Secret, verifiable auctions from elections

Elizabeth A. Quaglia[1] and Ben Smyth[2]

[1]Information Security Group, Royal Holloway, University of London
[2]Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg

March 28, 2018

Abstract

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

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

1 Introduction

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

Auctions. An auction is a process for the trade of goods and services from sellers to bidders [Kri00, MM87], with the support of an auctioneer. We study first-price sealed-bid auctions [Bra10], whereby bidders create bids which encapsulate the price they are willing to pay, and the auctioneer opens the bids to determine the winning price (namely, the highest price bid) and winning bidder.

Elections. An election is a decision-making procedure used by voters to choose a representative from some candidates [Gum05, AH10], with the support of a tallier. We study first-past-the-post secret ballot elections [LG84, Saa95], which are defined as follows. First, each voter creates a ballot which encapsulates the voter's chosen candidate (i.e., the voter's vote). Secondly, all ballots are tallied by the tallier to derive the distribution of votes. Finally, the representative – namely, the candidate with the most votes – is announced.

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

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

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

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

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

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

1.1 Constructing auctions from elections

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

1. We represent prices as candidates, and instruct bidders to create bids by "voting" for the candidate that represents the price they are willing to pay.
2. Bids are "tallied" to derive the distribution of prices and the winning price is determined from this distribution.

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

3. The tallier determines the winning bids and a winning bidder can be selected from these bids.[2]

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

1.2 Motivation and related work

There is an abundance of rich research on elections which can be capitalised upon to advance auctions. This statement can be justified with hindsight: Chaum [Cha81] exploited mixnets in elections twenty-three years before Peng *et al.* [PBDV04] made similar advances in auctions (Jakobsson & Juels [JJ00] use mixnets in a distinct way from Chaum and Peng *et al.*), Benaloh & Fischer [CF85] proposed using homomorphic encryption seventeen years before Abe & Suzuki [AS02a], and Okamoto [Oka96] demonstrated the use of trapdoor bit-commitments six years before Abe & Suzuki [AS02b].

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

1.3 Contribution

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

[1] Ballot secrecy does not prohibit such behaviour, because ballot secrecy assumes the tallier is trusted.

[2] Handling tie-breaks (i.e., selecting a winning bid from a set of winning bids) is straightforward. For instance, the set could be sampled uniformly at random. So this paper does not address this problem.

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

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

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

1.4 Comparison with McCarthy, Smyth & Quaglia

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

2 Auction schemes

2.1 Syntax

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

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

Setup*, denoted*[3] $(pk, sk, mb, mp) \leftarrow \mathsf{Setup}(\kappa)$*, is run by the auctioneer,* 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*.*

Bid*, denoted* $b \leftarrow \mathsf{Bid}(pk, np, p, \kappa)$*, is run by bidders.* Bid *takes as input a public key* pk*, an upper-bound* np *on the range of biddable prices,*[4] *a bidder's chosen price* p*, and a security parameter* κ*. A bidder's price should be selected from the range* $1, \dots, np$ *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 and* 0 *otherwise.*

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

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

2.1.1 Example: Enc2Bid

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

Definition 2 (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* $(pk, sk, poly(\kappa), |\mathfrak{m}|)$*.*

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

[4] An upper-bound on the range of biddable prices is sometimes useful for efficiency reasons, as can be observed from our case studies (§7).

- $\mathsf{Bid}(pk, np, p, \kappa)$ *computes* $b \leftarrow \mathsf{Enc}(pk, p)$ *and outputs* b *if* $1 \leq p \leq np \leq |\mathfrak{m}|$ *and* $\perp$ *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* $\mathfrak{m} = \{1, \ldots, |\mathfrak{m}|\}$ *to be the encryption scheme's plaintext space, and algorithm* Open *requires* ϵ *to be a constant symbol.*

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

Lemma 1. *Suppose* Π *is an asymmetric encryption scheme with perfect correctness that satisfies* IND-CPA*. 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.

2.2 Bid secrecy

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

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

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

The adversary has access to a left-right oracle which can compute bids on the adversary's behalf.[6] Bids can be computed by the left-right oracle in two ways, corresponding to a randomly chosen bit 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 (this may contain bids output by the oracle and bids generated by the adversary), which is opened by the challenger to reveal price p, set of winning bids $\mathfrak{b}$, 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 $\textit{correct-price}(pk, np, \mathfrak{bb}, p, \kappa)$, 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.[7]

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

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

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

[7] The existential quantifiers in *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.

of oracle call inputs and outputs.

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

Definition 3 (Bid secrecy)**.** *Let* $\Sigma = (\mathsf{Setup}, \mathsf{Bid}, \mathsf{Open}, \mathsf{Verify})$ *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.*[9]

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

$(pk, sk, mb, mp) \leftarrow \mathsf{Setup}(\kappa);$
$\beta \leftarrow_R \{0, 1\};\ L \leftarrow \emptyset;$
$np \leftarrow \mathcal{A}(pk, \kappa);\ \mathfrak{bb} \leftarrow \mathcal{A}^{\mathcal{O}}();$
$(p, \mathfrak{b}, pf) \leftarrow \mathsf{Open}(sk, np, \mathfrak{bb}, \kappa);$
$g \leftarrow \mathcal{A}(p, \mathfrak{b}, pf);$
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:*[10]

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

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

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

2.3 Auction verifiability

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

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

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

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

[11] We discuss our motivation to base the definitions on the notions of verifiability by Smyth, Frink & Clarkson in Section 6.3.

2.3.1 Individual verifiability

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

Definition 4 (Individual verifiability)**.** *Let* $\Sigma = (\mathsf{Setup}, \mathsf{Bid}, \mathsf{Open}, \mathsf{Verify})$ *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 ($\mathsf{Exp\text{-}IV}$)*, 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 $\mathsf{Exp\text{-}IV}$ 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.[12] 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 be parameterised with a public key constructed by the adversary, rather than one constructed by algorithm Setup (§2.2), we must adopt a stronger definition of injectivity which holds for adversarial keys. We define *strong injectivity* in Appendix B.1.

Definition 5 (Universal verifiability)**.** *Let* $\Sigma = (\mathsf{Setup}, \mathsf{Bid}, \mathsf{Open}, \mathsf{Verify})$ *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.*

[12] Universal verifiability captures a notion of non-repudiation, i.e., anyone can check whether the winning bids encapsulate the winning price.

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

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

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

Definition 6 (Election scheme [SFC16])**.** *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* $1, \ldots, nc$ *of candidates.*[14] Vote *outputs a ballot* b *or error symbol* $\perp$*.*

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*

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

[14] Votes are (abstractly) modelled as integers, rather than alphanumeric strings (such as representatives' names), for brevity.

$\mathfrak{bb}$*, 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*[15] *the number of votes for candidate* 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 and* 0 *otherwise.*

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

3.1.1 Example: Enc2Vote

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

Definition 7 (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.*

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

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

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

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

3.1.2 Comparing auction and election schemes

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

3.2 Reveal algorithm

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

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

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. A 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 that is correct with respect to that election scheme? We answer this question positively in Appendix D.

3.3 Construction

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

3.3.1 Non-verifiable auction schemes

Our first construction follows intuitively from our informal description (§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 9. *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* $(\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk, np, \mathfrak{bb}, \kappa)$. *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 3. *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 10. *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* $(\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk, np, \mathfrak{bb}, \kappa)$. *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* $pf' \leftarrow \mathsf{Prove}((pk, np, \mathfrak{bb}, p, \mathfrak{b}, \kappa), sk, \kappa)$*, 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 it as follows.

