IND-PA0}$, then $\text{Enc2Bid}(\Pi)$ satisfies $\text{Bid-Secrecy}$.

**Proof.** Let us suppose there exists a reveal algorithm $\text{Reveal-Enc2Bid}(\Pi)$ such that $\text{Enc2Bid}(\Pi)$ is equivalent to $\Lambda(\text{Enc2Vote}(\Pi), \text{Reveal-Enc2Bid}(\Pi))$. Hence, we can use Proposition 8 to prove that $\text{Enc2Bid}(\Pi)$ satisfies bid secrecy. We defer formalising a suitable reveal algorithm to Appendix C.

4.2.2 Bid secrecy for verifiable auction schemes

We generalise Proposition 8 to verifiable auction schemes, assuming the non-interactive proof system is zero-knowledge.

**Theorem 10.** Let $\Gamma$ be an election scheme, $\text{Reveal}$ be a reveal algorithm, and $\Delta$ be a non-interactive proof system for relation $\mathcal{R}(\Gamma, \text{Reveal})$. If $\Gamma$ satisfies $\text{Ballot-Secrecy}$ and $\text{W-Tally-Soundness}$, $\text{Reveal}$ satisfies reveal soundness with respect to $\Gamma$, and $\Delta$ is zero-knowledge, then $\Lambda(\Gamma, \text{Reveal}, \Delta)$ satisfies $\text{Bid-Secrecy}$.

In Section 5.1.2, we show that weak tally soundness is implied by universal verifiability, hence, a special case of the above theorem requires that $\Gamma$ satisfies universal verifiability, rather than weak tally soundness.

5 Verifiability results

We recall definitions of verifiability by Smyth, Frink & Clarkson [SFC16].\footnote{We discuss the suitability of the definition by Smyth, Frink & Clarkson in Section 6.3.}\footnote{Quaglia & Smyth present a tutorial-style introduction to modelling verifiability [QS17], and Smyth provides a technical introduction [Smy18a].}

We show that these definitions are sufficient to ensure that our construction produces schemes satisfying auction verifiability.

5.1 Election verifiability

5.1.1 Individual verifiability

Individual verifiability challenges the adversary to generate a collision from algorithm $\text{Vote}$. 
Definition 15 (Individual verifiability [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 $\text{Exp-IV-Ext}(\Gamma, A, \kappa)$ be the following game.

$$\text{Exp-IV-Ext}(\Gamma, A, \kappa) =$$
$$\begin{align*}
(pk, nc, v, v') &\leftarrow A(\kappa); \\
b &\leftarrow \text{Vote}(pk, nc, v, \kappa); \\
b' &\leftarrow \text{Vote}(pk, nc, v', \kappa); \\
\text{if } b = b' \land b \neq \bot \land b' \neq \bot &\text{ then} \\
&\text{return } 1 \\
\text{else} &\text{return } 0
\end{align*}$$

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

5.1.2 Universal verifiability

Universal verifiability challenges the adversary to concoct a scenario in which $\text{Verify}$ accepts, but the election outcome is not correct. Formally, we capture the correct election outcome using function $\text{correct-outcome}$. Since function $\text{correct-outcome}$ will now be parameterised with a public key constructed by the adversary, rather than a public key constructed by algorithm $\text{Setup}$ (cf. Section 4.2), we must adopt a stronger definition of injectivity which holds for adversarial keys. We define strong injectivity in Appendix B.2.

Definition 16 (Universal verifiability [SFC16]). Let $\Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$ be an election scheme satisfying strong injectivity, $A$ be an adversary, $\kappa$ be a security parameter, and $\text{Exp-UV-Ext}(\Gamma, A, \kappa)$ be the following game.

$$\text{Exp-UV-Ext}(\Gamma, A, \kappa) =$$
$$\begin{align*}
(pk, nc, bb, v, pf) &\leftarrow A(\kappa); \\
\text{if } v \neq \text{correct-outcome}(pk, nc, bb, \kappa) \land \text{Verify}(pk, nc, bb, v, pf, \kappa) = 1 &\text{ then} \\
&\text{return } 1 \\
\text{else} &\text{return } 0
\end{align*}$$

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

Universal verifiability is similar to weak tally soundness, in particular, both notions challenge the adversary to concoct a scenario in which the election outcome is not correct. The election outcome is computed by the challenger using algorithm $\text{Tally}$ in $W$-Tally-Soundness. By comparison, the outcome is chosen by the adversary in $\text{Exp-UV-Ext}$, under the condition that it must be accepted by algorithm $\text{Verify}$. Since completeness asserts that outcomes output by $\text{Tally}$ will be accepted by $\text{Verify}$, we have the following result.
Lemma 11. Let $\Gamma$ be an election scheme. If $\Gamma$ satisfies Exp-UV-Ext, then $\Gamma$ satisfies W-Tally-Soundness.

It is trivial to see that universal verifiability is strictly stronger than weak tally soundness, because Enc2Vote satisfies weak tally soundness (see proof of Proposition 9), but not universal verifiability (it accepts any election outcome).

Corollary 12. Exp-UV-Ext is strictly stronger than W-Tally-Soundness.

The proof of Corollary 12 follows from Lemma 11 and the above reasoning. We omit a formal proof.

5.2 Election verifiability implies auction verifiability

The following results demonstrate that our second construction ($\S$3.3.2) produces verifiable auction schemes from verifiable election schemes.

Theorem 13. Let $\Gamma$ be an election scheme, Reveal be a reveal algorithm, and $\Delta$ be a non-interactive proof system for relation $R(\Gamma, \text{Reveal})$, such that Reveal is correct with respect to $\Gamma$. If $\Gamma$ satisfies Exp-IV-Ext, then $\Lambda(\Gamma, \text{Reveal}, \Delta)$ satisfies Exp-IV.

The proof of Theorem 13 follows from Definitions 4, 10 & 15 and we omit a formal proof.

For universal verifiability, we require the non-interactive proof system to satisfy a notion of soundness. This notion can be captured by the following property on relation $R(\Gamma, \text{Reveal})$.

Definition 17. Given an election scheme $\Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$ and a reveal algorithm Reveal, we say relation $R(\Gamma, \text{Reveal})$ is $\Lambda$-suitable, if $(\text{pk}, \text{np}, \text{bb}, p, b, \kappa, \text{sk}) \in R(\Gamma, \text{Reveal})$ implies correct-bids $(\text{pk}, \text{np}, \text{bb}, p, b, \kappa)$ with overwhelming probability.

Theorem 14. Let $\Gamma$ be an election scheme, Reveal be a reveal algorithm that is correct with respect to $\Gamma$, and $\Delta$ be a non-interactive proof system for relation $R(\Gamma, \text{Reveal})$. If $\Gamma$ satisfies Exp-UV-Ext, $\Delta$ satisfies soundness, and $R(\Gamma, \text{Reveal})$ is $\Lambda$-suitable, then $\Lambda(\Gamma, \text{Reveal}, \Delta)$ satisfies Exp-UV.

6 Scope of results

Given an election scheme $\Gamma$, the scope of key results (Proposition 8 and Theorems 10 & 14) depend on the existence of a reveal algorithm Reveal that is correct with respect to $\Gamma$ and a non-interactive proof system $\Delta$ for relation $R(\Gamma, \text{Reveal})$, such that $\Delta$ satisfies soundness and zero-knowledge, Reveal satisfies reveal soundness with respect to $\Gamma$, and $R(\Gamma, \text{Reveal})$ is $\Lambda$-suitable. We show that suitable reveal algorithms exist for all election schemes in Appendix D. And we demonstrate that suitable non-interactive proof systems exist for a large class of election schemes ($\S$6.1). Furthermore, since our privacy results (Proposition
8 and Theorem 10) depend on a new notion of ballot secrecy, we provide conditions under which the new notion coincides with an existing notion (§6.2). Finally, our results apply to election schemes that satisfy definitions of ballot secrecy (Definition 12) and verifiability (Definitions 15 & 16), and we consider the suitability of these definitions (§6.3).

6.1 Non-interactive proof systems suitable for our construction

Intuitively, a non-interactive proof system for relation \( R(\Gamma, \text{Reveal}) \) must prove \( \text{Reveal}(sk, nc, bb, v, \kappa) \) outputs the set of ballots \( b \) on bulletin board \( bb \) for vote \( v \). Assuming \( \Gamma \) is a verifiable election scheme, this can be achieved by exploiting the scheme’s tallying algorithm \( \text{Tally} \). In particular, if \( b \) is a set of ballots for vote \( v \), then \( \text{Tally}(sk, nc, b, \kappa) \) will output \((v_b, pf_b)\) such that \( v_b \) is a zero-filled vector except for index \( v \) which will contain \(|b|\) (i.e., outcome \( v_b \) contains \(|b|\) votes for candidate \( v \) and no votes for the other candidates) and this can be witnessed by proof \( pf_b \). Moreover, if \( b \) is the set of ballots on the bulletin board \( bb \) for vote \( v \), then \( b \subseteq bb \), and \( \text{Tally}(sk, nc, bb, \kappa) \) will output \((v_{bb}, pf_{bb})\) such that \( v_{bb}[v] = v_b[v] \) and this can be witnessed by proof \( pf_{bb} \). Thus, to prove \( \text{Reveal}(sk, nc, bb, v, \kappa) \) outputs the set of ballots \( b \) on bulletin board \( bb \) for vote \( v \), it suffices to output proofs \( pf_b \) and \( pf_{bb} \). We formalise such a proof system.

**Definition 18.** Given an election scheme \( \Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}_\Gamma) \), we define \( \delta(\Gamma) = (\text{Prove}, \text{Verify}) \) as follows.

\begin{align*}
\text{Prove}(s, sk, \kappa) & \text{ parses } s \text{ as vector } (pk, nc, bb, v, b, \kappa), \text{ outputting } \bot \text{ if parsing fails; computes } (v_{bb}, pf_{bb}) \leftarrow \text{Tally}(sk, nc, bb, \kappa); (v_b, pf_b) \leftarrow \text{Tally}(sk, nc, b, \kappa); \text{ and outputs } (pf_{bb}, pf_b). \\
\text{Verify}(s, \sigma, \kappa) & \text{ proceeds as follows. Parse } s \text{ as vector } (pk, nc, bb, v, b, \kappa) \text{ and } \sigma \text{ as } (pf_{bb}, pf_b), \text{ outputting } 0 \text{ if parsing fails. Let } v_b \text{ be a vector of length } nc \text{ which is zero-filled, except for index } v \text{ which contains } |b|, \text{ and let } v_{bb} \text{ be the election outcome. If such a vector exists and } \text{Verify}_\Gamma(pk, nc, bb, v_b, v_{bb}, pf_{bb}, pf_b, \kappa) = 1 \land \text{Verify}_\Gamma(pk, nc, b, v_b, pf_b, \kappa) = 1 \land v_{bb}[v] = v_b[v] \land b \subseteq bb, \text{ then output } 1, \text{ otherwise, output } 0. 
\end{align*}

**Lemma 15.** Let \( \Gamma \) be an election scheme and \( \text{Reveal} \) be a reveal algorithm that is correct with respect to \( \Gamma \). If \( \Gamma \) satisfies \text{Exp-UV-Ext} and \( \text{Reveal} \) satisfies reveal soundness with respect to \( \Gamma \), then \( \delta(\Gamma) \) is a non-interactive proof system for relation \( R(\Gamma, \text{Reveal}) \).

For election schemes ensuring honest key generation (Definition 19), our construction \( \delta \) produces non-interactive proof systems that are sound (Lemma 16).

**Definition 19 (Honest key generation).** An election scheme \( (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}) \) ensures honest key generation, if for all probabilistic polynomial-time
adversaries $\mathcal{A}$, there exists a negligible function $\text{negl}$, such that for all security parameters $\kappa$, we have

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

Honest key generation assures that a public key is produced by an election scheme’s $\text{Setup}$ algorithm, parameterised by some security parameter and coins. There is no assurance that those coins were chosen uniformly at random. Correctness and completeness, however, assume coins are chosen uniformly at random, while perfect correctness and perfect completeness do not. Consequently, tallying produces the expected election outcome and verification succeeds for that outcome, when perfect correctness and perfect completeness are satisfied.

**Lemma 16.** Let $\Gamma$ be an election scheme with perfect correctness and perfect completeness, and let $\text{Reveal}$ be a reveal algorithm that is correct with respect to $\Gamma$. Suppose $\delta(\Gamma)$ is a non-interactive proof system for relation $R(\Gamma, \text{Reveal})$. If $\Gamma$ satisfies $\text{Exp-UV-Ext}$ and ensures honest key generation, and $\text{Reveal}$ satisfies reveal soundness with respect to $\Gamma$, then $\delta(\Gamma)$ is sound.

For election schemes that construct tallying proofs using a zero-knowledge non-interactive proof systems (Definition 20), our construction $\delta$ produces proof systems satisfying zero-knowledge (Lemma 17).

**Definition 20 (Tallying proof system).** Let $\Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}_\Gamma)$ be an election scheme and $\Delta = (\text{Prove}, \text{Verify})$ be a non-interactive proof system. We say $\Delta$ is a tallying proof system for $\Gamma$, if for all security parameters $\kappa$, integers $nc$, bulletin boards $bb$, outputs $(pk, sk, mb, mc)$ of $\text{Setup}(\kappa)$, and outputs $(v, pf)$ of $\text{Tally}(sk, nc, bb, nc, \kappa)$, we have $pf = \text{Prove}((pk, nc, bb, v), sk, \kappa; r)$, where coins $r$ are chosen uniformly at random by $\text{Tally}$.

**Lemma 17.** Let $\Gamma$ be an election scheme and $\text{Reveal}$ be a reveal algorithm that is correct with respect to $\Gamma$. Suppose $\delta(\Gamma)$ is a non-interactive proof system for relation $R(\Gamma, \text{Reveal})$. If there exists a tallying proof system for $\Gamma$ that satisfies zero-knowledge, then $\delta(\Gamma)$ satisfies zero-knowledge.

### 6.2 Notions of ballot secrecy

Proposition 5 shows that $\text{Ballot-Secrecy}$ is strictly stronger than the definition by Smyth ($\text{Smy-Ballot-Secrecy}$) [Smy16]. Intuitively, this is because it is not possible to simulate tallying in the general case (cf. Section 4.1). Nevertheless, if an election scheme proves correct key generation using a non-interactive proof system satisfying simulation sound extractability, then the witness used to construct the proof can be extracted. This typically enables the private key to be extracted too. Hence, it is possible to simulate tallying, which suffices to ensure $\text{Ballot-Secrecy}$ and $\text{Smy-Ballot-Secrecy}$ coincide. We prove this result (Proposition 18), using a suitably formulated precondition (Definition 21) that we straightforwardly derive from simulation sound extractability.

23
Definition 21. Let $\Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$ be an election scheme and $H$ be a random oracle. We say $\Gamma$ satisfies simulation sound private key extractibility, if there exists a probabilistic polynomial-time algorithm $K$, such that for all coins $r$, there exists a negligible function $\text{negl}$ and for all security parameters $\kappa$, we have $\Pr[(pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa; r); sk' \leftarrow K(\text{Setup}'(H, pk) : sk = sk')] > 1 - \text{negl}(\kappa)$, where $H$ is a transcript of the random oracle's input and output, and oracle $\text{Setup}'$ computes $(pk', sk', mb', mc') \leftarrow \text{Setup}(\kappa; r)$, forwarding any oracle queries by $\text{Setup}$ to $K$, and outputs $(pk', sk', mb', mc')$.

Proposition 18. Given an election scheme $\Gamma$ satisfying simulation sound private key extractibility, we have $\Gamma$ satisfies Ballot-Secrecy iff $\Gamma$ satisfies Smy-Ballot-Secrecy.

6.3 Suitability of security definitions

6.3.1 Ballot secrecy

Discussion of ballot secrecy originates from Chaum [Cha81] and the earliest definitions of ballot secrecy are due to Benaloh et al. [BY86, BT94b, Ben96]. More recently, Bernhard, Pereira & Warinschi [BPW12a] and Cortier et al. [CGGI13a, CGGI13b] propose definitions of ballot secrecy. Bernhard et al. [BCG+15] show that those definitions are too weak and propose a strengthening of the definition by Bernhard, Pereira & Warinschi. Smyth [Smy16] shows that the definition by Bernhard et al., and other definitions, do not detect attacks that arise when the adversary controls the bulletin board or the communication channel, and proposes a definition that does.27 We build upon Smyth’s definition of ballot secrecy because it appears to detect the largest class of attacks.

The aforementioned definitions of ballot secrecy all assume the tallier is trusted.28 Hence, an election scheme that leaks the ballot-vote relation during tallying can satisfy those definitions, because the tallier is assumed not to disclose mappings. Indeed, election scheme $\text{Enc2Vote}$ satisfies ballot secrecy (Proposition 6), despite leaking such a map to the tallier. It is desirable to distribute the tallier’s role amongst several talliers and define a definition of ballot secrecy that detects such mappings, assuming at least one tallier is honest. However, formulating such a definition would advance the state-of-the-art in a manner that is beyond the scope of this paper, and extending our results in this direction is left as a possibility for future work.

6.3.2 Election verifiability

Discussion of universal verifiability seems to originate from Cohen & Fischer [CF85] and advanced by Benaloh & Tuinstra [BT94a] and Sako & Kilian [SK95].

---

27Smyth’s definition is based upon a technical report by Smyth [Smy14] and an extended version of that technical report by Bernhard & Smyth [BS15].

28Perfect ballot secrecy formalises a privacy notion without trusting the tallier, but it is only known to be satisfied by decentralised voting systems, e.g., [Sch99, KY02, Gro04, HRZ10, KSRH12], which are unsuitable for large-scale elections.
More recently, Juels, Catalano & Jakobsson [JCJ02, JCJ10], Cortier et al. [CGGI14] and Kiayias, Zacharias & Zhang [KZZ15] present definitions of election verifiability. Smyth, Frink & Clarkson [SFC16] show that definitions by Juels, Catalano & Jakobsson and Cortier et al. do not detect attacks that arise when tallying and verification procedures collude nor when verification procedures reject legitimate outcomes. Moreover, they show that the definition by Kiayias, Zacharias & Zhang does not detect the latter class of attacks. Smyth, Frink & Clarkson propose a definition of election verifiability (§5.1) that detects these attacks.29 We adopt their definition because it appears to detect the largest class of attacks. Moreover, their definition has proven to be useful in correctly identifying three schemes that do not satisfy verifiability, and in identifying two schemes that do. Furthermore, Helios satisfies their definitions, which will be useful in our case study (§7).

Küsters et al. [KTV10, KTV11, KTV12] propose an alternative, holistic notion of verifiability called global verifiability, which must be instantiated with a goal. Smyth, Frink & Clarkson [SFC16] show that goals proposed by Küsters et al. [KTV15, §5.2] and by Cortier et al. [CGK+16, §10.2] are too strong. Moreover, Smyth, Frink & Clarkson propose a weakening of the goal by Küsters et al. and show that their definition of election verifiability (§5.1) is strictly stronger than global verifiability with that goal, which further motivates the adoption of their definition of election verifiability. Nonetheless, the “gap” exists due to an uninteresting technical detail, hence, similar verifiability results might well be derivable from global verifiability. Moreover, global verifiability would suffice if the gap is filled.

7 Case study: Helios

We demonstrate the applicability of our construction by deriving auction schemes from Helios [Adi08, AMPQ09, BGP11, Per16], an open-source, web-based electronic voting system, which has been used in binding elections. The International Association of Cryptologic Research has used Helios annually since 2010 to elect board members [BVQ10, HBH10],30 the ACM used Helios for their 2014 general election [Sta14], the Catholic University of Louvain used Helios to elect the university president in 2009 [AMPQ09], and Princeton University has used Helios since 2009 to elect student governments.31,32 Helios defines two modes of tallying: tallying by homomorphically combining ciphertexts [AMPQ09] and tallying by mixnet [Adi08, BGP11]. In the former mode, Helios has been proved to satisfy ballot secrecy and verifiability, hence our results are immediately applicable. In the latter mode, no such results exist, thus our results are only applicable if ballot secrecy and verifiability are satisfied.

29Cortier et al. [CGK+16, §8.5 & §10.1] claim that the definition by Smyth, Frink & Clarkson is flawed, but that claim is false [SFC16, §9].
This mode is nonetheless interesting, because the auction scheme we derive is more efficient.

7.1 Tallying by homomorphic combinations

Informally, Helios with tallying by homomorphically combining ciphertexts [AMPQ09] can be modelled as the following election scheme:

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

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

**Tally** proceeds as follows. First, any ballots on the bulletin board for which proofs do not hold are discarded. Secondly, the ciphertexts in the remaining ballots are homomorphically combined, the homomorphic combination is decrypted to reveal the election outcome, and correctness of decryption is proved in zero-knowledge. Finally, the election outcome and proof of correct decryption are output.

**Verify** recomputes the homomorphic combination, checks the proofs, and outputs 1 if these checks succeed and 0 otherwise.

The original scheme [AMPQ09] is known to be vulnerable to attacks against ballot secrecy and verifiability, and defences against those attacks have been proposed [CS11, SC11, Smy12, CS13, SB13, SB14, Smy16, BPW12a, CE16]. We adopt the formal definition of Helios proposed by Smyth, Frink & Clarkson [SFC16], which adopts non-malleable ballots [SHM15] and uses the Fiat–Shamir transformation with the inclusion of statements in hashes [BPW12a] to defend against those attacks. We recall that formalisation in Appendix F and henceforth refer to it as *Helios’16*.

We derive an auction scheme from *Helios’16* using our construction parameterised with a reveal algorithm *Helios-Reveal* and non-interactive proof system $\delta(Helios’16)$. Reveal algorithm *Helios-Reveal* exploits some technical details of *Helios’16* that we have not yet discussed, so we defer the formal description to Appendix G. Our privacy and verifiability results allow us to prove security of the derived auction scheme.

**Theorem 19.** Auction scheme $\Lambda(Helios’16, Helios-Reveal, \delta(Helios’16))$ satisfies Bid-Secrecy.

---

33Beyond secrecy and verifiability, attacks against eligibility are also known [SP13, SP15, MS17].

34Formally, $\Lambda(Helios’16, Helios-Reveal, \delta(Helios’16))$ is an auction scheme by Lemmata 4, 15, & 43.
Proof. We have $\delta(\text{Helios'16})$ is a non-interactive proof system for relation $R(\text{Helios'16}, \text{Helios-Reveal})$ (Lemma 15) satisfying zero-knowledge (Lemma 17), assuming there exists a tallying proof system for Helios'16 that satisfies zero-knowledge. Moreover, since Smyth has shown that Helios'16 satisfies Smy-Ballot-Secrecy [Smy16], we have Helios'16 satisfies Ballot-Secrecy (Proposition 18), assuming Helios'16 satisfies simulation sound private key extractibility. Furthermore, since Smyth, Frink & Clarkson have shown that Helios'16 satisfies Exp-UV-Ext [SFC16], we have Helios'16 satisfies W-Tally-Soundness (Lemma 11). Hence, by Theorem 10, it suffices to prove that reveal algorithm Helios-Reveal satisfies reveal soundness with respect to Helios'16 and to prove our earlier assumptions. We defer those proofs to Lemmata 40, 42 & 44 in Appendix G.

Theorem 20. Auction scheme $\Lambda(\text{Helios'16}, \text{Helios-Reveal}, \delta(\text{Helios'16}))$ satisfies Exp-IV and Exp-UV.

Proof. Smyth, Frink & Clarkson have shown that Helios'16 satisfies Exp-IV-Ext and Exp-UV-Ext [SFC16]. Hence, we have $\Lambda(\text{Helios'16}, \text{Helios-Reveal}, \delta(\text{Helios'16}))$ satisfies Exp-IV (Theorem 13). We have $\delta(\text{Helios'16})$ is a non-interactive proof system for relation $R(\text{Helios'16}, \text{Reveal})$ (Lemma 15) satisfying soundness (Lemma 16), assuming Helios'16 ensures honest key generation. Hence, to show Exp-UV is satisfied, it suffices (Theorem 14) to prove that Helios'16 satisfies perfect correctness and perfect completeness, $R(\text{Helios'16}, \text{Helios-Reveal})$ is $\Lambda$-suitable, and our previous assumption. We defer those proofs to Lemmata 38 & 39 in Appendix F and Lemmata 45 & 41 in Appendix G.

Our construction $\delta$ for non-interactive proof systems (§6) demonstrates the scope of our results. But, more efficient proof systems might exist. Indeed, we tailor a proof system for Helios'16 in Appendix G.2, which is more efficient than $\delta(\text{Helios'16})$.

Deriving auction schemes from Helios with tallying by homomorphically combining ciphertexts is not new. Indeed, McCarthy, Smyth & Quaglia [MSQ14] derive the Hawk auction scheme. However, they only provide an informal security analysis for Hawk. By contrast, we derive an auction scheme for which we provide formal security results.

7.2 Tallying by mixnet

Informally, Helios with tallying by mixnet [Adi08, BGP11] can be modelled as the following election scheme:

**Setup** as per above.

**Vote** encrypts the vote, proves correct ciphertext construction,$^{35}$ and outputs the ciphertext coupled with the proof.

---

$^{35}$The algorithm does not prove that the vote is selected from the sequence of candidates, because ciphertexts will be decrypted after mixing, thus, this check can be performed later.
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 mixed. Thirdly, the ciphertexts output by the mix are decrypted to reveal the election outcome and correctness of decryption is proved in zero-knowledge. Finally, the election outcome and proof of correct decryption are output.

Verify checks the proofs and outputs 1 if these checks succeed and 0 otherwise.

Unlike Helios with tallying by homomorphically combining ciphertexts, there is no description of Helios with tallying by mixnet in the cryptographic model (indeed, [Adi08] introduces the idea, and [BGP11] describes the general workflow of mixnet-based elections). Thus, we must formalise a suitable description.

