Constructing secret, verifiable auction schemes from election schemes

Elizabeth A. Quaglia and Ben Smyth

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

May 24, 2016

Abstract

Auctions and elections are seemingly disjoint research fields. Nevertheless, similar cryptographic primitives are used in both fields. For instance, mixnets, homomorphic encryption, and trapdoor bit-commitments, have been used by state-of-the-art schemes in both fields. 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 secrecy and verifiability. We demonstrate the applicability of our work by deriving an auction scheme from the Helios election scheme. Our results inaugurate the unification of auctions and elections, thereby facilitating the advancement of both fields.

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

1 Introduction

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

Auction schemes. 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.

Election schemes. 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 freely participate in auctions and elections; this can be achieved by participating in private [UN48, OAS69, OSC90, US90], which has led to the emergence of the following requirements [Smy15a, MSQ14a].

- 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, CRS05, Adi06, Dag07, Adi08, DJL13]; this is known as *verifiability.* Sometimes we write *auction verifiability* and *election verifiability* to distinguish verifiability in each field. Verifiability includes the following properties [KRS10, SFC15].

- 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 step is non-trivial, because election schemes satisfying ballot secrecy ensure voters cannot be linked to votes, hence, the bidder in the aforementioned steps 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 election scheme research which can be capitalised upon to advance auctions. This statement can be justified with hindsight: Chaum [Cha81] exploited mixnets in election schemes 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 auction and election schemes. Magkos, Alexandris & Chrissikopoulos remark that auction and election schemes have a similar structure and share similar security properties. And Her, Imamot & Sakurai contrast privacy properties of auction and election schemes, and compare the use of homomorphic encryption and mixnets between fields. More concretely, McCarthy, Smyth & Quaglia [MSQ14a] derive auction schemes from the Helios and Civitas election schemes. Lipmaa, Asokan & Niemi study the converse: they propose an auction scheme and claim that their scheme could be used to construct an election scheme [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 verifiability, assuming the underlying election scheme satisfies ballot secrecy and verifiability. To achieve this, we first 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] Selecting a winning bid from a set of winning bids – i.e., having a strategy to handle tie-breaks – is beyond the scope of this paper.

Summary of contributions and paper structure. We summarize our contributions as follows.

- We propose auction scheme syntax, and the first computational security definitions of bid secrecy and verifiability for auction schemes (Section 2).
- We present a construction for auction schemes from election schemes (Section 3).
- We prove that our construction guarantees bid secrecy (Section 4) and verifiability (Section 5), assuming the underlying election scheme satisfies analogous security properties.
- We use our construction to derive an auction scheme from the Helios election scheme (Section 6).

It follows from our results that secure auction schemes can immediately 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 [MSQ14a]. 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 any security proofs. Thus, the auction scheme we construct from Helios satisfies bid secrecy and verifiability, whereas the auction scheme that they derive from Helios has no such proofs. Moreover, we are the first to introduce computational security definitions of bid secrecy and verifiability for auction schemes.

2 Auction schemes

2.1 Syntax

We formulate syntax for *auction schemes.*

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

Setup*, denoted*[3] $(pk, sk, mb, mp) \leftarrow \mathsf{Setup}(\kappa)$*, is run by the auctioneer*[4]*.* Setup *takes a security parameter* κ *as input and outputs a key pair* pk, sk*, a maximum number of bids* mb*, and a maximum price* mp*.*

[3]Let $A(x_1, \ldots, x_n; r)$ denote the output of probabilistic algorithm A on inputs $x_1, \ldots, x_n$ and random coins r. 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]Some auction schemes permit the auctioneer's role to be distributed amongst several

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

Open, *denoted* $(p, \mathfrak{b}, pf) \leftarrow \mathsf{Open}(sk, np, \mathfrak{bb}, \kappa)$, *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* $\mathfrak{bb}$*, and a security parameter* κ*, where* $\mathfrak{bb}$ *is a set. It outputs a (winning) price p, a set of (winning) bids* $\mathfrak{b}$*, and a non-interactive proof pf of correct opening.*

Verify, *denoted* $s \leftarrow \mathsf{Verify}(pk, np, \mathfrak{bb}, p, \mathfrak{b}, pf, \kappa)$, *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* $\mathfrak{bb}$*, a price p, a set of bids* $\mathfrak{b}$*, a proof pf, and a security parameter* κ*. It outputs a bit s, which is* 1 *if the auction verifies successfully or* 0 *otherwise.*

Auction schemes must satisfy correctness, completeness, *and* injectivity, *which we define below.*

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

Definition 2 (Correctness)**.** *There exists a negligible function* negl*, such that for all security parameters* κ*, integers nb and np, and prices* $p_1, \ldots, p_{nb} \in \{1, \ldots, np\}$*, it holds that*

$$
\begin{aligned}
&\Pr[(pk, sk, mb, mp) \leftarrow \mathsf{Setup}(\kappa);\\
&\quad \textbf{for } 1 \leq i \leq nb \textbf{ do}\\
&\quad \lfloor\ b_i \leftarrow \mathsf{Bid}(pk, np, p_i, \kappa);\\
&\quad (p, \mathfrak{b}, pf) \leftarrow \mathsf{Open}(sk, np, \{b_1, \ldots, b_{nb}\}, \kappa)\\
&\quad : nb \leq mb \wedge np \leq mp \Rightarrow p = \mathtt{max}(0, p_1, \ldots, p_{nb}) \wedge \mathfrak{b} = \{b_i \mid p_i = p \wedge 1 \leq\\
&\quad i \leq nb\}] > 1 - \mathsf{negl}(\kappa).
\end{aligned}
$$

Completeness stipulates that outputs of algorithm Open will be accepted by algorithm Verify. This prevents *biasing attacks* [SFC15].

Definition 3 (Completeness)**.** *There exists a negligible function* negl*, such that for all security parameters* κ*, bulletin boards* $\mathfrak{bb}$*, and integers np, we have*

$$
\begin{aligned}
&\Pr[(pk, sk, mb, mp) \leftarrow \mathsf{Setup}(\kappa);\\
&\quad (p, \mathfrak{b}, pf) \leftarrow \mathsf{Open}(sk, np, \mathfrak{bb}, \kappa);\\
&\quad : |\mathfrak{bb}| \leq mb \wedge np \leq mp \Rightarrow \mathsf{Verify}(pk, np, \mathfrak{bb}, p, \mathfrak{b}, pf, \kappa) = 1] > 1 - \mathsf{negl}(\kappa).
\end{aligned}
$$

auctioneers. For simplicity, we consider only a single auctioneer in this paper. Generalising syntax and security definitions to multiple auctioneers is a possible direction for future work. Similarly, we consider only a single tallier in election schemes.

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

Definition 4 (Injectivity)**.** *For all security parameters* κ*, integers* np*, and prices* p *and* p'*, such that* $p \neq p'$*, we have*

$$\Pr[(pk, sk, mb, mp) \leftarrow \mathsf{Setup}(\kappa); b \leftarrow \mathsf{Bid}(pk, np, p, \kappa); \\ b' \leftarrow \mathsf{Bid}(pk, np, p', \kappa) : b \neq \perp \wedge b' \neq \perp \Rightarrow b \neq b'] = 1.$$

Our proposed syntax is based upon syntax for auction schemes by McCarthy, Smyth & Quaglia [MSQ14a] and syntax for election schemes by Smyth, Frink & Clarkson [SFC15]. Moreover, our correctness, completeness and injectivity properties are based upon similar properties of election schemes. (Cf. Section 3.1.)

2.2 Bid secrecy

We formalise *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 scheme's Setup algorithm, publishes the public key, and only uses the private key for opening.

The adversary has access to a left-right oracle [BDJR97, BR05] which can compute bids on the adversary's behalf. Bids can be computed by the left-right oracle in two ways, corresponding to a randomly chosen bit β. 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 (the bulletin board 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 $\mathfrak{b}$, 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 $\mathit{correct\text{-}price}(pk, np, \mathfrak{bb}, p, \kappa)$,

[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* $\mathsf{Succ}(\mathsf{Exp}(\cdot))$ in a game $\mathsf{Exp}(\cdot)$ as the probability that the adversary wins, that is, $\mathsf{Succ}(\mathsf{Exp}(\cdot)) = \Pr[\mathsf{Exp}(\cdot) = 1]$. Adversaries are assumed to be *stateful*, that is, information persists across invocations of the adversary in a single game, in particular, the adversary can access earlier assignments.

which holds when: $(p = 0 \vee (\exists r \ . \ \mathsf{Bid}(pk, np, p, \kappa; r) \in \mathfrak{bb} \setminus \{\perp\} \wedge 1 \leq p \leq np)) \wedge (\neg \exists p', r' \ . \ \mathsf{Bid}(pk, np, p', \kappa; r') \in \mathfrak{bb} \setminus \{\perp\} \wedge p < p' \leq np)$. 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.[6]

By design, our notion of bid secrecy is satisfiable by auction schemes which reveal losing prices, assuming that these prices cannot be linked to bidders. 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 formalized using predicate $\mathit{balanced}(\mathfrak{bb}, np, L)$, which holds when: for all prices $p \in \{1, \ldots, np\}$ we have $|\{b \mid b \in \mathfrak{bb} \wedge (b, p, p_1) \in L\}| = |\{b \mid b \in \mathfrak{bb} \wedge (b, p_0, p) \in L\}|$, where L is the set 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. On the other hand, if the adversary wins the game, then there exists a strategy to distinguish honestly cast bids.

Definition 5 (Bid secrecy)**.** *Let* $\Sigma = (\mathsf{Setup}, \mathsf{Bid}, \mathsf{Open}, \mathsf{Verify})$ *be an auction scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{Bid\text{-}Secrecy}(\Sigma, \mathcal{A}, \kappa)$ *be the following game.*[7]

$\mathsf{Bid\text{-}Secrecy}(\Sigma, \mathcal{A}, \kappa) =$

$(pk, sk, mb, mp) \leftarrow \mathsf{Setup}(\kappa)$;
$\beta \leftarrow_R \{0, 1\}$; $L \leftarrow \emptyset$;
$np \leftarrow \mathcal{A}(pk, \kappa)$; $\mathfrak{bb} \leftarrow \mathcal{A}^{\mathcal{O}}()$;
$(p, \mathfrak{b}, pf) \leftarrow \mathsf{Open}(sk, np, \mathfrak{bb}, \kappa)$;
$g \leftarrow \mathcal{A}(p, \mathfrak{b}, pf)$;
if $g = \beta \wedge \mathit{balanced}(\mathfrak{bb}, np, L) \wedge |\mathfrak{bb}| \leq mb \wedge np \leq mp$
$\wedge (\mathit{correct\text{-}price}(pk, np, \mathfrak{bb}, p, \kappa) \Rightarrow \forall b \in \mathfrak{bb} \ . \ (b, p, p_1) \notin L \wedge (b, p_0, p) \notin L)$
then
  return 1
else
  return 0

Oracle $\mathcal{O}$ *is defined as follows:*[8]

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

[6] The existential quantifiers in *correct-price* demonstrate the importance of defining injectivity *perfectly* rather than *computationally*. In particular, *correct-price* cannot interpret a bid for more than one price.

[7] We write $x \leftarrow_R S$ for the assignment to x of an element chosen uniformly at random from set S.

[8] The oracle may access game parameters, e.g., pk. Henceforth, we allow oracles to access game parameters without an explicit mention.

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

Our definition of bid secrecy is based upon the notion of ballot secrecy proposed by Smyth [Smy15a] (cf. Appendix A) and, roughly speaking, corresponds to a symbolic bid secrecy definition proposed by Dreier, Lafourcade & Lakhnech [DLL13, Definition 15].[9]

2.2.1 Example: Enc2Bid

We demonstrate the applicability of our definition with a construction (Enc2Bid) for auction schemes from asymmetric encryption schemes.[10]

Definition 6 (Enc2Bid)**.** *Given an asymmetric encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, we define* $\mathsf{Enc2Bid}(\Pi)$ *as follows.*

- $\mathsf{Setup}(\kappa)$ *computes* $(pk, sk) \leftarrow \mathsf{Gen}(\kappa)$ *and outputs* $(pk, sk, poly(\kappa), |\mathfrak{m}|)$*.*
- $\mathsf{Bid}(pk, np, p, \kappa)$ *computes* $b \leftarrow \mathsf{Enc}(pk, p)$ *and outputs* b*, if* $1 \leq p \leq np \leq |\mathfrak{m}|$*, and outputs* $\perp$*, otherwise.*
- $\mathsf{Open}(sk, np, \mathfrak{bb}, \kappa)$ *proceeds as follows. Computes* $\mathfrak{d} \leftarrow \{(b, \mathsf{Dec}(sk, b)) \mid b \in \mathfrak{bb}\}$*. Finds the largest integer* p *such that* $(b, p) \in \mathfrak{d} \wedge 1 \leq p \leq np$*, outputting* $(0, \emptyset, \epsilon)$ *if no such integer exists. Computes* $\mathfrak{b} \leftarrow \{b \mid (b, p') \in \mathfrak{d} \wedge p' = p\}$*. Outputs* $(p, \mathfrak{b}, \epsilon)$*.*
- $\mathsf{Verify}(pk, np, \mathfrak{bb}, p, \mathfrak{b}, pf, \kappa)$ *outputs 1.*

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

Lemma 1. *Suppose* Π *is an asymmetric encryption scheme with perfect correctness. We have* $\mathsf{Enc2Bid}(\Pi)$ *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.

Intuitively, given a non-malleable asymmetric encryption scheme Π, auction scheme $\mathsf{Enc2Bid}(\Pi)$ derives bid secrecy from the encryption scheme until opening and opening maintains bid secrecy by only disclosing winning bids and the winning price. We defer a formal proof of bid secrecy until Section 4.2.1, where we can use our election to auction scheme construction and accompanying security results.

[9] We discuss our motivation to base the definition of bid secrecy on the notion of ballot secrecy by Smyth in Section 4.1.

[10] We present definitions of cryptographic primitives and relevant security definitions in Appendix B.

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 [SFC15] (cf. Section 5.1).[11]

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

$\mathsf{Exp\text{-}IV}(\Sigma, \mathcal{A}, \kappa) =$

$(pk, np, p, p') \leftarrow \mathcal{A}(\kappa)$;
$b \leftarrow \mathsf{Bid}(pk, np, p, \kappa)$;
$b' \leftarrow \mathsf{Bid}(pk, np, p', \kappa)$;
if $b = b' \wedge b \neq \perp \wedge b' \neq \perp$ **then**
  return 1
else
  return 0

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

Individual verifiability resembles injectivity, but game 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 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. Formally, we check the validity of the winning price using predicate *correct-price*. And we check the validity of the set of winning bids using predicate *correct-bids*$(pk, np, \mathfrak{bb}, p, \mathfrak{b}, \kappa)$, which holds when $\mathfrak{b} = \mathfrak{bb} \cap \{b \mid b = \mathsf{Bid}(pk, np, p, \kappa; r)\}$, i.e., it holds when $\mathfrak{b}$ is the intersection of the bulletin board and the set of all bids for the winning price.

Since function *correct-price* will now be parameterised with a public key constructed by the adversary, rather than a public key constructed by algorithm Setup (cf. Section 2.2), we must strengthen injectivity to hold for adversarial keys.

[11] We discuss our motivation to base the definitions on the notions of verifiability by Smyth, Frink & Clarkson in Section 5.

Definition 8 (Strong injectivity)**.** *An auction scheme* $(\mathsf{Setup}, \mathsf{Bid}, \mathsf{Open}, \mathsf{Verify})$ *satisfies* strong injectivity*, if for all security parameters* κ*, public keys* pk*, integers* np*, and prices* p *and* p'*, such that* $p \neq p'$*, we have*

$$\Pr[b \leftarrow \mathsf{Bid}(pk, np, p, \kappa); b' \leftarrow \mathsf{Bid}(pk, np, p', \kappa) : b \neq \bot \ \wedge \ b' \neq \bot \Rightarrow b \neq b'] = 1.$$

Definition 9 (Universal verifiability)**.** *Let* $\Sigma = (\mathsf{Setup}, \mathsf{Bid}, \mathsf{Open}, \mathsf{Verify})$ *be an auction scheme satisfying strong injectivity,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{Exp\text{-}UV}(\Sigma, \mathcal{A}, \kappa)$ *be the following game.*

$\mathsf{Exp\text{-}UV}(\Sigma, \mathcal{A}, \kappa) =$

$(pk, np, \mathfrak{bb}, p, \mathfrak{b}, pf) \leftarrow \mathcal{A}(\kappa);$
if $(\neg \mathit{correct\text{-}price}(pk, np, \mathfrak{bb}, p, \kappa) \vee \neg \mathit{correct\text{-}bids}(pk, np, \mathfrak{bb}, p, \mathfrak{b}, \kappa)) \wedge$
$\mathsf{Verify}(pk, np, \mathfrak{bb}, p, \mathfrak{b}, pf, \kappa) = 1$ **then**
  return 1
else
  return 0

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

3 Auctions from elections

3.1 Election scheme syntax

We recall syntax for *election schemes* from Smyth, Frink & Clarkson [SFC15].