Definition 11. *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* $((pk, nc, \mathfrak{bb}, v, \mathfrak{b}, \kappa),\ sk) \in R(\Gamma, \mathsf{Reveal}) \Leftrightarrow \exists mb, mc, r, r' \,.\, \mathfrak{b} = \mathsf{Reveal}(sk, nc, \mathfrak{bb}, v, \kappa; r) \wedge (pk, sk, mb, mc) = \mathsf{Setup}(\kappa; r') \wedge 1 \leq v \leq nc \leq mc \wedge |\mathfrak{bb}| \leq mb$.

Lemma 4. *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).[17]

4.1 Ballot secrecy

Our definition of ballot secrecy strengthens an earlier definition by Smyth [Smy16].[18,19]

Definition 12 (Ballot secrecy)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *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.*

[17] Our privacy results could be extended to other definitions of bid secrecy and ballot secrecy, by modifying our proofs.

[18] We discuss the suitability of Smyth's definition in Section 6.3.

[19] Quaglia & Smyth present a tutorial-style introduction to modelling ballot secrecy [QS17], and Smyth provides a technical introduction [Smy18a].

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

$(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
$\beta \leftarrow_R \{0,1\}; L \leftarrow \emptyset; W \leftarrow \emptyset$;
$nc \leftarrow \mathcal{A}(pk, \kappa); \mathfrak{bb} \leftarrow \mathcal{A}^{\mathcal{O}}()$;
$(\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk, nc, \mathfrak{bb}, \kappa)$;
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 (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 board (intuitively, this captures a setting where the bulletin board is constructed by an adversary that casts ballots on behalf of a subset of voters and controls the distribution of votes cast by the remaining voters). The adversary is also given a mapping, denoted W in Definition 12, from ballots to votes, for all ballots on the bulletin board which were not output by the oracle. To avoid trivial distinctions, we insist that oracle queries are balanced, i.e., predicate *balanced* must hold. Intuitively, if the adversary does not succeed, then ballots for different votes cannot be distinguished, hence, a voter cannot be linked to a vote, i.e., ballot secrecy is preserved. By comparison, if the adversary succeeds, then ballots can be distinguished and ballot secrecy is not preserved.

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

Ballot-Secrecy (with access to W). Moreover, Ballot-Secrecy is strictly stronger (Proposition 5), because the outcome might leak information.[20]

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

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

4.1.1 Example: Enc2Vote satisfies ballot secrecy

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

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

4.2 Relations between ballot and bid secrecy

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

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

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

[20] Parallel decryption leaks information similarly [BPW12b, Appendix A].

[21] 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 6.

price. Hence, Bob is linked to his bid, which violates bid secrecy.) Tallying algorithms that output incorrect winning prices can satisfy correctness, because correctness does not consider an adversary, hence, tallying might produce correct output under ideal conditions and incorrect output in the presence of an adversary. This leads to a separation result (cf. Appendix E). Moreover, our construction is also reliant upon the underlying reveal algorithm, which might output an incorrect set of ballots (in the presence of an adversary), hence, there is a further separation. Nevertheless, we can formulate soundness conditions (that must hold in the presence of adversaries) which capture a class of election schemes for which our intuition holds.

Weak tally soundness. Correctness for election schemes ensures that algorithm Tally produces the expected election outcome under ideal conditions. A similar property, which we call *weak tally soundness*, can hold in the presence of an adversary. Our formulation of weak tally soundness (Definition 13) challenges the adversary to concoct a scenario in which the election outcome does not include the votes of all ballots on the bulletin board that were produced by Vote.[22]

We capture correct election outcomes using function *correct-outcome*,[23] which is defined such that for all pk, nc, $\mathfrak{bb}$, κ, ℓ, and $v \in \{1, \dots, nc\}$, we have $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)$ is a vector of length nc and $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)[v] = \ell \iff \exists^{=\ell} b \in \mathfrak{bb} \setminus \{\perp\} : \exists r : b = \mathsf{Vote}(pk, nc, v, \kappa; r)$. That is, component v of vector $\mathit{correct\text{-}outcome}(pk, \mathfrak{bb}, nc, k)$ equals ℓ iff there exist ℓ ballots on the bulletin board that are votes for candidate v.

Definition 13 (Weak 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* $\mathsf{W\text{-}Tally\text{-}Soundness}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

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

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

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

[23]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* weak tally soundness (W-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* $\mathsf{Succ}(\mathsf{W\text{-}Tally\text{-}Soundness}(\Gamma, \mathcal{A}, \kappa)) \leq \mathsf{negl}(\kappa)$*.*

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 output a vote and bulletin board for which the reveal algorithm produces a set of ballots that does not coincide with the set of ballots on the bulletin board that tally to that vote.

Definition 14 (Reveal soundness)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *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 7. *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 our construction presented in Section 3.3.1 produces auction schemes satisfying bid secrecy, assuming the underlying election scheme satisfies ballot secrecy and weak tally soundness, and the underlying reveal algorithm satisfies reveal soundness.

Proposition 8. *Let* Γ *be an election scheme and* Reveal *be a reveal algorithm. If* Γ *satisfies* Ballot-Secrecy *and* W-Tally-Soundness*, and* Reveal *satisfies reveal soundness with respect to* Γ*, then* $\Lambda(\Gamma, \mathsf{Reveal})$ *satisfies* Bid-Secrecy*.*

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

Example: Enc2Bid **satisfies bid secrecy**

Intuitively, given a non-malleable asymmetric encryption scheme 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 can use our construction and accompanying security results to formally prove this result.

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

Proof. Let us suppose there exists 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 8 to prove that $\mathsf{Enc2Bid}(\Pi)$ satisfies bid secrecy. We defer formalising a suitable reveal algorithm to Appendix C. $\square$

4.2.2 Bid secrecy for verifiable auction schemes

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

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

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

5 Verifiability results

We recall definitions of verifiability by Smyth, Frink & Clarkson [SFC16].[24,25] We show that these definitions are sufficient to ensure that our construction produces schemes satisfying auction verifiability.

5.1 Election verifiability

5.1.1 Individual verifiability

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

[24]We discuss the suitability of the definition by Smyth, Frink & Clarkson in Section 6.3.

[25]Quaglia & Smyth present a tutorial-style introduction to modelling verifiability [QS17], and Smyth provides a technical introduction [Smy18a].

Definition 15 (Individual verifiability [SFC16])**.** *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 ($\mathsf{Exp\text{-}IV\text{-}Ext}$)*, 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 adopt a stronger definition of injectivity which holds for adversarial keys. We define *strong injectivity* in Appendix B.2.