We formalise Helios with tallying by mixnet as the class of election schemes HeliosM’16 (see Appendix H). Given an election scheme Γ from HeliosM’16, we derive auction scheme Λ(Γ, HeliosM-Reveal, δ(Γ)) using a reveal algorithm that works as follows:

HeliosM-Reveal constructs a ciphertext for the winning bid, performs plaintext equality tests between that ciphertext and the ciphertexts input to the mix, and outputs any ballots for which the test succeeds.36

We defer a formal definition to Appendix I.1.

Our privacy and verifiability results allow us to prove security.

Theorem 21. Given an election scheme Γ ∈ HeliosM’16, auction scheme Λ(Γ, HeliosM-Reveal, δ(Γ)) satisfies Bid-Secrecy, Exp-IV, and Exp-UV.

The proof of Theorem 21 is similar in structure to the proofs of Theorems 19 & 20, and we defer the details to Appendix I.

8 Conclusion

We present a generic construction for auction schemes from election schemes, and we formulate precise conditions under which auction schemes produced by our construction are secure. Thereby demonstrating that the seemingly disjoint research fields of auctions and elections are related. Our results advance the unification of auctions and elections; facilitating the progression of both fields. In particular, secure auction schemes can now be constructed from election schemes, allowing advances in election schemes to be capitalised upon to advance auction schemes.

36 A plaintext equality test [JJ00] is a cryptographic primitive which allows a key holder to check whether two ciphertexts contain the same plaintext, without decrypting.
Acknowledgements

This research was conducted in part at École Normale Supérieure and 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 22 (Asymmetric encryption scheme [KL07]). An asymmetric encryption scheme is a tuple of efficient algorithms \((\text{Gen}, \text{Enc}, \text{Dec})\) such that:

- \(\text{Gen}\), denoted \((pk, sk) \leftarrow \text{Gen}(\kappa)\), takes a security parameter \(\kappa\) as input and outputs a key pair \((pk, sk)\).
- \(\text{Enc}\), denoted \(c \leftarrow \text{Enc}(pk, m)\), takes a public key \(pk\) and message \(m\) from the plaintext space\(^{37}\) as input, and outputs a ciphertext \(c\).
- \(\text{Dec}\), denoted \(m \leftarrow \text{Dec}(sk, c)\), takes a private key \(sk\), and ciphertext \(c\) as input, and outputs a message \(m\) or error symbol \(\perp\). We assume \(\text{Dec}\) is deterministic.

Moreover, the scheme must be correct: there exists a negligible function \(\text{negl}\), such that for all security parameters \(\kappa\) and messages \(m\) from the plaintext space, we have \(\Pr[(pk, sk) \leftarrow \text{Gen}(\kappa); c \leftarrow \text{Enc}(pk, m) : \text{Dec}(sk, c) = m] > 1 - \text{negl}(\kappa)\). We say correctness is perfect, if the aforementioned probability is one.

Definition 23 (IND-PA0 [BS99]). Let \(\Pi = (\text{Gen}, \text{Enc}, \text{Dec})\) be an asymmetric encryption scheme, \(A\) be an adversary, \(\kappa\) be a security parameter, and \(\text{IND-PA0}(\Pi, A, \kappa)\) be the following game.\(^{38}\)

\[\text{IND-PA0}(\Pi, A, \kappa) = \]

\(^{37}\)Definitions of asymmetric encryption schemes (including the definition by Katz & Lindell [KL07]) typically leave the set defining the plaintext space implicit. Such definitions can be extended to explicitly include the plaintext space, for instance, Smyth, Frink & Clarkson [SFC16] present a definition in which algorithm \(\text{Setup}\) outputs the plaintext space.

\(^{38}\)We extend set membership notation to vectors: we write \(x \in x\) if \(x\) is an element of the set \(\{x[i] : 1 \leq i \leq |x|\}\).
\[(pk, sk) \leftarrow \text{Gen}(\kappa);\]
\[\beta \leftarrow \{0, 1\};\]
\[(m_0, m_1) \leftarrow \mathcal{A}(pk, \kappa);\]
\[y \leftarrow \text{Enc}(pk, m_\beta);\]
\[c \leftarrow \mathcal{A}(y);\]
\[p \leftarrow (\text{Dec}(sk, c[1]), \ldots, \text{Dec}(sk, c[|c|]));\]
\[g \leftarrow \mathcal{A}(p);\]
\[\text{if } g = \beta \land y \not\in c\text{ then return } 1\]
\[\text{else return } 0\]

In the above game, we insist \(m_0\) and \(m_1\) are in the encryption scheme’s plaintext space and \(|m_0| = |m_1|\). We say \(\Pi\) satisfies indistinguishability under chosen plaintext and parallel chosen ciphertext attacks (IND-PA0), if for all probabilistic polynomial-time adversaries \(\mathcal{A}\), there exists a negligible function \(\text{negl}\), such that for all security parameters \(\kappa\), we have \(\text{Succ}(\text{IND-PA0}(\Pi, \mathcal{A}, \kappa)) \leq \frac{1}{2} + \text{negl}(\kappa)\).

**Definition 24** (Homomorphic encryption [SFC16]). An asymmetric encryption scheme \(\Pi = (\text{Gen}, \text{Enc}, \text{Dec})\) is homomorphic,\(^{39}\) with respect to ternary operators \(\odot, \oplus, \otimes\), if there exists a negligible function \(\text{negl}\), such that for all security parameters \(\kappa\), the following conditions are satisfied:\(^{41}\)

- For all messages \(m_1\) and \(m_2\) from \(\Pi\)’s plaintext space, we have \(\Pr[(pk, sk) \leftarrow \text{Gen}(\kappa); c_1 \leftarrow \text{Enc}(pk, m_1); c_2 \leftarrow \text{Enc}(pk, m_2); \text{Dec}(sk, c_1 \odot_{pk} c_2) = \text{Dec}(sk, c_2)] > 1 - \text{negl}(\kappa)\).

- For all messages \(m_1\) and \(m_2\) from \(\Pi\)’s plaintext space, and all coins \(r_1\) and \(r_2\), we have \(\Pr[(pk, sk) \leftarrow \text{Gen}(\kappa); \text{Enc}(pk, m_1; r_1) \odot_{pk} \text{Enc}(pk, m_2; r_2) = \text{Enc}(pk, m_1 \odot_{pk} m_2; r_1 \oplus_{pk} r_2)] > 1 - \text{negl}(\kappa)\).

We say \(\Pi\) is additively homomorphic, respectively multiplicatively homomorphic, if for all security parameters \(\kappa\) and key pairs \(pk, sk\), such that there exists coins \(r\) and \((pk, sk) = \text{Gen}(\kappa; r)\), we have \(\odot_{pk}\) is the addition operator, respectively multiplication operator, in the group defined by \(\Pi\)’s plaintext space and \(\odot_{pk}\).

**A.2 Proof systems**

**Definition 25** (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:

\(^{39}\)Our definition of an asymmetric encryption scheme leaves the plaintext space implicit, whereas, Smyth, Frink & Clarkson [SFC16] explicitly define the plaintext space; this change is reflected in our definition of homomorphic encryption.

\(^{40}\)Henceforth, we implicitly bind ternary operators, i.e., we write \(\Pi\) is a homomorphic asymmetric encryption scheme as opposed to the more verbose \(\Pi\) is a homomorphic asymmetric encryption scheme, with respect to ternary operators \(\odot, \oplus, \otimes\).

\(^{41}\)We write \(x \odot_{pk} y\) for the application of ternary operator \(\odot\) to inputs \(x, y,\) and \(pk\). We occasionally abbreviate \(x \odot_{pk} y\) as \(x \circ Y\), when \(pk\) is clear from the context.
• **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 \( \text{Verify} \) is deterministic.

Moreover, the system must be complete: there exists a negligible function \( \text{negl} \), such that for all statement 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 26** (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 \( H \), the Fiat-Shamir transformation, denoted \( \text{FS}(\Sigma, H) \), is the non-interactive proof system \((\text{Prove}, \text{Verify})\), defined as follows:

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

\[
\text{Verify}(s, (\text{comm}, \text{resp}), \kappa) =
\begin{align*}
&\text{chal} \leftarrow H(\text{comm}, s); \\
&\text{return} \ \text{Verify}_\Sigma(s, (\text{comm}, \text{chal}, \text{resp}), \kappa);
\end{align*}
\]

**Definition 27** (Soundness). Suppose \((\text{Prove}, \text{Verify})\) is a non-interactive proof system for relation \( R \). We say \((\text{Prove}, \text{Verify})\) is sound, if for all probabilistic polynomial-time adversaries \( A \), there exists a negligible function \( \text{negl} \), such that for all security parameters \( \kappa \), we have
\[
\Pr[(s,\sigma) \leftarrow A(s,\kappa) : (s,w) \not\in R \land \text{Verify}(s,\sigma) = 1] \leq \text{negl}(\kappa).
\]

**Definition 28** (Zero-knowledge). Let \( \Delta = (\text{Prove}, \text{Verify}) \) be a non-interactive proof system for a relation \( R \), derived by application of the Fiat-Shamir transformation to a random oracle \( H \) and a sigma protocol. Moreover, let \( S \) be an algorithm, \( A \) be an adversary, \( \kappa \) be a security parameter, and \( \text{ZK}(\Delta, A, H, S, \kappa) \) be the following game.

\[
\text{ZK}(\Delta, A, H, S, \kappa) =
\begin{align*}
&\beta \leftarrow_R \{0,1\}; \\
&g \leftarrow A^H, \mathcal{P}(\kappa); \\
&\text{if } g = \beta \text{ then return } 1 \\
&\text{else return } 0 \end{align*}
\]

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 $S$ can patch random oracle $H$.\footnote{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.} We say $\Delta$ satisfies zero-knowledge, if there exists a probabilistic polynomial-time algorithm $S$, 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 $\text{Succ}(\text{ZK}(\Delta, A, H, S, \kappa)) \leq \frac{1}{2} + \text{negl}(\kappa)$. An algorithm $S$ for which zero-knowledge holds is called a simulator for (Prove, Verify).

**Definition 29** (Simulation sound extractability [SFC16, BPW12a, Gro06]). Suppose $\Sigma$ is a sigma protocol for relation $R$, $H$ is a random oracle, and (Prove, Verify) is a non-interactive proof system, such that $FS(\Sigma, H) = (\text{Prove, Verify})$. Further suppose $S$ is a simulator for (Prove, Verify) and $H$ can be patched by $S$. Proof system (Prove, Verify) satisfies simulation sound extractability if there exists a probabilistic polynomial-time algorithm $K$, 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 $\footnote{We extend set membership notation to vectors: we write $x \in x$ if $x$ is an element of the set $\{x[i] : 1 \leq i \leq |x|\}$.}$

$$
\Pr[P \leftarrow \cdot; Q \leftarrow A^{H,P}(-;r); W \leftarrow K^{A'}(H,P,Q) : |Q| \neq |W| \lor \exists j \in \{1, \ldots, |Q|\} . (Q[j][1], W[j]) \notin R \land \\
\forall (s,\sigma) \in Q, (t,\tau) \in P . \text{Verify}(s,\sigma,\kappa) = 1 \land \sigma \neq \tau] \leq \text{negl}(\kappa)
$$

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

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

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

Algorithm $K$ is an extractor for (Prove, Verify).

**Theorem 22** ([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 Prove and Verify. We write $\text{Prove}(s, w, m, \kappa)$ and $\text{Verify}(s, (\text{comm}, \text{resp}), m, k)$ for invocations of Prove and Verify which include an optional string. When $m$ is provided, it is included in the hashes in both algorithms. That is, given $FS(\Sigma, H) = (\text{Prove, Verify})$, 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 22 can be extended to this generalisation.
B Correctness, completeness and injectivity

B.1 Definitions for auctions

**Correctness** asserts that the price and the set of bids output by algorithm \texttt{Open} correspond to the winning price and the set of winning bids, assuming the bids on the bulletin board were all produced by algorithm \texttt{Bid}.

**Definition 30 (Correctness).** There exists a negligible function \(\text{negl}\), such that for all security parameters \(\kappa\), integers \(nb\) and \(np\), and prices \(p_1, \ldots, p_{nb} \in \{1, \ldots, np\}\), it holds that

\[
\Pr[\text{pk}, \text{sk}, \text{mb}, \text{mp} \leftarrow \text{Setup}(\kappa);
\hspace{1cm} \text{for } 1 \leq i \leq nb \text{ do } b_i \leftarrow \text{Bid}(\text{pk}, np, p_i, \kappa);
\hspace{1cm} (p, b, pf) \leftarrow \text{Open}(\text{sk}, np, \{b_1, \ldots, b_{nb}\}, \kappa)
\hspace{1cm} : nb \leq mb \land np \leq mp \Rightarrow p = \max(0, p_1, \ldots, p_{nb}) \land b = \{b_i | p_i = p \land 1 \leq i \leq nb\}] > 1 - \text{negl}(\kappa).
\]

**Completeness** stipulates that outputs of algorithm \texttt{Open} will be accepted by algorithm \texttt{Verify}. This prevents biasing attacks [SFC16], which arise when algorithm \texttt{Verify} rejects legitimate outcomes, possibly due to presence of a bid on the bulletin board that was not produced by algorithm \texttt{Bid}.

**Definition 31 (Completeness).** There exists a negligible function \(\text{negl}\), such that for all security parameters \(\kappa\), bulletin boards \(\mathbb{b}\), and integers \(np\), we have

\[
\Pr[\text{pk}, \text{sk}, \text{mb}, \text{mp} \leftarrow \text{Setup}(\kappa); (p, b, pf) \leftarrow \text{Open}(\text{sk}, np, \mathbb{b}, \kappa)
\hspace{1cm} : |\mathbb{b}| \leq mb \land np \leq mp \Rightarrow \text{Verify}(\text{pk}, np, \mathbb{b}, p, b, pf, \kappa) = 1] > 1 - \text{negl}(\kappa).
\]

**Injectivity** asserts that a bid can only be interpreted for one price, assuming the public key input to algorithm \texttt{Bid} was produced by algorithm \texttt{Setup}. This ensures that distinct prices are not mapped to the same bid by algorithm \texttt{Bid}. Hence, a bid unambiguously encodes a price.

**Definition 32 (Injectivity).** For all security parameters \(\kappa\), integers \(np\), and prices \(p\) and \(p’\), such that \(p \neq p’\), we have

\[
\Pr[\text{pk}, \text{sk}, \text{mb}, \text{mp} \leftarrow \text{Setup}(\kappa); b \leftarrow \text{Bid}(\text{pk}, np, p, \kappa);
\hspace{1cm} b’ \leftarrow \text{Bid}(\text{pk}, np, p’, \kappa) : b \neq \bot \\
\hspace{1cm} \land b’ \neq \bot \\
\hspace{1cm} \Rightarrow b \neq b’] = 1.
\]

To formulate our definition of universal verifiability, we require strong injectivity, which asserts that a bid can only be interpreted for one price, even if the public key input to algorithm \texttt{Bid} was produced by the adversary.

**Definition 33 (Strong injectivity).** An auction scheme \((\text{Setup}, \text{Bid}, \text{Open}, \text{Verify})\) satisfies strong injectivity, if for all security parameters \(\kappa\), public keys \(pk\), integers \(np\), and prices \(p\) and \(p’\), such that \(p \neq p’\), we have \(\Pr[b \leftarrow \text{Bid}(pk, np, p, \kappa);
\hspace{1cm} b’ \leftarrow \text{Bid}(pk, np, p’, \kappa) : b \neq \bot \\
\hspace{1cm} \land b’ \neq \bot \\
\hspace{1cm} \Rightarrow b \neq b’] = 1.
\)
B.2 Definitions for elections

**Definition 34** (Correctness [SFC16]). 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 vector of length \(nc\) whose components are all 0, then

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

**Definition 35** (Completeness [SFC16]). There exists a negligible function \(\text{negl}\), such that for all security parameters \(\kappa\), bulletin boards \(\text{bb}\), and integers \(nc\), we have

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

**Definition 36** (Injectivity). For all security parameters \(\kappa\), integers \(nc\), and votes \(v\) and \(v'\), such that \(v \neq v'\), we have

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

Injectivity for election schemes (Definition 36) is analogous to injectivity for auction schemes (Definition 32) and is slightly weaker than the original definition (Definition 37).

**Definition 37** (Strong injectivity [SFC16]). An election scheme \((\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})\) satisfies strong injectivity, if for all security parameters \(\kappa\), public keys \(pk\), integers \(nc\), and votes \(v\) and \(v'\), such that \(v \neq v'\), we have

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

C Proofs

By Definitions 31 & 35, we have the following facts:

**Fact 23.** Suppose \(\Sigma = (\text{Setup}, \text{Bid}, \text{Open}, \text{Verify})\) is an auction scheme. Further suppose for all public keys \(pk\), integers \(p\) and \(np\), sets \(b\) and \(\text{bb}\), proofs \(pf\), and security parameters \(\kappa\), we have \(\text{Verify}(pk, np, \text{bb}, p, b, pf, \kappa) = 1\). It follows that \(\Sigma\) satisfies completeness.
Fact 24. Suppose $\Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$ is an election scheme. Further suppose for all public keys $pk$, integers $nc$, sets $bb$, vectors $v$, proofs $pf$, and security parameters $\kappa$, we have $\text{Verify}(pk, nc, bb, v, pf, \kappa) = 1$. It follows that $\Gamma$ satisfies completeness.

C.1 Proof of Lemma 1

Let $\text{Enc2Bid}(\Pi) = (\text{Setup}, \text{Bid}, \text{Open}, \text{Verify})$ and $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$. We prove that $\text{Enc2Bid}(\Pi)$ satisfies correctness, completeness, and injectivity.

First, correctness. Suppose $\kappa$ is a security parameter, $nb$ and $np$ are integers, and $p_1, \ldots, p_{nb} \in \{1, \ldots, np\}$ are prices. Further suppose $(pk, sk, mb, mp)$ is an output of $\text{Setup}(\kappa)$ such that $nb \leq mb \wedge np \leq mp$ and for each $1 \leq i \leq nb$ we have $\text{Bid}(pk, np, p_i, \kappa)$ outputs $b_i$. Let $bb = \{b_1, \ldots, b_{nb}\}$. Suppose $\text{Open}(sk, np, bb, \kappa)$ outputs $(p, b_1, pf)$. Let $d \leftarrow \{(b, \text{Dec}(sk, b)) \mid b \in bb\}$. Since $\Pi$ satisfies IND-CPA, we have $b_1, \ldots, b_{nb}$ are pairwise distinct with overwhelming probability. Moreover, since $(pk, sk)$ are outputs of $\text{Gen}$ and since $\Pi$ is perfectly correct, we have $d = \{(b_1, p_1), \ldots, (b_{np}, p_{np})\}$. By inspection of $\text{Open}$, we have $p$ is the largest integer such that $(b_i, p_i) \in d \wedge 1 \leq p \leq np$, or no such integer exists and $p = 0$. It follows that $p = \max(0, p_1, \ldots, p_{nb})$ in both cases. By further inspection of $\text{Open}$, we have $b = \{b \mid (b, p') \in d \wedge p' = p\}$ in the former case and $b = \emptyset$ in the latter case. In the former case, we have $b = \{b_i \mid p_i = p \wedge 1 \leq i \leq nb\}$. And, in the latter case, we have $0 \notin \{p_1, \ldots, p_{nb}\}$, hence, $b = \{b_i \mid p_i = p \wedge 1 \leq i \leq nb\} = \emptyset$. It follows that correctness is (perfectly) satisfied.

Secondly, completeness. Algorithm $\text{Verify}$ always outputs $1$, hence, the result follows from Fact 23.

Finally, injectivity. By contradiction, suppose there exists a security parameter $\kappa$, integer $p, p', np$, and coins $r, s, s'$ such that

$$(pk, sk, mb, mp) = \text{Setup}(\kappa; r) \wedge b = \text{Bid}(pk, np, p, \kappa; s) \wedge b' = \text{Bid}(pk, np, p', \kappa; s') \wedge b \neq \bot \wedge b' \neq \bot \wedge b = b' \wedge p \neq p'.$$

By definition of $\text{Setup}$, we have $(pk, sk) \leftarrow \text{Gen}(\kappa; r)$ and $mp = \{1, \ldots, |m|\}$, where $m$ is the encryption scheme’s plaintext space. Moreover, by definition of $\text{Bid}$, we have $b = \text{Enc}(pk, p; s)$ and $b' = \text{Enc}(pk, p'; s')$. Furthermore, since $b \neq \bot \wedge b' \neq \bot$, we have, by inspection of $\text{Bid}$, that $p$ and $p'$ are from the plaintext space. Since $\Pi$ is perfectly correct, we have

$$\text{Dec}(sk, b) = p = p' = \text{Dec}(sk, b'),$$

thus deriving a contradiction and concluding our proof. \qed

C.2 Proof of Lemma 2

Let $\text{Enc2Vote}(\Pi) = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$ and $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$. Algorithm $\text{Verify}$ always outputs $1$, hence, it follows from Fact 24 that $\text{Enc2Vote}(\Pi)$ satisfies completeness. The proof that $\text{Enc2Vote}(\Pi)$ satisfies injectivity is similar to the proof that $\text{Enc2Bid}(\Pi)$ satisfies injectivity (Appendix C.1), and we
omit a formal proof. We prove that Enc2Vote(\Pi) satisfies correctness. Suppose \(\kappa\) is a security parameter, \(nb\) and \(nc\) are integers, and \(v_1, \ldots, v_{nb} \in \{1, \ldots, nc\}\) are votes, and \(v\) is a vector of length \(nc\) whose components are all 0. Further suppose \((pk, sk, mb, mc)\) is an output of Setup(\(\kappa\)) such that \(nb \leq mb \land nc \leq mc\) and for each \(1 \leq i \leq nb\) we have \(b_i\) is an output of Vote(\(pk, nc, v_i, \kappa\)). Moreover, for each \(1 \leq i \leq nb\) compute \(v[i] \leftarrow v[i] + 1\). Suppose \((v', pf)\) is an output of Tally(\(sk, nc, \{b_1, \ldots, b_{nb}\}, \kappa\)). By inspection of algorithm Tally, we have \(v'\) is a vector of length \(nc\) computed as follows:

\[
\text{for } b \in \{b_1, \ldots, b_{nb}\} \text{ do }
\begin{align*}
&v \leftarrow \text{Dec}(sk, b); \\
&\text{if } 1 \leq v \leq nc \text{ then } \\
&\quad v[v] \leftarrow v[v] + 1;
\end{align*}
\]

Since \(\Pi\) satisfies IND-CPA, we have \(b_1, \ldots, b_{nb}\) are pairwise distinct with overwhelming probability. Moreover, since \(pk, sk\) are output by Gen and since \(\Pi\) is perfectly correct, we have \(\text{Dec}(sk, b_i) = v_i\) for all \(i \in \{1, \ldots, nb\}\). It follows that \(v = v'\). Hence, correctness is (perfectly) satisfied. \(\square\)

### C.3 Proof of Lemma 3

Let \(\Lambda(\Gamma, \text{Reveal}) = (\text{Setup}_\Lambda, \text{Bid}, \text{Open}, \text{Verify}_\Lambda)\) and \(\Gamma = (\text{Setup}_\Pi, \text{Vote}, \text{Tally}, \text{Verify}_\Pi)\). Algorithm Verify\(\_\)\(\Pi\) always outputs 1, hence, it follows from Fact 23 that \(\Lambda(\Gamma, \text{Reveal})\) satisfies completeness. Moreover, it follows from injectivity of \(\Gamma\) that \(\Lambda(\Gamma, \text{Reveal})\) satisfies injectivity. We show that \(\Lambda(\Gamma, \text{Reveal})\) satisfies correctness. Suppose \(\kappa\) is a security parameter, \(nb\) and \(np\) are integers, and \(p_1, \ldots, p_{nb} \in \{1, \ldots, np\}\) are prices. Further suppose \((pk, sk, mb, mp)\) is an output of Setup(\(\kappa\)) such that \(nb \leq mb \land np \leq mp\) and for each \(1 \leq i \leq nb\) we have Bid(\(pk, np, p_i, \kappa\)) outputs \(b_i\). Let \(bb = \{b_1, \ldots, b_{nb}\}\). Moreover, suppose Open(\(sk, np, bb, \kappa\)) outputs \((p, b, pf)\) and Tally(\(sk, np, bb, \kappa\)) outputs \((v, pf)\). Since \(\Gamma\) satisfies correctness, we have with overwhelming probability that \(v\) can be equivalently computed by initialising \(v\) as a zero-filled vector of length \(np\) and by performing the following computation:

\[
\text{for } 1 \leq i \leq nb \text{ do }
\begin{align*}
&v[p_i] \leftarrow v[p_i] + 1;
\end{align*}
\]

By inspection of Open, we have \(p\) is the largest integer such that \(v[p] > 0 \land 1 \leq p \leq np\), or no such integer exists and \(p = 0\). It follows that \(p = \max(0, p_1, \ldots, p_{nb})\) in both cases. By further inspection of Open, we have \(b\) is an output of Reveal(\(sk, np, bb, p, \kappa\)) in the former case and \(b = \emptyset\) in the latter. In the former case we have \(b = \{b_i \mid p_i = p \land 1 \leq i \leq nb\}\) with overwhelming probability, because reveal algorithm Reveal is correct with respect to \(\Gamma\). And in the latter case we have \(0 \not\in \{p_1, \ldots, p_{nb}\}\), hence, \(b = \{b_i \mid p_i = p \land 1 \leq i \leq nb\} = \emptyset\). Hence, correctness is satisfied with overwhelming probability. \(\square\)
C.4 Proof of Lemma 4

The proof that \( \Lambda(\Gamma, \text{Reveal}, \Delta) \) satisfies correctness and injectivity is similar to the proof that \( \Lambda(\Gamma, \text{Reveal}) \) satisfies correctness and injectivity (Appendix C.3), and we omit a formal proof. We prove that \( \Lambda(\Gamma, \text{Reveal}, \Delta) \) satisfies completeness.