Definition 10 (Election scheme [SFC15])**.** *An* election scheme *is a tuple of efficient algorithms* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *such that:*

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

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

Tally*, denoted* $(\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk, nc, \mathfrak{bb}, \kappa)$*, is run by the tallier.* Tally *takes as input a private key* sk*, some number of candidates* nc*, a bulletin board* $\mathfrak{bb}$*, and a security parameter* κ*, where* $\mathfrak{bb}$ *is a set. It outputs an election outcome* $\mathbf{v}$ *and a non-interactive proof* pf *that the outcome is correct. An election outcome is a vector* $\mathbf{v}$ *of length* nc *such that* $\mathbf{v}[v]$ *indicates*[12] *the number of votes for candidate* v*.*

[12] Let $\mathbf{v}[v]$ denote component v of vector $\mathbf{v}$.

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

Election schemes must satisfy correctness, completeness, *and* injectivity, *which are defined below.*

Definition 11 (Correctness [SFC15]). *There exists a negligible function* negl, *such that for all security parameters* κ, *integers* nb *and* nc, *and votes* $v_1, \ldots, v_{nb} \in \{1, \ldots, nc\}$, *it holds that if* $\mathbf{v}$ *is a vector of length* nc *whose components are all* 0, *then*

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

Definition 12 (Completeness [SFC15]). *There exists a negligible function* negl, *such that for all security parameters* κ, *bulletin boards* $\mathfrak{bb}$, *and integers* nc, *we have*

$$
\begin{aligned}
&\Pr[(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa);\\
&\quad (\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk, nc, \mathfrak{bb}, \kappa);\\
&\quad : |\mathfrak{bb}| \le mb \wedge nc \le mc \Rightarrow \mathsf{Verify}(pk, nc, \mathfrak{bb}, \mathbf{v}, pf, \kappa) = 1] > 1 - \mathsf{negl}(\kappa).
\end{aligned}
$$

Definition 13 (Injectivity). *For all security parameters* κ, *integers* nc, *and votes* v *and* v', *such that* $v \neq v'$, *we have*

$$
\begin{aligned}
\Pr[(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa); b \leftarrow \mathsf{Vote}(pk, nc, v, \kappa);&\\
b' \leftarrow \mathsf{Vote}(pk, nc, v', \kappa) : b \neq \bot \wedge b' \neq \bot \Rightarrow b \neq b'] = 1.&
\end{aligned}
$$

Injectivity for election schemes (Definition 13) is analogous to injectivity for auction schemes (Definition 4) and is slightly weaker than the original definition (cf. Definition 23).

Comparing auction and election schemes. Auction schemes are distinguished from election schemes in the final step of their execution: 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 do so by introducing *reveal algorithms.*

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 14 (Reveal algorithm)**.** *A* reveal algorithm *is an efficient algorithm* Reveal *defined as follows:*

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

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

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

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, such that the reveal algorithm 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 (Section 3.3.1) which can produce auction schemes satisfying bid secrecy. Building upon this result, we present our second construction (Section 3.3.2) which can produce auction schemes satisfying bid secrecy *and* auction verifiability.

3.3.1 Non-verifiable auction schemes

Our first construction follows intuitively from our informal description (Section 1.1). 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 15. *Given an election scheme* $\Gamma = (\mathsf{Setup}_\Gamma, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify}_\Gamma)$ *and a reveal algorithm* Reveal*, we define* $\Lambda(\Gamma, \mathsf{Reveal}) = (\mathsf{Setup}_\Lambda, \mathsf{Bid}, \mathsf{Open}, \mathsf{Verify}_\Lambda)$ *as follows.*

$\mathsf{Setup}_\Lambda(\kappa)$ *computes* $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}_\Gamma(\kappa)$ *and outputs* (pk, sk, mb, mc).

$\mathsf{Bid}(pk, np, p, \kappa)$ *computes* $b \leftarrow \mathsf{Vote}(pk, np, p, \kappa)$ *and outputs* b.

$\mathsf{Open}(sk, np, \mathfrak{bb}, \kappa)$ *proceeds as follows. Computes* $(\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk, np, \mathfrak{bb})$. *Finds the largest integer* p *such that* $\mathbf{v}[p] > 0 \ \wedge \ 1 \leq p \leq np$*, outputting* $(0, \emptyset, \epsilon)$ *if no such integer exists. Computes* $\mathfrak{b} \leftarrow \mathsf{Reveal}(sk, np, \mathfrak{bb}, p, \kappa)$. *And outputs* $(p, \mathfrak{b}, \epsilon)$.

$\mathsf{Verify}_\Lambda(pk, np, \mathfrak{bb}, p, \mathfrak{b}, pf', \kappa)$ *outputs 1.*

Algorithm Open *requires* ϵ *to be a constant symbol.*

Lemma 2. *Let* Γ *be an election scheme and* Reveal *be a reveal algorithm. Suppose* Reveal *is correct with respect to* Γ. *We have* $\Lambda(\Gamma, \mathsf{Reveal})$ *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 16. *Given an election scheme* $\Gamma = (\mathsf{Setup}_\Gamma, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify}_\Gamma)$*, a reveal algorithm* Reveal*, and a non-interactive proof system* $\Delta = (\mathsf{Prove}, \mathsf{Verify})$*, we define* $\Lambda(\Gamma, \mathsf{Reveal}, \Delta) = (\mathsf{Setup}_\Lambda, \mathsf{Bid}, \mathsf{Open}, \mathsf{Verify}_\Lambda)$ *as follows.*

$\mathsf{Setup}_\Lambda(\kappa)$ *computes* $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}_\Gamma(\kappa)$ *and outputs* (pk, sk, mb, mc).

$\mathsf{Bid}(pk, np, p, \kappa)$ *computes* $b \leftarrow \mathsf{Vote}(pk, np, p, \kappa)$ *and outputs* b.

$\mathsf{Open}(sk, np, \mathfrak{bb}, \kappa)$ *proceeds as follows. Computes* $(\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk, np, \mathfrak{bb})$. *Finds the largest integer* p *such that* $\mathbf{v}[p] > 0 \ \wedge \ 1 \leq p \leq np$*, outputting* $(0, \emptyset, \epsilon)$ *if no such integer exists. Computes* $\mathfrak{b} \leftarrow \mathsf{Reveal}(sk, np, \mathfrak{bb}, p, \kappa)$ *and* $pf' \leftarrow \mathsf{Prove}((pk, np, \mathfrak{bb}, p, \mathfrak{b}, \kappa), sk)$*, and outputs* $(p, \mathfrak{b}, (\mathbf{v}, pf, pf'))$.

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

1. *Checks that* $\mathsf{Verify}_\Gamma(pk, np, \mathfrak{bb}, \mathbf{v}, pf, \kappa) = 1$.
2. *Checks that* p *is the largest integer such that* $\mathbf{v}[p] > 0 \ \wedge \ 1 \leq p \leq np$ *or there is no such integer and* $(p, \mathfrak{b}, pf') = (0, \emptyset, \epsilon)$.
3. *Checks that* $\mathsf{Verify}((pk, np, \mathfrak{bb}, p, \mathfrak{b}, \kappa), pf', \kappa) = 1$.

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

Algorithms Tally *and* Verify *require* ϵ *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 such a relation as follows.

Definition 17. *Given an election scheme* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *and a reveal algorithm* Reveal*, we define binary relation* $R(\Gamma, \mathsf{Reveal})$ *over vectors of length* 6 *and bitstrings such that* $((pk, nc, \mathfrak{bb}, v, \mathfrak{b}, \kappa),\ sk) \in R(\Gamma, \mathsf{Reveal}) \Leftrightarrow \exists mb, mc, r, r' \,.\, \mathfrak{b} = \mathsf{Reveal}(sk, nc, \mathfrak{bb}, v, \kappa; r) \wedge (pk, sk, mb, mc) = \mathsf{Setup}(\kappa; r') \wedge 1 \leq v \leq nc \leq mc$.

Lemma 3. *Let* Γ *be an election scheme,* Reveal *be a reveal algorithm, and* Δ *be a non-interactive proof system for relation* $R(\Gamma, \mathsf{Reveal})$*. Suppose* Reveal *is correct with respect to* Γ*. We have* $\Lambda(\Gamma, \mathsf{Reveal}, \Delta)$ *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).

4.1 Ballot secrecy

Our definition of ballot secrecy strengthens an earlier definition by Smyth [Smy15a].[13]

Definition 18 (Ballot secrecy)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{Ballot\text{-}Secrecy}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

$\mathsf{Ballot\text{-}Secrecy}(\Gamma, \mathcal{A}, \kappa) =$

[13] We adopt the definition of ballot secrecy by Smyth because it strengthens earlier definitions by Bernhard *et al.* [BCP+11, BPW12b, SB13, SB14, BCG+15b] to detect attacks that arise when the adversary controls the bulletin board and the communication channel. Our privacy results could be extended to other definitions of bid secrecy and ballot secrecy, by modifying our proofs.

$(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
$\beta \leftarrow_R \{0, 1\}; L \leftarrow \emptyset; W \leftarrow \emptyset$;
$nc \leftarrow \mathcal{A}(pk, \kappa); \mathfrak{bb} \leftarrow \mathcal{A}^{\mathcal{O}}()$;
$(\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk, nc, \mathfrak{bb}, \kappa)$;
for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \notin L$ **do**
  $(\mathbf{v}', pf') \leftarrow \mathsf{Tally}(sk, nc, \{b\}, \kappa)$;
  $W \leftarrow W \cup \{(b, \mathbf{v}')\}$;
$g \leftarrow \mathcal{A}(\mathbf{v}, pf, W)$;
if $g = \beta \wedge balanced(\mathfrak{bb}, nc, L) \wedge |\mathfrak{bb}| \leq mb \wedge nc \leq mc$ **then**
  return 1
else
  return 0

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

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

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

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 adversary's 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 W 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. On the other hand, if the adversary does succeed, then ballots can be distinguished and ballot secrecy is not preserved.

Comparing notions of ballot secrecy. Our definition of ballot secrecy (Ballot-Secrecy) strengthens an earlier definition (Smy-Ballot-Secrecy) by Smyth [Smy15a] (recalled in Appendix A). In particular, in Ballot-Secrecy the adversary is given the vote corresponding to *any* ballot that was *not* computed by the oracle, whereas in Smy-Ballot-Secrecy the adversary does not have this capability. 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). In Appendix A, we show that Ballot-Secrecy is strictly stronger using a scheme that satisfies Smy-Ballot-Secrecy but not Ballot-Secrecy, hence separating the two notions.

4.1.1 Example: Enc2Vote satisfies ballot secrecy

We demonstrate the applicability of our definition using a construction (Enc2Vote) for election schemes from non-malleable public-key encryption schemes.[14]

Definition 19 (Enc2Vote). *Given an asymmetric encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, we define* $\mathsf{Enc2Vote}(\Pi)$ *as follows.*

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

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

Lemma 4. *Suppose* Π *is an asymmetric encryption scheme with perfect correctness. We have* $\mathsf{Enc2Vote}(\Pi)$ *is an election scheme.*

Intuitively, given an encryption scheme Π satisfying non-malleability, the election scheme $\mathsf{Enc2Vote}(\Pi)$ 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.[15]

Proposition 5. *Suppose* Π *is an asymmetric encryption scheme with perfect correctness. If* Π *satisfies IND-PA0, then* $\mathsf{Enc2Vote}(\Pi)$ *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.

[14] The construction was originally presented by Bernhard *et al.* [SB14, SB13, BPW12b, BCP+11] in a slightly different setting.

[15] Bellare & Sahai [BS99, §5] show that their notion of non-malleability (CNM-CPA) coincides with a simpler indistinguishability notion (IND-PA0), thus it suffices to consider IND-PA0 in Proposition 5.

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 producing the expected outcome, and the underlying reveal algorithm producing the expected set of ballots. Otherwise, a poorly designed tally algorithm could lead to the construction of auction schemes which do not satisfy bid secrecy, and similarly for a poorly designed reveal algorithm. This leads to a separation result (cf. Appendix E). Nevertheless, we can formulate soundness conditions which capture a class of election schemes for which our intuition holds.

Tally soundness. Correctness for election schemes ensures that algorithm Tally produces the expected election outcome under ideal conditions. A similar property, which we call *tally soundness*, can hold in the presence of an adversary. Our formulation of tally soundness (Definition 20) 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.

Formally, we capture the correct election outcome using function *correct-outcome*, which is defined such that for all pk, nc, $\mathfrak{bb}$, κ, ℓ, and $v \in \{1, \ldots, nc\}$, we have[16]

$$\begin{aligned}&\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)[v] = \ell\\ &\qquad\qquad\iff \exists^{=\ell} b \in \mathfrak{bb} \setminus \{\bot\} : \exists r : b = \mathsf{Vote}(pk, nc, v, \kappa; r)\end{aligned}$$

That is, component v of vector $\mathit{correct\text{-}outcome}(pk, \mathfrak{bb}, n_C, k)$ equals ℓ iff there exist ℓ ballots on the bulletin board that are votes for candidate v. The vector produced by *correct-outcome* must be of length n_C.

Definition 20 (Tally soundness)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{Tally\text{-}Soundness}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

$\mathsf{Tally\text{-}Soundness}(\Gamma, \mathcal{A}, \kappa) =$

$(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
$(nc, \mathfrak{bb}) \leftarrow \mathcal{A}(pk, \kappa)$;
$(\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk, nc, \mathfrak{bb}, \kappa)$;
if $\exists v \in \{1, \ldots, nc\} \,.\, \mathbf{v}[v] < \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)[v] \wedge |\mathfrak{bb}| \leq mb \wedge nc \leq mc$ **then**
  return 1
else
  return 0

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

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

Reveal soundness. Correctness for reveal algorithms ensures that algorithm 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 concoct a scenario in which the set of ballots for a particular vote is not correct, i.e., the set does not contain all the ballots for the specified vote.

Definition 21 (Reveal soundness)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme,* Reveal *be a reveal algorithm,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{Reveal\text{-}Soundness}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

$\mathsf{Reveal\text{-}Soundness}(\Gamma, \mathcal{A}, \kappa) =$

$(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
$(nc, \mathfrak{bb}, v) \leftarrow \mathcal{A}(pk, \kappa)$;
$\mathfrak{b} \leftarrow \mathsf{Reveal}(sk, nc, \mathfrak{bb}, v, \kappa)$;
$W \leftarrow \emptyset$;
for $b \in \mathfrak{bb}$ **do**
  $(\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk, nc, \{b\}, \kappa)$;
  $W \leftarrow W \cup \{(b, \mathbf{v})\}$;
if $\mathfrak{b} \neq \{b \mid (b, \mathbf{v}) \in W \wedge \mathbf{v}[v] = 1\} \wedge |\mathfrak{bb}| \leq mb \wedge 1 \leq v \leq nc \leq mc$ **then**
  return 1
else
  return 0

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

Lemma 6. *Let* Γ *be an election scheme and* Reveal *be a reveal algorithm. If* Reveal *satisfies reveal soundness with respect to* Γ*, then* Reveal *is correct with respect to* Γ*.*

4.2.1 Bid secrecy for non-verifiable auction schemes

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

Proposition 7. *Let* Γ *be an election scheme and* Reveal *be a reveal algorithm. Moreover, let* $\Sigma = \Lambda(\Gamma, \mathsf{Reveal})$*. If* Γ *satisfies ballot secrecy and tally soundness, and* Reveal *satisfies reveal soundness with respect to* Γ*, then* Σ *satisfies bid secrecy.*

We demonstrate the applicability of our result in the following example.

Example: $\mathsf{Enc2Bid}$ **satisfies bid secrecy**

In Appendix C we present a reveal algorithm $\mathsf{Reveal\text{-}Enc2Bid}(\Pi)$ such that $\mathsf{Enc2Bid}(\Pi)$ is equivalent to $\Lambda(\mathsf{Enc2Vote}(\Pi), \mathsf{Reveal\text{-}Enc2Bid}(\Pi))$. Hence, we can use Proposition 7 to prove that $\mathsf{Enc2Bid}(\Pi)$ satisfies bid secrecy, obtaining the following result.

Proposition 8. *Suppose* Π *is an asymmetric encryption scheme with perfect correctness. If* Π *satisfies IND-PA0, then* $\mathsf{Enc2Bid}(\Pi)$ *satisfies bid secrecy.*

4.2.2 Bid secrecy for verifiable auction schemes

We generalise Proposition 7 to verifiable auction schemes, assuming the non-interactive proof system is zero-knowledge.