Definition 16 (Universal verifiability [SFC16])**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *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 ($\mathsf{Exp\text{-}UV\text{-}Ext}$)*, 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 weak tally soundness, in particular, both notions challenge the adversary to concoct a scenario in which the election outcome is not correct. The election outcome is computed by the challenger using algorithm Tally in W-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 11. *Let* Γ *be an election scheme. If* Γ *satisfies* Exp-UV-Ext*, then* Γ *satisfies* W-Tally-Soundness*.*

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

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

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

5.2 Election verifiability implies auction verifiability

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

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* Exp-IV-Ext*, then* $\Lambda(\Gamma, \mathsf{Reveal}, \Delta)$ *satisfies* Exp-IV*.*

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

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

Definition 17. *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* $\textit{correct-bids}(pk, np, \mathfrak{bb}, p, \mathfrak{b}, \kappa)$ *with overwhelming probability.*

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

6 Scope of results

Given an election scheme Γ, the scope of key results (Proposition 8 and Theorems 10 & 14) depend on the existence of a reveal algorithm Reveal that is correct with respect to Γ and a non-interactive proof system Δ for relation $R(\Gamma, \mathsf{Reveal})$, such that Δ satisfies soundness and zero-knowledge, Reveal satisfies reveal soundness with respect to Γ, and $R(\Gamma, \mathsf{Reveal})$ is Λ-suitable. We show that suitable reveal algorithms exist for all election schemes in Appendix D. And we demonstrate that suitable non-interactive proof systems exist for a large class of election schemes (§6.1). Furthermore, since our privacy results (Proposition

8 and Theorem 10) depend on a new notion of ballot secrecy, we provide conditions under which the new notion coincides with an existing notion (§6.2).[26] Finally, our results apply to election schemes that satisfy definitions of ballot secrecy (Definition 12) and verifiability (Definitions 15 & 16), and we consider the suitability of these definitions (§6.3).

6.1 Non-interactive proof systems suitable for our construction

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

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

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

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

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

Definition 19 (Honest key generation)**.** *An election scheme* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *ensures* honest key generation*, if for all probabilistic polynomial-time*

[26] Helios satisfies the existing notion, which will be useful in our case study (§7).

adversaries $\mathcal{A}$, there exists a negligible function negl*, such that for all security parameters κ, we have*

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

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

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

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

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

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

6.2 Notions of ballot secrecy

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

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

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

6.3 Suitability of security definitions

6.3.1 Ballot secrecy

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

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

6.3.2 Election verifiability

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

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

[28] Perfect ballot secrecy formalises a privacy notion without trusting the tallier, but it is only known to be satisfied by decentralised voting systems, e.g., [Sch99, KY02, Gro04, HRZ10, KSRH12], which are unsuitable for large-scale elections.

More recently, Juels, Catalano & Jakobsson [JCJ02, JCJ10], Cortier *et al.* [CGGI14] and Kiayias, Zacharias & Zhang [KZZ15] present definitions of election verifiability. Smyth, Frink & Clarkson [SFC16] show that definitions by Juels, Catalano & Jakobsson and Cortier *et al.* do not detect attacks that arise when tallying and verification procedures collude nor when verification procedures reject legitimate outcomes. Moreover, they show that the definition by Kiayias, Zacharias & Zhang does not detect the latter class of attacks. Smyth, Frink & Clarkson propose a definition of election verifiability (§5.1) that detects these attacks.[29] We adopt their definition because it appears to detect the largest class of attacks. Moreover, their definition has proven to be useful in correctly identifying three schemes that do not satisfy verifiability, and in identifying two schemes that do. Furthermore, Helios satisfies their definitions, which will be useful in our case study (§7).

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

7 Case study: Helios

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

[29] Cortier *et al.* [CGK+16, §8.5 & §10.1] claim that the definition by Smyth, Frink & Clarkson is flawed, but that claim is false [SFC16, §9].

[30] `http://www.iacr.org/elections/`, accessed 13 Jul 2017.

[31] `https://heliosvoting.wordpress.com/2009/10/13/helios-deployed-at-princeton/`, accessed 13 Jul 2017.

[32] `https://princeton.heliosvoting.org/`, accessed 13 Jul 2017.

This mode is nonetheless interesting, because the auction scheme we derive is more efficient.

7.1 Tallying by homomorphic combinations

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

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

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

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

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

The original scheme [AMPQ09] is known to be vulnerable to attacks against ballot secrecy and verifiability,[33] and defences against those attacks have been proposed [CS11, SC11, Smy12, CS13, SB13, SB14, Smy16, BPW12a, CE16]. We adopt the formal definition of Helios proposed by Smyth, Frink & Clarkson [SFC16], which adopts non-malleable ballots [SHM15] and uses the Fiat–Shamir transformation with the inclusion of statements in hashes [BPW12a] to defend against those attacks. We recall that formalisation in Appendix F and henceforth refer to it as *Helios'16*.

We derive an auction scheme from Helios'16 using our construction parameterised with a reveal algorithm Helios-Reveal and non-interactive proof system δ(Helios'16).[34] Reveal algorithm Helios-Reveal exploits some technical details of Helios'16 that we have not yet discussed, so we defer the formal description to Appendix G. Our privacy and verifiability results allow us to prove security of the derived auction scheme.

Theorem 19. *Auction scheme Λ(Helios'16, Helios-Reveal, δ(Helios'16)) satisfies* Bid-Secrecy*.*

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

[34] Formally, Λ(Helios'16, Helios-Reveal, δ(Helios'16)) is an auction scheme by Lemmata 4, 15, & 43.

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

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

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

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

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

7.2 Tallying by mixnet

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

Setup as per above.

Vote encrypts the vote, proves correct ciphertext construction,[35] and outputs the ciphertext coupled with the proof.

[35] The algorithm does not prove that the vote is selected from the sequence of candidates, because ciphertexts will be decrypted after mixing, thus, this check can be performed later.

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

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

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

We formalise Helios with tallying by mixnet as the class of election schemes HeliosM'16 (see Appendix H). Given an election scheme Γ from HeliosM'16, we derive auction scheme $\Lambda(\Gamma, \mathsf{HeliosM\text{-}Reveal}, \delta(\Gamma))$ using a reveal algorithm that works as follows:

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

We defer a formal definition to Appendix I.1.

Our privacy and verifiability results allow us to prove security.

Theorem 21. *Given an election scheme $\Gamma \in$ HeliosM'16, auction scheme $\Lambda(\Gamma,$ $\mathsf{HeliosM\text{-}Reveal}, \delta(\Gamma))$ satisfies* Bid-Secrecy*,* Exp-IV*, and* Exp-UV*.*

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

8 Conclusion

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

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

Acknowledgements

This research was conducted in part at École Normale Supérieure and INRIA, with support from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC project *CRYSP* (259639).

A Cryptographic primitives

A.1 Asymmetric encryption

Definition 22 (Asymmetric encryption scheme [KL07])**.** *An* asymmetric encryption scheme *is a tuple of efficient algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *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*[37] *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.*

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 23 (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.*[38]

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

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

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

$(pk, sk) \leftarrow \mathsf{Gen}(\kappa)$;
$\beta \leftarrow_R \{0,1\}$;
$(m_0, m_1) \leftarrow \mathcal{A}(pk, \kappa)$;
$y \leftarrow \mathsf{Enc}(pk, m_\beta)$;
$\mathbf{c} \leftarrow \mathcal{A}(y)$;
$\mathbf{p} \leftarrow (\mathsf{Dec}(sk, \mathbf{c}[1]), \ldots, \mathsf{Dec}(sk, \mathbf{c}[|\mathbf{c}|]))$;
$g \leftarrow \mathcal{A}(\mathbf{p})$;
if $g = \beta \wedge y \notin \boldsymbol{c}$ **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* $\mathsf{Succ}(\textit{IND-PA0}(\Pi, \mathcal{A}, \kappa)) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$.