Let \( \Gamma = (\text{Setup}_\Gamma, \text{Vote}, \text{Tally}, \text{Verify}_\Gamma) \), \( \Delta = (\text{Prove}, \text{Verify}) \), and \( \Lambda(\Gamma, \text{Reveal}, \Delta) = (\text{Setup}_\Delta, \text{Bid}, \text{Open}, \text{Verify}_\Delta) \). Suppose \( \kappa \) is a security parameter, \( \bb \) is a bulletin board, and \( \np \) is an integer. Further suppose \( (pk, sk, mb, mp) \) is an output of \( \text{Setup}_\Delta(\kappa) \) such that \( |\bb| \leq mb \land np \leq mp \) and \( (p, b, \sigma) \) is an output of \( \text{Open}(sk, np, \bb, \kappa) \). It suffices to show that \( \text{Verify}_\Delta(pk, np, \bb, p, b, \sigma, \kappa) = 1 \) with overwhelming probability. By definition of \( \text{Verify}_\Delta \), we must show that checks (1) – (3) hold with overwhelming probability.

Check (1) succeeds with overwhelming probability, because \( \Gamma \) satisfies completeness. Check (2) succeeds by definition of \( \text{Open} \). We prove that Check (3) succeeds with overwhelming probability as follows. If \( p \not\in \{1, \ldots, np\} \), then the check vacuously holds, otherwise, we proceed as follows. Since \( \Delta \) satisfies completeness, it suffices to show that \( ((pk, np, \bb, p, b, \kappa), sk) \in R(\Gamma, \text{Reveal}) \). By aforementioned assumptions, we have \( 1 \leq p \leq np \leq mp \) and \( |\bb| \leq mb \), moreover, there exists coins \( r \) such that \( (pk, sk, mb, mp) = \text{Setup}_\Delta(\kappa; r) \). Furthermore, by inspection of \( \text{Open} \), there exist coins \( r' \) such that \( b = \text{Reveal}(sk, np, \bb, v, \kappa; r') \). The result \( (pk, np, \bb, p, b, \sigma, \kappa) = 1 \). We have that \( \text{Verify}_\Delta(pk, np, \bb, p, b, pf, \kappa) \) outputs 1 with overwhelming probability, hence, \( \Lambda(\Gamma, \text{Reveal}, \Delta) \) satisfies completeness.

C.5 Proof of Proposition 5

We introduce a construction for election schemes (Definition 38) which demonstrates that our notion of ballot secrecy (Ballot-Secrecy) is strictly stronger than Smyth’s notion (Smy- Ballot-Secrecy).

Definition 38. Suppose \( \Gamma = (\text{Setup}_\Gamma, \text{Vote}_\Gamma, \text{Tally}_\Gamma, \text{Verify}_\Gamma) \) is an election scheme, and \( \epsilon \) and \( \epsilon' \) are constant symbols that do not appear in the codomain of \( \text{Vote}_\Gamma \). Let \( \chi(\Gamma, \epsilon, \epsilon') = (\text{Setup}_\chi, \text{Vote}_\chi, \text{Tally}_\chi, \text{Verify}_\chi) \) be defined as follows.

\( \text{Setup}_\chi(\kappa) \). Computes \( (pk, sk, mb, mc) \leftarrow \text{Setup}_\Gamma(\kappa) \), generates a nonce \( k \) of the same length as \( sk \), and outputs \( (pk, (sk, k), mb, mc) \).

\( \text{Vote}_\chi(pk, nc, v, \kappa) \). Computes \( b \leftarrow \text{Vote}_\Gamma(pk, nc, v, \kappa) \) and outputs \( b \).

\( \text{Tally}_\chi(sk', nc, \bb, \kappa) \). Parses \( sk' \) as \( (sk, k) \), computes

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

if \( \bb = \{\epsilon\} \) then
\[
v[0] = sk \oplus k
\]
else if \( \bb = \{\epsilon'\} \) then
\[
v[0] = k
\]

and outputs \( (v, pf) \).

37
Lemma 25. Suppose $\Gamma = (\text{Setup}_\Gamma, \text{Vote}_\Gamma, \text{Tally}_\Gamma, \text{Verify}_\Gamma)$ is an election scheme, and $\epsilon$ and $\epsilon'$ are constant symbols that do not appear in the codomain of $\text{Vote}_\Gamma$. We have $\chi(\Gamma, \epsilon, \epsilon') = (\text{Setup}_\chi, \text{Vote}_\chi, \text{Tally}_\chi, \text{Verify}_\chi)$ is an election scheme.

Proof sketch. Since constant symbols $\epsilon$ and $\epsilon'$ do not appear in the codomain of $\text{Vote}_\Gamma$, correctness of $\chi(\Gamma, \epsilon, \epsilon')$ follows from correctness of $\Gamma$. Moreover, since algorithm $\text{Verify}_\chi$ always outputs 1, we have completeness of $\chi(\Gamma, \epsilon, \epsilon')$ by Fact 24. Furthermore, injectivity of $\chi(\Gamma, \epsilon, \epsilon')$ trivially follows from injectivity of $\Gamma$. □

Proof sketch of Proposition 5. Intuitively, given an election scheme $\Gamma$ satisfying $\text{Smy-Ballot-Secrecy}$ and constant symbols $\epsilon$ and $\epsilon'$, we have $\chi(\Gamma, \epsilon, \epsilon')$ satisfies $\text{Smy-Ballot-Secrecy}$, because tallying may only leak $sk \oplus k$ or $k$, but not both. By comparison, $\chi(\Gamma, \epsilon)$ does not satisfy $\text{Ballot-Secrecy}$, because of the following attack. The adversary outputs bulletin board $bb \cup \{\epsilon, \epsilon'\}$, recovers $sk \oplus k$ and $k$ from $W$, and obtains the private key. By election scheme correctness, this key can be used to recover votes from ballots. □

C.6 Proof of Proposition 6

Let BS0, respectively BS1, be the game derived from $\text{Ballot-Secrecy}$ by replacing $\beta \leftarrow_R \{0, 1\}$ with $\beta \leftarrow 0$, respectively $\beta \leftarrow 1$. These games are trivially related to $\text{Ballot-Secrecy}$, namely, $\text{Succ}(\text{Ballot-Secrecy}(\Gamma, \mathcal{A}, \kappa)) = \frac{1}{2} \cdot \text{Succ}(\text{BS0}(\Gamma, \mathcal{A}, \kappa)) + \frac{1}{2} \cdot \text{Succ}(\text{BS1}(\Gamma, \mathcal{A}, \kappa))$. Moreover, let BS1:0 be the game derived from BS1 by replacing $g = \beta$ with $g = 0$. We relate game BS1 to BS1:0, and we relate games BS0 and BS1:0 to the hybrid games $G_0, G_1, \ldots$ introduced in Definition 39. We use these relations to prove Proposition 6.

Lemma 26. Let $\Pi$ be an asymmetric encryption scheme satisfying IND-CPA and let $\Gamma = \text{Enc2Vote}(\Pi)$. If a probabilistic polynomial-time adversary $\mathcal{A}$ wins game $\text{Ballot-Secrecy}$, then for all security parameters $\kappa$ we have $\text{Succ}(\text{BS1}(\Gamma, \mathcal{A}, \kappa)) = 1 - \text{Succ}(\text{BS1:0}(\Gamma, \mathcal{A}, \kappa))$.

Definition 39. Let $\Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$ be an election scheme, $\mathcal{A}$ be a probabilistic polynomial-time adversary, $\epsilon$ be a constant symbol, and $\kappa$ be a security parameter. We introduce games $G_0, G_1, \ldots$ defined as follows.
\[ G_{i}(\Gamma, A, \epsilon, \kappa) = \]
\[
\begin{align*}
(pk, sk, mb, mc) &\leftarrow \text{Setup}(\kappa); \\
L &\leftarrow \emptyset; W &\leftarrow \emptyset; \\
nc &\leftarrow A(pk, \kappa); bb &\leftarrow A^{O}(); \\
v &\leftarrow (0, \ldots, 0); \\
\text{// vector of length } nc
\end{align*}
\]
\[
\text{for } b \in bb \land (b, v_0, v_1) \notin L \text{ do} \\
\quad (v', pf) &\leftarrow \text{Tally}(sk, nc, \{b\}, \kappa); \\
\quad W &\leftarrow W \cup \{(b, v')\}; \\
\quad v &\leftarrow v + v'; \\
\text{for } b \in bb \land (b, v_0, v_1) \in L \text{ do} \\
\quad v[v_0] &\leftarrow v[v_0] + 1; \\
\quad g &\leftarrow A(v, \epsilon, W); \\
\quad \text{if } g = 0 \land \text{balanced}(bb, nc, L) \land |bb| \leq mb \land nc \leq mc \text{ then} \\
\quad \quad \text{return } 1 \\
\quad \text{else} \\
\quad \quad L &\text{ return } 0
\end{align*}
\]

Oracle \( O \) is defined such that \( O(v_0, v_1) \) computes, on inputs \( v_0, v_1 \in \{1, \ldots, nc\} \), the following:

\[
\text{if } |L| < i \text{ then} \\
\quad b &\leftarrow \text{Vote}(pk, nc, v_1, \kappa); \\
\text{else} \\
\quad b &\leftarrow \text{Vote}(pk, nc, v_0, \kappa); \\
\quad L &\leftarrow L \cup \{(b, v_0, v_1)\}; \\
\text{return } b
\]

**Fact 27.** Let \( \Pi \) be an asymmetric encryption scheme. Suppose \( \text{Enc2Vote}(\Pi) = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}) \). There exists a negligible function \( \text{negl} \), such that for all security parameters \( \kappa \), bulletin boards \( bb_0 \) and \( bb_1 \) such that \( bb_0 \cap bb_1 = \emptyset \), and integers \( nc \), we have

\[
\Pr[(pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa); \\
(\mathbf{v}, pf) \leftarrow \text{Tally}(sk, nc, bb_0 \cup bb_1, \kappa); \\
(\mathbf{v}_0, pf_0) \leftarrow \text{Tally}(sk, nc, bb_0, \kappa); \\
(\mathbf{v}_1, pf_1) \leftarrow \text{Tally}(sk, nc, bb_1, \kappa); \\
|bb_0 \cup bb_1| \leq mb \land nc \leq mc \Rightarrow \mathbf{v} = \mathbf{v}_0 + \mathbf{v}_1 > 1 - \text{negl}(\kappa).
\]

Proof of Fact 27. The proof follows from Definition 7. \( \square \)

**Lemma 28.** Let \( \Pi \) be an asymmetric encryption scheme satisfying IND-CPA and let \( \Gamma = \text{Enc2Vote}(\Pi) \). Suppose \( \epsilon \) is the constant symbol used by \( \Gamma \). We have for all probabilistic polynomial-time adversaries \( A \) and security parameters \( \kappa \) that \( \text{Succ}(\text{BS0}(\Gamma, A, \kappa)) = \text{Succ}(G_0(\Gamma, A, \epsilon, \kappa)) \) and \( \text{Succ}(\text{BS1:0}(\Gamma, A, \kappa)) = \text{Succ}(G_q(\Gamma, A, \epsilon, \kappa)) \), where \( q \) is an upper-bound on adversary \( A \)’s 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 oracle calls $O(v_0, v_1)$ 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 oracle calls $O(v_0, v_1)$ 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 $G_q$, where $L$ is the set constructed by the oracle in $G_q$.

It follows that $|bb| \leq mb \land nc \leq mc$ in $BS_0$, respectively $BS_1:0$, iff $|bb| \leq mb \land nc \leq mc$ in $G_0$, respectively $G_q$. Moreover, predicate balanced is satisfied in $BS_0$, respectively $BS_1:0$, iff predicate balanced is satisfied in $G_0$, respectively $G_q$. Hence, if $|bb| \leq mb \land nc \leq mc$ is not satisfied or if predicate balanced is not satisfied, then $Succ(BS_0(\Gamma, A, \kappa)) = Succ(G_0(\Gamma, A, \epsilon, \kappa))$ and $Succ(BS_1:0(\Gamma, A, \kappa)) = Succ(G_q(\Gamma, A, \epsilon, \kappa))$, concluding our proof. Otherwise, it suffices to show that the inputs to the third adversary call are equivalent.

By inspection of games $BS_0$ and $G_0$, respectively $BS_1:0$ and $G_q$, it is trivial to see that the third element of the triple input to the adversary call is equivalently computed in each game. Furthermore, the second element of the triple input to the adversary call in $G_0$, respectively $G_q$, is $\epsilon$ and, by definition of $\Gamma$, it is also $\epsilon$ in $BS_0$, respectively $BS_1:0$. It remains to show that the first element of the triple input to the adversary call, namely the outcome, is equivalently computed in games $BS_0$ and $G_0$, respectively $BS_1:0$ and $G_q$.

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 individually tallying each ballot on the bulletin board that was constructed by the adversary (i.e., ballots in $\{b \mid b \in bb \land (b, v_0, v_1) \notin L\}$, where $bb$ is the bulletin board and $L$ is the set constructed by the oracle), and by simulating the tally of the remaining ballots (i.e., ballots constructed by the oracle, namely, ballots in $\{b \mid b \in bb \land (b, v_0, v_1) \in L\}$). By Fact 27, it suffices to prove that the simulations are valid, i.e., in $G_0$ and $G_q$, computing

$$\text{for } b \in bb \land (b, v_0, v_1) \in L \text{ do }$$
$$\begin{align*}
    &v[v_0] \leftarrow v[v_0] + 1 \\
\end{align*}$$

is equivalent to

$$\text{for } b \in bb \land (b, v_0, v_1) \in L \text{ do }$$
$$\begin{align*}
    &v \leftarrow \text{Dec}(sk, b); \\
    &\text{if } 1 \leq v \leq nc \text{ then } \\
    &\begin{align*}
        &v[v] \leftarrow v[v] + 1
    \end{align*}
\end{align*}$$

where $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$.

In $G_0$, we have for all $(b, v_0, v_1) \in L$ that $b$ is an output of $\text{Enc}(pk, v_0)$ such that $1 \leq v_0 \leq nc$. And $v_0$ is from the plaintext space, thus, $\text{Dec}(sk, b) = v_0$ by correctness of $\Pi$. Similarly, in $G_q$, we have for all $(b, v_0, v_1) \in L$ that $b$ is an output of $\text{Enc}(pk, v_1)$ such that $1 \leq v_1 \leq nc$. And $v_1$ is from the plaintext.
space, thus, \( \text{Dec}(sk, b) = v_1 \) by correctness of \( \Pi \). Hence, computing \( \text{for } b \in \mathbb{b} \land (b, v_0, v_1) \in L \text{ do } v \leftarrow \text{Dec}(sk, b); \text{if } 1 \leq v \leq nc \text{ then } v[v] \leftarrow v[v] + 1 \) is equivalent to

\[
\text{for } b \in \mathbb{b} \land (b, v_0, v_1) \in L \text{ do } v \leftarrow \text{Dec}(sk, b); \text{if } 1 \leq v \leq nc \text{ then } v[v] \leftarrow v[v] + 1
\]

In \( G_0 \), it follows by correctness of \( \Pi \) that the simulation is valid. Moreover, since predicate balanced holds in \( G_q \), we have for all \( v \in \{1, \ldots, nc\} \) that \( \{|b | b \in \mathbb{b} \land (b, v, v_1) \in L\} = \{|b | b \in \mathbb{b} \land (b, v_0, v) \in L\} \), where \( \mathbb{b} \) is the bulletin board and \( L \) is the set constructed by the oracle. Hence, in \( G_q \), computing

\[
\text{for } b \in \mathbb{b} \land (b, v_0, v_1) \in L \text{ do } v[v_0] \leftarrow v[v_0] + 1;
\]

is equivalent to

\[
\text{for } b \in \mathbb{b} \land (b, v_0, v_1) \in L \text{ do } v[v_1] \leftarrow v[v_1] + 1;
\]

Thus, the simulation is valid in \( G_q \) too, thereby concluding our proof.

**Proof of Proposition 6.** Let \( \Gamma = \text{Enc2Vote}(\Pi) \). Let us suppose \( \Gamma \) does not satisfy \textbf{Ballot-Secrecy}, i.e., there exists a probabilistic polynomial-time adversary \( A \), such that for all negligible 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 \( \text{BS0} \) and \( \text{BS1} \), we have

\[
= \frac{1}{2} \cdot (\text{Succ(\text{BS0}(\Gamma, A, \kappa)))} + \text{Succ(\text{BS1}(\Gamma, A, \kappa)))}
\]

And, by Lemma 26, we have

\[
= \frac{1}{2} \cdot (\text{Succ(\text{BS0}(\Gamma, A, \kappa))) + 1 - \text{Succ(\text{BS1:0}(\Gamma, A, \kappa)))})
= \frac{1}{2} + \frac{1}{2} \cdot (\text{Succ(\text{BS0}(\Gamma, A, \kappa))) - \text{Succ(\text{BS1:0}(\Gamma, A, \kappa)))})
\]

with non-negligible probability. Let \( \epsilon \) be the constant symbol used by \( \Gamma \) and let \( q \) be an upper-bound on the number of oracle queries made by \( A \). Hence, by Lemma 28, we have

\[
= \frac{1}{2} + \frac{1}{2} \cdot (\text{Succ(\text{G0}(\Gamma, A, \epsilon, \kappa))) - \text{Succ(\text{Gq}(\Gamma, A, \epsilon, \kappa)))})
\]

which can be rewritten as a telescoping series

\[
= \frac{1}{2} + \frac{1}{2} \cdot \sum_{0 \leq j < q} \text{Succ(\text{Gj}(\Gamma, A, \epsilon, \kappa))) - \text{Succ(\text{Gj+1}(\Gamma, A, \epsilon, \kappa)))}
\]
Suppose \( \text{Succ}(G_i(\Gamma, \mathcal{A}, \epsilon, \kappa)) - \text{Succ}(G_{i+1}(\Gamma, \mathcal{A}, \epsilon, \kappa)) \) is the largest term in the series, where \( i \in \{0, \ldots, q - 1\} \). Hence,

\[
\frac{1}{2} + \frac{1}{2} \cdot q \cdot (\text{Succ}(G_i(\Gamma, \mathcal{A}, \epsilon, \kappa)) - \text{Succ}(G_{i+1}(\Gamma, \mathcal{A}, \epsilon, \kappa)))
\]

Thus,

\[
\frac{1}{2} + \frac{1}{q} \cdot \text{negl}(\kappa) < \frac{1}{2} + \frac{1}{2} \cdot (\text{Succ}(G_i(\Gamma, \mathcal{A}, \epsilon, \kappa)) - \text{Succ}(G_{i+1}(\Gamma, \mathcal{A}, \epsilon, \kappa)))
\]

From \( \mathcal{A} \), we construct an adversary \( \mathcal{B} \) against \( \Pi \), and show that \( \mathcal{B} \) wins with probability at least \( \frac{1}{2} + \frac{1}{2} \cdot (\text{Succ}(G_i(\Gamma, \mathcal{A}, \epsilon, \kappa)) - \text{Succ}(G_{i+1}(\Gamma, \mathcal{A}, \epsilon, \kappa))) \).

Let \( \Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}) \) and \( \Pi = (\text{Gen}, \text{Enc}, \text{Dec}) \). We define adversary \( \mathcal{B} \) as follows.

- \( \mathcal{B}(pk, \kappa) \) computes \( nc \leftarrow \mathcal{A}(pk, \kappa); L \leftarrow \emptyset \) and runs \( \mathcal{A} \), handling oracle calls \( O(v_0, v_1) \) as follows, namely, if \( |L| < i \), then compute \( b \leftarrow \text{Enc}(pk, v_1); L \leftarrow L \cup \{(b, v_0, v_1)\} \) and return \( b \) to \( \mathcal{A} \), otherwise, assign \( v_0^* \leftarrow v_0; v_1^* \leftarrow v_1 \) and output \( (v_0, v_1) \).
- \( \mathcal{B}(y) \) assigns \( L \leftarrow L \cup \{(y, v_0^*, v_1^*)\}; \) returns \( y \) to \( \mathcal{A} \) and handles any further oracle calls \( O(v_0, v_1) \) as follows, namely, computes \( b \leftarrow \text{Enc}(pk, v_0); L \leftarrow L \cup \{(b, v_0, v_1)\} \) and returns \( b \) to \( \mathcal{A} \); assigns \( \mathcal{A} \)'s output to \( bb \); supposes \( \{b_1, \ldots, b_k\} = bb \setminus \{b \mid (b, v_0, v_1) \in L\} \); and outputs \( (b_1, \ldots, b_k) \) to the challenger.
- \( \mathcal{B}(p) \) initialises \( W \) as the empty set and \( v \) as a zero-filled vector of length \( nc \), computes

\[
\text{for } 1 \leq j \leq k \text{ do}
\]

\[
\text{v}' \leftarrow (0, \ldots, 0); \quad // \text{vector of length } nc
\]

\[
\text{if } 1 \leq p[j] \leq nc \text{ then}
\]

\[
\text{v}[p[j]] \leftarrow \text{v}[p[j]] + 1;
\]

\[
\text{v}'[p[j]] \leftarrow 1;
\]

\[
W \leftarrow W \cup \{(b_j, v')\};
\]

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

\[
\text{v}[v_0] \leftarrow \text{v}[v_0] + 1;
\]

\[
g \leftarrow \mathcal{A}(v, \epsilon, W);
\]

and outputs \( g \).

We prove that \( \mathcal{B} \) wins \text{IND-PA0} against \( \Pi \) with non-negligible probability.

Suppose \( (pk, sk) \) is an output of \( \text{Gen}(\kappa) \). Further suppose we run \( \mathcal{B}(pk, \kappa) \).

It is trivial to see that \( \mathcal{B}(pk, \kappa) \) simulates the challenger and oracle in both \( G_i \) and \( G_{i+1} \). In particular, \( \mathcal{B}(pk, \kappa) \) simulates the first \( i - 1 \) oracle calls. Since \( G_i \) and \( G_{i+1} \) are equivalent to adversaries that make less than \( i \) oracle queries, adversary \( \mathcal{A} \) must make at least \( i \) queries to ensure that \( \frac{1}{2} \cdot (\text{Succ}(G_i(\Gamma, \mathcal{A}, \epsilon, \kappa)) - \text{Succ}(G_{i+1}(\Gamma, \mathcal{A}, \epsilon, \kappa))) \)
Succ(G_{i+1}(\Gamma, \mathcal{A}, \epsilon, \kappa))) is non-negligible. Hence, termination of \mathcal{B} is guaranteed with non-negligible probability. Suppose \mathcal{B} terminates by outputting \((m_0, m_1)\), corresponding to the inputs of \mathcal{A}'s \(i\)th left-right oracle call. Further suppose \(y\) is an output of Enc\((pk, m_\beta)\), where \(\beta\) is a bit, and \(c\) is an output of \(\mathcal{B}(y)\). If \(\beta = 0\), then \(\mathcal{B}(y)\) simulates the oracle in \(G_i\), otherwise (\(\beta = 1\)), \(\mathcal{B}(y)\) simulates the oracle in \(G_{i+1}\). By definition of \(\mathcal{B}\), we have \(c = (b_1, \ldots, b_k)\) such that

\[
\{b_1, \ldots, b_k\} = \mathbb{b} \setminus \{b \mid (b, v_0, v_1) \in L\}
\]

where \(\mathbb{b}\) is \(\mathcal{A}\)'s output. Let \(p \leftarrow (\text{Dec}(sk, c[1]), \ldots, \text{Dec}(sk, c[|c|]))\). And suppose \(g\) is an output of \(\mathcal{B}(p)\). Let us assume that if \(\beta = 0\), then \(\mathcal{B}(p)\) simulates the challenger in \(G_i\), otherwise, \(\mathcal{B}(p)\) simulates the challenger in \(G_{i+1}\), i.e., we assume the following claims:

**Claim 29.** Computing \(W\) as

\[
W \leftarrow \emptyset; \quad \text{for } 1 \leq j \leq k \text{ do } \\
\quad \text{if } 1 \leq p[j] \leq nc \text{ then } \\
\quad \quad v[p[j]] \leftarrow 1; \\
\quad \quad W \leftarrow W \cup \{(b_j, v)\};
\]

is equivalent to computing \(W\) as

\[
W \leftarrow \emptyset; \quad \text{for } b \in \mathbb{b} \wedge (b, v_0, v_1) \notin L \text{ do } \\
\quad (v, pf) \leftarrow \text{Tally}(sk, nc, \{b\}, \kappa); \\
\quad W \leftarrow W \cup \{(b, v)\};
\]

**Claim 30.** Computing \(v\) as

\[
v \leftarrow (0, \ldots, 0); \quad \text{vector of length } nc \\
\text{for } 1 \leq j \leq k \text{ do } \\
\quad \text{if } 1 \leq p[j] \leq nc \text{ then } \\
\quad \quad v[p[j]] \leftarrow v[p[j]] + 1;
\]

is equivalent to computing \(v\) as

\[
v \leftarrow (0, \ldots, 0); \quad \text{vector of length } nc \\
\text{for } b \in \mathbb{b} \wedge (b, v_0, v_1) \notin L \text{ do } \\
\quad (v', pf) \leftarrow \text{Tally}(sk, nc, \{b\}, \kappa); \\
\quad v \leftarrow v + v';
\]

In the above claims, it suffices to consider set \(L\), since it corresponds to the set generated by the oracle in \(G_i\), if \(\beta = 0\), respectively \(G_{i+1}\) if \(\beta = 1\).