Theorem 9. *Let* Γ *be an election scheme,* Reveal *be a reveal algorithm, and* Δ *be a non-interactive proof system for relation* $R(\Gamma, \mathsf{Reveal})$*. Moreover, let* $\Sigma = \Lambda(\Gamma, \mathsf{Reveal}, \Delta)$*. If* Γ *satisfies ballot secrecy and tally soundness,* Reveal *satisfies reveal soundness with respect to* Γ*, and* Δ *is zero-knowledge, then* Σ *satisfies bid secrecy.*

We shall see that tally soundness is implied by universal verifiability (Section 5.1.2), hence, a special case of the above theorem requires that Γ satisfies universal verifiability, rather than tally soundness.

5 Verifiability results

We recall definitions of election verifiability by Smyth, Frink & Clarkson [SFC15].[17] We show that these definitions are sufficient to ensure that our construction produces schemes satisfying auction verifiability.

[17] Küsters et al. [KTV11] argue that decomposing verifiability into individual and universal verifiability is insufficient to detect certain attacks involving ill-formed ballots. Cortier et al. [CEK+15, §1] and Smyth, Frink & Clarkson [SFC15] have shown that this argument does not hold in general: they present definitions of universal verifiability that rule out such attacks. Nevertheless, Smyth, Frink & Clarkson acknowledge that "there [might] still lurk ... 'gaps' in [their] decomposition." But, we must concede that gaps might also lurk in alternative definitions of verifiability. In particular, those definitions that do not require decomposition, such as global verifiability. Indeed, Cortier et al. [CGK+16, §1] have observed "severe limitations and weaknesses" in some definitions of global verifiability.

We adopt verifiability definitions by Smyth, Frink & Clarkson [SFC15] because they improve upon earlier definitions, e.g., [JCJ10, CGGI14, KZZ15], to detect attacks that arise when tallying and verification procedures collude, when verification procedures reject legitimate outcomes, and when the adversary controls the bulletin board and the communication channel. Moreover, Smyth, Frink & Clarkson have shown verifiability results for Helios, which will be useful in our case study. Our verifiability results could be extended to other definitions of verifiability, e.g., [KTV10, CGK+16], by modifying our proofs.

5.1 Election verifiability

5.1.1 Individual verifiability

Individual verifiability challenges the adversary to generate a collision from algorithm Vote.

Definition 22 (Individual verifiability [SFC15]). *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{Exp\text{-}IV\text{-}Ext}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

$\mathsf{Exp\text{-}IV\text{-}Ext}(\Gamma, \mathcal{A}, \kappa) =$

$(pk, nc, v, v') \leftarrow \mathcal{A}(\kappa)$;
$b \leftarrow \mathsf{Vote}(pk, nc, v, \kappa)$;
$b' \leftarrow \mathsf{Vote}(pk, nc, v', \kappa)$;
if $b = b' \wedge b \neq \perp \wedge b' \neq \perp$ **then**
  return 1
else
  return 0

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

5.1.2 Universal verifiability

Universal verifiability challenges the adversary to concoct a scenario in which Verify accepts, but the election outcome is not correct.

Formally, we capture the correct election outcome using function *correct-outcome*. Since function *correct-outcome* will now be parameterised with a public key constructed by the adversary, rather than a public key constructed by algorithm Setup (cf. Section 4.2), we must strengthen injectivity to hold for adversarial keys.

Definition 23 (Strong injectivity [SFC15]). *An election scheme* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *satisfies* strong injectivity*, if for all security parameters* κ*, public keys* pk*, integers* nc*, and votes* v *and* v'*, such that* $v \neq v'$*, we have*

$$\Pr[b \leftarrow \mathsf{Vote}(pk, nc, v, \kappa); b' \leftarrow \mathsf{Vote}(pk, nc, v', \kappa) : b \neq \perp \wedge b' \neq \perp \Rightarrow b \neq b'] = 1.$$

Definition 24 (Universal verifiability [SFC15]). *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme satisfying strong injectivity,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{Exp\text{-}UV\text{-}Ext}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

$\mathsf{Exp\text{-}UV\text{-}Ext}(\Gamma, \mathcal{A}, \kappa) =$

$(pk, nc, \mathfrak{bb}, \mathbf{v}, pf) \leftarrow \mathcal{A}(\kappa)$;
if $\mathbf{v} \neq \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa) \wedge \mathsf{Verify}(pk, nc, \mathfrak{bb}, \mathbf{v}, pf, \kappa) = 1$
then
  return 1
else
  return 0

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

Universal verifiability is similar to 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 Tally in Tally-Soundness. By comparison, the outcome is chosen by the adversary in Exp-UV-Ext, under the condition that it must be accepted by algorithm Verify. Since completeness asserts that outcomes output by Tally will be accepted by Verify, we have the following result.

Lemma 10. *Let* Γ *be an election scheme. If* Γ *satisfies universal verifiability, then* Γ *satisfies tally soundness.*

It is trivial to see that universal verifiability is strictly stronger than tally soundness, because Enc2Vote satisfies tally soundness (see proof of Proposition 8), but not universal verifiability (it accepts any election outcome).

Corollary 11. *Universal verifiability is strictly stronger than tally soundness.*

The proof of Corollary 11 follows from Lemma 10 and the above reasoning; we omit a formal proof.

5.2 Election verifiability implies auction verifiability

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

Theorem 12. *Let* Γ *be an election scheme,* Reveal *be a reveal algorithm, and* Δ *be a non-interactive proof system for relation* $R(\Gamma, \mathsf{Reveal})$*, such that* Reveal *is correct with respect to* Γ*. If* Γ *satisfies individual verifiability, then* $\Lambda(\Gamma, \mathsf{Reveal}, \Delta)$ *satisfies individual verifiability.*

The proof of Theorem 12 follows from Definitions 7, 16 & 22 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 $R(\Gamma, \mathsf{Reveal})$.

Definition 25. *Given an election scheme* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *and a reveal algorithm* Reveal*, we say relation* $R(\Gamma, \mathsf{Reveal})$ *is* Λ-suitable*, if* $((pk, np, \mathfrak{bb}, p, \mathfrak{b}, \kappa), sk) \in R(\Gamma, \mathsf{Reveal})$ *implies* $\mathit{correct\text{-}bids}(pk, np, \mathfrak{bb}, p, \mathfrak{b}, \kappa)$ *with overwhelming probability.*

Theorem 13. *Let* Γ *be an election scheme,* Reveal *be a reveal algorithm, and* Δ *be a non-interactive proof system for relation* $R(\Gamma, \mathsf{Reveal})$*, such that* Reveal *is correct with respect to* Γ*. If* Γ *satisfies universal verifiability,* Δ *satisfies soundness, and* $R(\Gamma, \mathsf{Reveal})$ *is* Λ*-suitable, then* $\Lambda(\Gamma, \mathsf{Reveal}, \Delta)$ *satisfies universal verifiability.*

6 Case study: Helios

We demonstrate the applicability of our construction by deriving an auction scheme from Helios [AMPQ09].

6.1 Helios

Helios is an open-source, web-based electronic voting system, which has been deployed in the real-world: the International Association of Cryptologic Research has used Helios annually since 2010 to elect board members [BVQ10, HBH10],[18] 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.[19,20]

Informally, Helios can be modelled as an election scheme (Setup, Vote, Tally, Verify) such that:

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

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

Tally proceeds as follows. First, any ballots on the bulletin board for which proofs do not hold are discarded. Secondly, the ciphertexts in the remaining ballots are homomorphically combined, 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 Helios 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, Smy15b, BPW12a]. We adopt the formal definition of Helios proposed by Smyth, Frink & Clarkson

[18] `http://www.iacr.org/elections/`, accessed 3 Apr 2013.

[19] `http://heliosvoting.org/2009/10/13/helios-deployed-at-princeton/`, accessed 8 Feb 2013.

[20] `https://princeton.heliosvoting.org/`, accessed 8 Feb 2013.

[SFC15], 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. Henceforth, we write *Helios'16* to refer to that formalization.

6.2 An auction scheme from Helios'16

We derive an auction scheme from Helios'16 using our construction parameterised with a reveal algorithm and a non-interactive proof system. We formally describe that reveal algorithm and proof system in Appendix F, and refer to the resulting scheme as *the auction scheme from Helios'16*. Our privacy and verifiability results allow us to prove security of that scheme:

Theorem 14. *If Helios'16 satisfies ballot secrecy, then the auction scheme from Helios'16 satisfies bid secrecy.*

Proof. Smyth, Frink & Clarkson have shown that Helios'16 satisfies universal verifiability [SFC15]. It follows from Lemma 10 that Helios'16 satisfies tally soundness. Hence, by Theorem 9, it suffices to prove that the reveal algorithm satisfies reveal soundness and that the non-interactive proof system is zero-knowledge. We defer those proofs to Lemmata 30 & 31 in Appendix F.3. □

Proving that Helios'16 satisfies ballot secrecy would advance the state-of-the-art in a manner that is beyond the scope of this case study. Indeed, the only privacy results [BPW12a, Ber14, BCG$^+$15a] for Helios consider variants of Helios'16 and depend upon undesirable trust assumptions [Smy15a].

Theorem 15. *The auction scheme from Helios'16 satisfies individual and universal verifiability.*

Proof. Smyth, Frink & Clarkson have shown that Helios'16 satisfies individual and universal verifiability [SFC15]. Hence, by Theorem 12, the auction scheme from Helios'16 satisfies individual verifiability. To show universal verifiability, it suffices (Theorem 13) to prove that the non-interactive proof system satisfies soundness and the associated relation is Λ-suitable. We defer those proofs to Lemmata 33 & 32 in Appendix F.4. □

Deriving auction schemes from Helios is not new. Indeed, McCarthy, Smyth & Quaglia [MSQ14a] derive the *Hawk* auction scheme from Helios. Our auction scheme is distinguished from Hawk by formal security results, whereas Hawk only has an informal security analysis [MSQ14b, §4.4].

7 Conclusion

We demonstrate that the seemingly disjoint research fields of auctions and elections are actually related. In particular, 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. Our results inaugurate the unification of auctions and elections, thereby facilitating the advancement of both fields. In particular, secure auction schemes can be immediately constructed from election schemes, allowing advances in election schemes to be capitalised upon to advance auction schemes.

Acknowledgements

This research was largely conducted 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 Ballot secrecy

We recall the definition of ballot secrecy (Definition 26) by Smyth [Smy15a] (which is based upon an unpublished draft by Smyth [Smy14] and an extended version of that draft by Bernhard & Smyth [BS15]) and introduce a construction for election schemes (Definition 27) which demonstrates that our notion of ballot secrecy is strictly stronger than Smyth's notion (Proposition 16).

Definition 26 (Smy-Ballot-Secrecy [Smy15a])**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* Smy-Ballot-Secrecy$(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

Smy-Ballot-Secrecy$(\Gamma, \mathcal{A}, \kappa) =$

$(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
$nc \leftarrow \mathcal{A}(pk, \kappa)$;
$\beta \leftarrow_R \{0, 1\}; L \leftarrow \emptyset$;
$\mathfrak{bb} \leftarrow \mathcal{A}^{\mathcal{O}}()$;
$(\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk, nc, \mathfrak{bb}, \kappa)$;
$g \leftarrow \mathcal{A}(\mathbf{v}, pf)$;
if $g = \beta \wedge balanced(\mathfrak{bb}, nc, L) \wedge 1 \leq nc \leq mc \wedge |\mathfrak{bb}| \leq mb$ **then**
  return 1
else
  return 0

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

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

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

Definition 27. *Suppose* $\Gamma = (\mathsf{Setup}_\Gamma, \mathsf{Vote}_\Gamma, \mathsf{Tally}_\Gamma, \mathsf{Verify}_\Gamma)$ *is an election scheme and* ϵ *is a constant. Let* $\chi(\Gamma, \epsilon) = (\mathsf{Setup}_\chi, \mathsf{Vote}_\chi, \mathsf{Tally}_\chi, \mathsf{Verify}_\chi)$ *be the following election scheme.*

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

$\mathsf{Vote}_\chi(pk, n_C, v, \kappa)$. *Computes* $b \leftarrow \mathsf{Vote}_\Gamma(pk, n_C, v, \kappa)$ *and outputs* b.

$\mathsf{Tally}_\chi(sk', nc, \mathfrak{bb}, \kappa)$. *Parses* sk' *as* (sk, k), *computes* $(\mathbf{v}, pf) \leftarrow \mathsf{Tally}_\Gamma(sk, nc, \mathfrak{bb}, \kappa)$, *and outputs* $(\mathbf{v}, (pf, sk \oplus k))$, *if* $\mathfrak{bb} = \{\epsilon\}$, *and outputs* $(\mathbf{v}, (pf, k))$, *otherwise.*

$\mathsf{Verify}_\chi(pk, nc, \mathfrak{bb}, \mathbf{v}, pf', \kappa)$. *Parses* pf' *as* (pf, h), *computes* $s \leftarrow \mathsf{Verify}_\Gamma(pk, nc, \mathfrak{bb}, \mathbf{v}, pf, \kappa)$, *and outputs* s.

Proposition 16. Ballot-Secrecy *is strictly stronger than* Smy-Ballot-Secrecy.

Proof sketch. Intuitively, given an election scheme Γ satisfying Smy-Ballot-Secrecy and a constant ϵ, we have $\chi(\Gamma, \epsilon)$ satisfies Smy-Ballot-Secrecy, because tallying reveals either $(\mathbf{v}, (pf, sk \oplus k))$ or $(\mathbf{v}, (pf, k))$. By comparison, $\chi(\Gamma, \epsilon)$ does not satisfy Ballot-Secrecy, because of the following attack. The adversary outputs bulletin board $\mathfrak{bb} \cup \{\epsilon\}$ such that $\mathfrak{bb} \neq \emptyset$, 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. □

B Cryptographic primitives

B.1 Asymmetric encryption

Definition 28 (Asymmetric encryption scheme [KL07])**.** *An* asymmetric encryption scheme *is a tuple of efficient algorithms* (Gen, Enc, Dec) *such that:*

- **Gen**, *denoted* $(pk, sk) \leftarrow \mathsf{Gen}(\kappa)$, *takes a security parameter* κ *as input and outputs a key pair* (pk, sk).
- **Enc**, *denoted* $c \leftarrow \mathsf{Enc}(pk, m)$, *takes a public key* pk *and message* m *from the plaintext space*[21] *as input, and outputs a ciphertext* c.
- **Dec**, *denoted* $m \leftarrow \mathsf{Dec}(sk, c)$, *takes a private key* sk, *and ciphertext* c *as input, and outputs a message* m *or error symbol* $\perp$. *We assume* Dec *is deterministic.*

[21] 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 [SFC15] present a definition in which algorithm Setup outputs the plaintext space.

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

Definition 29 (IND-PA0 [BS99])**.** *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an asymmetric encryption scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and IND-PA0(*$\Pi, \mathcal{A}, \kappa$*) be the following game.*[22]

$\mathsf{IND\text{-}PA0}(\Pi, \mathcal{A}, \kappa) =$

$(pk, sk) \leftarrow \mathsf{Gen}(\kappa)$;
$\beta \leftarrow_R \{0, 1\}$;
$(m_0, m_1) \leftarrow \mathcal{A}(pk, \kappa)$;
$y \leftarrow \mathsf{Enc}(pk, m_\beta)$;
$\mathbf{c} \leftarrow \mathcal{A}(y)$;
$\mathbf{p} \leftarrow (\mathsf{Dec}(sk, \mathbf{c}[1]), \ldots, \mathsf{Dec}(sk, \mathbf{c}[|\mathbf{c}|]))$;
$g \leftarrow \mathcal{A}(\mathbf{p})$;
if $g = \beta \wedge y \notin \mathbf{c}$ **then**
  return 1
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* Π *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* negl*, such that for all security parameters* κ*, we have* $\mathsf{Succ}(\mathit{IND\text{-}PA0}(\Pi, \mathcal{A}, \kappa)) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$*.*

Definition 30 (Homomorphic encryption [SFC15])**.** *An asymmetric encryption scheme* $\Gamma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* homomorphic*,*[23] *with respect to ternary operators* $\odot$*,* $\oplus$*, and* $\otimes$*,*[24] *if there exists a negligible function* negl*, such that for all security parameters* κ*, the following conditions are satisfied:*[25]

- *For all messages* m_1 *and* m_2 *from* Γ*'s plaintext space, we have* $\Pr[(pk, sk) \leftarrow \mathsf{Gen}(\kappa); c_1 \leftarrow \mathsf{Enc}(pk, m_1); c_2 \leftarrow \mathsf{Enc}(pk, m_2) : \mathsf{Dec}(sk, c_1 \otimes_{pk} c_2) = \mathsf{Dec}(sk, c_1) \odot_{pk} \mathsf{Dec}(sk, c_2)] > 1 - \mathsf{negl}(\kappa)$*.*
- *For all messages* m_1 *and* m_2 *from* Γ*'s plaintext space, and all coins* r_1 *and* r_2*, we have* $\Pr[(pk, sk) \leftarrow \mathsf{Gen}(\kappa) : \mathsf{Enc}(pk, m_1; r_1) \otimes_{pk} \mathsf{Enc}(pk, m_2; r_2) = \mathsf{Enc}(pk, m_1 \odot_{pk} m_2; r_1 \oplus_{pk} r_2)] > 1 - \mathsf{negl}(k)$*.*

---

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

[23] Our definition of an asymmetric encryption scheme leaves the plaintext space implicit, whereas, Smyth, Frink & Clarkson [SFC15] explicitly define the plaintext space; this change is reflected in our definition of homomorphic encryption.