Definition 24 (Homomorphic encryption [SFC16])**.** *An asymmetric encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is* homomorphic,[39] *with respect to ternary operators $\odot$, $\oplus$, and $\otimes$,*[40] *if there exists a negligible function* negl*, such that for all security parameters κ, the following conditions are satisfied:*[41]

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

We say Π is additively homomorphic*, respectively* multiplicatively 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, respectively multiplication operator, in the group defined by Π's plaintext space and $\odot_{pk}$.*

A.2 Proof systems

Definition 25 (Non-interactive proof system [SFC16])**.** *A* non-interactive proof system *for a relation R is a tuple of algorithms* $(\mathsf{Prove}, \mathsf{Verify})$*, such that:*

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

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

[41] 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.

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

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

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

Definition 26 (Fiat-Shamir transformation [FS87])**.** *Given a sigma protocol* $\Sigma = (\mathsf{Comm}, \mathsf{Chal}, \mathsf{Resp}, \mathsf{Verify}_\Sigma)$ *for relation* R *and a hash function* $\mathcal{H}$*, the* Fiat-Shamir transformation*, denoted* $\mathsf{FS}(\Sigma, \mathcal{H})$*, is the non-interactive proof system* $(\mathsf{Prove}, \mathsf{Verify})$*, defined as follows:*

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

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

Definition 27 (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 28 (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 to a random oracle* $\mathcal{H}$ *and a sigma protocol. Moreover, let* $\mathcal{S}$ *be an algorithm,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{ZK}(\Delta, \mathcal{A}, \mathcal{H}, \mathcal{S}, \kappa)$ *be the following game.*

$\mathsf{ZK}(\Delta, \mathcal{A}, \mathcal{H}, \mathcal{S}, \kappa) =$
  $\beta \leftarrow_R \{0, 1\}$;
  $g \leftarrow \mathcal{A}^{\mathcal{H}, \mathcal{P}}(\kappa)$;
  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 $\mathcal{H}$.[42] *We say Δ satisfies* zero-knowledge*, if there exists a probabilistic polynomial-time algorithm $\mathcal{S}$, such that for all probabilistic polynomial-time algorithm adversaries $\mathcal{A}$, there exists a negligible function* negl*, and for all security parameters κ, we have* $\mathsf{Succ}(\mathsf{ZK}(\Delta, \mathcal{A}, \mathcal{H}, \mathcal{S}, \kappa)) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$*. An algorithm $\mathcal{S}$ for which zero-knowledge holds is called a* simulator *for* $(\mathsf{Prove}, \mathsf{Verify})$*.*

Definition 29 (Simulation sound extractability [SFC16, BPW12a, Gro06])**.** *Suppose Σ is a sigma protocol for relation R, $\mathcal{H}$ is a random oracle, and* $(\mathsf{Prove}, \mathsf{Verify})$ *is a non-interactive proof system, such that* $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{Prove}, \mathsf{Verify})$*. Further suppose $\mathcal{S}$ is a simulator for* $(\mathsf{Prove}, \mathsf{Verify})$ *and $\mathcal{H}$ can be patched by $\mathcal{S}$. Proof system* $(\mathsf{Prove}, \mathsf{Verify})$ *satisfies* simulation sound extractability *if there exists a probabilistic polynomial-time algorithm $\mathcal{K}$, such that for all probabilistic polynomial-time adversaries $\mathcal{A}$ and coins r, there exists a negligible function* negl*, such that for all security parameters κ, we have:*[43]

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

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

- *$\mathcal{A}'()$. Computes $\mathbf{Q}' \leftarrow \mathcal{A}(-; r)$, forwarding any of $\mathcal{A}$'s oracle queries to $\mathcal{K}$, and outputs $\mathbf{Q}'$. By running $\mathcal{A}(-; r)$, $\mathcal{K}$ is rewinding the adversary.*
- *$\mathcal{P}(s)$. Computes $\sigma \leftarrow \mathcal{S}(s, \kappa); \mathbf{P} \leftarrow (\mathbf{P}[1], \dots, \mathbf{P}[|\mathbf{P}|], (s, \sigma))$ and outputs σ.*

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

Theorem 22 ([BPW12a])**.** *Let Σ be a sigma protocol for relation R, and let $\mathcal{H}$ be a random oracle. Suppose Σ satisfies special soundness and special honest verifier zero-knowledge. Non-interactive proof system* $\mathsf{FS}(\Sigma, \mathcal{H})$ *satisfies zero-knowledge and simulation sound extractability.*

The Fiat-Shamir transformation can be generalised to include an optional string m in the hashes produced by functions Prove and Verify. We write $\mathsf{Prove}(s, w, m, \kappa)$ and $\mathsf{Verify}(s, (\mathsf{comm}, \mathsf{resp}), m, k)$ for invocations of Prove and Verify which include an optional string. When m is provided, it is included in the hashes in both algorithms. That is, given $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{Prove}, \mathsf{Verify})$, the hashes are computed as follows in both algorithms: $\mathsf{chal} \leftarrow \mathcal{H}(\mathsf{comm}, s, m)$. Simulators can be generalised to include an optional string m too. We write $\mathcal{S}(s, m, \kappa)$ for invocations of simulator $\mathcal{S}$ which include an optional string. Theorem 22 can be extended to this generalisation.

[42] Random oracles can be *programmed* or *patched*. We will not need the details of how patching works, so we omit them here; see Bernhard et al. [BPW12a] for a formalisation.

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

B Correctness, completeness and injectivity

B.1 Definitions for auctions

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 30 (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* [SFC16], which arise when algorithm Verify rejects legitimate outcomes, possibly due to presence of a bid on the bulletin board that was not produced by algorithm Bid.

Definition 31 (Completeness)**.** *There exists a negligible function* negl*, such that for all security parameters* κ*, bulletin boards* $\mathfrak{bb}$*, and integers* np*, we have*

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

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

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

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

Definition 33 (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* $\Pr[b \leftarrow \mathsf{Bid}(pk, np, p, \kappa); b' \leftarrow \mathsf{Bid}(pk, np, p', \kappa) : b \neq \perp \;\wedge\; b' \neq \perp \Rightarrow b \neq b'] = 1$.

B.2 Definitions for elections

Definition 34 (Correctness [SFC16])**.** *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*

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

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

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

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

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

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

Definition 37 (Strong injectivity [SFC16])**.** *An election scheme* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *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*

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

C Proofs

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

Fact 23. *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 24. *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 Π satisfies $\mathsf{IND\text{-}CPA}$, we have $b_1, \dots, b_{nb}$ are pairwise distinct with overwhelming probability. Moreover, 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 23.