By Claims 29 & 30, we have either:

- \(\beta = 0\) and \(\mathcal{B}(p)\) simulates the challenger in \(G_i\), thus, \(g = \beta\) with at least the probability that \(\mathcal{A}\) wins \(G_i\).
• $\beta = 1$ and $B(p)$ simulates the challenger in $G_{i+1}$, thus, $g \neq 0$ with at least the probability that $A$ looses $G_{i+1}$ and, since $A$ wins game \textbf{Ballot-Secrecy}, we have $g$ is a bit, hence, $g = \beta$.

It follows that $B$’s success is at least $\frac{1}{2} \cdot \text{Succ}(G_i(\Gamma, A, \epsilon, \kappa)) + \frac{1}{2} \cdot (1 - \text{Succ}(G_{i+1}(\Gamma, A, \epsilon, \kappa)))$, thus we conclude our proof by proving Claims 29 & 30.

\textbf{Proof of Claim 29.} By definition of $p$ and since $\text{Dec}$ is deterministic, the former computation is equivalent to

\begin{verbatim}
W $\leftarrow \emptyset$
for $1 \leq j \leq k$ do
    \begin{align*}
        v &\leftarrow (0, \ldots, 0); \text{ // vector of length } nc \\
        \text{if } 1 \leq \text{Dec}(sk, b_j) \leq nc \text{ then} \\
        \quad [v[\text{Dec}(sk, b_j)] \leftarrow 1; \\
        \quad W \leftarrow W \cup \{(b_j, v)\};
    \end{align*}
\end{verbatim}

Moreover, by definition of \textbf{Tally} and properties of addition, and since $\text{Dec}$ is deterministic, the later computation is equivalent to

\begin{verbatim}
W $\leftarrow \emptyset$
for $b \in bb \wedge (b, v_0, v_1) \notin L$ do
    \begin{align*}
        v &\leftarrow (0, \ldots, 0); \text{ // vector of length } nc \\
        \text{if } 1 \leq \text{Dec}(sk, b) \leq nc \text{ then} \\
        \quad [v[\text{Dec}(sk, b)] \leftarrow 1 \\
        \quad W \leftarrow W \cup \{(b, v)\};
    \end{align*}
\end{verbatim}

Hence, we conclude by (1).

\textbf{Proof of Claim 30.} By definition of $p$ and since $\text{Dec}$ is deterministic, the former computation computes vector $v$ as

\begin{verbatim}
v $\leftarrow (0, \ldots, 0); \text{ // vector of length } nc 
for 1 \leq j \leq k do
    \begin{align*}
        \text{if } 1 \leq \text{Dec}(sk, b_j) \leq nc \text{ then} \\
        \quad [v[\text{Dec}(sk, b_j)] \leftarrow v[\text{Dec}(sk, b_j)] + 1;
    \end{align*}
\end{verbatim}

Moreover, by definition of \textbf{Tally} and since $\text{Dec}$ is deterministic, the latter computation computes vector $v$ as

\begin{verbatim}
v $\leftarrow (0, \ldots, 0); \text{ // vector of length } nc 
for b \in bb \wedge (b, v_0, v_1) \notin L do
    v' $\leftarrow (0, \ldots, 0); \text{ // vector of length } nc 
    \text{if } 1 \leq \text{Dec}(sk, b) \leq nc \text{ then} \\
    \quad [v'[\text{Dec}(sk, b)] \leftarrow v'[\text{Dec}(sk, b)] + 1 \\
    \quad v \leftarrow v + v';
\end{verbatim}

which is equivalent to
C.7 Proof of Lemma 7

Let $\Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$. Suppose $\kappa$ is a security parameter, $nb$ and $nc$ are integers, $v, v_1, \ldots, v_{nb} \in \{1, \ldots, nc\}$ are votes, and $\text{Setup}(\kappa)$ outputs $(pk, sk, mb, mc)$. Suppose algorithm $\text{Reveal}$ is not correct with respect to $\Gamma$. We construct an adversary $A$ against game $\text{Reveal-Soundness}$.

- $A(pk, \kappa)$ computes for $1 \leq i \leq nb$ do $b_i \leftarrow \text{Vote}(pk, nc, v_i, \kappa)$ and outputs $(nc, \{b_1, \ldots, b_{nb}\}, v)$.

We consider the interesting case: $nb \leq mb \wedge nc \leq mc$. Since $\text{Setup}$ is efficient, integers $mb$ and $mc$ can be efficiently computed. Moreover, since $\text{Vote}$ is efficient, $nb \leq mb \wedge nc \leq mc$, and $v \in \{1, \ldots, nc\}$, adversary $A$ is efficient, i.e., $A$ is a probabilistic polynomial-time adversary.

Suppose $A(pk, \kappa)$ outputs $(nc, \{b_1, \ldots, b_{nb}\}, v)$ and $W$ is computed as follows.

$W \leftarrow \emptyset$; for $b \in bb$ do $(v, pf) \leftarrow \text{Tally}(sk, nc, \{b\}, \kappa)$; $W \leftarrow W \cup \{(b, v)\}$;

By correctness of $\Gamma$, we have for all $1 \leq i \leq nb$ that $\text{Tally}(sk, nc, \{b_i\}, \kappa)$ outputs $(v, pf)$ such that $v[v_i] = 1$. Suppose $\text{Reveal}(sk, nc, \{b_1, \ldots, b_{nb}\}, v, \kappa)$ outputs $b$. Since $\text{Reveal}$ is not correct with respect to $\Gamma$, we have $b \neq \{b_i \mid v_i = v \land 1 \leq i \leq nb\} = \{b \mid (b, v) \in W \land v[x] = 1\}$, with non-negligible probability. Hence, $A$ wins game $\text{Reveal-Soundness}$, concluding our proof.

C.8 Proof of Proposition 8

Let $\Gamma = (\text{Setup}_\Gamma, \text{Vote}, \text{Tally}, \text{Verify}_\Gamma)$, $\Sigma = \Lambda(\Gamma, \text{Reveal}) = (\text{Setup}_\Sigma, \text{Bid}, \text{Open}, \text{Verify}_\Sigma)$, and $\epsilon$ be the constant used by algorithm $\text{Open}$. Suppose $\Sigma$ does not satisfy bid secrecy, hence, there exists an adversary $A$, such that for all negligible functions $\text{negl}$, there exists a security parameter $\kappa$ and $\text{Succ}(\text{Bid-Secrecy}(\Sigma, A, \kappa)) > \frac{1}{2} + \text{negl}(\kappa)$. We construct an adversary $B$ that wins $\text{Ballot-Secrecy}(\Gamma, B, \kappa)$:

- $B(pk, \kappa)$ computes $np \leftarrow A(pk, \kappa)$ and outputs $np$.
- $B()$ initialises $L \leftarrow \emptyset$, computes $bb \leftarrow A()$, and outputs $bb$. Any oracle calls from $A$ on inputs $(p_0, p_1)$ are forwarded to $B$’s oracle and a transcript of calls is maintained, i.e., $B$ computes $b \leftarrow O(p_0, p_1)$; $L \leftarrow L \cup \{(b, p_0, p_1)\}$ and returns $b$ to $A$.  

45
\* \(B(v, pf, W)\) proceeds as follows. Finds the largest integer \(p\) such that \(v[p] > 0 \land 1 \leq p \leq np\); if no such integer exists, then algorithm \(B\) computes \(g \leftarrow A(0, \emptyset, \epsilon)\), and outputs \(g\). If \((b, p, p_1) \in L \land b \in \{0\},\) then abort. Otherwise, algorithm \(B\) assigns \(b \leftarrow \{b \mid (b, v') \in W \land v'[p] = 1\}\), computes \(g \leftarrow A(p, b, \epsilon)\), and outputs \(g\).

It is trivial to see that \(B(pk, \kappa)\) and \(B()\) simulate \(A\)'s challenger to \(A\). Let us prove that \(B(v, pf, W)\) simulates \(A\)'s challenger. In essence, we must prove that \(B\) simulates algorithm \(Open\). By inspection of \(Ballot\)-\texttt{Secrecy}, we have \(v\) and \(pf\) are output by algorithm \(Tally\). By inspection of adversary \(B\) and algorithm \(Open\), if there is no integer \(b\) such that \(v[p] > 0 \land 1 \leq p \leq np\), then it is trivial to see that \(B\) simulates algorithm \(Open\). Otherwise, it suffices to prove that: 1) \(B\) aborts with negligible probability, and 2) \(B\) simulates \(Reveal\) to produce \(b\) with overwhelming probability. We prove each condition as follows.

1. We will prove this by contradiction. Suppose \(B\) aborts with non-negligible probability, hence, \((b, p, p_1) \in L \land b \in \{0\},\) where \(p\) is the largest integer such that \(v[p] > 0 \land 1 \leq p \leq np\). By definition of \(Ballot\)-\texttt{Secrecy}, we have \(b\) was produced by the oracle. And by definition of the oracle, there exists coins \(r\) such that \(b = Vote(pk, np, p, \kappa; r) \lor b = Vote(pk, np, p_1, \kappa; r)\) and \(1 \leq p, p_1 \leq nc\). Since \(A\) wins the \(Bid\)-\texttt{Secrecy} game, we infer balanced\((bb, np, L)\), hence, there exists \(b', p_0, r\) such that \((b', p_0, p) \in L \land b' \in \{0\} \land np \leq nc \land ((b' = Vote(pk, np, p, \kappa; r) \lor b = Vote(pk, np, p, \kappa; r)) \lor (b' = Vote(pk, np, p, \kappa; r') \land b = Vote(pk, np, p_1, \kappa; r))).\)

Let \(v_0\) and \(v_1\) be zero-filled vectors of length \(np\). By correctness of \(\Gamma\), the computation \(v_0[p] ← 1; v_1[p] ← 1; v_0[p_0] ← v_0[p_0] + 1; v_1[p_1] ← v_1[p_1] + 1; (v', p'') ← Tally(sk, np, \{b, b'\}, \kappa)\) ensures \(v' = v_0 \lor v' = v_1\), with overwhelming probability. Moreover, we have \(v'[p] ≥ correct\text{-}outcome(pk, np, \{b, b'\}, \kappa)[p]\) by weak tally soundness, and we also have \(v'[p_0] ≥ correct\text{-}outcome(pk, np, \{b, b'\}, \kappa)[p_0] \lor v'[p_1] ≥ correct\text{-}outcome(pk, np, \{b, b'\}, \kappa)[p_1].\) Thus, by definition of \(correct\text{-}outcome\), we have

\[
\text{If follows that }
\exists r . \text{ Bid}(pk, np, p, \kappa; r) ∈ \{0\} \land 1 \leq p \leq np
\]

(3)

Since \(\Gamma\) satisfies weak tally soundness, we have for all \(p' \in \{1, \ldots, np\}\) that \(v[p'] ≥ correct\text{-}outcome(pk, np, \{b, b'\}, \kappa)[p']\), with overwhelming probability. Moreover, since \(p\) is the largest integer such that \(v[p] > 0 \land 1 \leq p \leq np\), we have for all \(p' \in \{p + 1, \ldots, np\}\) that \(v[p'] ≤ 0\). Hence, by definition of \(correct\text{-}outcome\), we have, with overwhelming probability, that:

\[
\neg \exists p', r' . \text{ Bid}(pk, np, p', \kappa; r') ∈ \{0\} \land p < p' \leq np
\]

(4)

By (3) \& (4), we derive that \(correct\text{-}price(pk, np, \{b, b'\}, \kappa)\) holds with overwhelming probability. Furthermore, since \(A\) wins the \(Ballot\)-\texttt{Secrecy} game
it follows for all \( b \in \mathbb{b} \) that \( (b, p, p_1) \not\in L \) with overwhelming probability. However, we have assumed \( (b, p, p_1) \in L \land b \in \mathbb{b} \) with non-negligible probability, hence we derive a contradiction.

2. Since \( \mathcal{B} \) aborts with negligible probability, we can infer \( b \in \mathbb{b} \) implies \( (b, p, p_1) \not\in L \) with overwhelming probability. By this inference and by definition of Ballot-Secrecy, we have \( W \) is a set of pairs \( (b, v') \) such that \( b \in \mathbb{b} \) and \( (v', p') \) is output by Tally for some \( p' \). It follows by definition of \( \mathcal{B} \) that \( b = \{ b \mid (b, v') \in W \land v'[p] = 1 \} \). Since Reveal satisfies reveal soundness with respect to \( \Gamma \), we have \( \mathcal{B} \) simulates Reveal.

We have shown that \( \mathcal{B} \) simulates \( \mathcal{A}'s \) challenger with overwhelming probability. It follows that \( \mathcal{B} \) determines \( \beta \) correctly with the same success as \( \mathcal{A} \) with overwhelming probability, hence, \( \mathcal{B} \) wins Ballot-Secrecy(\( \Gamma, \mathcal{B}, \kappa \)) with overwhelming probability, thereby deriving a contradiction and concluding our proof. \( \square \)

### C.9 Proof of Proposition 9

Let \( \text{Enc2Vote}(\mathcal{I}) = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}) \) and \( \mathcal{I} = (\text{Gen}, \text{Enc}, \text{Dec}) \). Moreover, let \( \text{Reveal-Enc2Bid}(\mathcal{I}) \) be algorithm \( \text{Reveal-Enc2Bid} \) such that:

- \( \text{Reveal-Enc2Bid}(sk, nc, \mathbb{b}, v, \kappa) \) computes \( b \leftarrow \{ b \mid b \in \mathbb{b} \land \text{Dec}(sk, b) = v \} \) and outputs \( b \).

It follows from Definitions 2, 9 & 7 that auction schemes \( \text{Enc2Bid}(\mathcal{I}) \) and \( \Lambda(\text{Enc2Vote}(\mathcal{I}), \text{Reveal-Enc2Bid}(\mathcal{I})) \) are equivalent, assuming the same constant is used by \( \text{Enc2Vote}(\mathcal{I}), \text{Enc2Bid}(\mathcal{I}) \), and \( \Lambda(\text{Enc2Vote}(\mathcal{I}), \text{Reveal-Enc2Bid}(\mathcal{I})) \). Hence, by Proposition 6 and 8, to show that \( \text{Enc2Bid}(\mathcal{I}) \) satisfies bid secrecy, it suffices to show that \( \text{Enc2Vote}(\mathcal{I}) \) satisfies weak tally soundness and that \( \text{Reveal-Enc2Bid}(\mathcal{I}) \) satisfies reveal soundness with respect to \( \text{Enc2Vote}(\mathcal{I}) \).

We prove \( \text{Enc2Vote}(\mathcal{I}) \) satisfies weak tally soundness by contradiction. Suppose \( \kappa \) is a security parameter and \( \text{Setup}(\kappa) \) outputs \( (pk, sk, mb, mc) \). Further suppose \( nc \) is an integer and \( \mathbb{b} \) is a set such that \( |\mathbb{b}| \leq mb \land nc \leq mc \). Moreover, suppose \( \text{Tally}(sk, nc, \mathbb{b}, \kappa) \) outputs \( (v, pf) \). Let \( \ell = \text{correct-outcome}(pk, nc, \mathbb{b}, \kappa, v) \). Suppose there exists \( v \in \{1, \ldots, nc\} \) such that \( v[v] < \ell \). By definition of \( \text{correct-outcome} \), we have \( \exists b \in \mathbb{b} \setminus \{ \bot \} : \exists r : b = \text{Enc}(pk, v, r) \). And by definition of Vote, bulletin board \( \mathbb{b} \) contains \( \ell \) ciphertexts for plaintext \( v \). Since \( pk, sk \) are outputs of \( \text{Gen} \) and since \( \mathcal{I} \) is perfectly correct, we have that those \( \ell \) ciphertexts all decrypt to \( v \). By definition of Tally, it follows that \( v[v] \geq \ell \), thereby deriving a contradiction.

We prove \( \text{Reveal-Enc2Bid}(\mathcal{I}) \) satisfies reveal soundness with respect to election scheme \( \text{Enc2Vote}(\mathcal{I}) \). Suppose \( \kappa \) is a security parameter and \( \text{Setup}(\kappa) \) outputs \( (pk, sk, mb, mc) \). Further suppose \( \mathbb{b} \) is a set and \( nc \) and \( \ell \) are integers such that \( |\mathbb{b}| \leq mb \land 1 \leq \ell \leq nc \leq mc \). Moreover, suppose \( \text{Reveal-Enc2Bid}(sk, nc, \mathbb{b}, v, \kappa) \) outputs \( b \). By definition of \( \text{Reveal-Enc2Bid} \), we have

\[
\mathbb{b} = \{ b \mid b \in \mathbb{b} \land \text{Dec}(sk, b) = v \}.
\]
Suppose $W$ is computed as follows.

$$W \leftarrow \emptyset;$$

for $b \in \mathbb{b}$ do

$$\begin{align*}
(v, pf) &\leftarrow \text{Tally}(sk, nc, \{b\}, \kappa); \\
W &\leftarrow W \cup \{(b, v)\};
\end{align*}$$

Let $v_0$ be a zero-filled vector of length $nc$. By definition of Tally, it follows that $W$ can be equivalently computed as follows.

$$W \leftarrow \emptyset;$$

for $b \in \mathbb{b}$ do

$$\begin{align*}
v &\leftarrow v_0; \\
v' &\leftarrow \text{Dec}(sk, b); \\
\text{if } 1 \leq v' \leq nc \text{ then} \\
v[v'] &\leftarrow 1; \\
W &\leftarrow W \cup \{(b, v)\};
\end{align*}$$

We have for all $(b, v) \in W$ that $v[v] = 1$ if $\text{Dec}(sk, b) = v$, hence, we derive $b = \{b \mid (b, v) \in W \land v[v] = 1\}$). It follows that reveal soundness with respect to $\text{Enc2Vote}(\Pi)$ is satisfied.

\section*{C.10 Proof of Theorem 10}

Let $\Sigma = \Lambda(\Gamma, \text{Reveal})$ and $\Sigma' = \Lambda(\Gamma, \text{Reveal}, \Delta)$. By Proposition 8, we have that $\Sigma$ satisfies bid secrecy. We prove $\Sigma'$ satisfies bid secrecy by contradiction. Suppose $\Sigma'$ does not satisfy bid secrecy, hence, there exists an adversary $A$, such that for all negligible functions $\text{negl}$, there exists a security parameter $\kappa$ and $\text{Succ(Bid-Secrecy}(\Sigma', A, \kappa)) > \frac{1}{2} + \text{negl}(\kappa)$. Let us construct an adversary $B$ that wins $\text{Bid-Secrecy}(\Sigma, B, \kappa)$.

- $B(pk, \kappa)$ computes $nc \leftarrow A(pk, \kappa)$ and outputs $nc$.
- $B()$ computes $\mathbb{b} \leftarrow A()$, forwarding any oracle calls to its own oracle, and outputs $\mathbb{b}$.
- $B(p, b, pf)$ computes $pf' \leftarrow S((pk, nc, \mathbb{b}, p, b, \kappa), \kappa); g \leftarrow A(p, b, pf')$ and outputs $g$, where $S$ is a simulator for $\Delta$.

It is trivial to see that $B(pk, \kappa)$ and $B()$ simulate $A$’s challenger to $A$. Moreover, there exists a negligible function $\text{negl}'$ such that $B(p, b, pf)$ simulates $A$’s challenger to $A$ with overwhelming probability $1 - \text{negl}'(\kappa)$, because outputs of $S$ are indistinguishable from proofs output by $\Delta$. Let $q$ be the probability that $A$ determines $\beta$ correctly when $A$ does not see the same distribution of inputs as in $\text{Bid-Secrecy}(\Sigma', A, \kappa)$. The success probability of $B$ is greater than $(1 - \text{negl}'(\kappa)) \cdot (\frac{1}{2} + \text{negl}(\kappa)) + \text{negl}'(\kappa) \cdot q$, hence, $B$ wins $\text{Bid-Secrecy}(\Sigma, B, \kappa)$, deriving a contradiction and concluding our proof.
C.11 Proof of Lemma 11

Let \( \Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}) \). Suppose \( \Gamma \) does not satisfy tally soundness, hence, there exists an adversary \( A \), such that for all negligible functions \( \text{negl} \), there exists a security parameter \( \kappa \) and \( \text{Succ}(\text{W-Tally-Soundness}(\Gamma, A, \kappa)) > \text{negl}(\kappa) \). We construct an adversary \( B \) that wins \( \text{Exp-UV-Ext}(\Gamma, B, \kappa) \):

- \( B(\kappa) \) computes \((pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa); (nc, bb) \leftarrow A(pk, \kappa); (v, pf) \leftarrow \text{Tally}(sk, nc, bb, \kappa) \) and outputs \((pk, nc, bb, v, pf)\).

Since \( A \) wins \( \text{W-Tally-Soundness}(\Gamma, A, \kappa) \), we have: \( \Pr[\{(pk, nc, bb, v, pf) \leftarrow B(\kappa) : v \neq \text{correct-outcome}(pk, nc, bb, \kappa) \wedge [bb \leq mb \wedge nc \leq mc] > \text{negl}(\kappa) \} > 1 - \text{negl}(\kappa) \). It follows that: \( \Pr[\{(pk, nc, bb, v, pf) \leftarrow B(\kappa) : v \neq \text{correct-outcome}(pk, nc, bb, \kappa) \wedge \text{Verify}(pk, nc, bb, v, pf, \kappa) = 1] > \text{negl}(\kappa) \). Hence, \( B \) wins \( \text{Exp-UV-Ext}(\Gamma, B, \kappa) \). \( \square \)

C.12 Proof of Theorem 14

Let \( \Sigma = \Lambda(\Gamma, \text{Reveal}, \Delta) = (\text{Setup}_\Sigma, \text{Bid}, \text{Open}, \text{Verify}_\Sigma) \), \( \Gamma = (\text{Setup}_\Gamma, \text{Vote}, \text{Tally}, \text{Verify}_\Gamma) \), and \( \Delta = (\text{Prove}, \text{Verify}) \).

Suppose \( \Gamma \) satisfies universal verifiability. By definition of universal verifiability, we have \( \Gamma \) satisfies strong injectivity. And, by definition of strong injectivity and by Definition 10, it is trivial to see that \( \Sigma \) satisfies strong injectivity. We proceed by contradiction. Suppose \( \Sigma \) does not satisfy universal verifiability, hence, there exists an adversary \( A \), negligible function \( \text{negl} \), and security parameter \( \kappa \), such that \( \text{Succ}(\text{Exp-UV}(\Sigma, A, \kappa)) > \text{negl}(\kappa) \), i.e.,

\[
\Pr[(pk, np, bb, p, b, \sigma) \leftarrow A(\kappa) \setminus \text{correct-price}(pk, np, bb, p, \kappa) \vee \text{correct-bids}(pk, np, bb, p, b, \kappa)) \setminus \text{Verify}_\Sigma(pk, np, bb, p, b, \sigma, \kappa) = 1] > \text{negl}(\kappa)
\]

(5)

We construct adversaries \( B \) and \( C \), from adversary \( A \), such that either \( B \) wins \( \text{Exp-UV-Ext}(\Gamma, B, \kappa) \) or \( \text{Pr}((s, \tau) \leftarrow C(\kappa) : (s, w) \notin R(\Gamma, \text{Reveal}) \wedge \text{Verify}(s, \tau, \kappa) = 1] \) is non-negligible:

- \( B(\kappa) \) computes \((pk, np, bb, p, b, \sigma) \leftarrow A(\kappa), \) parses \( \sigma \) as \((v, pf, pf')\), and outputs \((pk, np, bb, v, pf')\).

- \( C(\kappa) \) computes \((pk, np, bb, p, b, \sigma) \leftarrow A(\kappa), \) parses \( \sigma \) as \((v, pf, pf')\), assigns \( s \leftarrow (pk, np, bb, p, b, \kappa) \), and outputs \((s, pf')\).

Henceforth, we assume that adversaries \( B \) and \( C \) successfully parse \( \sigma \). This assumption is necessary for \( A \) to win \( \text{Exp-UV}(\Sigma, A, \kappa) \), hence we do not lose generality.
First, we consider adversary $B$’s success. Let $\psi(v, p, np)$ hold if $p$ is the largest integer such that $v[p] > 0$ and $1 \leq p \leq np$, or there is no such integer and $p = 0$. By definition of $\psi$ and by inspection of $\text{Verify}_\Sigma$, we have:

$$\text{Verify}_\Sigma(pk, np, bb, p, b, \sigma, \kappa) = 1 \Rightarrow \text{Verify}(pk, np, bb, \sigma[1], \sigma[2], \kappa) = 1 \land \psi(\sigma[1], p, np) \quad (6)$$

Let us assume the following:

$$\psi(v, p, np) \land \neg \text{correct-price}(pk, np, bb, p, \kappa)$$

$$\Rightarrow v \neq \text{correct-outcome}(pk, np, bb, \kappa) \quad (7)$$

By (6) & (7) and logical reasoning, we have: $\text{Verify}_\Sigma(pk, np, bb, p, \sigma, \kappa) = 1 \land \neg \text{correct-price}(pk, np, bb, p, \kappa) \Rightarrow \text{Verify}(pk, np, bb, \sigma[1], \sigma[2], \kappa) = 1 \land \sigma[1] \neq \text{correct-outcome}(pk, np, bb, \kappa)$. It follows that:

$$\Pr[\{pk, nc, bb, v, pf\} \leftarrow B(\kappa) : \text{Verify}(pk, nc, bb, v, pf, \kappa) = 1$$

$$\land v \neq \text{correct-outcome}(pk, nc, bb, \kappa)]$$

$$\geq \Pr[\{pk, np, bb, p, b, \sigma\} \leftarrow A(\kappa) : \text{Verify}_\Sigma(pk, np, bb, p, b, \sigma, \kappa) = 1$$

$$\land \neg \text{correct-price}(pk, np, bb, p, \kappa)] \quad (8)$$

Equation (8) relates $B$’s success to $A$’s success.