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

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

We say Γ *is* additively homomorphic*, if for all security parameters* κ *and key pairs* pk, sk*, such that there exists coins* r *and* $(pk, sk) = \mathsf{Gen}(\kappa; r)$*, we have* $\odot_{pk}$ *is the addition operator in the group defined by* Γ*'s plaintext space and* $\odot_{pk}$*.*

B.2 Proof systems

Definition 31 (Non-interactive proof system [SFC15])**.** *A* non-interactive proof system *for a relation* R *is a tuple of algorithms* $(\mathsf{Prove}, \mathsf{Verify})$*, such that:*

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

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

Definition 32 (Soundness)**.** *Suppose* $(\mathsf{Prove}, \mathsf{Verify})$ *is a non-interactive proof system for relation* R*. We say* $(\mathsf{Prove}, \mathsf{Verify})$ *is* sound*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such that for all security parameters* κ*, we have* $\Pr[(s, \sigma) \leftarrow \mathcal{A}(\kappa) : (s, w) \notin R \wedge \mathsf{Verify}(s, \sigma) = 1] \leq \mathsf{negl}(\kappa)$*.*

Definition 33 (Zero knowledge)**.** *Let* $\Delta = (\mathsf{Prove}, \mathsf{Verify})$ *be a non-interactive proof system for a relation* R*, derived by application of the Fiat-Shamir transformation [FS87] to a random oracle* $\mathcal{H}$ *and the sigma protocol. Moreover, let* $\mathcal{S}$ *be an algorithm,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{ZK}(\Delta, \mathcal{A}, \mathcal{H}, \mathcal{S}, \kappa)$ *be the following game.*

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

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

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

And algorithm $\mathcal{S}$ *can patch random oracle* $\mathcal{H}$*.*[26] *We say* Δ *satisfies* zero knowledge*, if there exists a probabilistic polynomial-time algorithm* $\mathcal{S}$*, such that for all probabilistic polynomial-time algorithm adversaries* $\mathcal{A}$*, there exists a negligible*

[26] 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 formalization.

function negl*, and for all security parameters* κ*, we have* $\mathsf{Succ}(\mathit{ZK}(\Delta, \mathcal{A}, \mathcal{H}, \mathcal{S}, \kappa)) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$*. An algorithm* $\mathcal{S}$ *for which zero knowledge holds is called a* simulator *for* $(\mathsf{Prove}, \mathsf{Verify})$*.*

C Proofs

By Definitions 3 & 12, we have the following facts:

Fact 17. *Suppose* $\Sigma = (\mathsf{Setup}, \mathsf{Bid}, \mathsf{Open}, \mathsf{Verify})$ *is an auction scheme. Further suppose for all public keys* pk*, integers* p *and* np*, sets* $\mathfrak{b}$ *and* $\mathfrak{bb}$*, proofs* pf*, and security parameters* κ*, we have* $\mathsf{Verify}(pk, np, \mathfrak{bb}, p, \mathfrak{b}, pf, \kappa) = 1$*. It follows that* Σ *satisfies completeness.*

Fact 18. *Suppose* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *is an election scheme. Further suppose for all public keys* pk*, integers* nc*, sets* $\mathfrak{bb}$*, vectors* $\mathbf{v}$*, proofs* pf*, and security parameters* κ*, we have* $\mathsf{Verify}(pk, nc, \mathfrak{bb}, \mathbf{v}, pf, \kappa) = 1$*. It follows that* Γ *satisfies completeness.*

C.1 Proof of Lemma 1

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

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

Secondly, completeness. Algorithm Verify always outputs 1, hence, the result follows from Fact 17.

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

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

By definition of Setup, we have $(pk, sk) \leftarrow \mathsf{Gen}(\kappa; r)$ and $mp = \{1, \dots, |\mathfrak{m}|\}$, where $\mathfrak{m}$ is the encryption scheme's plaintext space. Moreover, by definition of Bid, we have $b = \mathsf{Enc}(pk, p; s)$ and $b' = \mathsf{Enc}(pk, p'; s')$. Furthermore, since

$b \neq \perp \wedge b' \neq \perp$, we have, by inspection of Bid, that p and p' are from the plaintext space. Since Π is perfectly correct, we have

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

thus deriving a contradiction and concluding our proof. □

C.2 Proof of Lemma 2

Let $\Lambda(\Gamma, \mathsf{Reveal}) = (\mathsf{Setup}_\Lambda, \mathsf{Bid}, \mathsf{Open}, \mathsf{Verify}_\Lambda)$ and $\Gamma = (\mathsf{Setup}_\Gamma, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify}_\Gamma)$. Algorithm Verify_Γ always outputs 1, hence, it follows from Fact 17 that $\Lambda(\Gamma, \mathsf{Reveal})$ satisfies completeness. Moreover, it follows from injectivity of Γ that $\Lambda(\Gamma, \mathsf{Reveal})$ satisfies injectivity. We show that $\Lambda(\Gamma, \mathsf{Reveal})$ satisfies correctness. Suppose κ 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 $\mathsf{Setup}(\kappa)$ such that $nb \leq mb \wedge np \leq mp$ and for each $1 \leq i \leq nb$ we have $\mathsf{Bid}(pk, np, p_i, \kappa)$ outputs b_i. Let $\mathfrak{bb} = \{b_1, \ldots, b_{nb}\}$. Moreover, suppose $\mathsf{Open}(sk, np, \mathfrak{bb}, \kappa)$ outputs $(p, \mathfrak{b}, pf)$ and $\mathsf{Tally}(sk, np, \mathfrak{bb}, \kappa)$ outputs $(\mathbf{v}, pf)$. Since Γ satisfies correctness, we have with overwhelming probability that $\mathbf{v}$ can be equivalently computed by initialising $\mathbf{v}$ as a zero-filled vector of length np and by performing the following computation:

for $1 \leq i \leq nb$ **do**
  $\mathbf{v}[p_i] \leftarrow \mathbf{v}[p_i] + 1$;

By inspection of Open, we have p is the largest integer such that $\mathbf{v}[p] > 0 \wedge 1 \leq p \leq np$, or no such integer exists and $p = 0$. It follows that $p = \mathtt{max}(0, p_1, \ldots, p_{nb})$ in both cases. By further inspection of Open, we have $\mathfrak{b}$ is an output of $\mathsf{Reveal}(sk, np, \mathfrak{bb}, p, \kappa)$ in the former case and $\mathfrak{b} = \emptyset$ in the latter. In the former case we have $\mathfrak{b} = \{b_i \mid p_i = p \wedge 1 \leq i \leq nb\}$ with overwhelming probability, because reveal algorithm Reveal is correct with respect to Γ. And in the latter case we have $0 \notin \{p_1, \ldots, p_{nb}\}$, hence, $\mathfrak{b} = \{b_i \mid p_i = p \wedge 1 \leq i \leq nb\} = \emptyset$. Hence, correctness is satisfied with overwhelming probability. □

C.3 Proof of Lemma 3

The proof that $\Lambda(\Gamma, \mathsf{Reveal}, \Delta)$ satisfies correctness and injectivity is similar to the proof that $\Lambda(\Gamma, \mathsf{Reveal})$ satisfies correctness and injectivity (Appendix C.2), and we omit a formal proof. We prove that $\Lambda(\Gamma, \mathsf{Reveal}, \Delta)$ satisfies completeness.

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

Check (1) succeeds with overwhelming probability, because Γ satisfies completeness. Check (2) succeeds by definition of Open. We prove that Check (3) succeeds with overwhelming probability as follows. If $p \notin \{1, \ldots, np\}$, then the check vacuously holds, otherwise, we proceed as follows. Since Δ satisfies completeness, it suffices to show that $((pk, np, \mathfrak{bb}, p, \mathfrak{b}, \kappa),\ sk) \in R(\Gamma, \mathsf{Reveal})$. By aforementioned assumptions, we have $1 \le p \le np \le mp$, moreover, there exists coins r such that $(pk, sk, mb, mp) = \mathsf{Setup}_\Lambda(\kappa; r)$. Furthermore, by inspection of Open, there exist coins r' such that $\mathfrak{b} = \mathsf{Reveal}(sk, np, \mathfrak{bb}, v, \kappa; r')$. The result $((pk, np, \mathfrak{bb}, p, \mathfrak{b}, \kappa),\ sk) \in R(\Gamma, \mathsf{Reveal})$ follows.

We have that $\mathsf{Verify}_\Lambda(pk, np, \mathfrak{bb}, p, \mathfrak{b}, pf, \kappa)$ outputs 1 with overwhelming probability, hence, $\Lambda(\Gamma, \mathsf{Reveal}, \Delta)$ satisfies completeness. □

C.4 Proof of Lemma 4

Let $\mathsf{Enc2Vote}(\Pi) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ and $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Algorithm Verify always outputs 1, hence, it follows from Fact 18 that $\mathsf{Enc2Vote}(\Pi)$ satisfies completeness. The proof that $\mathsf{Enc2Vote}(\Pi)$ satisfies injectivity is similar to the proof that $\mathsf{Enc2Bid}(\Pi)$ satisfies injectivity (Appendix C.1), and we omit a formal proof. We prove that $\mathsf{Enc2Vote}(\Pi)$ satisfies correctness. Suppose κ is a security parameter, nb and nc are integers, and $v_1, \ldots, v_{nb} \in \{1, \ldots, nc\}$ are votes, and $\mathbf{v}$ is a vector of length nc whose components are all 0. Further suppose (pk, sk, mb, mc) is an output of $\mathsf{Setup}(\kappa)$ such that $nb \le mb \wedge nc \le mc$ and for each $1 \le i \le nb$ we have b_i is an output of $\mathsf{Vote}(pk, nc, v_i, \kappa)$. Moreover, for each $1 \le i \le nb$ compute $\mathbf{v}[v_i] \leftarrow \mathbf{v}[v_i] + 1$. Suppose $(\mathbf{v}', pf)$ is an output of $\mathsf{Tally}(sk, nc, \{b_1, \ldots, b_{nb}\}, \kappa)$. By inspection of algorithm Tally, we have $\mathbf{v}'$ is a vector of length nc computed as follows:

```
for b ∈ {b_1, ..., b_nb do
    v ← Dec(sk, b);
    if 1 ≤ v ≤ nc then
        v[v] ← v[v] + 1;
```

Since pk, sk are output by Gen and since Π is perfectly correct, we have $\mathsf{Dec}(sk, b_i) = v_i$ for all $i \in \{1, \ldots, nb\}$. It follows that $\mathbf{v} = \mathbf{v}'$. Hence, correctness is (perfectly) satisfied. □

C.5 Proof of Proposition 5

Let $\mathsf{BS0}$, respectively $\mathsf{BS1}$, be the game derived from $\mathsf{Ballot\text{-}Secrecy}$ by replacing $\beta \leftarrow_R \{0,1\}$ with $\beta \leftarrow 0$, respectively $\beta \leftarrow 1$. These games are trivially related to $\mathsf{Ballot\text{-}Secrecy}$, namely, $\mathsf{Succ}(\mathsf{Ballot\text{-}Secrecy}(\Gamma, \mathcal{A}, \kappa)) = \frac{1}{2} \cdot \mathsf{Succ}(\mathsf{BS0}(\Gamma, \mathcal{A}, \kappa)) + \frac{1}{2} \cdot \mathsf{Succ}(\mathsf{BS1}(\Gamma, \mathcal{A}, \kappa))$. Moreover, let $\mathsf{BS1{:}0}$ be the game derived from $\mathsf{BS1}$ by replacing $g = \beta$ with $g = 0$. We relate game $\mathsf{BS1}$ to $\mathsf{BS1{:}0}$, and we relate games $\mathsf{BS0}$ and $\mathsf{BS1{:}0}$ to the hybrid games $\mathsf{G}_0, \mathsf{G}_1, \ldots$ introduced in Definition 34. We use these relations to prove Proposition 5.

Lemma 19. *Let Π be an asymmetric encryption scheme and let $\Gamma = \mathsf{Enc2Vote}(\Pi)$. If a probabilistic polynomial-time adversary $\mathcal{A}$ wins game* $\mathsf{Ballot\text{-}Secrecy}$*, then* *for all security parameters* κ *we have* $\mathsf{Succ}(\mathsf{BS1}(\Gamma, \mathcal{A}, \kappa)) = 1 - \mathsf{Succ}(\mathsf{BS1{:}0}(\Gamma, \mathcal{A}, \kappa))$.

Definition 34. *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme,* $\mathcal{A}$ *be a probabilistic polynomial-time adversary,* ϵ *be a constant symbol, and* κ *be a security parameter. We introduce games* $\mathsf{G}_0, \mathsf{G}_1, \ldots$ *defined as follows.*

$\mathsf{G}_i(\Gamma, \mathcal{A}, \epsilon, \kappa) =$

```
(pk, sk, mb, mc) ← Setup(κ);
L ← ∅; W ← ∅;
nc ← A(pk, κ); bb ← A^O();
v ← (0, ..., 0); // vector of length nc
for b ∈ bb ∧ (b, v_0, v_1) ∉ L do
    (v', pf) ← Tally(sk, nc, {b}, κ);
    W ← W ∪ {(b, v')};
    v ← v + v';
for b ∈ bb ∧ (b, v_0, v_1) ∈ L do
    v[v_0] ← v[v_0] + 1;
g ← A(v, ε, W);
if g = 0 ∧ balanced(bb, nc, L) ∧ |bb| ≤ mb ∧ nc ≤ mc then
    return 1
else
    return 0
```

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

```
if |L| < i then
    b ← Vote(pk, nc, v_1, κ);
else
    b ← Vote(pk, nc, v_0, κ);
L ← L ∪ {(b, v_0, v_1)};
return b;
```

Fact 20. *Let* Π *be an asymmetric encryption scheme. Suppose* $\mathsf{Enc2Vote}(\Pi) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$*. There exists a negligible function* negl*, such that for all security parameters* κ*, bulletin boards* $\mathfrak{bb}_0$ *and* $\mathfrak{bb}_1$ *such that* $\mathfrak{bb}_0 \cap \mathfrak{bb}_1 = \emptyset$*, and integers* nc*, we have*

$$\begin{aligned}\Pr[&(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa);\\ &(\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk, nc, \mathfrak{bb}_0 \cup \mathfrak{bb}_1, \kappa);\\ &(\mathbf{v}_0, pf_0) \leftarrow \mathsf{Tally}(sk, nc, \mathfrak{bb}_0, \kappa);\\ &(\mathbf{v}_1, pf_1) \leftarrow \mathsf{Tally}(sk, nc, \mathfrak{bb}_1, \kappa)\\ &: |\mathfrak{bb}_0 \cup \mathfrak{bb}_1| \le mb \wedge nc \le mc \Rightarrow \mathbf{v} = \mathbf{v}_0 + \mathbf{v}_1] > 1 - \mathsf{negl}(\kappa).\end{aligned}$$

Proof of Fact 20. The proof follows from Definition 19. □

Lemma 21. *Let* Π *be an asymmetric encryption scheme and let* $\Gamma = \mathsf{Enc2Vote}(\Pi)$*. Suppose* ϵ *is the constant symbol used by* Γ*. We have for all probabilistic polynomial-time adversaries* $\mathcal{A}$ *and security parameters* κ *that* $\mathsf{Succ}(\mathsf{BS0}(\Gamma, \mathcal{A}, \kappa)) = \mathsf{Succ}(\mathsf{G}_0(\Gamma, \mathcal{A}, \epsilon, \kappa))$ *and* $\mathsf{Succ}(\mathsf{BS1{:}0}(\Gamma, \mathcal{A}, \kappa)) = \mathsf{Succ}(\mathsf{G}_q(\Gamma, \mathcal{A}, \epsilon, \kappa))$*, where* q *is an upper-bound on adversary* $\mathcal{A}$*'s oracle queries.*

Proof. The challengers in games $\mathsf{BS0}$ and G_0, respectively $\mathsf{BS1{:}0}$ and G_q, both construct 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 $\mathcal{O}(v_0, v_1)$ in games $\mathsf{BS0}$ and G_0 output ballots for vote v_0, hence, the bulletin boards are equivalent in both games. The bulletin boards in $\mathsf{BS1{:}0}$ and G_q are similarly equivalent, in particular, left-right oracle calls $\mathcal{O}(v_0, v_1)$ in both games output ballots for vote v_1, because q is an upper-bound on the left-right oracle queries, therefore, $|L| < q$ in G_q, where L is the set constructed by the oracle in G_q.