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

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

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 $\mathsf{Enc2Vote}(\Pi) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ and $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Algorithm Verify always outputs 1, hence, it follows from Fact 24 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 $nb \leq mb \wedge nc \leq mc$ and for each $1 \leq i \leq nb$ we have b_i is an output of $\mathsf{Vote}(pk, nc, v_i, \kappa)$. Moreover, for each $1 \leq i \leq nb$ 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 Π satisfies $\mathsf{IND\text{-}CPA}$, we have $b_1, \ldots, b_{nb}$ are pairwise distinct with overwhelming probability. Moreover, 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.3 Proof of Lemma 3

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 23 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 ≤ i ≤ nb do
    v[p_i] ← 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.4 Proof of Lemma 4

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.3), 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 $1 \leq p \leq np \leq mp$ and $|\mathfrak{bb}| \leq mb$, 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. $\square$

C.5 Proof of Proposition 5

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

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

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

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

$\mathsf{Tally}_\chi(sk', nc, \mathfrak{bb}, \kappa)$. *Parses* sk' *as* (sk, k)*, computes*

$(\mathbf{v}, pf) \leftarrow \mathsf{Tally}_\Gamma(sk, nc, \mathfrak{bb}, \kappa)$;
if $\mathfrak{bb} = \{\epsilon\}$ **then**
| $\mathbf{v}[0] = sk \oplus k$
else if $\mathfrak{bb} = \{\epsilon'\}$ **then**
⌊ $\mathbf{v}[0] = k$

and outputs $(\mathbf{v}, pf)$*.*

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

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

Proof sketch. Since constant symbols ϵ and ϵ' do not appear in the codomain of Vote_Γ, correctness of $\chi(\Gamma, \epsilon, \epsilon')$ follows from correctness of Γ. Moreover, since algorithm Verify_χ always outputs 1, we have completeness of $\chi(\Gamma, \epsilon, \epsilon')$ by Fact 24. Furthermore, injectivity of $\chi(\Gamma, \epsilon, \epsilon')$ trivially follows from injectivity of Γ. $\square$

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

C.6 Proof of Proposition 6

Let BS0, respectively BS1, be the game derived from Ballot-Secrecy by replacing $\beta \leftarrow_R \{0,1\}$ with $\beta \leftarrow 0$, respectively $\beta \leftarrow 1$. These games are trivially related to Ballot-Secrecy, namely, $\mathsf{Succ}(\mathsf{Ballot\text{-}Secrecy}(\Gamma, \mathcal{A}, \kappa)) = \frac{1}{2} \cdot \mathsf{Succ}(\mathsf{BS0}(\Gamma, \mathcal{A}, \kappa)) + \frac{1}{2} \cdot \mathsf{Succ}(\mathsf{BS1}(\Gamma, \mathcal{A}, \kappa))$. Moreover, let BS1:0 be the game derived from BS1 by replacing $g = \beta$ with $g = 0$. We relate game BS1 to BS1:0, and we relate games BS0 and BS1:0 to the hybrid games $\mathsf{G}_0, \mathsf{G}_1, \ldots$ introduced in Definition 39. We use these relations to prove Proposition 6.

Lemma 26. *Let* Π *be an asymmetric encryption scheme satisfying* IND-CPA *and let* $\Gamma = \mathsf{Enc2Vote}(\Pi)$*. If a probabilistic polynomial-time adversary* $\mathcal{A}$ *wins game* Ballot-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 39. *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.*

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

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

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

if $|L| < i$ **then**
  $b \leftarrow \mathsf{Vote}(pk, nc, v_1, \kappa);$
else
  $b \leftarrow \mathsf{Vote}(pk, nc, v_0, \kappa);$
$L \leftarrow L \cup \{(b, v_0, v_1)\};$
return b;

Fact 27. *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*

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

Proof of Fact 27. The proof follows from Definition 7. □

Lemma 28. *Let* Π *be an asymmetric encryption scheme satisfying* IND-CPA *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 public keys using the same algorithm and provide those keys, along with the security parameter, as input to the first adversary call, thus, these inputs and corresponding outputs are equivalent.

Left-right oracle calls $\mathcal{O}(v_0, v_1)$ 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 $\{b \mid b \in \mathfrak{bb} \wedge (b, v_0, v_1) \notin L\}$, 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 $\{b \mid b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L\}$). By Fact 27, it suffices to prove that the simulations are valid, i.e., in G_0 and G_q, computing

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**
    $v \leftarrow \mathsf{Dec}(sk, b)$;
    if $1 \leq v \leq nc$ **then**
        $\mathbf{v}[v] \leftarrow \mathbf{v}[v] + 1$

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

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

By definition of 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 26, 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 28, 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

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

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,

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

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 $1 \leq j \leq k$ **do**
      $\mathbf{v}' \leftarrow (0, \ldots, 0)$; `// vector of length` nc
      if $1 \leq \boldsymbol{p}[j] \leq nc$ **then**
          $\mathbf{v}[\mathbf{p}[j]] \leftarrow \mathbf{v}[\mathbf{p}[j]] + 1$;
          $\mathbf{v}'[\mathbf{p}[j]] \leftarrow 1$;
      $W \leftarrow W \cup \{(b_j, \mathbf{v}')\}$;
  for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ **do**
      $\mathbf{v}[v_0] \leftarrow \mathbf{v}[v_0] + 1$;
  $g \leftarrow \mathcal{A}(\mathbf{v}, \epsilon, W)$;

  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 $\mathbf{c} = (b_1, \ldots, b_k)$ such that

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

where $\mathfrak{bb}$ is $\mathcal{A}$'s output. Let $\mathbf{p} \leftarrow (\mathsf{Dec}(sk, \mathbf{c}[1]), \ldots, \mathsf{Dec}(sk, \mathbf{c}[|\mathbf{c}|]))$. 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 29. *Computing W as*

$W \leftarrow \emptyset$;
for $1 \leq j \leq k$ **do**
  $\mathbf{v} \leftarrow (0, \ldots, 0)$; `// vector of length` nc
  if $1 \leq \boldsymbol{p}[j] \leq nc$ **then**
    $\mathbf{v}[\mathbf{p}[j]] \leftarrow 1$;
  $W \leftarrow W \cup \{(b_j, \mathbf{v})\}$;

is equivalent to computing W as

$W \leftarrow \emptyset$;
for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \notin L$ **do**
  $(\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk, nc, \{b\}, \kappa)$;
  $W \leftarrow W \cup \{(b, \mathbf{v})\}$;

Claim 30. *Computing $\mathbf{v}$ as*

$\mathbf{v} \leftarrow (0, \ldots, 0)$; `// vector of length` nc
for $1 \leq j \leq k$ **do**
  if $1 \leq \boldsymbol{p}[j] \leq nc$ **then**
    $\mathbf{v}[\mathbf{p}[j]] \leftarrow \mathbf{v}[\mathbf{p}[j]] + 1$;

is equivalent to computing $\mathbf{v}$ 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**
  $(\mathbf{v}', pf) \leftarrow \mathsf{Tally}(sk, nc, \{b\}, \kappa)$;
  $\mathbf{v} \leftarrow \mathbf{v} + \mathbf{v}'$;

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

By Claims 29 & 30, we have either:

- $\beta = 0$ and $\mathcal{B}(\mathbf{p})$ 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 29 & 30.

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

$W \leftarrow \emptyset$;
for $1 \leq j \leq k$ **do**
  $\mathbf{v} \leftarrow (0, \ldots, 0)$; `// vector of length` nc
  if $1 \leq \mathsf{Dec}(sk, b_j) \leq nc$ **then**
    $\mathbf{v}[\mathsf{Dec}(sk, b_j)] \leftarrow 1$;
  $W \leftarrow W \cup \{(b_j, \mathbf{v})\}$;

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

$W \leftarrow \emptyset$;
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**
    $\mathbf{v}[\mathsf{Dec}(sk, b)] \leftarrow 1$
  $W \leftarrow W \cup \{(b, \mathbf{v})\}$;

Hence, we conclude by (1).