Secondly, we consider adversary $C$’s success. By further inspection of $\text{Verify}_\Sigma$, we have:

$$\text{Verify}_\Sigma(pk, np, bb, p, b, \sigma, \kappa) = 1 \Rightarrow \text{Verify}(pk, np, bb, p, b, \sigma, \kappa[3], \kappa) = 1$$

Moreover, since relation $R(\Gamma, \text{Reveal})$ is $\Lambda$-suitable, we have:

$$\neg \text{correct-bids}(pk, np, bb, p, b, \kappa) \Rightarrow ((pk, np, bb, p, b, \kappa), sk) \not\in R(\Gamma, \text{Reveal})$$

with overwhelming probability. It follows that:

$$\Pr[(s, \tau) \leftarrow C(\kappa) : (s, w) \notin R(\Gamma, \text{Reveal}) \land \text{Verify}(s, \tau, \kappa) = 1]$$

$$\geq \Pr[(pk, np, bb, p, b, \sigma) \leftarrow A(\kappa) : \text{Verify}_\Sigma(pk, np, bb, p, b, \sigma, \kappa) = 1$$

$$\land \neg \text{correct-bids}(pk, np, bb, p, b, \kappa)] \quad (9)$$

with overwhelming probability. Equation (9) relates $C$’s success to $A$’s success.

Finally, we use the relations with $A$’s success to show that either adversary $B$ wins $\text{Exp-UV-Ext}(\Gamma, B, \kappa)$ or $\Pr[(s, \tau) \leftarrow C(\kappa) : (s, w) \notin R(\Gamma, \text{Reveal}) \land \text{Verify}(s, \tau, \kappa) = 1] = 1$ is non-negligible, thereby deriving a contradiction. By (5), (8), & (9), we have:

$$\Pr[(pk, nc, bb, v, pf) \leftarrow B(\kappa) : \text{Verify}(pk, nc, bb, v, pf, \kappa) = 1$$

$$\land v \neq \text{correct-outcome}(pk, nc, bb, \kappa)] > \text{negl}(\kappa)$$

$$\lor \Pr[(s, \tau) \leftarrow C(\kappa) : (s, w) \notin R(\Gamma, \text{Reveal}) \land \text{Verify}(s, \tau, \kappa) = 1] > \text{negl}(\kappa)$$

The above equation shows that $A$’s success provides an advantage for adversary $B$ or $C$. To conclude, it remains to prove (7).
\textbf{Proof of (7).} By inspection of correct-price, we have:

\[
\psi(v, p, np) \lor \neg\text{correct-price}(pk, np, bb, p, \kappa)
\]

\[
= \psi(v, p, np) \lor ((\exists p', r'). \text{Bid}(pk, np, p', \kappa; r') \in bb \setminus \{\bot\} \land p < p' \leq np)
\]

\[
\lor p \notin \{0, \ldots, np\}
\]

\[
\lor (p \neq 0 \land \neg \exists r. \text{Bid}(pk, np, p, \kappa; r) \in bb \setminus \{\bot\})
\]

Moreover, since \(\psi(v, p, np) \land p \notin \{0, \ldots, np\}\) is false, we have:

\[
= \psi(v, p, np) \land ((\exists p', r'). \text{Bid}(pk, nc, p', \kappa; r') \in bb \setminus \{\bot\} \land p < p' \leq np)
\]

\[
\lor (p \neq 0 \land \neg \exists r. \text{Bid}(pk, np, p, \kappa; r) \in bb \setminus \{\bot\})
\]

Furthermore, we have \(\psi(v, p, np) \land \exists p', r'. \text{Bid}(pk, nc, p', \kappa; r') \in bb \setminus \{\bot\} \land p < p' \leq np\) implies \(v \neq \text{correct-outcome}(pk, np, bb, \kappa)\), because \(v[p'] = 0\) by definition of \(\psi\). We also have \(\psi(v, p, np) \land p \neq 0 \land \neg \exists r. \text{Bid}(pk, np, p, \kappa; r) \in bb \setminus \{\bot\}\) implies \(v \neq \text{correct-outcome}(pk, np, bb, \kappa)\), because \(v[p] > 0\). It follows that:

\[
\Rightarrow v \neq \text{correct-outcome}(pk, np, bb, \kappa),
\]

thereby concluding our proof.

\(\square\)

\textbf{C.13 Proof of Lemma 15}

Let \(\Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}_T)\) and \(\delta(\Gamma) = (\text{Prove}, \text{Verify})\). Suppose \((s, sk) \in R(\Gamma, \text{Reveal})\), i.e., \(s\) is a vector \((pk, nc, bb, v, b, \kappa)\) and there exists \(mb, mc, r, r'\) such that \(b = \text{Reveal}(sk, nc, bb, v, b, \kappa; r), (pk, sk, mb, mc) = \text{Setup}(\kappa; r'), 1 \leq v \leq nc \leq mc\), and \(|bb| \leq mb\). Further suppose \(\sigma\) is an output of \(\text{Prove}(s, sk, \kappa)\). By definition of \(\text{Prove}\), we have \(\sigma\) is a pair \((pf_{bb}, pf_b)\) such that \((v_{bb}, pf_{bb})\) is an output of \(\text{Tally}(sk, nc, bb, \kappa)\) and \((v_b, pf_b)\) is an output of \(\text{Tally}(sk, nc, b, \kappa)\), for some \(v_{bb}\) and \(v_b\).

By completeness of election scheme \(\Gamma\), we have \(\text{Verify}(pk, nc, bb, v_{bb}, pf_{bb}, \kappa) = 1\) and \(\text{Verify}(pk, nc, b, v_b, pf_b, \kappa) = 1\), with overwhelming probability. And, since \(\Gamma\) satisfies universal verifiability, we have \(v_{bb} = \text{correct-outcome}(pk, nc, bb, \kappa)\) and \(v_b = \text{correct-outcome}(pk, nc, b, \kappa)\), with overwhelming probability.

Since \(\text{Reveal}\) satisfies reveal soundness with respect to \(\Gamma\), it is trivial to see \(b \leq bb\), because \(b\) is required to be the largest subset of \(bb\) such that each element tallies to a vote for \(v\), i.e., for all \(b \in b\) and outputs \((v, pf)\) of \(\text{Tally}(sk, nc, \{b\})\), we have \(v[v] = 1\), with overwhelming probability. By completeness of \(\Gamma\), we have \(\text{Verify}_T(pk, nc, \{b\}, v, pf, \kappa) = 1\), with overwhelming probability. And, since \(\Gamma\) satisfies universal verifiability, we have \(v[v] = \text{correct-outcome}(pk, nc, \{b\}, \kappa)[v]\), with overwhelming probability. By definition of \(\text{correct-outcome}\), we have \(\exists r : b = \text{Vote}(pk, nc, v, \kappa; r)\), with overwhelming probability. Moreover, by strong injectivity, we have \(\exists r : b = \text{Vote}(pk, nc, v, \kappa; r)\), with overwhelming probability.

51
Hence, \( \text{correct-outcome}(pk, nc, b, \kappa)[v] = |b| \), with overwhelming probability. Furthermore, for all \( v' \in \{1,\ldots, nc\} \setminus \{v\} \) we have \( \text{correct-outcome}(pk, nc, b, \kappa)[v'] = 0 \). Thus, \( \mathbf{v}_b \) is a vector of length \( nc \) which is zero-filled, except for index \( v \) which contains \( |b| \).

Since \( b \) is required to be the largest subset of \( \mathbf{b} \) such that each element tallies to a vote for \( v \), we have for all \( b \in \mathbf{b} \setminus b \) and outputs \( (v, pf) \) of \( \text{Tally}(sk, nc, \{b\}, \kappa) \) that \( \mathbf{v}[v] \neq 1 \). \( \text{Verify}_T(pk, nc, \{b\}, v, pf, \kappa) = 1 \), \( \mathbf{v}[v] = \text{correct-outcome}(pk, nc, \{b\}, \kappa)[v] \), and \( \exists^{v|b} b \in \{b\} \setminus \{\bot\} : \exists r : b = \text{Vote}(pk, nc, v, \kappa; r) \), with overwhelming probability. It follows that \( \mathbf{v}[v] = 0 \), with overwhelming probability. Thus, \( \mathbf{v}_{bb}[v] = \text{correct-outcome}(pk, nc, bb, \kappa)[v] = \text{correct-outcome}(pk, nc, b, \kappa)[v] = \mathbf{v}_b[v] \). \( \square \)

C.14 Proof of Lemma 16

Let \( \Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}_T) \) and \( \delta(\Gamma) = (\text{Prove}, \text{Verify}) \). Suppose \( \delta(\Gamma) \) is not sound, i.e., there exists a probabilistic polynomial-time adversaries \( A \) such that for all negligible function \( \text{negl} \), there exists a security parameters \( \kappa \), and \( \Pr[(s, \sigma) \leftarrow A(\kappa) : (s, w) \notin R(\Gamma, \text{Reveal}) \land \text{Verify}(s, \sigma) = 1] > \text{negl}(\kappa) \). Further suppose \( (s, \sigma) \) is an output of \( A(\kappa) \) such that \( (s, w) \notin R(\Gamma, \text{Reveal}) \) and \( \text{Verify}(s, \sigma) = 1 \).

By definition of \( \text{Verify} \), we have \( s \) parses as vector \( (pk, nc, bb, v, b, b, \kappa) \) and \( \sigma \) parses as \( (pf_{bb}, pf_{b}) \). Let \( \mathbf{v}_b \) be a vector of length \( nc \) that is zero-filled, except for index \( v \in \{1,\ldots, nc\} \) which contains \( |b| \), i.e., \( \mathbf{v}_b[v] = |b| \). Since \( \text{Verify}(s, \sigma) = 1 \), we have \( \text{Verify}_T(pk, nc, b, v, pf_{b}, \kappa) = 1 \) and \( \text{Verify}_T(pk, nc, bb, v, bb, pf_{bb}, \kappa) = 1 \). And, since \( \Gamma \) ensures honest key generation, there exist integers \( mb \) and \( mc \), a private key \( sk \) and coins \( r \) such that \( (pk, sk, mb, mc) = \text{Setup}(\kappa; r), \) \( |b| \leq mb, |bb| \leq mb, \) and \( nc \leq mc \), with non-negligible probability. If \( mb = 0 \), then \( |b| = mb \) and \( |bb| = mb \), hence, \( b = \emptyset \) and \( bb = \emptyset \), and by correctness of \( \text{Reveal} \) we have \( \exists r : b = \text{Reveal}(sk, nc, bb, v, \kappa; r) \), thereby deriving a contradiction and concluding our proof. Otherwise (\( 1 \leq mb \)), we proceed as follows.

Since \( \Gamma \) satisfies universal verifiability, we have \( \mathbf{v}_b = \text{correct-outcome}(pk, nc, b, \kappa) \), hence, \( \text{correct-outcome}(pk, nc, b, \kappa)[v] = |b| \), with overwhelming probability. By definition of \( \text{correct-outcome} \), we have \( \exists^{v|b} b \in b \setminus \{\bot\} : \exists r : b = \text{Vote}(pk, nc, v, \kappa; r) \), with overwhelming probability. Moreover, by strong injectivity, we have \( \exists^{v|b} b \in b \setminus \{\bot\} : \exists r : b = \text{Vote}(pk, nc, v, \kappa; r) \), with overwhelming probability. Thus, \( \bot \notin b \), hence,

\[
\exists^{v|b} b \in b : \exists r : b = \text{Vote}(pk, nc, v, \kappa; r)
\]

(10)

That is, \( b \) is a set of ballots for vote \( v \), with overwhelming probability. Moreover, by perfect correctness of \( \Gamma \), we have for all \( b \in \mathbf{b} \) and outputs \( (v, pf) \) of \( \text{Tally}(sk, nc, \{b\}, \kappa) \) that \( \mathbf{v}[v] = 1 \), with overwhelming probability. Hence, by definition of reveal soundness, we have

\[
\exists r : b = \text{Reveal}(sk, nc, b, v, \kappa; r)
\]

(11)
with overwhelming probability.

Since Verify(s, σ) = 1, we have \( v_{bb} = v_b \), hence, \( v_{bb}[v] = |b| \). Moreover, since \( \Gamma \) satisfies universal verifiability, we have \( v_{bb} = \text{correct-outcome}(pk, nc, bb, \kappa) \), hence, \( \text{correct-outcome}(pk, nc, bb, \kappa)[v] = |b| \), with overwhelming probability. By definition of \text{correct-outcome}, we have \( \exists^{|b|} b \in bb \setminus \{ \bot \} : \exists r : b = \text{Vote}(pk, nc, v, \kappa; r) \). Since Verify(s, σ) = 1, we have \( b \subseteq bb \). And, since (10), we have \( \exists^b b \in bb \setminus \{ b \cup \{ \bot \} \} : \exists r : b = \text{Vote}(pk, nc, v, \kappa; r), i.e., set \( bb \setminus \{ b \cup \{ \bot \} \} \) does not contain any ballot for vote \( v \), with overwhelming probability. It follows for all \( b \in bb \setminus \{ b \cup \{ \bot \} \} \) that \text{correct-outcome}(pk, nc, {b}, \kappa)[v] = 0, with overwhelming probability. By perfect completeness of \( \Gamma \), we have for all outputs \( (v, pf) \) of Tally\( (sk, nc, \{ b \}, \kappa) \) that Verify\( (pk, nc, \{ b \}, v, pf, \kappa) = 1 \). Moreover, since \( \Gamma \) satisfies universal verifiability, we have \( v[v] = 0 \). Hence, by (11) and definition of reveal soundness, we have

\[ \exists r : b = \text{Reveal}(sk, nc, bb \setminus \{ \bot \}, v, \kappa; r) \]  

with overwhelming probability.

By perfect completeness of \( \Gamma \), we have for all outputs \( (v, pf) \) of Tally\( (sk, nc, \{ \bot \}, \kappa) \) that Verify\( (pk, nc, \{ \bot \}, v, pf, \kappa) = 1 \). And, since \( \Gamma \) satisfies universal verifiability, we have \( v[v] = \text{correct-outcome}(pk, nc, bb, \kappa)[v] \). By definition of \text{correct-outcome}, we have \( \exists^{|v[\bot]|} b \in \{ \bot \} \setminus \{ \bot \} : \exists r : b = \text{Vote}(pk, nc, v, \kappa; r) \), hence, \( \exists^0 b \in \emptyset : \exists r : b = \text{Vote}(pk, nc, v, \kappa; r) \) and \( v[v] = 0 \). Thus, by (12) and definition of reveal soundness, we have \( \exists r : b = \text{Reveal}(sk, nc, bb, v, \kappa; r) \), thereby deriving a contradiction and concluding our proof.

C.15 Proof of Lemma 17

Let \( \Delta = (\text{Prove}_1, \text{Verify}) \) and \( \delta(\Gamma) = (\text{Prove}_3, \text{Verify}_3) \). Moreover, let \( S \) be the simulator for \( \Delta \). We derive a simulator \( S_\delta \) from \text{Prove}_3 by replacing

\[ (v_{bb}, pf_{bb}) \leftarrow \text{Tally}(sk, nc, bb, \kappa); (v_b, pf_b) \leftarrow \text{Tally}(sk, nc, b, \kappa) \]

with

\[ (v_{bb}, pf_{bb}) \leftarrow \text{Tally}(sk, nc, bb, \kappa); (v_b, pf_b) \leftarrow \text{Tally}(sk, nc, b, \kappa); \]

\[ pf_{bb} \leftarrow S((pk, nc, bb, v_{bb}, \kappa)); pf_b \leftarrow S((pk, nc, b, v_{b}, \kappa)) \]

Suppose \( \delta(\Gamma) \) does not satisfy zero-knowledge, hence, there exists a probabilistic polynomial-time adversary \( A \), such that for all negligible functions negl, there exists a security parameter \( \kappa \) and Succ\( (ZK(\delta(\Gamma), A, H, S_\delta, \kappa)) > \frac{1}{2} + \text{negl}(\kappa) \). We construct an adversary \( B \) against \( \Delta \) from \( A, S_\delta, \) and \( S \). (For clarity, we rename \( B \)'s oracle \( Q \).)

- \( B(\kappa) \) computes \( g \leftarrow A^{H, P}(\kappa) \) and outputs \( g \), handling \( A \)'s oracle calls to \( P(s, w) \) by computing \( \sigma \leftarrow Q'(s, w, \kappa) \) and returning \( \sigma \) to \( A \), where \( Q' \) is derived from \( S_\delta \) by replacing \( S \) with \( Q \).
We prove the following contradiction: \( \text{Succ}(\mathcal{ZK}(\Delta, \mathcal{B}, \mathcal{H}, S, \kappa)) > \frac{1}{2} + \text{negl}'(\kappa) \), for some negligible function \( \text{negl}' \). It suffices to show that adversary \( \mathcal{B} \) simulates \( \mathcal{A} \)'s oracle \( \mathcal{P} \) to \( \mathcal{A} \).

Suppose adversary \( \mathcal{A} \) calls \( \mathcal{P}(s, sk) \). We have \((s, sk) \in R(\Gamma, \text{Reveal}) \), hence, \( s \) is a vector \((pk, nc, bb, v, b, \kappa)\) such that \( \exists mb, mc, r, r' \), \( b = \text{Reveal}(sk, nc, bb, v, \kappa; r) \wedge (pk, sk, mb, mc) = \text{Setup}(\kappa; r') \wedge 1 \leq v \leq nc \leq mc \wedge |bb| \leq mb \). We distinguish two cases:

- **Case I:** \( \beta = 0 \). Adversary \( \mathcal{A} \) expects \( \mathcal{P}(s, sk) \) to compute \( \sigma \leftarrow \text{Prove}_{\mathcal{A}}(s, sk, \kappa) \) and return \( \sigma \). By definition of \( \text{Prove}_{\mathcal{A}} \), we know \( \sigma \) will be a vector \((\pi_{bb}^b, \pi_f^b)\) such that \((\pi_{bb}^b, \pi_f^b)\) is an output of \( \text{Tally}(sk, nc, bb, \kappa) \) for some \( \pi_{bb}^b \) and \((\pi_b^b, \pi_f^b)\) is an output of \( \text{Tally}(sk, nc, b, \kappa) \) for some \( \pi_b^b \).

  Since there exists a tallying proof system for Helios'16 that satisfies zero-knowledge, we have \( \pi_f^b \) is an output of \( \text{Prove}((pk, nc, bb, \pi_{bb}^b), sk, \kappa; r_{bb}) \), for some coins \( r_{bb} \) chosen uniformly at random. Similarly, we have \( \pi_f^b \) is an output of \( \text{Prove}((pk, nc, bb, \pi_b^b), sk, \kappa; r_b) \), for some coins \( r_b \) chosen uniformly at random. Thus, computing \( \sigma \leftarrow \text{Prove}_{\mathcal{A}}(s, sk, \kappa) \) and returning \( \sigma \), is equivalent to computing \( \sigma \leftarrow Q'(s, w, \kappa) \) and returning \( \sigma \), because the only distinction between \( Q' \) and \( \text{Prove}_{\mathcal{A}} \) is resampling \( \delta \).

- **Case II:** \( \beta = 1 \). Adversary \( \mathcal{A} \) expects \( \mathcal{P}(s, sk) \) to compute \( \sigma \leftarrow S_\delta(s, \kappa) \) and return \( \sigma \), which is trivially equivalent to computing \( \sigma \leftarrow Q'(s, w, \kappa) \) and returning \( \sigma \), because \( Q' \) and \( S_\delta \) are identical in this case.

Hence, in both cases, adversary \( \mathcal{B} \) simulates \( \mathcal{A} \)'s oracle \( \mathcal{P} \) to \( \mathcal{A} \), concluding our proof. \( \square \)

### C.16 Proof of Proposition 18

The if implication follows from Proposition 5, and we consider the only if implication. Suppose \( \Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}) \) is an election scheme. Further suppose \( \Gamma \) satisfies simulation sound private key extractability, hence, there exists an algorithm \( \mathcal{K} \) satisfying the conditions of Definition 21. Moreover, suppose \( \Gamma \) does not satisfy \textbf{Ballot-Secrecy}, i.e., there exists a probabilistic polynomial-time adversary \( \mathcal{A} \), such that for all negligible functions \( \text{negl} \), there exists a security parameter \( \kappa \) and \( \text{Succ}(\text{Ballot-Secrecy}(\Gamma, \mathcal{A}, \kappa)) \leq \frac{1}{2} + \text{negl}(\kappa) \). We construct an adversary \( \mathcal{B} \) against \textbf{Smy-Ballot-Secrecy}.

- \( \mathcal{B}(pk, \kappa) \) computes \( nc \leftarrow \mathcal{A}(pk, \kappa) \), initialises \( \mathcal{H} \) as a transcript of the random oracle’s input and output, computes \( sk \leftarrow \mathcal{K}(\mathcal{H}, pk) \), and outputs \( nc \).

- \( \mathcal{B}(\cdot) \) initialises \( L \leftarrow \emptyset \), computes \( bb \leftarrow \mathcal{A}(\cdot) \), and outputs \( bb \). Any oracle calls from \( \mathcal{A} \) on inputs \((v_0, v_1)\) are forwarded to \( \mathcal{B} \)'s oracle and a transcript of calls is maintained, i.e., \( \mathcal{B} \) computes \( b \leftarrow \mathcal{O}(v_0, v_1) \); \( L \leftarrow L \cup \{(b, v_0, v_1)\} \) and returns \( b \) to \( \mathcal{A} \).
\[ B(v, pf) \text{ computes} \]
\[
W \leftarrow \emptyset; \\
\text{for } b \in bb \land (b, v_0, v_1) \notin L \text{ do} \\
\quad (v', pf') \leftarrow \text{Tally}(sk, nc, \{b\}, \kappa); \\
\quad W \leftarrow W \cup \{(b, v')\}; \\
\quad g \leftarrow A(v, pf, W); \\
\text{and outputs } g.
\]

It is trivial to see that \( B(pk, \kappa) \) and \( B() \) simulate \( A \)'s challenger to \( A \). Moreover, \( B \)'s challenger computes \( pk \) such that \((pk, sk', mb, mc)\) is an output of \( \text{Setup}(\kappa) \), for some \( sk', mb \), and \( mc \). And, since \( \Gamma \) satisfies simulation sound private key extractibility, we have \( B(pk, \kappa) \) computes \( sk \) such that \( sk = sk' \), with overwhelming probability. Hence, \( B(v, pf) \) simulates \( A \)'s challenger to \( A \) too. It follows that \( B \)'s success is \( \text{Succ(Ballot-Secrecy}(\Gamma, A, \kappa)) \) with overwhelming probability, thereby deriving a contradiction and concluding our proof.

D Reveal algorithms exist

We prove that every election scheme has a reveal algorithm (Lemma 31) that is correct with respect to that election scheme (Proposition 32). Our proof follows from election scheme correctness: algorithm Tally can be applied to every ballot on the bulletin board to link votes to ballots. The result is largely theoretical, because the class of reveal algorithms introduced in the proof leak the ballot-vote mapping for every ballot on the bulletin board during execution. This does not violate ballot secrecy, because the tallier is assumed to be trusted, i.e., the tallier is assumed not to disclose mappings. Nevertheless, reveal algorithms which only disclose a set of ballots for a particular vote, i.e., revealing the minimal amount of information, are preferable for privacy, and we demonstrate the existence of such algorithms in the context of our case study (§7).

**Definition 40.** \emph{Given an election scheme} \( \Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}) \), we define \( \rho(\Gamma) = \text{Reveal} \) as follows.

\[
\text{Reveal}(sk, nc, bb, v, \kappa) = \\
\quad b \leftarrow \emptyset; \\
\quad \text{for } b \in bb \text{ do} \\
\quad \quad (v, pf) \leftarrow \text{Tally}(sk, nc, \{b\}, \kappa); \\
\quad \quad \text{if } v[v] = 1 \text{ then} \\
\quad \quad \quad b \leftarrow b \cup \{b\}; \\
\quad \quad \text{return } b
\]

**Lemma 31.** \emph{Given an election scheme} \( \Gamma \), we have \( \rho(\Gamma) \) \emph{is a reveal algorithm}.

**Proposition 32.** \emph{Given an election scheme} \( \Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}) \), we have \( \rho(\Gamma) \) \emph{is correct with respect to} \( \Gamma \).
Lemma 33. Given an election scheme \( \Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}) \), we have \( \rho(\Gamma) \) satisfies reveal soundness with respect to \( \Gamma \).

Proof sketch. It is trivial to see that the set \( \{ b \mid (b, v) \in W \land v[v] = 1 \} \) constructed in game \text{Reveal-Soundness} is equal to the set output by \( \rho(\Gamma) \).

Lemma 34. Given an election scheme \( \Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}) \) satisfying perfect correctness, perfect completeness, and universal verifiability, we have \( R(\Gamma, \rho(\Gamma)) \) is \( \Lambda \)-suitable.

Proof. Let reveal algorithm \( \rho(\Gamma) = \text{Reveal} \). Suppose \( (pk, \kappa) \in R(\Gamma, \text{Reveal}) \), i.e., there exist \( mb, mc, r \) and \( r' \) such that \( (pk, sk, mb, mc) = \text{Setup}(\kappa; r') \), \( b = \text{Reveal}(sk, mc, \kappa; r) \), \( 1 \leq v \leq nc \leq mc \), and \( |bb| \leq mb \). Let \( b' = bb \cap \{ b \mid b = \text{Vote}(pk', \kappa; r'') \} \). To prove relation \( R(\Gamma, \text{Reveal}) \) is \( \Lambda \)-suitable, we need to show that predicate \text{correct-bids} holds, i.e., \( b = b' \). It suffices to prove \( b \in b' \) iff \( b \in b' \).