It follows that $|\mathfrak{bb}| \leq mb \wedge nc \leq mc$ in $\mathsf{BS0}$, respectively $\mathsf{BS1{:}0}$, iff $|\mathfrak{bb}| \leq mb \wedge nc \leq mc$ in G_0, respectively G_q. Moreover, predicate *balanced* is satisfied in $\mathsf{BS0}$, respectively $\mathsf{BS1{:}0}$, iff predicate *balanced* is satisfied in G_0, respectively G_q. Hence, if $|\mathfrak{bb}| \leq mb \wedge nc \leq mc$ is not satisfied or if predicate *balanced* is not satisfied, then $\mathsf{Succ}(\mathsf{BS0}(\Gamma, \mathcal{A}, \kappa)) = \mathsf{Succ}(\mathsf{G}_0(\Gamma, \mathcal{A}, \epsilon, \kappa))$ and $\mathsf{Succ}(\mathsf{BS1{:}0}(\Gamma, \mathcal{A}, \kappa)) = \mathsf{Succ}(\mathsf{G}_q(\Gamma, \mathcal{A}, \epsilon, \kappa))$, concluding our proof. Otherwise, it suffices to show that the inputs to the third adversary call are equivalent.

By inspection of games $\mathsf{BS0}$ and G_0, respectively $\mathsf{BS1{:}0}$ 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 ϵ and, by definition of Γ, it is also ϵ in $\mathsf{BS0}$, respectively $\mathsf{BS1{:}0}$. It remains to show that the first element of the triple input to the adversary call, namely the outcome, is equivalently computed in games $\mathsf{BS0}$ and G_0, respectively $\mathsf{BS1{:}0}$ and G_q.

In $\mathsf{BS0}$, respectively $\mathsf{BS1{:}0}$, 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 $\{b \in \mathfrak{bb} \wedge (b, v_0, v_1) \notin L\}$, where $\mathfrak{bb}$ 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 $\{b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L\}$). By Fact 20, it suffices to prove that the simulations are valid, i.e., in G_0 and G_q, computing

```
for b ∈ bb ∧ (b, v0, v1) ∈ L do
  v[v0] ← v[v0] + 1
```

is equivalent to

```
for b ∈ bb ∧ (b, v0, v1) ∈ L do
  v ← Dec(sk, b);
  if 1 ≤ v ≤ nc then
    v[v] ← v[v] + 1
```

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

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

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

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

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

is equivalent to

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

Thus, the simulation is valid in G_q too, thereby concluding our proof. □

Proof of Proposition 5. Let $\Gamma = \mathsf{Enc2Vote}(\Pi)$. Suppose Γ does not satisfy $\mathsf{Ballot\text{-}Secrecy}$, i.e., there exists a probabilistic polynomial-time adversary $\mathcal{A}$, such that for all negligible functions negl, there exists a security parameter and

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

By definition of BS0 and BS1, we have

$$= \frac{1}{2} \cdot (\mathsf{Succ}(\mathsf{BS0}(\Gamma, \mathcal{A}, \kappa)) + \mathsf{Succ}(\mathsf{BS1}(\Gamma, \mathcal{A}, \kappa)))$$

And, by Lemma 19, we have

$$\begin{aligned} &= \frac{1}{2} \cdot (\mathsf{Succ}(\mathsf{BS0}(\Gamma, \mathcal{A}, \kappa)) + 1 - \mathsf{Succ}(\mathsf{BS1{:}0}(\Gamma, \mathcal{A}, \kappa))) \\ &= \frac{1}{2} + \frac{1}{2} \cdot (\mathsf{Succ}(\mathsf{BS0}(\Gamma, \mathcal{A}, \kappa)) - \mathsf{Succ}(\mathsf{BS1{:}0}(\Gamma, \mathcal{A}, \kappa))) \end{aligned}$$

with non-negligible probability. Let ϵ be the constant symbol used by Γ and let q be an upper-bound on the number of oracle queries made by $\mathcal{A}$. Hence, by Lemma 21, we have

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

which can be rewritten as a telescoping series

$$= \frac{1}{2} + \frac{1}{2} \cdot \sum_{0 \le j < q} \mathsf{Succ}(\mathsf{G}_j(\Gamma, \mathcal{A}, \epsilon, \kappa)) - \mathsf{Succ}(\mathsf{G}_{j+1}(\Gamma, \mathcal{A}, \epsilon, \kappa))$$

Suppose $\mathsf{Succ}(\mathsf{G}_i(\Gamma, \mathcal{A}, \epsilon, \kappa)) - \mathsf{Succ}(\mathsf{G}_{i+1}(\Gamma, \mathcal{A}, \epsilon, \kappa))$ is the largest term in the series, where $i \in \{0, \ldots, q-1\}$. Hence,

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

Thus,

$$\frac{1}{2} + \frac{1}{q} \cdot \mathsf{negl}(\kappa) \;<\; \frac{1}{2} + \frac{1}{2} \cdot (\mathsf{Succ}(\mathsf{G}_i(\Gamma, \mathcal{A}, \epsilon, \kappa)) - \mathsf{Succ}(\mathsf{G}_{i+1}(\Gamma, \mathcal{A}, \epsilon, \kappa)))$$

From $\mathcal{A}$, we construct an adversary $\mathcal{B}$ against Π, and show that $\mathcal{B}$ wins with probability at least $\frac{1}{2} + \frac{1}{2} \cdot (\mathsf{Succ}(\mathsf{G}_i(\Gamma, \mathcal{A}, \epsilon, \kappa)) - \mathsf{Succ}(\mathsf{G}_{i+1}(\Gamma, \mathcal{A}, \epsilon, \kappa)))$.

Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ and $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. 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 $\mathcal{O}(v_0, v_1)$ as follows, namely, if $|L| < i$, then compute $b \leftarrow \mathsf{Enc}(pk, v_1); L \leftarrow L \cup \{(b, v_0, v_1)\}$ 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 $\mathcal{O}(v_0, v_1)$ as follows, namely, computes $b \leftarrow \mathsf{Enc}(pk, v_0); L \leftarrow L \cup \{(b, v_0, v_1)\}$ and returns b to $\mathcal{A}$; assigns $\mathcal{A}$'s output to $\mathfrak{bb}$; supposes $\{b_1, \ldots, b_k\} = \mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\}$; and outputs $(b_1, \ldots, b_k)$ to the challenger.
- $\mathcal{B}(\mathbf{p})$ initialises W as the empty set and $\mathbf{v}$ as a zero-filled vector of length nc, computes

```
for 1 ≤ j ≤ k do
    v' ← (0, ..., 0); // vector of length nc
    if 1 ≤ p[j] ≤ nc then
        v[p[j]] ← v[p[j]] + 1;
        v'[p[j]] ← 1;
    W ← W ∪ {(b_j, v')};
for b ∈ bb ∧ (b, v_0, v_1) ∈ L do
    v[v_0] ← v[v_0] + 1;
g ← A(v, ε, W);
```

  and outputs g.

We prove that $\mathcal{B}$ wins $\mathsf{IND\text{-}PA0}$ against Π with non-negligible probability.

Suppose (pk, sk) is an output of $\mathsf{Gen}(\kappa)$. 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 $\frac{q}{2} \cdot (\mathsf{Succ}(\mathsf{G}_i(\Gamma, \mathcal{A}, \epsilon, \kappa)) - \mathsf{Succ}(\mathsf{G}_{i+1}(\Gamma, \mathcal{A}, \epsilon, \kappa)))$ 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 ith left-right oracle call. Further suppose y is an output of $\mathsf{Enc}(pk, m_\beta)$, where β is a bit, and $\mathbf{c}$ 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 $\mathbf{c} = (b_1, \dots, b_k)$ such that

$$\{b_1, \dots, b_k\} = \mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\} \tag{1}$$

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

Claim 22. *Computing W as*

```
W ← ∅;
for 1 ≤ j ≤ k do
    v ← (0, ..., 0); // vector of length nc
    if 1 ≤ p[j] ≤ nc then
        v[p[j]] ← 1;
    W ← W ∪ {(b_j, v)};
```

is equivalent to computing W as

```
W ← ∅;
for b ∈ bb ∧ (b, v_0, v_1) ∉ L do
    (v, pf) ← Tally(sk, nc, {b}, κ);
    W ← W ∪ {(b, v)};
```

Claim 23. *Computing $\mathbf{v}$ as*

```
v ← (0, ..., 0); // vector of length nc
for 1 ≤ j ≤ k do
    if 1 ≤ p[j] ≤ nc then
        v[p[j]] ← v[p[j]] + 1;
```

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

```
v ← (0, ..., 0); // vector of length nc
for b ∈ bb ∧ (b, v_0, v_1) ∉ L do
    (v', pf) ← Tally(sk, nc, {b}, κ);
    v ← 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 22 & 23, we have either:

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

It follows that $\mathcal{B}$'s success is at least $\frac{1}{2} \cdot \mathsf{Succ}(\mathsf{G}_i(\Gamma, \mathcal{A}, \epsilon, \kappa)) + \frac{1}{2} \cdot (1 - \mathsf{Succ}(\mathsf{G}_{i+1}(\Gamma, \mathcal{A}, \epsilon, \kappa)))$, thus we conclude our proof by proving Claims 22 & 23.

Proof of Claim 22. By definition of $\mathbf{p}$ and since Dec is deterministic, the former computation is equivalent to

```
W ← ∅;
for 1 ≤ j ≤ k do
    v ← (0, ..., 0); // vector of length nc
    if 1 ≤ Dec(sk, b_j) ≤ nc then
        v[Dec(sk, b_j)] ← 1;
    W ← W ∪ {(b_j, v)};
```

Moreover, by definition of Tally and properties of addition, and since Dec is deterministic, the later computation is equivalent to

```
W ← ∅;
for b ∈ 𝔟𝔟 ∧ (b, v_0, v_1) ∉ L do
    v ← (0, ..., 0); // vector of length nc
    if 1 ≤ Dec(sk, b) ≤ nc then
        v[Dec(sk, b)] ← 1
    W ← W ∪ {(b, v)};
```

Hence, we conclude by (1).

Proof of Claim 23. By definition of $\mathbf{p}$ and since Dec is deterministic, the former computation computes vector $\mathbf{v}$ as