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

$\mathbf{v} \leftarrow (0, \ldots, 0)$; `// vector of length` nc
for $1 \leq j \leq k$ **do**
  if $1 \leq \mathsf{Dec}(sk, b_j) \leq nc$ **then**
    $\mathbf{v}[\mathsf{Dec}(sk, b_j)] \leftarrow \mathbf{v}[\mathsf{Dec}(sk, b_j)] + 1$;

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.7 Proof of Lemma 7

Let $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$. 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). 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 $(nc, \{b_1, \ldots, b_{nb}\}, v)$.

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.8 Proof of Proposition 8

Let $\Gamma = (\mathsf{Setup}_\Gamma, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify}_\Gamma)$, $\Sigma = \Lambda(\Gamma, \mathsf{Reveal}) = (\mathsf{Setup}_\Sigma, \mathsf{Bid}, \mathsf{Open}, \mathsf{Verify}_\Sigma)$, 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 $\mathit{balanced}(\mathfrak{bb}, np, L)$, 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, we have $\mathbf{v}'[p] \geq \mathit{correct\text{-}outcome}(pk, np, \{b, b'\}, \kappa)[p]$ by weak tally soundness, and we also have $\mathbf{v}'[p_0] \geq \mathit{correct\text{-}outcome}(pk, np, \{b, b'\}, \kappa)[p_0] \vee \mathbf{v}'[p_1] \geq \mathit{correct\text{-}outcome}(pk, np, \{b, b'\}, \kappa)[p_1]$. Thus, by definition of *correct-outcome*, we have

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

   It follows that

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

   Since Γ satisfies weak 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:

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

   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 $\mathsf{Bid\text{-}Secrecy}$ 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 $\mathcal{B}$ determines β correctly with the same success as $\mathcal{A}$ with overwhelming probability, hence, $\mathcal{B}$ wins $\mathsf{Ballot\text{-}Secrecy}(\Gamma, \mathcal{B}, \kappa)$ with overwhelming probability, thereby deriving a contradiction and concluding our proof. □

C.9 Proof of Proposition 9

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

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

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

We prove $\mathsf{Enc2Vote}(\Pi)$ satisfies weak tally soundness by contradiction. Suppose κ is a security parameter and $\mathsf{Setup}(\kappa)$ 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 $\mathsf{Tally}(sk, nc, \mathfrak{bb}, \kappa)$ 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 election scheme $\mathsf{Enc2Vote}(\Pi)$. 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 Reveal-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 $1 \le v' \le nc$ **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.10 Proof of Theorem 10

Let $\Sigma = \Lambda(\Gamma, \mathsf{Reveal})$ and $\Sigma' = \Lambda(\Gamma, \mathsf{Reveal}, \Delta)$. By Proposition 8, 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 $\mathcal{A}$ determines 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.11 Proof of Lemma 11

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 $\mathsf{Succ}(\mathsf{W\text{-}Tally\text{-}Soundness}(\Gamma, \mathcal{A}, \kappa)) > \mathsf{negl}(\kappa)$. 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 $\mathsf{W\text{-}Tally\text{-}Soundness}(\Gamma, \mathcal{A}, \kappa)$, 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.12 Proof of Theorem 14

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 10, 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.,

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

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 $\mathbf{v}[p] > 0 \ \wedge\ 1 \le p \le np$, or there is no such integer and $p = 0$. By definition of ψ and by inspection of Verify_Σ, we have:

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

Let us assume the following:

$$\begin{aligned}&\psi(\mathbf{v}, p, np) \wedge \neg \mathit{correct\text{-}price}(pk, np, \mathfrak{bb}, p, \kappa)\\ &\qquad\qquad \Rightarrow \mathbf{v} \neq \mathit{correct\text{-}outcome}(pk, np, \mathfrak{bb}, \kappa)\end{aligned} \tag{7}$$

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:

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

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:

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

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:

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

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

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

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. □

C.13 Proof of Lemma 15

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

By completeness of election scheme Γ, we have $\mathsf{Verify}_\Gamma(pk, nc, \mathfrak{bb}, \mathbf{v}_{\mathfrak{bb}}, pf_{\mathfrak{bb}}, \kappa) = 1$ and $\mathsf{Verify}_\Gamma(pk, nc, \mathfrak{b}, \mathbf{v}_{\mathfrak{b}}, pf_{\mathfrak{b}}, \kappa) = 1$, with overwhelming probability. And, since Γ satisfies universal verifiability, we have $\mathbf{v}_{\mathfrak{bb}} = \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)$ and $\mathbf{v}_{\mathfrak{b}} = \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{b}, \kappa)$, with overwhelming probability.

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

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

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

C.14 Proof of Lemma 16

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

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

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

$$\exists^{=|\mathfrak{b}|} b \in \mathfrak{b} : \exists! r : b = \mathsf{Vote}(pk, nc, v, \kappa; r) \tag{10}$$

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

$$\exists r \,.\, \mathfrak{b} = \mathsf{Reveal}(sk, nc, \mathfrak{b}, v, \kappa; r) \tag{11}$$

with overwhelming probability.

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

$$\exists r \,.\, \mathfrak{b} = \mathsf{Reveal}(sk, nc, \mathfrak{bb} \setminus \{\perp\}, v, \kappa; r) \tag{12}$$

with overwhelming probability.

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

C.15 Proof of Lemma 17

Let $\Delta = (\mathsf{Prove}, \mathsf{Verify})$ and $\delta(\Gamma) = (\mathsf{Prove}_\delta, \mathsf{Verify}_\delta)$. Moreover, let $\mathcal{S}$ be the simulator for Δ. We derive a simulator $\mathcal{S}_\delta$ from Prove_δ by replacing

$$(\mathbf{v}_{\mathfrak{bb}}, \mathit{pf}_{\mathfrak{bb}}) \leftarrow \mathsf{Tally}(sk, nc, \mathfrak{bb}, \kappa); (\mathbf{v}_{\mathfrak{b}}, \mathit{pf}_{\mathfrak{b}}) \leftarrow \mathsf{Tally}(sk, nc, \mathfrak{b}, \kappa)$$

with

$$\begin{aligned}(\mathbf{v}_{\mathfrak{bb}}, \mathit{pf}_{\mathfrak{bb}}) \leftarrow \mathsf{Tally}(sk, nc, \mathfrak{bb}, \kappa); (\mathbf{v}_{\mathfrak{b}}, \mathit{pf}_{\mathfrak{b}}) \leftarrow \mathsf{Tally}(sk, nc, \mathfrak{b}, \kappa);\\ \mathit{pf}_{\mathfrak{bb}} \leftarrow \mathcal{S}((pk, nc, \mathfrak{bb}, \mathbf{v}_{\mathfrak{bb}}), \kappa); \mathit{pf}_{\mathfrak{b}} \leftarrow \mathcal{S}((pk, nc, \mathfrak{b}, \mathbf{v}_{\mathfrak{b}}), \kappa)\end{aligned}$$

Suppose $\delta(\Gamma)$ 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 $\mathsf{Succ}(\mathsf{ZK}(\delta(\Gamma), \mathcal{A}, \mathcal{H}, \mathcal{S}_\delta, \kappa)) > \frac{1}{2} + \mathsf{negl}(\kappa)$. We construct an adversary $\mathcal{B}$ against Δ from $\mathcal{A}$, $\mathcal{S}_\delta$, 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 $\mathcal{S}_\delta$ by replacing $\mathcal{S}$ with $\mathcal{Q}$.