Case I: \( b \in b' \). By definition of \text{Reveal}, we have \( b \in bb \) and \( \text{Tally}(sk, nc, \{ b \}, \kappa) \) outputs a vector \( v \) such that \( v[v] = 1 \). Moreover, we have \( \text{Verify}(pk, nc, \{ b \}, v, pf, \kappa) = 1 \), by perfect completeness. And we have \( v[v] = \text{correct-outcome}(pk, nc, bb, \kappa)[v] \), by universal verifiability. It follows by definition of \text{correct-outcome} that \( \exists b \in bb \setminus \{ \bot \} : \exists \gamma : b = \text{Vote}(pk, nc, \kappa; r) \), hence, \( \exists r : b = \text{Vote}(pk, nc, \kappa; r) \). Thus, \( b \in b' \), concluding the proof of Case I.

Case II: \( b \in b' \). By definition of \text{b'}, we have \( b \in bb \) and \( b \) is an output of algorithm \text{Vote}. And, by perfect correctness of \( \Gamma \), we have \( \text{Tally}(sk, nc, \{ b \}, \kappa) \) outputs a vector \( v \) such that \( v[v] = 1 \). Thus, by definition of \text{Reveal}, we have \( b \in b \), concluding our proof.
E Separation result

We prove that every election scheme satisfying ballot secrecy can be modified such that ballot secrecy is preserved, yet the auction scheme derived from the modified scheme, using our construction, does not satisfy bid secrecy (Proposition 35). Our proof exploits our construction’s reliance on the tallying algorithm (§4.2): we modify the election scheme’s tallying algorithm such that it announces an incorrect outcome in the presence of an adversary. The modification preserves ballot secrecy, because ballot secrecy does not depend on the correctness of the outcome. However, auction schemes derived from the modified scheme do not satisfy bid secrecy, because the adversary can cause the announcement of an incorrect winning price, which causes the reveal algorithm to disclose the set of bids for that price, which enables losing bidders that bid at that price to be identified.

Proposition 35. There exists a function incorrect-price, such that for all election schemes \( \Gamma \) (that permits at least two prices and at least three bids for some security parameter) satisfying ballot secrecy, we have election scheme incorrect-price(\( \Gamma \)) satisfies ballot secrecy, yet auction scheme \( \Lambda(\text{incorrect-price}(\Gamma), \text{Reveal}) \) does not satisfy bid secrecy, for some reveal algorithm Reveal that is correct with respect to incorrect-price(\( \Gamma \)).

Definition 41. Let \( \Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}) \) be an election scheme. Suppose \( \omega \) and \( \epsilon \) are constant symbols that cannot be output by Vote. We define incorrect-price(\( \Gamma \)) = (\( \text{Setup}, \text{Vote}, \text{Tally}', \text{Verify}' \)), where Tally' and Verify' are defined as follows.

\[
\begin{align*}
\text{Tally}'(pk, sk, nc, bb, \kappa) & \text{ initialises } v \text{ as a zero-filled vector of length } nc, \text{ computes } \\
& \text{if } \omega \in bb \text{ then } v[1] \leftarrow 1; pf \leftarrow \epsilon \text{ else } (v, pf) \leftarrow \text{Tally}(pk, sk, nc, bb, \kappa), \text{ and outputs } (v, pf).
\end{align*}
\]

\[
\text{Verify}'(pk, nc, bb, v, pf, \kappa) \text{ outputs } 1.
\]

Lemma 36. Given an election scheme \( \Gamma \), we have incorrect-price(\( \Gamma \)) is an election scheme.

Proof sketch. It suffices to show that incorrect-price(\( \Gamma \)) satisfies correctness, completeness, and injectivity. Let \( \Gamma = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify}) \). Correctness follows from the underlying scheme, because \( \omega \) cannot be output by Vote. Completeness follows from Fact 24. And Injectivity follows from the underlying scheme, because we do not modify Setup nor Vote. 

Lemma 37. Given an election scheme \( \Gamma \) satisfying ballot secrecy, we have that incorrect-price(\( \Gamma \)) satisfies ballot secrecy.

Proof sketch. Suppose incorrect-price(\( \Gamma \)) does not satisfy ballot secrecy, i.e., there exists an adversary that wins game Ballot-Secrecy against incorrect-price(\( \Gamma \)). From this adversary we can construct an adversary that wins Ballot-Secrecy
against Γ, simulating the tally algorithm if necessary (i.e., in cases when the bulletin board contains the constant used in set membership tests by incorrect-price), hence deriving a contradiction.

Proof of Proposition 35. Suppose Γ is an election scheme satisfying ballot secrecy. By Lemmata 36 & 37, we have incorrect-price(Γ) is an election scheme satisfying ballot secrecy. And, by Proposition 32, there exists a reveal algorithm Reveal that is correct with respect to incorrect-price(Γ). By Lemma 3, we have Λ(incorrect-price(Γ), Reveal) is an auction scheme. And it remains to show that Λ(incorrect-price(Γ), Reveal) does not satisfy bid secrecy.

Let incorrect-price(Γ) = (Setup_Γ, Vote, Tally, Verify_Γ), Λ(incorrect-price(Γ), Reveal) = (Setup, Bid, Open, Verify), and ω be the constant used by the set membership test introduced by incorrect-price. We construct an adversary A against game Bid-Secrecy.

- A(p, κ) outputs 2.
- A() computes \( b_0 \leftarrow \mathcal{O}(1, 2); b_1 \leftarrow \mathcal{O}(2, 1); b_b \leftarrow \{b_0, b_1, \omega\} \) and outputs \( b_b \).
- A(p, b, pf) outputs 0 if \( b_0 \in b \), and 1 otherwise.

Suppose κ is a security parameter and Setup(κ) outputs \((pk, sk, mb, mp)\) such that \(3 \leq mb\) and \(2 \leq mp\), i.e., the scheme permits at least three bids and two prices. Further suppose \( A(pk, κ) \) outputs \( np \) and \( A() \) outputs \( b_b \), hence, we have \( b_b = \{b_0, b_1, \omega\} \), such that

\[
b_0 = \text{Vote}(pk, np, 1 + \beta, κ; r_0) \land b_1 = \text{Vote}(pk, np, 2 - \beta, κ; r_1),
\]

for some coins \( r_0 \) and \( r_1 \), where \( \beta \) is the bit chosen by the challenger. Moreover, suppose Open(sk, np, bb, κ) outputs \((p, b, pf)\), hence, we have \( b \) is an output of Reveal(sk, np, bb, p, κ), where \( p = 1 \), since \( \omega \in bb \). By definition of Reveal, set \( b \) is computed as follows:

\[
b \leftarrow \emptyset;
\]

\[
\text{for } b \in bb \text{ do }
\]

\[
\begin{align*}
& (v, pf) \leftarrow \text{Tally}(sk, np, \{b\}, κ);
& \text{if } v[p] = 1 \text{ then }
& b \leftarrow b \cup \{b\};
\end{align*}
\]

By correctness of incorrect-price(Γ), we have Tally(sk, np, \{b\}, κ) outputs \((v, pf)\) such that \( v[p] = 1 \) iff \( \beta = 0 \), with overwhelming probability. It follows that \( b_0 \in b \) iff \( \beta = 0 \), with overwhelming probability. Hence, \( A(p, b, pf) \) outputs \( g = \beta \), with overwhelming probability. Moreover, we have balanced(bb, np, L). And

\[
correct-price(pk, np, bb, p, κ) \Rightarrow \forall b \in bb . \ (b, p, p_1) \notin L \land (b, p_0, p) \notin L
\]

holds vacuously, because \( b_1 - \beta \in bb \) is a bid for \( 2 > p \), hence, correct-price(pk, np, bb, p, κ) does not hold. Thus, the adversary wins against game Bid-Secrecy, concluding our proof.

\[
58
\]
Helios with tallying by homomorphic combinations

Smyth, Frink & Clarkson [SFC16, SFC17] formalise a generic construction for Helios-like election schemes (Definition 43), which is instantiated on the choice of a homomorphic encryption scheme and sigma protocols for the relations introduced in the following definition.  

**Definition 42 ([SFC17]).** Let $(\Gen, \Enc, \Dec)$ be a homomorphic asymmetric encryption scheme and $\Sigma$ be a sigma protocol for a binary relation $R$.

- $\Sigma$ proves correct key construction if $a ((\kappa, \pk, m), (sk, s)) \in R \iff (pk, sk) = \Gen(\kappa; s)$ and $m$ is the encryption scheme’s plaintext space.
- Suppose $(pk, sk) = \Gen(\kappa; s)$, for some security parameter $\kappa$ and coins $s$, and $m$ is the encryption scheme’s plaintext space.
- $\Sigma$ proves plaintext knowledge in a subspace if $((pk, c, m'), (m, r)) \in R \iff c = \Enc(pk, m; r) \land m \in m' \land m' \subseteq m$.
- $\Sigma$ proves correct decryption if $((pk, c, m), sk) \in R \iff m = \Dec(pk, s)$.

**Definition 43 (Generalised Helios [SFC17]).** Suppose $\Pi = (\Gen, \Enc, \Dec)$ is an additively homomorphic asymmetric encryption scheme, $\Sigma_1$ is a sigma protocol that proves correct key construction, $\Sigma_2$ is a sigma protocol that proves plaintext knowledge in a subspace, $\Sigma_3$ is a sigma protocol that proves correct decryption, and $\mathcal{H}$ is a hash function. Let $\FS(\Sigma_1, \mathcal{H}) = (\ProveKey, \VerKey)$, $\FS(\Sigma_2, \mathcal{H}) = (\ProveCiph, \VerifyCiph)$, and $\FS(\Sigma_3, \mathcal{H}) = (\ProveDec, \VerifyDec)$. We define election scheme generalised Helios, denoted $\Helios(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}) = (\Setup, \Vote, \Tally, \Verify)$, as follows.

**Setup($\kappa$).** Select coins $s$ uniformly at random, compute $(pk, sk) \leftarrow \Gen(\kappa; s)$; $\rho \leftarrow \ProveKey((\kappa, pk, m), (sk, s), \kappa); pk' \leftarrow (pk, m, \rho); sk' \leftarrow (pk, sk)$, let $m$ be the largest integer such that $\{0, \ldots, m\} \subseteq \{0\} \cup m$, and output $(pk', sk', m, m)$, where $m$ is the encryption scheme’s plaintext space.

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

---

Our presentation of Helios extends algorithm Verify to check that the number of candidates is less than the maximum number of candidates. Moreover, we insist that the number of ballots on the bulletin board is less than the maximum number of ballots. This is necessary to ensure Helios produces election schemes which ensure honest key generation.

Given a binary relation $R$, we write $((s_1, \ldots, s_l), (w_1, \ldots, w_k)) \in R \iff P(s_1, \ldots, s_l, w_1, \ldots, w_k)$ for $(s, w) \in R \iff P(s_1, \ldots, s_l, w_1, \ldots, w_k) \land s = (s_1, \ldots, s_l)$ and $w = (w_1, \ldots, w_k)$, hence, $R$ is only defined over pairs of vectors of lengths $l$ and $k$.  

59
for \(1 \leq j \leq nc - 1\) do
  if \(j = v\) then \(m_j \leftarrow 1\); else \(m_j \leftarrow 0\);
  \(c_j \leftarrow \text{Enc}(pk, m_j; r_j);\)
  \(\sigma_j \leftarrow \text{ProveCiph}((pk, c_j, \{0, 1\}), (m_j, r_j), j, \kappa);\)
\(c \leftarrow c_1 \otimes \cdots \otimes c_{nc-1};\)
\(m \leftarrow m_1 \otimes \cdots \otimes m_{nc-1};\)
\(r \leftarrow r_1 \otimes \cdots \otimes r_{nc-1};\)
\(\sigma_{nc} \leftarrow \text{ProveCiph}((pk, c, \{0, 1\}), (m, r), \kappa);\)
Output ballot \((c_1, \ldots, c_{nc-1}, \sigma_1, \ldots, \sigma_{nc});\)

Tally\((sk', nc, bb, \kappa)\). Initialise vectors \(v\) of length \(nc\) and \(pf\) of length \(nc - 1\).
Compute for \(1 \leq j \leq nc\) do \(v[j] \leftarrow 0\). Parse \(sk'\) 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 \(\bigwedge_{j=1}^{nc-1} \text{VerifyCiph}((pk, b_i[j], \{0, 1\}), (b_i[j] + nc - 1), j, \kappa) = 1 \wedge \text{VerifyCiph}((pk, b_i[1] \otimes \cdots \otimes b_i[nc - 1], \{0, 1\}), (b_i[2 \cdot nc - 1], \kappa) = 1\). If \(\{b_1, \ldots, b_\ell\} = \emptyset\), then output \((v, pf)\), otherwise, compute:
for \(1 \leq j \leq nc - 1\) do
  \(c \leftarrow b_1[j] \otimes \cdots \otimes b_\ell[j];\)
  \(v[j] \leftarrow \text{Dec}(sk, c);\)
  \(pf[j] \leftarrow \text{ProveDec}((pk, c, v[j]), sk, \kappa);\)
\(v[nc] \leftarrow \ell - \sum_{j=1}^{nc-1} v[j];\)
Output \((v, pf)\).

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

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

**Lemma 38.** Suppose \(\Pi, \Sigma_1, \Sigma_2, \Sigma_3\) and \(\mathcal{H}\) satisfy the preconditions of Definition 43. Further suppose \(\Pi\) satisfies perfect correctness and is perfectly homomorphic. Moreover, suppose \(\Sigma_1\) and \(\Sigma_2\) satisfy perfect completeness. We have Helios\((\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})\) satisfies perfect correctness.

**Proof sketch.** Smyth, Frink & Clarkson shown that Helios\((\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})\) satisfies correctness. And their proof can be adapted to show perfect correctness. In particular, perfect completeness of \(\Sigma_1\) ensures that algorithm \text{Vote} does not
Lemma 39. Suppose $\Pi$, $\Sigma_1$, $\Sigma_2$, $\Sigma_3$ and $\mathcal{H}$ satisfy the preconditions of Definition 43. Further $\Sigma_1$, $\Sigma_2$ and $\Sigma_3$ satisfy perfect completeness, moreover, $\Sigma_2$ perfectly satisfies special soundness and special honest verifier zero-knowledge, and $\mathcal{H}$ is a random oracle. We have $\text{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ satisfies perfect completeness.

Proof sketch. Smyth, Frink & Clarkson shown that $\text{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ satisfies completeness. And their proof can be adapted to show perfect completeness (assuming the proof of Theorem 22 can be adapted to show $\text{FS}(\Sigma_2, \mathcal{H})$ satisfies perfect simulation sound extractability), when $\Sigma_1$ and $\Sigma_3$ are perfectly complete.

Instantiations of generalised Helios work as follows [SFC16].

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

- **Vote** takes a vote $v \in \{1, \ldots, nc\}$ 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 $\sigma_1, \ldots, \sigma_{nc}$ so that this can be verified. In particular, proof $\sigma_j$ demonstrates ciphertext $c_j$ contains 0 or 1, for all $1 \leq j \leq nc-1$. And proof $\sigma_{nc}$ demonstrates that the homomorphic combination of ciphertexts $c_1 \otimes \cdots \otimes c_{nc-1}$ contains 0 or 1. (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.

Generalised Helios can be instantiated to derive Helios’16, i.e., a formal definition of Helios with tallying by homomorphically combining ciphertexts (§7.1).

Definition 44 (Helios’16 [SFC17]). Election scheme Helios’16 is $\text{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$, where $\Pi$ is additively homomorphic El Gamal [CGS97, §2], $\Sigma_1$ is the sigma protocol for proving knowledge of discrete logarithms by Chaum et al. [CEGP87, Protocol 2], $\Sigma_2$ is the sigma protocol for proving knowledge of...
disjunctive equality between discrete logarithms by Cramer et al. [CFSY96, Figure 1], $\Sigma_3$ is the sigma protocol for proving knowledge of equality between discrete logarithms by Chaum & Pedersen [CP93, §3.2], and $H$ is a random oracle.

Although Helios actually uses SHA-256 [NIS12], we assume that $H$ is a random oracle to prove our results. Moreover, we assume the sigma protocols used by Helios’16 satisfy the preconditions of generalised Helios, that is, [CEGP87, Protocol 2] is a sigma protocol for proving correct key construction, [CFSY96, Figure 1] is a sigma protocol for proving plaintext knowledge in a subspace, and [CP93, §3.2] is a sigma protocol for proving decryption. Furthermore, we assume applying the Fiat-Shamir transformation to those sigma protocols results in non-interactive proof systems satisfying perfect completeness. We leave formally proving these assumptions as future work.

Bernhard et al. [BPW12a, §4] remark that the sigma protocols used by Helios’16 to prove discrete logarithms and equality between discrete logarithms both satisfy special soundness and special honest verifier zero-knowledge, hence, Theorem 22 is applicable. Bernhard et al. also remark that the sigma protocol for proving knowledge of disjunctive equality between discrete logarithms 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 22] still hold.” We adopt the same position and assume that Theorem 22 is applicable.

### G Auction schemes from Helios with tallying by homomorphic combinations

In this appendix, let $(\text{Setup, Vote, Tally, Verify})$ be Helios’16. And let $(\text{Gen, Enc, Dec})$ be additively homomorphic El Gamal [CGS97, §2]. Moreover, let $(\text{ProveKey, VerKey})$, respectively $(\text{ProveDec, VerifyDec})$ and $(\text{ProveCiph, VerifyCiph})$, be 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 equality between discrete logarithms by Chaum & Pedersen [CP93, §3.2], and the sigma protocol for proving knowledge of disjunctive equality between discrete logarithms by Cramer et al. [CFSY96].

**Lemma 40.** There exists a tallying proof system for Helios’16 that satisfies zero-knowledge.

**Proof sketch.** Suppose $\kappa$ is a security parameter, $nc$ is an integer, $bb$ is a bulletin board, $(pk, sk, mb, mc)$ is an output of $\text{Setup}(\kappa)$, and $(v, pf)$ is an output of $\text{Tally}(sk, nc, bb, nc, \kappa)$. By inspection of Tally, we have $pf$ is a vector of proofs produced by $\text{ProveDec}$. And, since $(\text{ProveDec, VerifyDec})$ satisfies zero-knowledge, there trivially exists a tallying proof system for Helios’16 that satisfies zero-knowledge. \hfill \square
Lemma 41. Helios’16 ensures honest key generation.

Proof. Suppose Helios’16 does not ensure honest key generation, hence, there exists a probabilistic polynomial-time adversary, such that for all negligible functions \( \text{negl} \), there exists a security parameter \( \kappa \) and \( \Pr[(pk, nc, bb, v, pf) \leftarrow A(\kappa) : \text{Verify}(pk, nc, bb, v, pf, \kappa) = 1 \Rightarrow \exists sk, mb, mc . (pk, sk, mb, mc) = \text{Setup}(\kappa; r) \wedge 1 \leq mb \wedge |bb| \leq mb \wedge nc \leq mc] \leq 1 - \text{negl}(\kappa) \). Further suppose \( (pk', nc, bb, v, pf) \) is an output of \( A(\kappa) \) such that \( \text{Verify}(pk', nc, bb, v, pf, \kappa) = 1 \). By definition of \text{Verify}, we have \( pk' \) is a vector \( (pk, m, \rho) \) and \( \text{VerKey}(\kappa, pk, m), \rho, \kappa) = 1 \). By Theorem 22, we have \( (\text{ProveKey}, \text{VerKey}) \) satisfies simulation sound extractability, hence, there exists a private key \( sk \) and coins \( s \) such that \( (pk, sk) = \text{Gen}(\kappa; s) \) and \( m \) is the plaintext space, with overwhelming probability. Let \( m \) be the largest integer such that \( \{0, \ldots, m\} \subseteq \mathbb{m} \). By definition of \text{Setup}, there exists \( r \) such that \( (pk', (pk, sk), m, m) = \text{Setup}(\kappa; r) \), with overwhelming probability. Moreover, by definition of \text{Verify}, we have \( |bb| \leq m \) and \( nc \leq m \). Thus, we derive a contradiction and conclude our proof.

Lemma 42. Helios’16 satisfies simulation sound private key extractability.

Proof. We have Helios’16 = Helios\( (\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}) \), where \( \Pi, \Sigma_1, \Sigma_2, \Sigma_3 \) and \( \mathcal{H} \) satisfy the conditions given in Definition 44. Let \( F_S(\Sigma_1, \mathcal{H}) = (\text{ProveKey}, \text{VerKey}) \). Since \( \Sigma_1 \) satisfies special soundness and special honest verifier zero-knowledge, there exists an extractor for \( (\text{ProveKey}, \text{VerKey}) \) by Theorem 22. Let \text{ExtProve} be such an extractor. We define algorithm \( \mathcal{K} \) using \text{ExtProve}.

- \( \mathcal{K}(\mathbf{H}, pk') \) parses \( pk' \) as a vector \( (pk, m, \rho) \), initialises \( Q \) as vector \( ((\kappa, pk, m, \rho)) \) and \( P \) as an empty vector, computes \( W \leftarrow \text{ExtProve}(\mathbf{H}, P, Q), \)
  parses \( W[1] \) as a vector \( (sk, s) \), and outputs \( (pk, sk) \).

Suppose \( (pk', sk', mb, mc) \) is an output of \( \text{Setup}(\kappa) \). By definition of algorithm \text{Setup}, we have \( pk' = (pk, m, \rho), sk' = (pk, sk), \) and \( \rho \) is an output of \( \text{ProveKey}(\kappa, pk, m, (sk, s), \kappa) \), where \( (pk, sk) = \text{Gen}(\kappa; s) \) for some coins \( s \). And, since \text{ExtProve} is an extractor for \( (\text{ProveKey}, \text{VerKey}) \), applying \text{ExtProve} to statement \( (\kappa, pk, m) \) and proof \( \rho \) allows us to extract witness \( (sk, s) \), with overwhelming probability. Hence, \( \mathcal{K}(\mathbf{H}, pk') \) outputs \( sk' \), with overwhelming probability, which concludes our proof.

G.1 Reveal algorithm

Definition 45. We define reveal algorithm \text{Helios-Reveal} as follows.

\text{Helios-Reveal}(sk', nc, bb, v, \kappa) proceeds as follows. Parse \( sk' \) as a vector \( (pk, sk) \).
Let \( \{b_1, \ldots, b_\ell\} \) be the largest subset of \( bb \) satisfying the conditions given by algorithm \text{Tally}. Compute:
Lemma 43. Reveal algorithm Helios-Reveal is correct with respect to Helios’16.

Proof. Suppose $\kappa$ is a security parameter, $nb$ and $nc$ are integers, and $v, v_1, \ldots, v_{nb} \in \{1, \ldots, nc\}$ are votes. Further suppose $(pk', sk', mb, mc)$ is an output of Setup$(\kappa)$ such that $nb \leq mb \land nc \leq mc$, hence $sk'$ is a tuple $(pk, sk)$. Moreover, suppose for each $1 \leq i \leq nb$ that $b_i$ is an output of Vote$(pk', nc, v_i, \kappa)$. Let $bb = \{b_1, \ldots, b_{nb}\}$. Suppose $b$ is an output of Helios-Reveal$(sk', nc, bb, v, \kappa)$. By definition of Helios’16, the largest subset of $bb$ satisfying the conditions given by algorithm Tally, hence, Helios-Reveal operates on $bb$, rather than a strict subset of $bb$. We distinguish two cases.

- Case I: $1 \leq v < nc$. By definition of Vote, we have for all $b \in bb$ that $b[v]$ is an El Gamal ciphertext. Moreover, if $v = v_i$ then $b[v]$ enciphers 1, otherwise, $b[v]$ enciphers 0. By correctness of El Gamal, we have with overwhelming probability that $v_i = v$ implies $\text{Dec}(sk, b[v]) = 1$. Hence, by definition of Helios-Reveal, we have $b \in b$.

- Case II: $v = nc$. By definition of Vote, we have for all $b \in bb$ that $b[1], \ldots, b[nc - 1]$ are El Gamal ciphertexts, each enciphering 0. Given that El Gamal is homomorphic, we have with overwhelming probability that $\text{Dec}(sk, b[1] \otimes \cdots \otimes b[nc - 1]) = 0$. Hence, by definition of Helios-Reveal, we have $b \in b$.

In both cases, it follows that $b = \{b_i \mid v_i = v \land 1 \leq i \leq nb\}$, with overwhelming probability, thereby concluding our proof.

Lemma 44. Reveal algorithm Helios-Reveal satisfies reveal soundness with respect to Helios’16.

Proof. Suppose $\kappa$ is a security parameter, $(pk', sk', mb, mc)$ is an output of Setup$(\kappa)$, and $(nc, bb, v)$ is an output of $\mathcal{A}(pk', \kappa)$, such that $1 \leq v \leq nc \leq mc$ and $|bb| \leq mb$. By definition of algorithm Setup, we have $pk'$ is a tuple $(pk, m, \rho)$, such that $(pk, sk)$ is an output of Gen, $m$ is the plaintext space, and $\rho$ is a proof of correct key construction. Further suppose that $b$ is an output of Helios-Reveal. To prove that Helios-Reveal satisfies reveal soundness with respect to Helios’16, it suffices to show $b = \{b \mid (b, v) \in W \land \text{v}[v] = 1\}$ with overwhelming probability, where $W$ is computed as follows: $W \leftarrow \emptyset; \text{for } b \in bb$ do $(v, pf) \leftarrow \text{Tally}(sk', nc, \{b\}, \kappa); W \leftarrow W \cup \{(b, v)\}$.