```
v ← (0, ..., 0); // vector of length nc
for 1 ≤ j ≤ k do
    if 1 ≤ Dec(sk, b_j) ≤ nc then
        v[Dec(sk, b_j)] ← v[Dec(sk, b_j)] + 1;
```

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

$\mathbf{v} \leftarrow (0, \ldots, 0)$; `// vector of length` nc
for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \notin L$ **do**
  $\mathbf{v}' \leftarrow (0, \ldots, 0)$; `// vector of length` nc
  if $1 \leq \mathsf{Dec}(sk, b) \leq nc$ **then**
    $\mathbf{v}'[\mathsf{Dec}(sk, b)] \leftarrow \mathbf{v}'[\mathsf{Dec}(sk, b)] + 1$
  $\mathbf{v} \leftarrow \mathbf{v} + \mathbf{v}'$;

which is equivalent to

$\mathbf{v} \leftarrow (0, \ldots, 0)$; `// vector of length` nc
for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \notin L$ **do**
  if $1 \leq \mathsf{Dec}(sk, b) \leq nc$ **then**
    $\mathbf{v}[\mathsf{Dec}(sk, b)] \leftarrow \mathbf{v}[\mathsf{Dec}(sk, b)] + 1$

Hence, we conclude by (1). □

C.6 Proof of Lemma 6

Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$. Suppose algorithm Reveal is not correct with respect to Γ. We construct an adversary $\mathcal{A}$ against game Reveal-Soundness.

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

Suppose κ is a security parameter, nb and nc are integers, $v, v_1, \ldots, v_{nb} \in \{1, \ldots, nc\}$ are votes, and $\mathsf{Setup}(\kappa)$ outputs (pk, sk, mb, mc). We consider the interesting case: $nb \leq mb \wedge nc \leq mc$. Since Setup is efficient, integers mb and mc can be efficiently computed. Moreover, since Vote is efficient, $nb \leq mb \wedge nc \leq mc$, and $v \in \{1, \ldots, nc\}$, adversary $\mathcal{A}$ is efficient, i.e., $\mathcal{A}$ is a probabilistic polynomial-time adversary.

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

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

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

C.7 Proof of Proposition 7

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

- $\mathcal{B}(pk, \kappa)$ computes $np \leftarrow \mathcal{A}(pk, \kappa)$ and outputs np.
- $\mathcal{B}()$ initialises $L \leftarrow \emptyset$, computes $\mathfrak{bb} \leftarrow \mathcal{A}()$, and outputs $\mathfrak{bb}$. Any oracle calls from $\mathcal{A}$ on inputs (p_0, p_1) are forwarded to $\mathcal{B}$'s oracle and a transcript of calls is maintained, i.e., $\mathcal{B}$ computes $b \leftarrow \mathcal{O}(p_0, p_1); L \leftarrow L \cup \{(b, p_0, p_1)\}$ and returns b to $\mathcal{A}$.
- $\mathcal{B}(\mathbf{v}, pf, W)$ proceeds as follows. Finds the largest integer p such that $\mathbf{v}[p] > 0 \wedge 1 \leq p \leq np$; if no such integer exists, then algorithm $\mathcal{B}$ computes $g \leftarrow \mathcal{A}(0, \emptyset, \epsilon)$ and outputs g. If $(b, p, p_1) \in L \wedge b \in \mathfrak{bb}$, then abort. Otherwise, algorithm $\mathcal{B}$ assigns $\mathfrak{b} \leftarrow \{b \mid (b, \mathbf{v}') \in W \wedge \mathbf{v}'[p] = 1\}$, computes $g \leftarrow \mathcal{A}(p, \mathfrak{b}, \epsilon)$, and outputs g.

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

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

   Let $\mathbf{v}_0$ and $\mathbf{v}_1$ be zero-filled vectors of length np. By correctness of Γ, the computation $\mathbf{v}_0[p] \leftarrow 1; \mathbf{v}_1[p] \leftarrow 1; \mathbf{v}_0[p_0] \leftarrow \mathbf{v}_0[p_0] + 1; \mathbf{v}_1[p_1] \leftarrow \mathbf{v}_1[p_1] + 1; (\mathbf{v}', pf') \leftarrow \mathsf{Tally}(sk, np, \{b, b'\}, \kappa)$ ensures $\mathbf{v}' = \mathbf{v}_0 \vee \mathbf{v}' = \mathbf{v}_1$, with overwhelming probability. Moreover, by tally soundness, we have $\mathbf{v}'[p] \geq$ *correct-outcome*$(pk, np, \{b, b'\}, \kappa)[p]$ and we have $\mathbf{v}'[p_0] \geq$ *correct-outcome*$(pk, np, \{b, b'\}, \kappa)[p_0] \vee \mathbf{v}'[p_1] \geq$ *correct-outcome*$(pk, np, \{b, b'\}, \kappa)[p_1]$. Thus, by definition of *correct-outcome*, we have

   $$b \neq \bot \wedge b' \neq \bot \tag{2}$$

   It follows that

   $$\exists r \,.\, \mathsf{Bid}(pk, np, p, \kappa; r) \in \mathfrak{bb} \setminus \{\bot\} \wedge 1 \leq p \leq np \tag{3}$$

Since Γ satisfies tally soundness, we have for all $p' \in \{1, \ldots, np\}$ that $\mathbf{v}[p'] \geq \mathit{correct\text{-}outcome}(pk, np, \mathfrak{bb}, \kappa)[p']$, with overwhelming probability. Moreover, since p is the largest integer such that $\mathbf{v}[p] > 0 \wedge 1 \leq p \leq np$, we have for all $p' \in \{p+1, \ldots, np\}$ that $\mathbf{v}[p'] \leq 0$. Hence, by definition of *correct-outcome*, we have, with overwhelming probability, that:

$$\neg \exists p', r' \,.\, \mathsf{Bid}(pk, np, p', \kappa; r') \in \mathfrak{bb} \setminus \{\perp\} \wedge p < p' \leq np \qquad (4)$$

By (3) & (4), we derive that $\mathit{correct\text{-}price}(pk, np, \mathfrak{bb}, p, \kappa)$ holds with overwhelming probability. Furthermore, since $\mathcal{A}$ wins the Bid-Secrecy game it follows for all $b \in \mathfrak{bb}$ that $(b, p, p_1) \notin L$ with overwhelming probability. However, we have assumed $(b, p, p_1) \in L \wedge b \in \mathfrak{bb}$ with non-negligible probability, hence we derive a contradiction.

2. Since $\mathcal{B}$ aborts with negligible probability, we can infer $b \in \mathfrak{bb}$ implies $(b, p, p_1) \notin L$ with overwhelming probability. By this inference and by definition of Ballot-Secrecy, we have W is a set of pairs $(b, \mathbf{v}')$ such that $b \in \mathfrak{bb}$ and $(\mathbf{v}', pf')$ is output by Tally for some pf'. It follows by definition of $\mathcal{B}$ that $\mathfrak{b} = \{b \mid (b, \mathbf{v}') \in W \wedge \mathbf{v}'[p] = 1\}$. Since Reveal satisfies reveal soundness with respect to Γ, we have $\mathcal{B}$ simulates Reveal.

We have shown that $\mathcal{B}$ simulates $\mathcal{A}$'s challenger with overwhelming probability. It follows that $\mathcal{B}$ guesses β 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. □

C.8 Proof of Proposition 8

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

- Reveal-Enc2Bid$(sk, nc, \mathfrak{bb}, v, \kappa)$ computes $\mathfrak{b} \leftarrow \{b \mid b \in \mathfrak{bb} \wedge \mathsf{Dec}(sk, b) = v\}$ and outputs $\mathfrak{b}$.

It follows from Definitions 6, 15 & 19 that Enc2Bid(Π) and Λ(Enc2Vote(Π), Reveal-Enc2Bid(Π)) are equivalent, assuming the same constant is used by Enc2Vote(Π), Enc2Bid(Π), and Λ(Enc2Vote(Π), Reveal-Enc2Bid(Π)). Hence, by Proposition 5 and 7, to show that Enc2Bid(Π) satisfies bid secrecy, it suffices to show that Enc2Vote(Π) satisfies tally soundness and Reveal-Enc2Bid(Π) satisfies reveal soundness with respect to Enc2Vote(Π).

We prove Enc2Vote(Π) satisfies tally soundness by contradiction. Suppose κ is a security parameter and Setup(κ) outputs (pk, sk, mb, mc). Further suppose nc is an integer and $\mathfrak{bb}$ is a set such that $|\mathfrak{bb}| \leq mb \wedge nc \leq mc$. Moreover, suppose Tally$(sk, nc, \mathfrak{bb}, \kappa)$ outputs $(\mathbf{v}, pf)$. Let $\ell = \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)[v]$. Suppose there exists $v \in \{1, \ldots, nc\}$ such that $\mathbf{v}[v] < \ell$. By definition of *correct-outcome*, we have $\exists^{=\ell} b \in \mathfrak{bb} \setminus \{\perp\} : \exists r : b = \mathsf{Enc}(pk, v; r)$. And by definition of Vote, bulletin board $\mathfrak{bb}$ contains ℓ ciphertexts for plaintext v. Since

pk, sk are outputs of Gen and since Π is perfectly correct, we have that those ℓ ciphertexts all decrypt to v. By definition of Tally, it follows that $\mathbf{v}[v] \geq \ell$, thereby deriving a contradiction.

We prove $\mathsf{Reveal\text{-}Enc2Bid}(\Pi)$ satisfies reveal soundness with respect to $\mathsf{Enc2Vote}(\Pi)$. Suppose κ is a security parameter and $\mathsf{Setup}(\kappa)$ outputs (pk, sk, mb, mc). Further suppose $\mathfrak{bb}$ is a set and nc and v are integers such that $|\mathfrak{bb}| \leq mb \wedge 1 \leq v \leq nc \leq mc$. Moreover, suppose $\mathsf{Reveal\text{-}Enc2Bid}(sk, nc, \mathfrak{bb}, v, \kappa)$ outputs $\mathfrak{b}$. By definition of $\mathsf{Reveal\text{-}Enc2Bid}$, we have

$$\mathfrak{b} = \{b \mid b \in \mathfrak{bb} \wedge \mathsf{Dec}(sk, b) = v\}.$$

Suppose W is computed as follows.

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

Let $\mathbf{v}_0$ 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 $b \in \mathfrak{bb}$ **do**
    $\mathbf{v} \leftarrow \mathbf{v}_0$;
    $v' \leftarrow \mathsf{Dec}(sk, b)$;
    if $1 \leq v' \leq nc$ **then**
        $\mathbf{v}[v'] \leftarrow 1$;
    $W \leftarrow W \cup \{(b, \mathbf{v})\}$;

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

C.9 Proof of Theorem 9

Let $\Sigma = \Lambda(\Gamma, \mathsf{Reveal})$ and $\Sigma' = \Lambda(\Gamma, \mathsf{Reveal}, \Delta)$. By Proposition 7, we have that Σ satisfies bid secrecy. We prove Σ' satisfies bid secrecy by contradiction. Suppose Σ' does not satisfy bid secrecy, hence, there exists an adversary $\mathcal{A}$, such that for all negligible functions negl, there exists a security parameter κ and $\mathsf{Succ}(\mathsf{Bid\text{-}Secrecy}(\Sigma', \mathcal{A}, \kappa)) > \frac{1}{2} + \mathsf{negl}(\kappa)$. Let us construct an adversary $\mathcal{B}$ that wins $\mathsf{Bid\text{-}Secrecy}(\Sigma, \mathcal{B}, \kappa)$.

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

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

C.10 Proof of Lemma 10

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

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

Since $\mathcal{A}$ wins $\mathsf{Tally\text{-}Soundness}(\Gamma, \mathcal{A}, \kappa)$, we have: $\Pr[(pk, nc, \mathfrak{bb}, \mathbf{v}, pf) \leftarrow \mathcal{B}(\kappa) : \mathbf{v} \neq \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa) \wedge |\mathfrak{bb}| \leq mb \wedge nc \leq mc] > \mathsf{negl}(\kappa)$. Moreover, by completeness, there exists a negligible function negl' such that: $\Pr[(pk, nc, \mathfrak{bb}, \mathbf{v}, pf) \leftarrow \mathcal{B}(\kappa) : |\mathfrak{bb}| \leq mb \wedge nc \leq mc \Rightarrow \mathsf{Verify}(pk, nc, \mathfrak{bb}, \mathbf{v}, pf, \kappa) = 1] > 1 - \mathsf{negl}'(\kappa)$. It follows that: $\Pr[(pk, nc, \mathfrak{bb}, \mathbf{v}, pf) \leftarrow \mathcal{B}(\kappa) : \mathbf{v} \neq \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa) \wedge \mathsf{Verify}(pk, nc, \mathfrak{bb}, \mathbf{v}, pf, \kappa) = 1] > \mathsf{negl}(\kappa) \cdot (1 - \mathsf{negl}'(\kappa))$. Hence, $\mathcal{B}$ wins $\mathsf{Exp\text{-}UV\text{-}Ext}(\Gamma, \mathcal{B}, \kappa)$. □

C.11 Proof of Theorem 13

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

Suppose Γ satisfies universal verifiability. By definition of universal verifiability, we have Γ satisfies strong injectivity. And, by definition of strong injectivity and by Definition 16, it is trivial to see that Σ satisfies strong injectivity. We proceed by contradiction. Suppose Σ does not satisfy universal verifiability, hence, there exists an adversary $\mathcal{A}$, negligible function negl, and security parameter κ, such that $\mathsf{Succ}(\mathsf{Exp\text{-}UV}(\Sigma, \mathcal{A}, \kappa)) > \mathsf{negl}(\kappa)$, i.e.,

$$\begin{aligned}
&\Pr[(pk, np, \mathfrak{bb}, p, \mathfrak{b}, \sigma) \leftarrow \mathcal{A}(\kappa) \\
&\quad : (\neg \mathit{correct\text{-}price}(pk, np, \mathfrak{bb}, p, \kappa) \vee \neg \mathit{correct\text{-}bids}(pk, np, \mathfrak{bb}, p, \mathfrak{b}, \kappa)) \\
&\quad \wedge \mathsf{Verify}_\Sigma(pk, np, \mathfrak{bb}, p, \mathfrak{b}, \sigma, \kappa) = 1] > \mathsf{negl}(\kappa) \qquad (5)
\end{aligned}$$

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

- $\mathcal{B}(\kappa)$ computes $(pk, np, \mathfrak{bb}, p, \mathfrak{b}, \sigma) \leftarrow \mathcal{A}(\kappa)$, parses σ as $(\mathbf{v}, pf, pf')$, and outputs $(pk, np, \mathfrak{bb}, \mathbf{v}, pf)$.
- $\mathcal{C}(\kappa)$ computes $(pk, np, \mathfrak{bb}, p, \mathfrak{b}, \sigma) \leftarrow \mathcal{A}(\kappa)$, parses σ as $(\mathbf{v}, pf, pf')$, assigns $s \leftarrow (pk, np, \mathfrak{bb}, p, \mathfrak{b}, \kappa)$, and outputs (s, pf').

Henceforth, we assume that adversaries $\mathcal{B}$ and $\mathcal{C}$ successfully parse σ. This assumption is necessary for $\mathcal{A}$ to win $\mathsf{Exp\text{-}UV}(\Sigma, \mathcal{A}, \kappa)$, hence we do not lose generality.

First, we consider adversary $\mathcal{B}$'s success. Let $\psi(\mathbf{v}, p, np)$ hold if p is the largest integer such that $\mathbf{v}[p] > 0 \wedge 1 \leq p \leq np$, or there is no such integer and $p = 0$. By definition of ψ and by inspection of Verify_Σ, we have:

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

Let us assume the following:

$$\psi(\mathbf{v}, p, np) \wedge \neg \mathit{correct\text{-}price}(pk, np, \mathfrak{bb}, p, \kappa) \\ \Rightarrow \mathbf{v} \neq \mathit{correct\text{-}outcome}(pk, np, \mathfrak{bb}, \kappa) \quad (7)$$

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

$$\begin{aligned} &\Pr[(pk, nc, \mathfrak{bb}, \mathbf{v}, pf) \leftarrow \mathcal{B}(\kappa) : \mathsf{Verify}_\Gamma(pk, nc, \mathfrak{bb}, \mathbf{v}, pf, \kappa) = 1 \\ &\qquad \wedge \mathbf{v} \neq \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)] \\ &\geq \Pr[(pk, np, \mathfrak{bb}, p, \mathfrak{b}, \sigma) \leftarrow \mathcal{A}(\kappa) : \mathsf{Verify}_\Sigma(pk, np, \mathfrak{bb}, p, \mathfrak{b}, \sigma, \kappa) = 1 \\ &\qquad \wedge \neg \mathit{correct\text{-}price}(pk, np, \mathfrak{bb}, p, \kappa)] \end{aligned} \quad (8)$$

Equation (8) relates $\mathcal{B}$'s success to $\mathcal{A}$'s success.

Secondly, we consider adversary $\mathcal{C}$'s success. By further inspection of Verify_Σ, we have:

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

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

$$\neg \mathit{correct\text{-}bids}(pk, np, \mathfrak{bb}, p, \mathfrak{b}, \kappa) \Rightarrow ((pk, np, \mathfrak{bb}, p, \mathfrak{b}, \kappa), sk) \notin R(\Gamma, \mathsf{Reveal})$$

with overwhelming probability. It follows that:

$$\begin{aligned} &\Pr[(s, \tau) \leftarrow \mathcal{C}(\kappa) : (s, w) \notin R(\Gamma, \mathsf{Reveal}) \wedge \mathsf{Verify}(s, \tau, \kappa) = 1] \\ &\geq \Pr[(pk, np, \mathfrak{bb}, p, \mathfrak{b}, \sigma) \leftarrow \mathcal{A}(\kappa) : \mathsf{Verify}_\Sigma(pk, np, \mathfrak{bb}, p, \mathfrak{b}, \sigma, \kappa) = 1 \\ &\qquad \wedge \neg \mathit{correct\text{-}bids}(pk, np, \mathfrak{bb}, p, \mathfrak{b}, \kappa)] \end{aligned} \quad (9)$$

with overwhelming probability. Equation (9) relates $\mathcal{C}$'s success to $\mathcal{A}$'s success.

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

$$\begin{aligned}&\Pr[(pk, nc, \mathfrak{bb}, \mathbf{v}, pf) \leftarrow \mathcal{B}(\kappa) : \mathsf{Verify}_\Gamma(pk, nc, \mathfrak{bb}, \mathbf{v}, pf, \kappa) = 1 \\ &\qquad \wedge \mathbf{v} \neq \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)] > \mathsf{negl}(\kappa) \\ &\vee \Pr[(s, \tau) \leftarrow \mathcal{C}(\kappa) : (s, w) \notin R(\Gamma, \mathsf{Reveal}) \wedge \mathsf{Verify}(s, \tau, \kappa) = 1] > \mathsf{negl}(\kappa)\end{aligned}$$

The above equation shows that $\mathcal{A}$'s success provides an advantage for adversary $\mathcal{B}$ or $\mathcal{C}$. To conclude, it remains to prove (7).

Proof of (7). By inspection of *correct-price*, we have:

$$\begin{aligned}&\psi(\mathbf{v}, p, np) \wedge \neg \mathit{correct\text{-}price}(pk, np, \mathfrak{bb}, p, \kappa) \\ &= \psi(\mathbf{v}, p, np) \wedge ((\exists p', r' \,.\, \mathsf{Bid}(pk, np, p', \kappa; r') \in \mathfrak{bb} \setminus \{\bot\} \wedge p < p' \leq np) \\ &\qquad \vee p \notin \{0, \dots, np\} \\ &\qquad \vee (p \neq 0 \wedge \neg \exists r \,.\, \mathsf{Bid}(pk, np, p, \kappa; r) \in \mathfrak{bb} \setminus \{\bot\}))\end{aligned}$$

Moreover, since $\psi(\mathbf{v}, p, np) \wedge p \notin \{0, \dots, np\}$ is false, we have:

$$\begin{aligned}&= \psi(\mathbf{v}, p, np) \wedge ((\exists p', r' \,.\, \mathsf{Bid}(pk, nc, p', \kappa; r') \in \mathfrak{bb} \setminus \{\bot\} \wedge p < p' \leq np) \\ &\qquad \vee (p \neq 0 \wedge \neg \exists r \,.\, \mathsf{Bid}(pk, np, p, \kappa; r) \in \mathfrak{bb} \setminus \{\bot\}))\end{aligned}$$

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

$$\Rightarrow \mathbf{v} \neq \mathit{correct\text{-}outcome}(pk, np, \mathfrak{bb}, \kappa),$$

thereby concluding our proof. □

D Reveal algorithms exist

We prove that every election scheme has a reveal algorithm that is correct with respect to that election scheme (Proposition 24). 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.

Proposition 24. *Given an election scheme, there exists a reveal algorithm that is correct with respect to that election scheme.*

Proof. Suppose $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ is an election scheme. Let reveal algorithm Reveal be defined as follows:

$\mathsf{Reveal}(sk, nc, \mathfrak{bb}, v, \kappa) =$

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

We prove that Reveal is correct with respect to Γ.

Suppose κ is a security parameter, nb and nc are integers, and $v, v_1, \ldots, v_{nb} \in \{1, \ldots, nc\}$ are votes. Moreover, suppose $\mathsf{Setup}(\kappa)$ outputs (pk, sk, mb, mc) such that $nb \leq mb \wedge nc \leq mc$ and for each $1 \leq i \leq nb$ we have $\mathsf{Vote}(pk, nc, v_i, \kappa)$ outputs b_i. Further suppose that $\mathsf{Reveal}(sk, nc, \{b_1, \ldots, b_{nb}\}, v, \kappa)$ outputs $\mathfrak{b}$. By definition of Reveal, we have $b_i \in \mathfrak{b}$ if $\mathsf{Tally}(sk, nc, \{b\}, \kappa)$ outputs $(\mathbf{v}, pf)$ such that $\mathbf{v}[v] = 1$. By correctness of Γ, we have $\mathbf{v}[v] = 1$ if $v_i = v$, with overwhelming probability. Furthermore, by definition of Reveal, we have $\mathfrak{b} \subseteq \{b_1, \ldots, b_{nb}\}$. It follows that $\mathfrak{b} = \{b_i \mid v_i = v\}$ with overwhelming probability, hence Reveal satisfies reveal algorithm correctness. □

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 25). Our proof exploits our construction's reliance on the tallying algorithm producing the expected outcome (§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, the auction scheme derived from the modified scheme does not satisfy bid secrecy, because the adversary can cause the announcement of an incorrect winning price, which causes the reveal algorithm to link bidders that bid at that price.

Proposition 25. *There exists a function* incorrect-price*, such that for all election schemes Γ (that permits at least two prices and at least three bids for some security parameter) satisfying ballot secrecy, we have election scheme* $\mathsf{incorrect\text{-}price}(\Gamma)$ *satisfies ballot secrecy, yet auction scheme* $\Lambda(\mathsf{incorrect\text{-}price}(\Gamma)$,

Reveal) *does not satisfy bid secrecy, for some reveal algorithm* Reveal *that is correct with respect to* $\mathsf{incorrect\text{-}price}(\Gamma)$.

Definition 35. *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme. Suppose* ω *and* ϵ *are constant symbols that cannot be output by* Vote. *We define* $\mathsf{incorrect\text{-}price}(\Gamma) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}', \mathsf{Verify}')$, *where* Tally' *and* Verify' *are defined as follows.*

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

$\mathsf{Verify}'(pk, nc, \mathfrak{bb}, \mathbf{v}, pf, \kappa)$ *outputs* 1.

Lemma 26. *Given an election scheme* Γ, *we have* $\mathsf{incorrect\text{-}price}(\Gamma)$ *is an election scheme.*

Proof sketch. It suffices to show that $\mathsf{incorrect\text{-}price}(\Gamma)$ satisfies correctness, completeness, and injectivity. Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$. Correctness follows from the underlying scheme, because ω cannot be output by Vote. Completeness follows from Fact 18. And Injectivity follows from the underlying scheme, because we do not modify Setup nor Vote. □

Lemma 27. *Given an election scheme* Γ *satisfying ballot secrecy, we have* $\mathsf{incorrect\text{-}price}(\Gamma)$ *satisfies ballot secrecy.*

Proof sketch. Suppose $\mathsf{incorrect\text{-}price}(\Gamma)$ does not satisfy ballot secrecy, i.e., there exists an adversary that wins game $\mathsf{Ballot\text{-}Secrecy}$ against $\mathsf{incorrect\text{-}price}(\Gamma)$. From this adversary we can construct an adversary that wins $\mathsf{Ballot\text{-}Secrecy}$ 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 $\mathsf{incorrect\text{-}price}$), hence deriving a contradiction. □

Proof of Proposition 25. Suppose Γ is an election scheme satisfying ballot secrecy. By Lemma 26 & 27, we have $\mathsf{incorrect\text{-}price}(\Gamma)$ is an election scheme satisfying ballot secrecy. And, by Proposition 24, there exists a reveal algorithm Reveal that is correct with respect to $\mathsf{incorrect\text{-}price}(\Gamma)$. By Lemma 2, we have $\Lambda(\mathsf{incorrect\text{-}price}(\Gamma), \mathsf{Reveal})$ is an auction scheme. And it remains to show that $\Lambda(\mathsf{incorrect\text{-}price}(\Gamma), \mathsf{Reveal})$ does not satisfy bid secrecy.

Let $\mathsf{incorrect\text{-}price}(\Gamma) = (\mathsf{Setup}_\Gamma, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify}_\Gamma)$, $\Lambda(\mathsf{incorrect\text{-}price}(\Gamma), \mathsf{Reveal}) = (\mathsf{Setup}, \mathsf{Bid}, \mathsf{Open}, \mathsf{Verify})$, and ω be the constant used by the set membership test introduced by $\mathsf{incorrect\text{-}price}$. We construct an adversary $\mathcal{A}$ against game $\mathsf{Bid\text{-}Secrecy}$.

- $\mathcal{A}(pk, \kappa)$ outputs 2.
- $\mathcal{A}()$ computes $b_0 \leftarrow \mathcal{O}(1, 2); b_1 \leftarrow \mathcal{O}(2, 1); \mathfrak{bb} \leftarrow \{b_0, b_1, \omega\}$ and outputs $\mathfrak{bb}$.

- $\mathcal{A}(p, \mathfrak{b}, pf)$ outputs 0 if $b_0 \in \mathfrak{b}$, and 1 otherwise.

Suppose κ is a security parameter and $\mathsf{Setup}(\kappa)$ 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 $\mathcal{A}(pk, \kappa)$ outputs np and $\mathcal{A}()$ outputs $\mathfrak{bb}$, hence, we have $\mathfrak{bb} = \{b_0, b_1, \omega\}$, such that

$$b_0 = \mathsf{Vote}(pk, np, 1 + \beta, \kappa; r_0) \wedge b_1 = \mathsf{Vote}(pk, np, 2 - \beta, \kappa; r_1),$$

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