We prove the following contradiction: $\mathsf{Succ}(\mathsf{ZK}(\Delta, \mathcal{B}, \mathcal{H}, \mathcal{S}, \kappa)) > \frac{1}{2} + \mathsf{negl}'(\kappa)$, for some negligible function negl'. It suffices to show that adversary $\mathcal{B}$ simulates $\mathcal{A}$'s oracle $\mathcal{P}$ to $\mathcal{A}$.

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

- Case I: $\beta = 0$. Adversary $\mathcal{A}$ expects $\mathcal{P}(s, sk)$ to compute $\sigma \leftarrow \mathsf{Prove}_\delta(s, sk, \kappa)$ and return σ. By definition of Prove_δ, we know σ will be a vector $(pf_{\mathfrak{bb}}, pf_{\mathfrak{b}})$ such that $(\mathbf{v}_{\mathfrak{bb}}, pf_{\mathfrak{bb}})$ is an output of $\mathsf{Tally}(sk, nc, \mathfrak{bb}, \kappa)$ for some $\mathbf{v}_{\mathfrak{bb}}$ and $(\mathbf{v}_{\mathfrak{b}}, pf_{\mathfrak{b}})$ is an output of $\mathsf{Tally}(sk, nc, \mathfrak{b}, \kappa)$ for some $\mathbf{v}_{\mathfrak{b}}$. Since there exists a tallying proof system for Helios'16 that satisfies zero-knowledge, we have $pf_{\mathfrak{bb}}$ is an output of $\mathsf{Prove}((pk, nc, \mathfrak{bb}, \mathbf{v}_{\mathfrak{bb}}), sk, \kappa; r_{\mathfrak{bb}})$, for some coins $r_{\mathfrak{bb}}$ chosen uniformly at random. Similarly, we have $pf_{\mathfrak{b}}$ is an output of $\mathsf{Prove}((pk, nc, \mathfrak{bb}, \mathbf{v}_{\mathfrak{b}}), sk, \kappa; r_{\mathfrak{b}})$, for some coins $r_{\mathfrak{b}}$ chosen uniformly at random. Thus, computing $\sigma \leftarrow \mathsf{Prove}_\delta(s, sk, \kappa)$ and returning σ, is equivalent to computing $\sigma \leftarrow \mathcal{Q}'(s, w, \kappa)$ and returning σ, because the only distinction between Q' and Prove_δ is resampling Prove.
- Case II: $\beta = 1$. Adversary $\mathcal{A}$ expects $\mathcal{P}(s, sk)$ to compute $\sigma \leftarrow \mathcal{S}_\delta(s, \kappa)$ and return σ, which is trivially equivalent to computing $\sigma \leftarrow \mathcal{Q}'(s, w, \kappa)$ and returning σ, because $\mathcal{Q}'$ and S_δ are identical in this case.

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

C.16 Proof of Proposition 18

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

- $\mathcal{B}(pk, \kappa)$ computes $nc \leftarrow \mathcal{A}(pk, \kappa)$, initialises $\mathbf{H}$ as a transcript of the random oracle's input and output, computes $sk \leftarrow \mathcal{K}(\mathbf{H}, pk)$, and outputs nc.
- $\mathcal{B}()$ initialises $L \leftarrow \emptyset$, computes $\mathfrak{bb} \leftarrow \mathcal{A}()$, and outputs $\mathfrak{bb}$. Any oracle calls from $\mathcal{A}$ on inputs (v_0, v_1) are forwarded to $\mathcal{B}$'s oracle and a transcript of calls is maintained, i.e., $\mathcal{B}$ computes $b \leftarrow \mathcal{O}(v_0, v_1); L \leftarrow L \cup \{(b, v_0, v_1)\}$ and returns b to $\mathcal{A}$.

- $\mathcal{B}(\mathbf{v}, pf)$ computes

  $W \leftarrow \emptyset$;
  for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \notin L$ **do**
      $(\mathbf{v}', pf') \leftarrow \mathsf{Tally}(sk, nc, \{b\}, \kappa)$;
      $W \leftarrow W \cup \{(b, \mathbf{v}')\}$;
  $g \leftarrow \mathcal{A}(\mathbf{v}, pf, W)$;

  and outputs g.

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

D Reveal algorithms exist

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

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

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

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

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

Proposition 32. *Given an election scheme* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$*, we have* $\rho(\Gamma)$ *is correct with respect to* Γ*.*

Proof. Let $\rho(\Gamma) = \mathsf{Reveal}$. 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. □

Given an election scheme Γ, the scope of Proposition 8 & Theorem 10 depends on the existence of a reveal algorithm Reveal satisfying reveal soundness with respect to Γ. Moreover, the scope of Theorem 14 depends on relation $R(\Gamma, \mathsf{Reveal})$ being Λ-suitable. We show that ρ constructs suitable reveal algorithms.

Lemma 33. *Given an election scheme* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$*, we have* $\rho(\Gamma)$ *satisfies reveal soundness with respect to* Γ*.*

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

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

Proof. Let reveal algorithm $\rho(\Gamma) = \mathsf{Reveal}$. Suppose $((pk, nc, \mathfrak{bb}, v, \mathfrak{b}, \kappa), sk) \in R(\Gamma, \mathsf{Reveal})$, i.e., there exist mb, mc, r and r' such that $(pk, sk, mb, mc) = \mathsf{Setup}(\kappa; r')$, $\mathfrak{b} = \mathsf{Reveal}(sk, nc, \mathfrak{bb}, v, \kappa; r)$, $1 \leq v \leq nc \leq mc$, and $|\mathfrak{bb}| \leq mb$. Let $\mathfrak{b}' = \mathfrak{bb} \cap \{b \mid b = \mathsf{Vote}(pk', nc, v, \kappa; r'')\}$. To prove relation $R(\Gamma, \mathsf{Reveal})$ 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 Reveal, we have $b \in \mathfrak{bb}$ and $\mathsf{Tally}(sk, nc, \{b\}, \kappa)$ outputs a vector $\mathbf{v}$ such that $\mathbf{v}[v] = 1$. Moreover, we have $\mathsf{Verify}(pk, nc, \{b\}, \mathbf{v}, pf, \kappa) = 1$, by perfect completeness. And we have $\mathbf{v}[v] = \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)[v]$, by universal verifiability. It follows by definition of *correct-outcome* that $\exists^{=1} b \in \mathfrak{bb} \setminus \{\perp\} : \exists r : b = \mathsf{Vote}(pk, nc, v, \kappa; r)$, hence, $\exists r : b = \mathsf{Vote}(pk, nc, v, \kappa; r)$. Thus, $b \in \mathfrak{b}'$, concluding the proof of Case I.

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

E Separation result

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

Proposition 35. *There exists a function* $\mathsf{incorrect\text{-}price}$*, 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* $\Lambda(\mathsf{incorrect\text{-}price}(\Gamma), \mathsf{Reveal})$ *does not satisfy bid secrecy, for some reveal algorithm* Reveal *that is correct with respect to* $\mathsf{incorrect\text{-}price}(\Gamma)$.