By definition of algorithm Tally, we have for all $(b, v) \in W$ that $b \in bb$ and either $\emptyset$ or $\{b\}$ is the largest subset of $\{b\}$ satisfying the conditions given by
algorithm Tally, moreover, in the former case $v$ is a zero-filled vector of length $nc$ and in the latter case $b[1], \ldots, b[nc-1]$ are ciphertexts on plaintexts in $\{0,1\}$ and $v = (\text{Dec}(sk, b[1]), \ldots, \text{Dec}(sk, b[nc-1]), 1 - \sum_{j=1}^{nc-1} v[j])$, with overwhelming probability.

Let $W'$ be the largest subset of $W$ such that for all $(b, v) \in W'$ we have $v$ is not a zero-filled vector. It follows that:

$$\{ b | \ (b, v) \in W' \land v[v] = 1 \} = \{ b | \ (b, v) \in W' \land v[v] = 1 \}$$

(13)

Let $\{b_1, \ldots, b_\ell\}$ be the largest subset of $b\mathbb{b}$ satisfying the tallying conditions. It follows that:

$$\{b_1, \ldots, b_\ell\} = \{ b | \ (b, v) \in W' \}$$

(14)

We distinguish two cases.

- **Case I:** $1 \leq v \leq nc - 1$. We have for all $(b, v) \in W'$ that $v[v] = 1 - \sum_{j=1}^{nc-1} v[j]$. By syntactic equality and (13), it suffices to prove $b = \{ b | \ (b, v) \in W' \land b[v] = 1 \}$. By definition of Helios-Reveal, we have $b = \{ b_i | 1 \leq i \leq \ell \land \text{Dec}(sk, b[v]) = 1 \}$. Hence, we conclude by (14).

- **Case II:** $v = nc$. We have for all $(b, v) \in W'$ that $v[nc] = 1 - \sum_{j=1}^{nc-1} v[j]$. By syntactic equality and (13), it suffices to prove $b = \{ b | \ (b, v) \in W' \land b[v] = 0 \}$. By definition of Helios-Reveal, we have $b = \{ b_i | 1 \leq i \leq \ell \land \text{Dec}(sk, b[nc]) = 0 \}$. Hence, we conclude by (14).

Hence, Helios-Reveal satisfies reveal soundness with respect to Helios'16.

**Lemma 45.** Relation $R(\text{Helios'16, Helios-Reveal})$ is $\Lambda$-suitable.

**Proof.** Suppose $((pk', nc, \mathbb{b}, v, b, \kappa), sk') \in R(\text{Helios'16, Helios-Reveal})$. By definition of $R(\text{Helios'16, Helios-Reveal})$, there exist $mb, mc, r', r''$ such that $(pk', sk', mb, mc) = \text{Setup}(\kappa; r')$, $b = \text{Helios-Reveal}(sk', nc, \mathbb{b}, v, b, r')$, and $1 \leq v \leq nc \leq mc$. Let $b' = \mathbb{b} \cap \{ b | b = \text{Vote}(pk', nc, v, b, \kappa; r'') \}$. To prove relation $R(\text{Helios'16, Helios-Reveal})$ is $\Lambda$-suitable, we need to show that predicate correct-bids holds, i.e., $b = b'$. It suffices to prove $b \in b$ iff $b \in b'$.

65
Case I: $b \in b$. By definition of Helios-Reveal, private key $sk'$ parses as a vector $(pk, sk)$ and $b \in b'$, hence, it remains to prove $b$ is an output of algorithm Vote for vote $v$.

By definition of Helios-Reveal, we have that $b$ satisfies the conditions given by algorithm Tally. Thus, $b$ is a vector of length $2 \cdot nc - 1$ and $\bigwedge^{nc-1}_{j=1} \text{VerifyCiph}(pk, b[j], \{0,1\}, (b[j + nc - 1]), j) = 1 \wedge \text{VerifyCiph}(pk, b[1] \otimes \cdots \otimes b[nc - 1], \{0,1\}, b[2 \cdot nc - 1]), nc) = 1$. In their proof that Helios’16 satisfies universal verifiability, Smyth, Frink & Clarkson [SFC16, SFC17] show:

1. Simulation sound extractability of $(\text{ProveCiph}, \text{VerifyCiph})$ implies the existence of messages $m_1, \ldots, m_{nc-1} \in \{0,1\}$ and coins $r_1, \ldots, r_{2 \cdot nc - 2}$ such that for all $1 \leq j \leq nc - 1$ we have $b[j + nc - 1] = \text{ProveCiph}(pk, b[j], \{0,1\}, (m_1, r_j), j, m; r_{j + nc - 1})$ and $b[j] = \text{Enc}(pk, m; r_j)$ with overwhelming probability.

2. There exist coins $r_{1, 2 \cdot nc - 1}$ such that $b[2 \cdot nc - 1] = \text{ProveCiph}(pk, c, \{0,1\}, (m, r), nc; r_{2 \cdot nc - 1})$ with overwhelming probability, where $c \leftarrow b[1] \otimes \cdots \otimes b[nc - 1]$, $m \leftarrow m_1 \otimes \cdots \otimes m_{nc-1}$, and $r \leftarrow r_1 \oplus \cdots \oplus r_{nc-1}$.

Thus, there exists $\beta, r$ such that

$$b = \text{Vote}(pk', nc, \beta, r)$$

and either $\beta = nc \wedge \bigwedge^{nc-1}_{j=1} m_j = 0$ or $\beta_i \in \{1, \ldots, nc - 1\} \wedge m_{\beta} = 1 \wedge \bigwedge_{j \in \{1, \ldots, \beta-1, \beta+1, \ldots, nc-1\}} m_j = 0$. And

$$\forall j \in \{1, \ldots, nc - 1\}, m_j = 1 \iff \beta = j \quad (15)$$

$$\sum^{nc-1}_{j=1} m_j = 0 \iff \beta = nc \quad (16)$$

Hence, it suffices to prove $\beta = v$.

By definition of Helios-Reveal, we have either: 1) $\text{Dec}(sk, b[v]) = 1$, hence, $m_{nc} = 1$ by perfect correctness of El Gamal, and $\beta = v$ by (15); or 2) $\text{Dec}(sk, b_1[1] \otimes \cdots \otimes b_1[nc - 1]) = 0$, hence, $m_1 \otimes \cdots \otimes m_{nc-1} = 0$ by the perfect correctness and perfectly homomorphic properties of El Gamal, and since $nc - 1 \leq mc$, we have $m_{1, j} \otimes \cdots \otimes m_{nc-1, j} = \sum^d_{i=1} m_{i, j}$, thus, $\sum^d_{i=1} m_{i, j} = 0$, and $\beta = v$ by (16). Hence, we conclude Case I.

Case II: $b \in b'$. By definition of $b'$, there exists $r$ such that $b = \text{Vote}(pk', nc, v, \kappa; r)$, and by perfect correctness of Helios’16, we have $b$ satisfies the conditions given by algorithm Tally. Moreover, by definition of algorithm Vote, if $1 \leq v < nc$, then there exist coins $r$ such that $b[v] = \text{Enc}(pk, v; r)$, and by perfect correctness of El Gamal, we have $\text{Dec}(sk, b[v]) = 1$, thus, $b \in b$. Otherwise ($v = nc$), for $1 \leq j \leq nc - 1$, there exist coins $r$ such that $b[j] = \text{Enc}(pk, 0; r)$, hence, $\text{Dec}(sk, b_1[1] \otimes \cdots \otimes b_1[nc - 1]) = 0 \otimes \cdots \otimes 0 = 0$ since El Gamal is perfectly correct and perfectly homomorphic, thus, $b \in b$, concluding Case II, and our proof.
G.2 Alternative non-interactive proof system

In Section 7.1, we constructed an auction scheme from Helios'16 using reveal algorithm Helios-Reveal and non-interactive proof system $\delta$(Helios'16). We show that a more efficient auction scheme can be derived from Helios'16 using an alternative non-interactive proof system, designed ad hoc to optimise the number of checks required.

**Definition 46.** We define the tuple of algorithms $(\text{ProveReveal}, \text{VerifyReveal})$ as follows:

\begin{align*}
\text{ProveReveal}(s, sk, \kappa) & \text{ proceeds as follows. Parse } s \text{ as } (pk', nc, bb, v, b, \kappa) \text{ and } pk' \text{ as } (pk, m, \rho). \text{ Output } \bot \text{ if parsing fails or if } \text{VerKey}(\kappa, pk, m, \rho, \kappa) \neq 1 \lor v \not\in \{1, \ldots, nc\} \lor \{1, \ldots, nc\} \not\subseteq m. \text{ Let } \{b_1, \ldots, b_\ell\} \text{ be the largest subset of } bb \text{ satisfying the conditions given by algorithm Tally. Initialise vector } Q \text{ of length } \ell \text{ and compute:}
\end{align*}

\begin{align*}
&\text{for } 1 \leq i \leq \ell \text{ do} \\
&\quad \text{if } 1 \leq v < nc \text{ then} \\
&\quad \quad Q[i] \leftarrow \text{ProveDec}((pk, b_i[v], \text{Dec}(sk, b_i[v])), sk, \kappa); \\
&\quad \text{else} \\
&\quad \quad c \leftarrow b_i[1] \otimes \cdots \otimes b_i[nc - 1]; \\
&\quad \quad Q[i] \leftarrow \text{ProveDec}((pk, c, \text{Dec}(sk, c)), sk, \kappa);
\end{align*}

Output $Q$.

\begin{align*}
\text{VerifyReveal}(s, Q) & \text{ proceeds as follows. Parse } s \text{ as } (pk', nc, bb, v, b, \kappa) \text{ and } pk' \text{ as } (pk, m, \rho). \text{ Output } 0 \text{ if parsing fails or if } \text{VerKey}(\kappa, pk, m, \rho, \kappa) \neq 1 \lor v \not\in \{1, \ldots, nc\} \lor \{1, \ldots, nc\} \not\subseteq m. \text{ Let } \{b_1, \ldots, b_\ell\} \text{ be the largest subset of } bb \text{ satisfying the conditions given by algorithm Tally. Output } 1 \text{ if any of the following checks hold.}
\end{align*}

1. $\{b_1, \ldots, b_\ell\} = \emptyset$, $|Q| = 0$, and $b = \emptyset$.
2. $1 \leq v < nc$, $|Q| = \ell$, $b \subseteq \{b_1, \ldots, b_\ell\}$, and for all $1 \leq i \leq \ell$, if $b_i \in b$, then $\text{VerifyDec}((pk, b_i[v], 1), Q[i], \kappa) = 1$, otherwise, $\text{VerifyDec}((pk, b_i[v], 0), Q[i], \kappa) = 1$.
3. $v = nc$, $|Q| = \ell$, $b \subseteq \{b_1, \ldots, b_\ell\}$, and for all, $1 \leq i \leq \ell$ if $b_i \in b$, then $\text{VerifyDec}((pk, b_i[1] \otimes \cdots \otimes b_i[nc - 1], 0), Q[i], \kappa) = 1$, otherwise, $\text{VerifyDec}((pk, b_i[1] \otimes \cdots \otimes b_i[nc - 1], 1), Q[i], \kappa) = 1$.

Output 0 if all of the checks fail.

**Lemma 46.** The tuple of algorithms $(\text{ProveReveal}, \text{VerifyReveal})$ is a non-interactive proof system for relation $R$(Helios'16, Helios-Reveal) (i.e., it satisfies completeness).

**Proof sketch.** Suppose $(s, sk') \in R$(Helios'16, Helios-Reveal) and $\kappa$ is a security parameter. Since $R$(Helios'16, Helios-Reveal) is defined over vectors of length 6
and bitstrings, we can parse $s$ as $(pk', nc, bb, v, b, \kappa)$. Moreover, by definition of $R(\text{Helios}'16, \text{Helios-}Reveal)$, there exists $mb, mc, r$, and $r'$, such that $b = \text{Helios-}Reveal(sk', nc, bb, v, \kappa; r')$, $(pk', sk', mb, mc) = \text{Setup}(\kappa; r')$ and $1 \leq v \leq nc \leq mc$. And, by definition of \text{Setup}, we have $pk'$ is a vector $(pk, m, \rho)$, where $(pk, sk)$ is an output of \text{Gen}, $m$ is the encryption scheme’s message space, $\rho$ is an output of \text{ProveKey}, and $mc$ is the largest integer such that $\{0, \ldots, mc\} \subseteq \{0\} \cup m$.

We have $\text{VerKey}(\kappa, pk, m, \rho, \kappa) = 1$, by perfect completeness of $(\text{ProveKey}, \text{VerKey})$. We also have $v \in \{1, \ldots, nc\}$ and $\{1, \ldots, nc\} \subseteq m$. Let $\{b_1, \ldots, b_{\ell}\}$ be the largest subset of $bb$ satisfying the conditions given by the Helios’16 tallying algorithm. Suppose \text{ProveReveal}$(s, sk, \kappa)$ outputs $Q$. By definition of algorithm \text{ProveReveal}, we have $Q$ is a vector of length $\ell$. If $\{b_1, \ldots, b_{\ell}\} = \emptyset$, then $|Q| = 0$, and $b = \emptyset$, by definition of algorithms \text{ProveReveal} and Helios-Reveal, hence, $\text{VerifyReveal}(s, Q) = 1$, by definition of algorithm \text{VerifyReveal}, concluding our proof. Otherwise, $b \subseteq \{b_1, \ldots, b_{\ell}\}$ and we proceed by distinguishing two cases.

- **Case I**: $1 \leq v < nc$. Suppose $i \in \{1, \ldots, \ell\}$. We have $Q[i]$ is an output of \text{ProveDec}($pk, b_i[v], \text{Dec}(sk, b_i[v])$), $sk, \kappa$ by definition of \text{ProveReveal}. If $b_i \in b$, then $\text{Dec}(sk, b_i[v]) = 1$ by definition of Helios-Reveal and, with overwhelming probability, $\text{VerifyDec}((pk, b_i[v], 1), Q[i], \kappa) = 1$ by perfect correctness of El Gamal and by perfect completeness of $\Delta$. Otherwise ($b_i \not\in b$), we proceed as follows. We have $\text{Dec}(sk, b_i[v]) \neq 1$ by definition of Helios-Reveal, hence, $\text{Dec}(sk, b_i[v]) = 0$, because $b_i[v]$ is a encryption of a plaintext in $\{0,1\}$, by the tallying conditions of Helios’16. By perfect correctness of El Gamal and by perfect completeness of $\Delta$, we have $\text{VerifyDec}((pk, b_i[v], 0), Q[i], \kappa) = 1$, with overwhelming probability.

- **Case II**: $v = nc$. Suppose $i \in \{1, \ldots, \ell\}$. Let $c = b_i[1] \odot \cdots \odot b_i[nc - 1]$. We have $Q[i]$ is an output of \text{ProveDec}($pk, c, \text{Dec}(sk, c)$), $sk, \kappa$ by definition of \text{ProveReveal}. If $b_i \in b$, then $\text{Dec}(sk, c) = 0$ by definition of Helios-Reveal and, with overwhelming probability, $\text{VerifyDec}((pk, c, 0), Q[i], \kappa) = 1$ by perfect correctness of El Gamal and by perfect completeness of $\Delta$. Otherwise ($b_i \not\in b$), we proceed as follows. We have $\text{Dec}(sk, c) = 1$ by definition of Helios-Reveal and the tallying conditions of Helios’16. By perfect correctness of El Gamal and by perfect completeness of $\Delta$, we have $\text{VerifyDec}((pk, c, 1), Q[i], \kappa) = 1$.

In both cases, one of the checks in \text{VerifyReveal} will succeed, hence, \text{VerifyReveal}$(s, Q) = 1$, with overwhelming probability. \hfill $\square$

**Lemma 47.** Non-interactive proof system $(\text{ProveReveal}, \text{VerifyReveal})$ is zero-knowledge.

**Proof sketch.** Bernhard et al. [BPW12a, §4] remark that $(\text{ProveDec}, \text{VerifyDec})$ is zero-knowledge. Let $S$ be the simulator for $(\text{ProveDec}, \text{VerifyDec})$. Suppose $(\text{ProveReveal}, \text{VerifyReveal})$ does not satisfy zero-knowledge, hence, there exists a probabilistic polynomial-time adversary $A$, such that for all negligible functions $\text{negl}$, there exists a security parameter $\kappa$ and $\text{Succ}(ZK((\text{ProveReveal}, \text{VerifyReveal})$,
We prove the following contradiction:

$$\text{Succ}(ZK((\text{ProveDec}, \text{ProveDec}), B, \mathcal{H}, S, \kappa)) > \frac{1}{2} + \text{negl}(\kappa)$$

We construct an adversary $B$ against $(\text{ProveDec}, \text{VerifyDec})$ from $\mathcal{A}$ and $S$. (For clarity, we rename $B$'s oracle $Q$.)

- $B(\kappa)$ computes $g \leftarrow \mathcal{A}^{K, \mathcal{P}}(\kappa)$ and outputs $g$, handling $\mathcal{A}$'s oracle calls to $\mathcal{P}(s, v)$ by computing $\sigma \leftarrow Q'(s, w, \kappa)$ and returning $\sigma$ to $\mathcal{A}$, where $Q'$ is derived from $\text{ProveReveal}$ by replacing all occurrences of $\text{ProveDec}(s', w', \kappa)$ with $Q(s', w')$.

We prove the following contradiction: $\text{Succ}(ZK((\text{ProveDec}, \text{ProveDec}), B, \mathcal{H}, S, \kappa)) > \frac{1}{2} + \text{negl}(\kappa)$, for some negligible function $\text{negl}$. It suffices to show that adversary $B$ simulates $\mathcal{A}$'s oracle $\mathcal{P}$ to $\mathcal{A}$. It is trivial to see that $\mathcal{P}$ is simulated when $\beta = 0$, because $\mathcal{P}$ and $Q'$ are identical in this case. Moreover, $\mathcal{P}$ is simulated when $\beta = 1$, because $S$ is indistinguishable from $\text{ProveDec}$.

**Lemma 48.** Non-interactive proof system $(\text{ProveReveal}, \text{VerifyReveal})$ is sound.

**Proof sketch.** Suppose $(\text{ProveReveal}, \text{VerifyReveal})$ is not sound, hence, there exists a probabilistic polynomial-time adversary $\mathcal{A}$, such that for all negligible functions $\text{negl}$, there exists a security parameter $\kappa$ and if $\mathcal{A}(\kappa)$ outputs $(s, \sigma)$, then $(s, w) \not\in R(\text{Helios'16}, \text{Helios-Reveal})$ and $\text{VerifyReveal}(s, \sigma) = 1$, with probability greater than $\text{negl}(\kappa)$.

By definition of $\text{VerifyReveal}$, we have $s$ parses as $(pk', nc, bb, v, b, \kappa)$ and $pk'$ as $(pk, m, \rho)$. Moreover, $\text{VerKey}(\kappa, pk, m, \rho, \kappa) = 1 \land v \in \{1, \ldots, nc\} \land \{1, \ldots, nc\} \subseteq m$. Bernhard et al. [BPW12a, §4] remark that $(\text{ProveKey}, \text{VerKey})$ satisfies their notion of simulation sound extractability, hence, $(\text{ProveKey}, \text{VerKey})$ satisfies soundness too. Thus, $w$ parses as $(sk, r)$ such that $(pk, sk) = \text{Gen}(\kappa; r)$ and $m$ is the encryption scheme's message space. Let $sk' = (pk, sk)$ and let $m$ be the largest integer such that $\{0, \ldots, m\} \subseteq \{0\} \cup m$. By definition of $\text{Setup}$, there exists $r'$ such that $(pk, sk, m, m) = \text{Setup}(\kappa; r')$. We have $\{1, \ldots, nc\} \subseteq m$ and $\{0, \ldots, m\} \subseteq m$, hence, $nc \leq m$ by definition of $m$. It follows that $\exists mb, mc, r', (pk, sk, mb, mc) = \text{Setup}(\kappa; r') \land 1 \leq v \leq nc \leq mc$.

Since $(s, w) \not\in R(\text{Helios'16}, \text{Helios-Reveal})$, we have $b$ is not an output of $\text{Helios-Reveal}(sk, nc, bb, v, \kappa)$. We proceed by contradiction: we show that if any of the three checks in $\text{VerifyReveal}$ holds, then $b$ is an output of $\text{Helios-Reveal}(sk, nc, bb, v, \kappa)$. We proceed by case analysis on the three checks.

1. By definition of $\text{Helios-Reveal}$, we have $\{b_1, \ldots, b_{\ell}\} = \emptyset \land b = \emptyset$ implies $b$ is an output of $\text{Helios-Reveal}(sk, nc, bb, v, \kappa)$.

Let $\{b_1, \ldots, b_{\ell}\}$ be the largest subset of $bb$ satisfying the tallying conditions of $\text{Helios'16}$. Hence, $b_i[v], \ldots, b_{\ell}[v]$ are ciphertexts on plaintexts in $\{0, 1\}$. Suppose $b \subseteq \{b_1, \ldots, b_{\ell}\}$ and $|Q| = \ell$. We consider the two remaining checks.

2. Suppose $1 \leq v < nc$ and for all $1 \leq i \leq \ell$, if $b_i \in b$, then $\text{VerifyDec}(pk, b_i[v], 1), Q[i], \kappa) = 1$, otherwise, $\text{VerifyDec}(pk, b_i[v], 0), Q[i], \kappa) = 1$. Bernhard et al. [BPW12a, §4] remark that $(\text{ProveDec}, \text{VerifyDec})$ satisfies their notion of simulation sound extractability, hence, $(\text{ProveDec}, \text{VerifyDec})$ satisfies soundness too. Thus, for all $1 \leq i \leq \ell$, if $b_i \in b$, then $\text{Dec}(sk, b_i[v]) =
1, otherwise, \( \text{Dec}(sk, b_i[v]) = 0 \), with overwhelming probability. It follows that \( b \) is a subset of \( \{ b_1, \ldots, b_\ell \} \) such that for all \( b \in b \) we have \( b[v] \) decrypts to 1, and for all \( b \not\in b \) \( b[v] \) decrypts to 0. Since the tallying conditions of Helios’16 ensure that \( b_1[v], \ldots, b_\ell[v] \) are ciphertexts on plaintexts in \( \{0, 1\} \), we have \( b \) is the largest subset of \( \{ b_1, \ldots, b_\ell \} \) such that for all \( b \in b \) we have \( b[v] \) decrypts to 1. Thus, \( b \) is an output of Helios-Reveal(\( sk, nc, bb, v, \kappa \)).

3. Suppose \( v = nc \) and for all \( 1 \leq i \leq \ell \), if \( b_i \in b \), then \( \text{VerifyDec}((pk, b_i[1] \otimes \cdots \otimes b_i[nc-1], 0), Q[i], \kappa) = 1 \), otherwise, \( \text{VerifyDec}((pk, b_i[1] \otimes \cdots \otimes b_i[nc-1], 1), Q[i], \kappa) = 1 \). Bernhard et al. [BPW12a, §4] remark that (\text{ProveDec, VerifyDec}) satisfies their notion of simulation sound extractability, hence, (\text{ProveDec, VerifyDec}) satisfies soundness too. Thus, for all \( 1 \leq i \leq \ell \), if \( b_i \in b \), then \( \text{Dec}(sk, b_i[1] \otimes \cdots \otimes b_i[nc-1]) = 0 \), otherwise, \( \text{Dec}(sk, b_i[1] \otimes \cdots \otimes b_i[nc-1]) = 1 \), with overwhelming probability. It follows that \( b \) is a subset of \( \{ b_1, \ldots, b_\ell \} \) such that for all \( b \in b \) we have \( b[1] \otimes \cdots \otimes b[nc-1] \) decrypts to 0, and for all \( b \not\in b \) \( b[1] \otimes \cdots \otimes b[nc-1] \) decrypts to 1. Since the tallying conditions of Helios’16 ensure that \( b[1] \otimes \cdots \otimes b[nc-1] \) is a ciphertext on a plaintext in \( \{0, 1\} \), we have \( b \) is the largest subset of \( \{ b_1, \ldots, b_\ell \} \) such that for all \( b \in b \) we have \( b[1] \otimes \cdots \otimes b[nc-1] \) decrypts to 0. Thus, \( b \) is an output of Helios-Reveal(\( sk, nc, bb, v, \kappa \)).

We have shown that if any of the three checks in \text{VerifyReveal} hold, then \( b \) is an output of Helios-Reveal(\( sk, nc, bb, v, \kappa \)), thereby deriving a contradiction, and concluding our proof.

\[ \Box \]

**Theorem 49.** If Helios’16 satisfies ballot secrecy, then auction scheme \( \Lambda(\text{Helios’16, Helios-Reveal, (ProveReveal, VerifyReveal)}) \) satisfies bid secrecy.\(^{46}\)

**Proof.** Smyth, Frink & Clarkson have shown that Helios’16 satisfies universal verifiability [SFC16, SFC17]. It follows from Lemma 11 that Helios’16 satisfies tally soundness. Reveal algorithm Helios-Reveal satisfies reveal soundness with respect to Helios’16 (Lemma 44). And (ProveReveal, VerifyReveal) is a non-interactive proof system for relation \( R(\text{Helios’16, Reveal}) \) (Lemma 46) satisfying zero-knowledge (Lemma 47). Hence, \( \Lambda(\text{Helios’16, Helios-Reveal, (ProveReveal, VerifyReveal)}) \) satisfies bid secrecy (Theorem 10).

\[ \Box \]

**Theorem 50.** Auction scheme \( \Lambda(\text{Helios’16, Helios-Reveal, (ProveReveal, VerifyReveal)}) \) satisfies individual and universal verifiability.