```
𝔟 ← ∅;
for b ∈ 𝔟𝔟 do
    (v, pf) ← Tally(sk, np, {b}, κ);
    if v[p] = 1 then
        𝔟 ← 𝔟 ∪ {b};
```

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

$$correct\text{-}price(pk, np, \mathfrak{bb}, p, \kappa) \Rightarrow \forall b \in \mathfrak{bb} \,.\, (b, p, p_1) \notin L \wedge (b, p_0, p) \notin L$$

holds vacuously, because $b_{1-\beta} \in \mathfrak{bb}$ is a bid for $2 > p$, hence, $correct\text{-}price(pk, np, \mathfrak{bb}, p, \kappa)$ does not hold. Thus, the adversary wins against game $\mathsf{Bid\text{-}Secrecy}$, concluding our proof. □

F An auction scheme from Helios

Our construction is parameterised by an election scheme, a reveal algorithm, and a non-interactive proof system. Hence, we derive an auction scheme from Helios as follows.

Definition 36. *Let* Helios'16 *be the election scheme defined by Smyth, Frink & Clarkson [SFC15],* Reveal *be the reveal algorithm given in Definition 37, and* Δ *be the proof system defined in Definition 38. We define* the auction scheme from Helios'16 *as* $\Lambda(\text{Helios'16}, \mathsf{Reveal}, \Delta)$.[27]

In this appendix, let $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ be Helios'16 and $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be the additively homomorphic asymmetric encryption scheme used by Helios'16. Moreover, let $(\mathsf{ProveKey}, \mathsf{VerKey})$, respectively $(\mathsf{ProveDec}, \mathsf{VerifyDec})$

[27] Formally, $\Lambda(\text{Helios'16}, \mathsf{Reveal}, \Delta)$ is an auction scheme by Lemmata 3, 28, & 29.

and $(\mathsf{ProveCiph}, \mathsf{VerifyCiph})$, be the non-interactive proof system derived by application of the Fiat-Shamir transformation [FS87] to a random oracle $\mathcal{H}$ and the sigma protocol for proving knowledge of discrete logarithms by Chaum et al. [CEGP87, Protocol 2], respectively the sigma protocol for proving knowledge of 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].

F.1 Reveal algorithm

Definition 37. *We define reveal algorithm* Helios-Reveal *as follows.*

Helios-Reveal$(sk', nc, \mathfrak{bb}, v, \kappa)$ *proceeds as follows. Parse sk' as a vector (pk, sk). Let $\{b_1, \dots, b_\ell\}$ be the largest subset of $\mathfrak{bb}$ satisfying the conditions given by the* Helios'16 *tallying algorithm (see [SFC15] for details). Compute:*

$\mathfrak{b} \leftarrow \emptyset$;
for $1 \leq i \leq \ell$ **do**
  if $(v = nc \wedge \mathsf{Dec}(sk, b_i[1] \otimes \cdots \otimes b_i[nc-1]) = 0)$
  $\vee\ (1 \leq v < nc \wedge \mathsf{Dec}(sk, b_i[v]) = 1)$ **then**
    $\mathfrak{b} \leftarrow \mathfrak{b} \cup \{b_j\}$;

Output $\mathfrak{b}$.

Lemma 28. *Reveal algorithm* Helios-Reveal *is correct with respect to* Helios'16.

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

- Case I: $1 \leq v < nc$. By definition of Vote, we have for all $b \in \mathfrak{bb}$ that b is an output of the asymmetric encryption scheme used by Helios'16. Moreover, if $v_i = v$, then $b[v]$ enciphers 1, otherwise, $b[v]$ enciphers 0. By correctness of the encryption scheme, we have with overwhelming probability that $v_i = v$ implies $\mathsf{Dec}(sk, b[v]) = 1$. Hence, by definition of Helios-Reveal, we have $b \in \mathfrak{b}$.

- Case II: $v = nc$. As per the first case, we have for all $b \in \mathfrak{bb}$ that b is an output of the encryption scheme used by Helios'16, but the construction of b differs from the previous case, namely, we have $b[1], \dots, b[nc-1]$ each encipher 0. Given that the encryption scheme is homomorphic, we have with overwhelming probability that $\mathsf{Dec}(sk, b[1] \otimes \cdots \otimes b[nc-1]) = 0$. Hence, by definition of Helios-Reveal, we have $b \in \mathfrak{b}$.

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

F.2 Non-interactive proof system

Definition 38. *We define the tuple of algorithms* ($\mathsf{ProveReveal}, \mathsf{VerifyReveal}$) *as follows:*

$\mathsf{ProveReveal}(s, sk, \kappa)$ *proceeds as follows. Parse* s *as* $(pk', nc, \mathfrak{bb}, v, \mathfrak{b}, \kappa)$ *and* pk' *as* $(pk, \mathfrak{m}, \rho)$. *Output* $\perp$ *if parsing fails or if* $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) \neq 1 \vee v \notin \{1, \ldots, nc\} \vee \{1, \ldots, nc\} \not\subseteq \mathfrak{m}$. *Let* $\{b_1, \ldots, b_\ell\}$ *be the largest subset of* $\mathfrak{bb}$ *satisfying the conditions given by the* Helios'16 *tallying algorithm (see [SFC15] for details). Initialise vector* $\mathbf{Q}$ *of length* ℓ *and compute:*

for $1 \leq i \leq \ell$ **do**
  if $1 \leq v < nc$ **then**
    $\mathbf{Q}[i] \leftarrow \mathsf{ProveDec}((pk, b_i[v], \mathsf{Dec}(sk, b_i[v])), sk, \kappa)$;
  else
    $c \leftarrow b_i[1] \otimes \cdots \otimes b_i[nc-1]$;
    $\mathbf{Q}[i] \leftarrow \mathsf{ProveDec}((pk, c, \mathsf{Dec}(sk, c)), sk, \kappa)$;

Output $\mathbf{Q}$.

$\mathsf{VerifyReveal}(s, \mathbf{Q})$ *proceeds as follows. Parse* s *as* $(pk', nc, \mathfrak{bb}, v, \mathfrak{b}, \kappa)$ *and* pk' *as* $(pk, \mathfrak{m}, \rho)$. *Output* 0 *if parsing fails or if* $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) \neq 1 \vee v \notin \{1, \ldots, nc\} \vee \{1, \ldots, nc\} \not\subseteq \mathfrak{m}$. *Let* $\{b_1, \ldots, b_\ell\}$ *be the largest subset of* $\mathfrak{bb}$ *satisfying the conditions given by the* Helios'16 *tallying algorithm. Output* 1 *if any of the following checks hold.*

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

Output 0 *if all of the checks fail.*

Lemma 29. *The tuple of algorithms* ($\mathsf{ProveReveal}, \mathsf{VerifyReveal}$) *is a non-interactive proof system for relation* $R(\text{Helios'16}, \mathsf{Helios\text{-}Reveal})$ *(i.e., it satisfies completeness).*

Proof sketch. Suppose $(s, sk') \in R(\text{Helios'16}, \mathsf{Helios\text{-}Reveal})$ and κ is a security parameter. Since $R(\text{Helios'16}, \mathsf{Helios\text{-}Reveal})$ is defined over vectors of length 6 and bitstrings, we can parse s as $(pk', nc, \mathfrak{bb}, v, \mathfrak{b}, \kappa)$. Moreover, by definition of $R(\text{Helios'16}, \mathsf{Helios\text{-}Reveal})$, there exists mb, mc, r, and r', such that $\mathfrak{b} =$

$\mathsf{Helios\text{-}Reveal}(sk', nc, \mathfrak{bb}, v, \kappa; r)$, $(pk', sk', mb, mc) = \mathsf{Setup}(\kappa; r')$ and $1 \leq v \leq nc \leq mc$. And, by definition of Setup, we have pk' is a vector $(pk, \mathfrak{m}, \rho)$, where (pk, sk) is an output of Gen, $\mathfrak{m}$ is the encryption scheme's message space, ρ is an output of ProveKey, and mc is the largest integer such that $\{0, ..., mc\} \subseteq \mathfrak{m}$.

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

- Case I: $1 \leq v < nc$. Suppose $i \in \{1, \ldots, \ell\}$. We have $\mathbf{Q}[i]$ is an output of $\mathsf{ProveDec}((pk, b_i[v], \mathsf{Dec}(sk, b_i[v])), sk, \kappa)$ by definition of ProveReveal. If $b_i \in \mathfrak{b}$, then $\mathsf{Dec}(sk, b_i[v]) = 1$ by definition of Helios-Reveal and, with overwhelming probability, $\mathsf{VerifyDec}((pk, b_i[v], 1), \mathbf{Q}[i], \kappa) = 1$ by correctness of the encryption scheme and by completeness of Δ. Otherwise ($b_i \notin \mathfrak{b}$,), we proceed as follows. We have $\mathsf{Dec}(sk, b_i[v]) \neq 1$ by definition of Helios-Reveal, hence, $\mathsf{Dec}(sk, b_i[v]) = 0$, because $b_i[v]$ is a encryption of a plaintext in $\{0, 1\}$, by the tallying conditions of Helios'16. By correctness of the encryption scheme and by completeness of Δ, we have $\mathsf{VerifyDec}((pk, b_i[v], 0), \mathbf{Q}[i], \kappa) = 1$, with overwhelming probability.
- Case II: $v = nc$. Suppose $i \in \{1, \ldots, \ell\}$. Let $c = b_i[1] \otimes \cdots \otimes b_i[nc-1]$. We have $\mathbf{Q}[i]$ is an output of $\mathsf{ProveDec}((pk, c, \mathsf{Dec}(sk, c)), sk, \kappa)$ by definition of ProveReveal. If $b_i \in \mathfrak{b}$, then $\mathsf{Dec}(sk, c) = 0$ by definition of Helios-Reveal and, with overwhelming probability, $\mathsf{VerifyDec}((pk, c, 0), \mathbf{Q}[i], \kappa) = 1$ by correctness of the encryption scheme and by completeness of Δ. Otherwise ($b_i \notin \mathfrak{b}$), we proceed as follows. We have $\mathsf{Dec}(sk, c) = 1$ by definition of Helios-Reveal and the tallying conditions of Helios'16. By correctness of the encryption scheme and by completeness of Δ, we have $\mathsf{VerifyDec}((pk, c, 1), \mathbf{Q}[i], \kappa) = 1$.

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

F.3 Lemmata supporting Theorem 14

Lemma 30. *Reveal algorithm* Helios-Reveal *satisfies reveal soundness with respect to* Helios'16.

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

By definition of algorithm Tally, we have for all $(b, \mathbf{v}) \in W$ that $b \in \mathfrak{bb}$ and either $\emptyset$ or $\{b\}$ is the largest subset of $\{b\}$ satisfying the tallying conditions given in [SFC15], moreover, in the former case $\mathbf{v}$ 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 $\mathbf{v} = (\mathsf{Dec}(sk, b[1]), \ldots, \mathsf{Dec}(sk, b[nc-1]), 1 - \sum_{j=1}^{nc-1} \mathbf{v}[j])$, with overwhelming probability.

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

$$\{b \mid (b, \mathbf{v}) \in W \wedge \mathbf{v}[v] = 1\} = \{b \mid (b, \mathbf{v}) \in W' \wedge \mathbf{v}[v] = 1\} \tag{10}$$

Let $\{b_1, \ldots, b_\ell\}$ be the largest subset of $\mathfrak{bb}$ satisfying the tallying conditions given in [SFC15]. It follows that:

$$\{b_1, \ldots, b_\ell\} = \{b \mid (b, \mathbf{v}) \in W'\} \tag{11}$$

We distinguish two cases.

- Case I: $1 \leq v \leq nc - 1$. We have for all $(b, \mathbf{v}) \in W'$ that $\mathbf{v}[v] = \mathsf{Dec}(sk, b[v])$. By syntactic equality and (10), it suffices to prove $\mathfrak{b} = \{b \mid (b, \mathbf{v}) \in W' \wedge \mathsf{Dec}(sk, b[v]) = 1\}$. By definition of Helios-Reveal, we have $\mathfrak{b} = \{b_i \mid 1 \leq i \leq \ell \wedge \mathsf{Dec}(sk, b_i[v]) = 1\}$. Hence, we conclude by (11).
- Case II: $v = nc$. We have for all $(b, \mathbf{v}) \in W'$ that $\mathbf{v}[nc] = 1 - \sum_{j=1}^{nc-1} \mathbf{v}[j]$. By syntactic equality and (10), it suffices to prove $\mathfrak{b} = \{b \mid (b, \mathbf{v}) \in W' \wedge 1 - \sum_{j=1}^{nc-1} \mathbf{v}[j] = 1\} = \{b \mid (b, \mathbf{v}) \in W' \wedge \sum_{j=1}^{nc-1} \mathbf{v}[j] = 0\}$. By definition of Helios-Reveal, we have $\mathfrak{b} = \{b_i \mid 1 \leq i \leq \ell \wedge \mathsf{Dec}(sk, b_i[1] \otimes \cdots \otimes b_i[nc-1]) = 0\}$. By (11), it suffices to prove $\bigwedge_{b \in \{b_1, \ldots, b_\ell\}} \mathsf{Dec}(sk, b[1] \otimes \cdots \otimes b[nc-1]) = \sum_{j=1}^{nc-1} \mathbf{v}[j] = \sum_{j=1}^{nc-1} \mathsf{Dec}(sk, b[j])$. We have for all $b \in \{b_1, \ldots, b_\ell\}$ that $b[1], \ldots, b[nc-1]$ are ciphertexts on plaintexts in $\{0,1\}$. Moreover, by definition of Setup, we have $\{0, \ldots, nc-1\} \subseteq \mathfrak{m}$. It follows that $\sum_{j=1}^{nc-1} \mathsf{Dec}(sk, b[j]) \in \mathfrak{m}$. Furthermore, since the encryption scheme is additively homomorphic, we have $\sum_{j=1}^{nc-1} \mathsf{Dec}(sk, b[j]) = \mathsf{Dec}(sk, b[1]) \odot \cdots \odot \mathsf{Dec}(sk, b[nc-1])$, hence, we conclude $\bigwedge_{b \in \{b_1, \ldots, b_\ell\}} \mathsf{Dec}(sk, b[1] \otimes \cdots \otimes b[nc-1]) = \sum_{j=1}^{nc-1} \mathsf{Dec}(sk, b[j])$, with overwhelming probability.

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

Lemma 31. *Non-interactive proof system* (ProveReveal, VerifyReveal) *is zero knowledge.*

Proof sketch. Bernhard *et al.* [BPW12a, §4] remark that (ProveDec, VerifyDec) is zero knowledge. Let $\mathcal{S}$ be the simulator for (ProveDec, VerifyDec). Suppose (ProveReveal, VerifyReveal) does not satisfy zero knowledge, hence, there exists a probabilistic polynomial-time adversary $\mathcal{A}$, such that for all negligible functions negl, there exists a security parameter κ and $\mathsf{Succ}(\mathsf{ZK}((\mathsf{ProveReveal}, \mathsf{VerifyReveal}), \mathcal{A}, \mathcal{H}, \mathcal{S}, \kappa)) > \frac{1}{2} + \mathsf{negl}(\kappa)$. We construct an adversary $\mathcal{B}$ against $(\mathsf{ProveDec}, \mathsf{VerifyDec})$ from $\mathcal{A}$ and $\mathcal{S}$. (For clarity, we rename $\mathcal{B}$'s oracle $\mathcal{Q}$.)

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

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

F.4 Lemmata supporting Theorem 15

Lemma 32. *Relation R(Helios'16, $\mathsf{Helios\text{-}Reveal}$) is Λ-suitable.*

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

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

By definition of $\mathsf{Helios\text{-}Reveal}$, we have that b satisfies the conditions given by the Helios'16 tallying algorithm. Thus, b is a vector of length $2 \cdot n_C - 1$ and $\bigwedge_{j=1}^{n_C-1} \mathsf{VerifyCiph}(pk, b[j], \{0,1\}, b[j + n_C - 1], j) = 1 \wedge \mathsf{VerifyCiph}(pk, b[1] \otimes \cdots \otimes b[n_C - 1], \{0,1\}, b[2 \cdot n_C - 1], n_C) = 1$. In their proof that Helios'16 satisfies universal verifiability, Smyth, Frink & Clarkson [SFC15] show:

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

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

Thus,

3. There exists β, r such that