**Proof.** Smyth, Frink & Clarkson have shown that Helios’16 satisfies individual and universal verifiability [SFC16, SFC17]. Hence, we have \( \Lambda(\text{Helios’16, Helios-Reveal}) \) (Helios’16) satisfies individual verifiability (Theorem 13). We have (ProveReveal, VerifyReveal) is a non-interactive proof system for relation \( R(\text{Helios’16, Reveal}) \) (Lemma 46) satisfying soundness (Lemma 48). And R(\( ^{46}\)Formally, \( \Lambda(\text{Helios’16, Helios-Reveal, (ProveReveal, VerifyReveal)}) \) is an auction scheme by Lemmata 4, 43, & 46.)
Helios’16, Helios-Reveal) is \( \Lambda \)-suitable (Lemmata 45). Hence, we have \( \Lambda \) (Helios’16, Helios-Reveal, (ProveReveal, VerifyReveal)) satisfies universal verifiability (Theorem 14).

\[ \square \]

## H Helios with tallying by mixnet

We formalise Helios with tallying by mixnet (§7.2) as the class of election schemes HeliosM’16 (Definition 49).\(^{47}\) We define that class using construction HeliosM (Definition 48), which is instantiated on the choice of a homomorphic encryption scheme and sigma protocols for the relations introduced in the following definition.

**Definition 47** ([SFC16]). Let \((Gen, Enc, Dec)\) be a homomorphic asymmetric encryption scheme and \(\Sigma\) be a sigma protocol for a binary relation \(R\). Suppose that \((pk, sk) = Gen(\kappa; s)\), for some security parameter \(\kappa\) and coins \(s\), and \(m\) is the encryption scheme’s plaintext space.

- \(\Sigma\) proves plaintext knowledge if \(((p, c), (m, r)) \in R \iff c = Enc(pk, m; r) \land m \in m\).
- \(\Sigma\) proves mixing if \(((p, c, c'), (r, \chi)) \in R \iff \bigwedge_{1 \leq i \leq |c|} c'[\chi(i)] = c[i] \otimes Enc(pk, c; r[i]) \land |c| = |c'| = |r|\), where \(c\) and \(c'\) are both vectors of ciphertexts encrypted under \(pk\), \(r\) is a vector of coins, \(\chi\) is a permutation on \(\{1, \ldots, |c|\}\), and \(e\) is an identity element of the encryption scheme’s message space with respect to \(\circ\).

**Definition 48** (Generalised Helios with tallying by mixnet). Suppose \(\Pi = (Gen, Enc, Dec)\) is a multiplicative homomorphic asymmetric encryption scheme, \(e\) is an identity element of \(\Pi\)’s message space with respect to \(\circ\), \(\Sigma_1\) proves correct key construction, \(\Sigma_2\) proves plaintext knowledge, \(\Sigma_3\) proves correct decryption, \(\Sigma_4\) proves mixing, and \(H\) is a hash function. Let \(FS(\Sigma_1, H) = (ProveKey, VerKey)\), \(FS(\Sigma_2, \mathcal{H}) = (ProveCiph, VerifyCiph)\), \(FS(\Sigma_3, \mathcal{H}) = (ProveDec, VerifyDec)\), \(FS(\Sigma_4, \mathcal{H}) = (ProveMix, VerifyMix)\), and \(p\) be some polynomial function. We define election scheme generalised Helios with tallying by mixnet, denoted \(HeliosM(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H}) = (Setup, Vote, Tally, Verify)\), as follows.

**Setup(\(\kappa\)).** Select coins \(s\) uniformly at random, compute \((pk, sk) \leftarrow Gen(\kappa; s)\);
\(\rho \leftarrow ProveKey((\kappa, pk, m), (sk, s), \kappa)\); \(pk' \leftarrow (pk, m, \rho)\); \(sk' \leftarrow (sk, sk)\); let \(mc\) be the largest integer such that \(\{0, \ldots, mc\} \subseteq \{0\} \cup m\), and output \((pk', sk', p(\kappa), mc)\), where \(m\) is the encryption scheme’s plaintext space.

**Vote(\(pk', nc, v, \kappa\)).** Parse \(pk'\) as a vector \((pk, m, \rho)\), output \(\perp\) if parsing fails or \(\text{VerKey}(\kappa, pk, m), \rho, \kappa) \neq 1 \lor v \notin \{1, \ldots, nc\} \lor \{1, \ldots, nc\} \not\subseteq m\). Select coins \(r\) uniformly at random, compute \(c \leftarrow Enc(pk, v; r)\); \(\sigma \leftarrow ProveCiph((pk, c), (v, r), \kappa)\), and output ballot \((c, \sigma)\).

\(^{47}\)Some aspects of Definition 49 first appeared in [Smy18b].
Tally$(sk', nc, bb, \kappa)$. Initialise $v$ as a zero-filled vector of length $nc$. Parse $sk'$ as a vector $(pk, sk)$, output $(v, \bot)$ if parsing fails. Proceed as follows.

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

2. Mix. Select a permutation $\chi$ on $\{1, \ldots, \ell\}$ uniformly at random, initialise $bb$ and $r$ as vectors of length $\ell$, fill $r$ with coins chosen uniformly at random, and compute
   
   for $1 \leq i \leq \ell$ do
   
   $bb[\chi(i)] \leftarrow b_i[1] \otimes \text{Enc}(pk, r[i])$

   $pf_1 \leftarrow \text{ProveMix}((pk, (b_1[1], \ldots, b_\ell[1]), bb), (r, \chi), \kappa)$

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

   for $1 \leq i \leq \ell$ do
   
   $W[i] \leftarrow \text{Dec}(sk, bb[i]);$

   $pf_2[i] \leftarrow \text{ProveDec}((pk, bb[i], W[i]), sk, \kappa);$

   if $1 \leq W[i] \leq nc$ then

   $v[W[i]] \leftarrow v[W[i]] + 1$

Output $(v, (bb, pf_1, W, pf_2))$.

Verify$(pk', nc, bb, v, pf, \kappa)$. Let $mc$ be the largest integer such that $\{0, \ldots, mc\} \subseteq \{0\} \cup m$. Parse $pk'$ as a vector $(pk, m, \rho)$ and $v$ parses as a vector of length $nc$, output 0 if parsing fails, $\text{VerKey}((\kappa, pk, m), \rho, \kappa) \neq 1$, $|bb| \leq p(\kappa)$, or $nc \leq mc$. Proceed as follows.

1. Remove invalid ballots. Compute $\{b_1, \ldots, b_\ell\}$ as per Step (1) of the tallying algorithm. If $\{b_1, \ldots, b_\ell\} = \emptyset$ and $v$ is a zero-filled vector, then output $1$. Otherwise, perform the following checks.

2. Check mixing. Parse $pf$ as a vector $(bb, pf_1, W, pf_2)$, output 0 if parsing fails. Check $\text{VerMix}((pk, (b_1[1], \ldots, b_\ell[1]), bb), pf_1, \kappa) = 1$.

3. Check decryption. Check $W$ and $pf_2$ are vectors of length $\ell$, $\bigwedge_{i=1}^{\ell} \text{VerifyDec}$( $pk, bb[i], W[i], pf_2[i]) = 1$, and $\bigwedge_{i=1}^{nc} \exists v[i] \in \{1, \ldots, \ell\} : v = W[i]$.

If the above checks hold, then output $1$, otherwise, output $0$.

Lemma 51, 53 & 55 demonstrate that HeliosM is a construction for election schemes, proofs of those lemmata appear in [Smy18b]. And Lemma 52 & 54 provide conditions under which HeliosM produces election schemes satisfying perfect correctness and perfect completeness.

**Lemma 51.** Suppose $\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4$ and $H$ satisfy the preconditions of Definition 43. We have HeliosM$(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, H)$ satisfies correctness.
Lemma 52. Suppose \(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4\) and \(\mathcal{H}\) satisfy the preconditions of Definition 48. Further suppose \(\Pi\) satisfies perfect correctness and is perfectly homomorphic. Moreover, suppose \(\Sigma_1, \Sigma_2\) and \(\Sigma_4\) satisfy perfect completeness. We have \(\Gamma = \text{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})\) satisfies perfect correctness.

Proof sketch. Lemma 51 shows that \(\text{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})\) satisfies correctness. And that proof can be adapted to show perfect correctness. In particular, perfect completeness of \(\Sigma_1\) ensures that algorithm \(\text{Vote}\) does not output \(\bot\) and perfect completeness of \(\Sigma_2\) ensures that algorithm \(\text{Tally}\) considers \(bb\) as the largest subset of \(bb\) satisfying the necessary conditions. Since \(\Pi\) is perfectly homomorphic, the mix outputs ciphertexts. And, since \(\Pi\) is perfectly correct, the outcome represents the votes.

Lemma 53. Suppose \(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4\) and \(\mathcal{H}\) satisfy the preconditions of Definition 43. Further suppose \(\Sigma_2\) satisfies special soundness and special honest verifier zero-knowledge, and \(\mathcal{H}\) is a random oracle. We have \(\text{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})\) satisfies completeness.

Proof sketch. Lemma 53 shows that \(\text{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})\) satisfies completeness. And that proof can be adapted to show perfect completeness (assuming the proof of Theorem 22 can be adapted to show FS(\(\Sigma_2, \mathcal{H}\) satisfies perfect simulation sound extractability), when \(\Sigma_1, \Sigma_3\) and \(\Sigma_4\) are perfectly complete.

Lemma 54. Suppose \(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4\) and \(\mathcal{H}\) satisfy the preconditions of Definition 48. Further suppose \(\Sigma_1, \Sigma_2, \Sigma_3\) and \(\Sigma_4\) satisfy perfect completeness, moreover, \(\Sigma_2\) perfectly satisfies special soundness and special honest verifier zero-knowledge, and \(\mathcal{H}\) is a random oracle. We have \(\text{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})\) satisfies perfect completeness.

Proof sketch. Lemma 53 shows that \(\text{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})\) satisfies completeness. And that proof can be adapted to show perfect completeness (assuming the proof of Theorem 22 can be adapted to show FS(\(\Sigma_2, \mathcal{H}\) satisfies perfect simulation sound extractability), when \(\Sigma_1, \Sigma_3\) and \(\Sigma_4\) are perfectly complete.

Lemma 55. Suppose \(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4\) and \(\mathcal{H}\) satisfy the preconditions of Definition 43. Further suppose \(\Pi\) is perfectly correct. We have \(\text{HeliosM}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})\) satisfies Injectivity.

Proof. Let \(\Pi = (\text{Gen}, \text{Enc}, \text{Dec})\) and \(\text{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H}) = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})\). Suppose \(\kappa\) is a security parameter, \(nc\) is an integer, and \(v\) and \(v'\) are votes such that \(v \neq v'\). Further suppose \((pk', sk', mb, nc)\) is an output of \(\text{Setup}(\kappa)\), \(b\) is an output of \(\text{Vote}(pk', nc, v, \kappa)\) and \(b'\) is an output of \(\text{Vote}(pk', nc, v', \kappa)\) such that \(b \neq \bot\) and \(b' \neq \bot\). By definition of \(\text{Setup}\), we have \(pk'\) is a vector \((pk, m, \rho)\) such that \((pk, sk)\) is an output of \(\text{Gen}(\kappa)\) and \(\{1, \ldots, nc\} \subseteq \{0\} \cup m\), where \(m\) is the encryption scheme’s plaintext space. By definition of \(\text{Vote}\), we have \(v, v' \in \{1, \ldots, nc\} \wedge nc \leq |m|\), hence, \(v, v' \in m\). Moreover, there exist coins \(r\) and \(r'\) such that \(b[1] = \text{Enc}(pk, v; r)\) and \(b'[1] = \text{Enc}(pk, v'; r)\). Since \(\Pi\) is perfectly correct, we have \(\text{Dec}(sk, b[1]) = v \neq v' = \text{Dec}(sk, b'[1])\). It follows that \(b[1] \neq b'[1]\), hence, \(b \neq b'\), concluding our proof.

73
Definition 49. HeliosM’16 is the class of election schemes that includes every scheme \( \Gamma \) = HeliosM(\( \Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H} \)) such that: \( \Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4 \) satisfy the preconditions of Definition 48; \( \Pi \) is perfectly correct; \( \Sigma_1, \Sigma_2, \Sigma_3 \) and \( \Sigma_4 \) satisfy perfect completeness, moreover, \( \Sigma_2, \Sigma_3 \) and \( \Sigma_4 \) satisfy special soundness and special honest verifier zero-knowledge; \( \mathcal{H} \) is a random oracle; \( \Gamma \) ensures honest key generation and satisfies Smy-\textit{Ballot-Secrecy}, Exp-IV-Ext, and Exp-UV-Ext.

Smyth has shown that there exists an election scheme in HeliosM’16 that satisfies Smy-\textit{Ballot-Secrecy} [Smy16] and Exp-IV-Ext & Exp-UV-Ext [Smy18b]. Hence, set HeliosM’16 is not empty.

I Auction schemes from Helios with tallying by mixnet

In this appendix, we suppose \( \Gamma = (\text{Setup, Vote, Tally, Verify}) \) is in the class of election schemes HeliosM’16. Moreover, we suppose \( \Pi = (\text{Gen, Enc, Dec}) \) is the asymmetric encryption scheme underlying \( \Gamma \). Furthermore, we suppose PET is a plaintext equality test that inputs a key pair and two ciphertexts, and outputs 1 if the ciphertexts contain the same plaintext.

Lemma 56. Given an election scheme \( \Gamma \) produced by HeliosM(\( \Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H} \)), there exists a tallying proof system for \( \Gamma \) that satisfies zero-knowledge.

Proof sketch. Let \( \text{FS}(\Sigma_3, \mathcal{H}) = (\text{ProveDec, VerifyDec}) \) and \( \text{FS}(\Sigma_4, \mathcal{H}) = (\text{ProveMix, VerMix}) \). Suppose \( \kappa \) is a security parameter, \( nc \) is an integer, \( \mathbf{bb} \) is a bulletin board, \( (pk, sk, mb, mc) \) is an output of \( \text{Setup}(\kappa) \), and \( (V, pf) \) is an output of \( \text{Tally}(sk, nc, \mathbf{bb}, nc, \kappa) \). If \( pf = \bot \), then there trivially exists a tallying proof system for \( \Gamma \) that satisfies zero-knowledge, otherwise, we proceed as follows. By inspection of \( \text{Tally} \), we have \( pf \) is a vector \( (\mathbf{bb}, pf_1, W, pf_2) \) such that \( \mathbf{bb} \) is a vector of mixed ciphertexts, \( pf_1 \) is a proof produced by \( (\text{ProveMix, VerMix}) \), \( W \) is a vector of plaintexts, and \( pf_1 \) is a proof produced by \( (\text{ProveDec, VerifyDec}) \).

By Theorem 22, we have \( (\text{ProveMix, VerMix}) \) and \( (\text{ProveDec, VerifyDec}) \) satisfy zero-knowledge, hence, it suffices to show that \( \mathbf{bb} \) and \( W \) can be constructed by a simulator. (Formally, a single simulator is needed, but it is straightforward to see simulators can be combined, so we omit these details for brevity.)

Let \( \{b_1, \ldots, b_\ell\} \) be the largest subset of \( \mathbf{bb} \) such that for all \( 1 \leq i \leq \ell \) we have \( b_i \) is a pair and \( \text{VerifyCiph}((pk, b_i[1]), b_i[2], \kappa) = 1 \). We can simulate the construction of \( \mathbf{bb} \) as follows: select a permutation \( \chi \) on \( \{1, \ldots, \ell\} \) uniformly at random, initialise \( \mathbf{bb} \) as a vector of length \( \ell \) and compute for \( 1 \leq i \leq \ell \) do \( \mathbf{bb}[\chi(i)] \leftarrow b_i[1] \oplus \text{Enc}(pk, \epsilon) \). Moreover, we can simulate the construction of \( W \) too. Let \( \text{ExtProve} \) be the extractor for \( \text{FS}(\Sigma_2, \mathcal{H}) \). Initialise \( H \) as a transcript of the random oracle’s input and output, and \( P \) as a transcript of simulated proofs. Compute:

\[
\begin{align*}
Q & \leftarrow ((pk, b_1[1]), b_1[2]), \ldots, ((pk, b_\ell[1]), b_\ell[2])); \\
W & \leftarrow \text{ExtProve}(H, P, Q); \\
\text{for } 1 \leq i \leq \ell & \text{ do } W[i] \leftarrow W[i][1];
\end{align*}
\]
Since ExtProve is an extractor, we have ExtProve(H, Π, Q) outputs a vector of witnesses associated with statements \((pk, b_1), \ldots (pk, b_l)\), i.e., a vector of pairs such that the first element is the plaintext corresponding to the ciphertext in the associated statement. Thus, the simulation is valid.

\[\square\]

**Lemma 57.** Given an election scheme \(\Gamma\) produced by HeliosM(\(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H}\)), we have \(\Gamma\) satisfies simulation sound private key extractibility.

A proof of Lemma 57 is similar to our proof of Lemma 42; we omit a formal proof.

### I.1 Reveal algorithm

**Definition 50.** We define reveal algorithm HeliosM-Reveal as follows.

\[\text{HeliosM-Reveal}(sk', nc, bb, v, \kappa)\] parses \(sk'\) as a vector \((pk, sk)\), initialises \(\{b_1, \ldots, b_l\}\) as the largest subset of \(bb\) such that each element is a pair and \(\bigwedge_{1 \leq i \leq l} \text{VerifyCiph}((pk, b_i[1]), b_i[2], \kappa) = 1\), computes \(b \leftarrow \emptyset; c \leftarrow \text{Enc}(pk, v); b \leftarrow \{b_i \mid \text{PET}(pk, sk, b_i[1], c) = 1 \land 1 \leq i \leq l\}\), and outputs \(b\).

**Lemma 58.** Reveal algorithm HeliosM-Reveal is correct with respect to \(\Gamma\).

**Proof sketch.** Suppose \(\kappa\) is a security parameter, \(nb\) and \(nc\) are integers, and \(v, v_1, \ldots, v_{nb} \in \{1, \ldots, nc\}\) are votes. Further suppose \((pk', sk', mb, mc)\) is an output of \(\text{Setup}(\kappa)\) such that \(nb \leq mb \land nc \leq mc\). Moreover, suppose for each \(1 \leq i \leq nb\) that \(b_i\) is an output of \(\text{Vote}(pk', nc, v_i, \kappa)\). Let \(bb = \{b_1, \ldots, b_{nb}\}\). Suppose \(b\) is an output of \(\text{HeliosM-Reveal}(sk', nc, bb, v, \kappa)\). By definition of HeliosM, the largest subset of \(bb\) satisfying the conditions given by algorithm Tally is \(bb\), hence, HeliosM-Reveal operates on \(bb\), rather than a strict subset of \(bb\). By definition of Vote, we have for all \(1 \leq i \leq nb\) that \(b_i[1]\) is a ciphertext on plaintext \(v_i\). Suppose \(c\) is an output of \(\text{Enc}(pk, v)\), where \(sk' = (pk, sk)\). Hence, by correctness of PET, we have \(b = \{b_i \mid v_i = v \land 1 \leq i \leq nb\}\), with overwhelming probability, thereby concluding our proof.

\[\square\]

**Lemma 59.** Reveal algorithm HeliosM-Reveal satisfies reveal soundness with respect to \(\Gamma\), assuming \(\Pi\) is perfectly correct and PET is perfectly correct.

**Proof sketch.** Suppose \(\kappa\) is a security parameter, \((pk', sk', mb, mc)\) is an output of \(\text{Setup}(\kappa)\), and \((nc, bb, v)\) is an output of \(A(pk', \kappa)\), such that \(1 \leq v \leq nc \leq mc\) and \(|bb| \leq mb\). By definition of algorithm Setup, we have \(pk'\) is a triple \((pk, sk, \rho)\), such that \((pk, sk)\) is an output of \(\text{Gen}\), \(m\) is the plaintext space, and \(\rho\) is a proof of correct key construction. Further suppose that \(b\) is an output of HeliosM-Reveal. To prove that HeliosM-Reveal satisfies reveal soundness with respect to \(\Gamma\), it suffices to show \(b = \{b \mid (b, v) \in W \land v[v] = 1\}\) with overwhelming probability, where \(W\) is computed as follows: \(W \leftarrow \emptyset;\) for \(b \in bb\) do \((v, pf) \leftarrow \text{Tally}(sk', nc, \{b\}, \kappa); W \leftarrow W \cup \{(b, v)\}\). We have for all \((b, v) \in W\) that \(b \in bb\) and, by definition of algorithm Tally, either \(\emptyset\) or \(\{b\}\) is the largest subset of \(\{b\}\) such that \(b\) is a pair and VerifyCiph((pk,
b[1]), b[2], κ) = 1. In the former case, we have v is a zero-filled vector of length nc. In the latter case, we have b[1] is a ciphertext, with overwhelming probability. And, if Dec(sk, b[1] ⊗ Enc(pk, ε; r)) ∉ {1, . . . , nc}, then v is a zero-filled vector of length nc, otherwise, v is a zero-filled vector of length nc, except for index Dec(sk, b[1] ⊗ Enc(pk, ε; r)) which contains 1, where r is chosen uniformly at random by algorithm Tally. Since Γ is homomorphic, we have b[1] ⊗ Enc(pk, ε; r) is a ciphertext with overwhelming probability. And, by perfect correctness, we have Dec(sk, b[1]) ⊗ pk Dec(sk, Enc(pk, ε; r)) and Dec(sk, b[1]) ⊗ pk Dec(sk, Enc(pk, ε; r)) = Dec(sk, b[1]) ⊗ pk ε, with overwhelming probability. And, since ε is an identity element, we have Dec(sk, b[1]) ⊗ pk ε = Dec(sk, b[1]), with overwhelming probability. It follows that

\[ \{ b | (b, v) ∈ W ∧ v[v] = 1 \} = \{ b | b ∈ \{b_1, . . . , b_l\} ∧ Dec(sk, b[1]) = v \} \]

where \{b_1, . . . , b_l\} is the largest subset of bb satisfying the tallying conditions. Suppose c is an output of Enc(pk, v). By perfect correctness of PET, we have for all b ∈ \{b_1, . . . , b_l\} that Dec(sk, b[1]) = v iff PET(pk, sk, b[1], c) = 1, with overwhelming probability. Thus, we conclude our proof by definition of HeliosM-Reveal.

Lemma 60. Relation R(Γ, HeliosM-Reveal) is Λ-suitable.

Proof. Suppose ((pk', nc, bb, v, b, κ), sk') ∈ R(Γ, HeliosM-Reveal). By definition of R(Γ, HeliosM-Reveal), there exist mb, mc, r, r' such that (pk', sk', mb, mc) = Setup(κ; r', b = HeliosM-Reveal(sk', nc, bb, v, κ; r), 1 ≤ v ≤ nc ≤ mc, and |bb| ≤ mb. Let b' = bb ∩ \{ b | b = Vote(pk', nc, v, κ; r'') \}. To prove relation R(Γ, HeliosM-Reveal) is Λ-suitable, we need to show that predicate correct-bids holds, i.e., b = b'. It suffices to prove b ∈ b iff b ∈ b'.

Case I: b ∈ b. By definition of HeliosM-Reveal, private key sk' parses as a vector (pk, sk) and b ∈ bb, hence, it remains to prove b is an output of algorithm Vote for vote v. By definition of HeliosM-Reveal, we have that b is a pair such that VerifyCiph((pk, b[1]), b[2], κ) = 1. By Theorem 22, we have (ProveCiph, VerifyCiph) satisfies simulation sound extractability, hence, there exists coins r, r' and a plaintext m from the message space such that b[2] = ProveCiph((pk, b[1], \{0, 1\}), (m, r), κ; r'') and b[1] = Enc(pk, m; r) with overwhelming probability. It follows that b is an output of Vote. Moreover, by perfect correctness of PET, we have m = v, with overwhelming probability. Hence, we conclude Case I.

Case II: b ∈ b'. By definition of b', we have b ∈ bb and b is an output of Vote(pk'), nc, v, κ). Since Γ satisfies perfect correctness (Lemma 52), we have Tally(sk', nc, \{b, κ\}) outputs (v, pf) such that v[v] = 1, thus, b must satisfy the conditions given by algorithm Tally (otherwise, algorithm Tally would output a zero-filled vector). Suppose c is an output of Enc(pk, v). By definition of Vote, we have b[1] is a ciphertext on plaintext v. And, by perfect correctness of PET, we have PET(pk, sk, b[1], c) = 1, thus b ∈ b, concluding Case II, and our proof. □
I.2 Proof of Theorem 21

Election scheme $\Gamma$ is perfectly correct (Lemma 52) and perfectly complete (Lemma 54). And there exists a tallying proof system for $\Gamma$ that satisfies zero-knowledge (Lemma 56). Moreover, $\Gamma$ satisfies simulation sound private key extractibility (Lemma 57). Since $\Gamma$ satisfies Smy-Ballot-Secrecy [Smy16], we have $\Gamma$ satisfies Ballot-Secrecy (Proposition 18). Furthermore, reveal algorithm HeliosM-Reveal($sk', nc, bb, v, \kappa$) satisfies correctness (Lemma 58) and reveal soundness (Lemma 59) with respect to $\Gamma$. And relation $R(\Gamma, \text{HeliosM-Reveal})$ is $A$-suitable (Lemma 60).

We have $\Gamma$ satisfies Smy-Ballot-Secrecy [Smy16] and Exp-IV-Ext & Exp-UV-Ext [Smy18b]. It follows that $\delta(\Gamma)$ is a non-interactive proof system for relation $R(\Gamma, \text{Reveal})$ (Lemma 15) satisfying soundness (Lemma 16) and zero-knowledge (Lemma 17). Hence, we have bid secrecy by Theorem 10, individual verifiability by Theorem 13, and universal verifiability by Theorem 14. 

References


[BCG+15] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. SoK: A comprehensive analysis of game-


78