$$b = \mathsf{Vote}(pk', nc, \beta, \kappa; r)$$

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

4. And

$$\forall j \in \{1, \ldots, nc-1\} \;.\; m_j = 1 \Longleftrightarrow \beta = j \tag{12}$$

$$\sum_{j=1}^{n_C-1} m_j = 0 \Longleftrightarrow \beta = n_C \tag{13}$$

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

By definition of Helios-Reveal, we have either: 1) $\mathsf{Dec}(sk, b[v]) = 1$, hence, $m_v = 1$ by correctness of the encryption scheme, and $\beta = v$ by (12); or 2) $\mathsf{Dec}(sk, b_i[1] \otimes \cdots \otimes b_i[nc-1]) = 0$, hence, $m_1 \odot \cdots \odot m_{nc-1} = 0$, and since $nc - 1 \leq mc$, we have $m_{1,j} \odot \cdots \odot m_{\ell,j} = \sum_{i=1}^{\ell} m_{i,j}$, thus, $\sum_{i=1}^{\ell} m_{i,j} = 0$, and $\beta = v$ by (13). Hence, we conclude Case I.

Case II: $b \in \mathfrak{b}'$. By definition of $\mathfrak{b}'$, there exists r such that $b = \mathsf{Vote}(pk', nc, v, \kappa; r) \in \mathfrak{bb}$. And by correctness of Helios'16, we have b satisfies the conditions given by the Helios'16 tallying algorithm. Moreover, by definition of algorithm Vote, if $1 \leq v < nc$, then there exist coins r such that $b[v] = \mathsf{Enc}(pk, 1; r)$, and by correctness of the encryption scheme, we have $\mathsf{Dec}(sk, b[v]) = 1$, thus, $b \in \mathfrak{b}$. Otherwise ($v = nc$), for $1 \leq j \leq nc - 1$, there exist coins r such that $b[j] = \mathsf{Enc}(pk, 0; r)$, hence, $\mathsf{Dec}(sk, b_i[1] \otimes \cdots \otimes b_i[nc-1]) = 0 \odot \cdots \odot 0 = 0$, thus, $b \in \mathfrak{b}$, concluding Case II, and our proof. □

Lemma 33. *Non-interactive proof system* (ProveReveal, VerifyReveal) *is sound.*

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

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

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

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

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

2. Suppose $1 \leq v < nc$ and for all $1 \leq i \leq \ell$, if $b_i \in \mathfrak{b}$, then $\mathsf{VerifyDec}((pk, b_i[v], 1), \mathbf{Q}[i], \kappa) = 1$, otherwise, $\mathsf{VerifyDec}((pk, b_i[v], 0), \mathbf{Q}[i], \kappa) = 1$. Bernhard *et al.* [BPW12a, §4] remark that (ProveDec, VerifyDec) satisfies their notion of simulation sound extractability, hence, (ProveDec, VerifyDec) satisfies soundness too. Thus, for all $1 \leq i \leq \ell$, if $b_i \in \mathfrak{b}$, then $\mathsf{Dec}(sk, b_i[v]) = 1$, otherwise, $\mathsf{Dec}(sk, b_i[v]) = 0$, with overwhelming probability. It follows that $\mathfrak{b}$ is a subset of $\{b_1, \ldots, b_\ell\}$ such that for every element b in $\mathfrak{b}$ we have $b[v]$ decrypts to 1, and for every element b in $\mathfrak{b} \setminus \{b_1, \ldots, b_\ell\}$ we have $b[v]$ 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 $\mathfrak{b}$ is the largest subset of $\{b_1, \ldots, b_\ell\}$ such that for every element b in $\mathfrak{b}$ we have $b[v]$ decrypts to 1. Thus, $\mathfrak{b}$ is an output of $\mathsf{Helios\text{-}Reveal}(sk, nc, \mathfrak{bb}, v, \kappa)$.

3. Suppose $v = nc$ and for all $1 \leq i \leq \ell$, if $b_i \in \mathfrak{b}$, then $\mathsf{VerifyDec}((pk, b_i[1] \otimes \cdots \otimes b_i[nc-1], 0), \mathbf{Q}[i], \kappa) = 1$, otherwise, $\mathsf{VerifyDec}((pk, b_i[1] \otimes \cdots \otimes b_i[nc-1], 1), \mathbf{Q}[i], \kappa) = 1$. Bernhard *et al.* [BPW12a, §4] remark that (ProveDec, VerifyDec) satisfies their notion of simulation sound extractability, hence, (ProveDec, VerifyDec) satisfies soundness too. Thus, for all $1 \leq i \leq \ell$, if $b_i \in \mathfrak{b}$, then $\mathsf{Dec}(sk, b_i[1] \otimes \cdots \otimes b_i[nc-1]) = 0$, otherwise, $\mathsf{Dec}(sk, b_i[1] \otimes \cdots \otimes b_i[nc-1]) = 1$, with overwhelming probability. It follows that $\mathfrak{b}$ is a subset of $\{b_1, \ldots, b_\ell\}$ such that for every element b in $\mathfrak{b}$ we have $b[1] \otimes \cdots \otimes b[nc-1]$ decrypts to 0, and for every element b in $\mathfrak{b} \setminus \{b_1, \ldots, b_\ell\}$ we have $b[1] \otimes \cdots \otimes b[nc-1]$ 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 $\mathfrak{b}$ is the largest subset of $\{b_1, \ldots, b_\ell\}$ such that for every element b in $\mathfrak{b}$ we have $b[1] \otimes \cdots \otimes b[nc-1]$ decrypts to 0. Thus, $\mathfrak{b}$ is an output of $\mathsf{Helios\text{-}Reveal}(sk, nc, \mathfrak{bb}, v, \kappa)$.

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

References

[Adi06] Ben Adida. *Advances in Cryptographic Voting Systems.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2006.

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

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

[AMPQ09] Ben Adida, Olivier de Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios. In *EVT/WOTE'09: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections.* USENIX Association, 2009.

[AS02a] Masayuki Abe and Koutarou Suzuki. M+1-st price auction using homomorphic encryption. In *PKC'02: 5th International Workshop on Practice and Theory in Public Key Cryptography*, volume 2274 of *LNCS*, pages 115–124. Springer, 2002.

[AS02b] Masayuki Abe and Koutarou Suzuki. Receipt-free sealed-bid auction. In *Information Security*, volume 2433 of *LNCS*, pages 191–199. Springer, 2002.

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

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

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

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

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

[BPW12a] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In *ASIACRYPT'12: 18th International Conference on the Theory and Application of Cryptology and Information Security*, volume 7658 of *LNCS*, pages 626–643. Springer, 2012.

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

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

[Bra10] Felix Brandt. Auctions. In Burton Rosenberg, editor, *Handbook of Financial Cryptography and Security*, pages 49–58. CRC Press, 2010.

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

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

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

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

[CEK+15] Véronique Cortier, Fabienne Eigner, Steve Kremer, Matteo Maffei, and Cyrille Wiedling. Type-Based Verification of Electronic Voting Protocols. In *POST'15: 4th Conference on Principles of Security and Trust*, LNCS, pages 303–323. Springer, 2015.

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

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

[CGGI14] Véronique Cortier, David Galindo, Stephane Glondu, and Malika Izabachène. Election Verifiability for Helios under Weaker Trust Assumptions. In *ESORICS'14: 19th European Symposium on Research in Computer Security*, volume 8713 of *LNCS*, pages 327–344. Springer, 2014.

[CGK+16] Veronique Cortier, David Galindo, Ralf Kuesters, Johannes Mueller, and Tomasz Truderung. Verifiability Notions for E-Voting Protocols. Cryptology ePrint Archive, Report 2016/287 (version 20160317:161048), 2016.

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

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

[CRS05] David Chaum, Peter Y. A. Ryan, and Steve Schneider. A Practical Voter-Verifiable Election Scheme. In *ESORICS'05: 10th European Symposium On Research In Computer Security*, volume 3679 of *LNCS*, pages 118–139. Springer, 2005.

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

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

[Dag07] Participants of the Dagstuhl Conference on Frontiers of E-Voting. *Dagstuhl Accord*, 2007. `http://www.dagstuhlaccord.org/`.

[DJL13] Jannik Dreier, Hugo Jonker, and Pascal Lafourcade. Defining verifiability in e-auction protocols. In *ASIACCS'13: 8th ACM Symposium on Information, Computer and Communications Security*, pages 547–552. ACM Press, 2013.

[DLL13] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. Formal Verification of e-Auction Protocols. In *POST'13: 2nd International Conference on Principles of Security and Trust*, volume 7796 of *LNCS*, pages 247–266. Springer, 2013.

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

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

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

[HIS05] Yong-Sork Her, Kenji Imamoto, and Kouichi Sakurai. Analysis and comparison of cryptographic techniques in e-voting and e-auction. Technical Report 10(2), Information Science and Electrical Engineering, Kyushu University, September 2005.

[JCJ02] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. Cryptology ePrint Archive, Report 2002/165, 2002.

[JCJ10] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. In David Chaum, Markus Jakobsson, Ronald L. Rivest, and Peter Y. A. Ryan, editors, *Towards Trustworthy Elections: New Directions in Electronic Voting*, volume 6000 of *LNCS*, pages 37–63. Springer, 2010.

[JJ00] Markus Jakobsson and Ari Juels. Mix and match: Secure function evaluation via ciphertexts. In *ASIACRYPT'00: 6th International Conference on the Theory and Application of Cryptology and Information Security*, pages 162–177. Springer, 2000.

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

[Kri00] Vijay Krishna. *Auction Theory.* Academic Press, second edition, 2000.

[KRS10] Steve Kremer, Mark D. Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In *ESORICS'10: 15th European Symposium on Research in Computer Security*, volume 6345 of *LNCS*, pages 389–404. Springer, 2010.

[KTV10] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and relationship to verifiability. In *CCS'10: 17th ACM Conference on Computer and Communications Security*, pages 526–535. ACM Press, 2010.

[KTV11] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Verifiability, Privacy, and Coercion-Resistance: New Insights from a Case Study. In *S&P'11: 32nd IEEE Symposium on Security and Privacy*, pages 538–553. IEEE Computer Society, 2011.

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

[LAN02] Helger Lipmaa, N. Asokan, and Valtteri Niemi. Secure Vickrey Auctions without Threshold Trust. In *FC'02: 6th International Conference on Financial Cryptography and Data Security*, volume 2357 of *LNCS*, pages 87–101. Springer, 2002.

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

[MAC02] Emmanouil Magkos, Nikos Alexandris, and Vassilis Chrissikopoulos. A Common Security Model for Conducting e-Auctions and e-Elections. CSCC'02: 6th WSEAS International Multiconference on Circuits, Systems, Communications and Computers `http://www.wseas.us/e-library/conferences/crete2002/papers/444-766.pdf`, 2002.

[MM87] R. Preston McAfee and John McMillan. Auctions and bidding. *Journal of Economic Literature*, 25(2):699–738, 1987.

[MSQ14a] Adam McCarthy, Ben Smyth, and Elizabeth A. Quaglia. Hawk and Aucitas: e-auction schemes from the Helios and Civitas e-voting schemes. In *FC'14: 18th International Conference on Financial Cryptography and Data Security*, volume 8437 of *LNCS*, pages 51–63. Springer, 2014.

[MSQ14b] Adam McCarthy, Ben Smyth, and Elizabeth A. Quaglia. Hawk and Aucitas: e-auction schemes from the Helios and Civitas e-voting schemes. `http://bensmyth.com/publications/2014-Hawk-and-Aucitas-auction-schemes/`, 2014. Long version of [MSQ14a].

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

[Oka96] Tatsuaki Okamoto. An electronic voting scheme. In *Advanced IT Tools: IFIP World Conference on IT Tools*, IFIP Advances in Information and Communication Technology, pages 21–30, 1996.

[OSC90] Document of the Copenhagen Meeting of the Conference on the Human Dimension of the CSCE, 1990.

[PBDV04] Kun Peng, Colin Boyd, Ed Dawson, and Kapalee Viswanathan. Efficient implementation of relative bid privacy in sealed-bid auction. In *Information Security Applications*, volume 2908 of *LNCS*, pages 244–256. Springer, 2004.

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

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

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

[SC11] Ben Smyth and Véronique Cortier. A note on replay attacks that violate privacy in electronic voting schemes. Technical Report RR-7643, INRIA, June 2011. `http://hal.inria.fr/inria-00599182/`.

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

[SFC15] Ben Smyth, Steven Frink, and Michael R. Clarkson. Election Verifiability: Definitions and an Analysis of Helios and JCJ. Cornell's digital repository, `https://ecommons.cornell.edu/handle/1813/39908` and Cryptology ePrint Archive, Report 2015/233, 2015.

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

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

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

[Smy15a] Ben Smyth. Secrecy and independence for election schemes. Cryptology ePrint Archive, Report 2015/942, 2015.

[Smy15b] Ben Smyth. Secrecy and independence for election schemes. Cryptology ePrint Archive, Report 2015/942, 2015.

[UN48] Universal Declaration of Human Rights, 1948.

[US90] Sherman Antitrust Act, 1890.