Definition 41. *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 36. *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 24. And Injectivity follows from the underlying scheme, because we do not modify Setup nor Vote. □

Lemma 37. *Given an election scheme* Γ *satisfying ballot secrecy, we have that* $\mathsf{incorrect\text{-}price}(\Gamma)$ *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. $\square$

Proof of Proposition 35. Suppose Γ is an election scheme satisfying ballot secrecy. By Lemmata 36 & 37, we have $\mathsf{incorrect\text{-}price}(\Gamma)$ is an election scheme satisfying ballot secrecy. And, by Proposition 32, there exists a reveal algorithm Reveal that is correct with respect to $\mathsf{incorrect\text{-}price}(\Gamma)$. By Lemma 3, 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 $\mathit{balanced}(\mathfrak{bb}, np, L)$. And

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

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

F Helios with tallying by homomorphic combinations

Smyth, Frink & Clarkson [SFC16, SFC17] formalise a generic construction for Helios-like election schemes (Definition 43), which is instantiated on the choice of a homomorphic encryption scheme and sigma protocols for the relations introduced in the following definition.[44]

Definition 42 ([SFC17])**.** *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a homomorphic asymmetric encryption scheme and* Σ *be a sigma protocol for a binary relation* R.[45]

- Σ proves correct key construction *if a* $((\kappa, pk, \mathfrak{m}), (sk, s)) \in R \Leftrightarrow (pk, sk) = \mathsf{Gen}(\kappa; s)$ *and* $\mathfrak{m}$ *is the encryption scheme's plaintext space.*

Suppose $(pk, sk) = \mathsf{Gen}(\kappa; s)$*, for some security parameter* κ *and coins* s*, and* $\mathfrak{m}$ *is the encryption scheme's plaintext space.*

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

- Σ proves correct decryption *if* $((pk, c, m), sk) \in R \Leftrightarrow m = \mathsf{Dec}(sk, c)$.

Definition 43 (Generalised Helios [SFC17])**.** *Suppose* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is an additively homomorphic asymmetric encryption scheme,* Σ_1 *is a sigma protocol that proves correct key construction,* Σ_2 *is a sigma protocol that proves plaintext knowledge in a subspace,* Σ_3 *is a sigma protocol that proves correct decryption, and* $\mathcal{H}$ *is a hash function. Let* $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$, $\mathsf{FS}(\Sigma_2, \mathcal{H}) = (\mathsf{ProveCiph}, \mathsf{VerifyCiph})$*, and* $\mathsf{FS}(\Sigma_3, \mathcal{H}) = (\mathsf{ProveDec}, \mathsf{VerifyDec})$*. We define election scheme* generalised Helios*, denoted* $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$*, as follows.*

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

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

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

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

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

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

$\mathsf{Tally}(sk', nc, \mathfrak{bb}, \kappa)$**.** *Initialise vectors* $\mathbf{v}$ *of length* nc *and* pf *of length* $nc-1$. *Compute* **for** $1 \leq j \leq nc$ **do** $\mathbf{v}[j] \leftarrow 0$. *Parse* sk' *as a vector* (pk, sk). *Output* $(\mathbf{v}, pf)$ *if parsing fails. Let* $\{b_1, \ldots, b_\ell\}$ *be the largest subset of* $\mathfrak{bb}$ *such that* $b_1 < \cdots < b_\ell$ *and for all* $1 \leq i \leq \ell$ *we have* b_i *is a vector of length* $2 \cdot nc - 1$ *and* $\bigwedge_{j=1}^{nc-1} \mathsf{VerifyCiph}((pk, b_i[j], \{0,1\}), (b_i[j+nc-1]), j, \kappa) = 1 \wedge \mathsf{VerifyCiph}((pk, b_i[1] \otimes \cdots \otimes b_i[nc-1], \{0,1\}), (b_i[2 \cdot nc-1]), nc, \kappa) = 1$. *If* $\{b_1, \ldots, b_\ell\} = \emptyset$, *then output* $(\mathbf{v}, pf)$, *otherwise, compute:*

$\quad$ **for** $1 \leq j \leq nc-1$ **do**
$\quad\quad$ $c \leftarrow b_1[j] \otimes \cdots \otimes b_\ell[j]$;
$\quad\quad$ $\mathbf{v}[j] \leftarrow \mathsf{Dec}(sk, c)$;
$\quad\quad$ $pf[j] \leftarrow \mathsf{ProveDec}((pk, c, \mathbf{v}[j]), sk, \kappa)$;
$\quad$ $\mathbf{v}[nc] \leftarrow \ell - \sum_{j=1}^{nc-1} \mathbf{v}[j]$;

Output $(\mathbf{v}, pf)$.

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

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

Lemma 38. *Suppose* Π, Σ_1, Σ_2, Σ_3 *and* $\mathcal{H}$ *satisfy the preconditions of Definition 43. Further suppose* Π *satisfies perfect correctness and is perfectly homomorphic. Moreover, suppose* Σ_1 *and* Σ_2 *satisfy perfect completeness. We have* $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ *satisfies perfect correctness.*

Proof sketch. Smyth, Frink & Clarkson shown that $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ satisfies correctness. And their proof can be adapted to show perfect correctness. In particular, perfect completeness of Σ_1 ensures that algorithm Vote does not output $\perp$, perfect completeness of Σ_2 ensures that algorithm Tally considers $\mathfrak{bb}$ as the largest subset of $\mathfrak{bb}$ satisfying the tallying conditions, and perfect correctness of Π ensures that the outcome represents the votes. Moreover, since Π is perfectly homomorphic, homomorphic combinations are valid. $\square$

Lemma 39. *Suppose Π, Σ_1, Σ_2, Σ_3 and $\mathcal{H}$ satisfy the preconditions of Definition 43. Further Σ_1, Σ_2 and Σ_3 satisfy perfect completeness, moreover, Σ_2 perfectly satisfies special soundness and special honest verifier zero-knowledge, and $\mathcal{H}$ is a random oracle. We have* $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ *satisfies perfect completeness.*

Proof sketch. Smyth, Frink & Clarkson shown that $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ satisfies completeness. And their proof can be adapted to show perfect completeness (assuming the proof of Theorem 22 can be adapted to show $\mathsf{FS}(\Sigma_2, \mathcal{H})$ satisfies perfect simulation sound extractability), when Σ_1 and Σ_3 are perfectly complete. $\square$

Instantiations of generalised Helios work as follows [SFC16].

- Setup generates the tallier's key pair. The public key includes a non-interactive proof demonstrating that the key pair is correctly constructed.
- Vote takes a vote $v \in \{1, \ldots, nc\}$ and outputs ciphertexts $c_1, \ldots, c_{nc-1}$ such that if $v < nc$, then ciphertext c_v contains plaintext 1 and the remaining ciphertexts contain plaintext 0, otherwise, all ciphertexts contain plaintext 0. Vote also outputs proofs $\sigma_1, ..., \sigma_{nc}$ so that this can be verified. In particular, proof σ_j demonstrates ciphertext c_j contains 0 or 1, for all $1 \leq j \leq nc - 1$. And proof σ_{nc} demonstrates that the homomorphic combination of ciphertexts $c_1 \otimes \cdots \otimes c_{nc-1}$ contains 0 or 1. (It follows that the voter's ballot contains a vote for exactly one candidate.)
- Tally homomorphically combines ciphertexts representing votes for a particular candidate and decrypts the homomorphic combinations. The number of votes for a candidate $v \in \{1, \ldots, nc - 1\}$ is simply the homomorphic combination of ciphertexts representing votes for that candidate. The number of votes for candidate nc is equal to the number of votes for all other candidates subtracted from the total number of valid ballots on the bulletin board.
- Verify checks that each of the above steps has been performed correctly.

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

Definition 44 (Helios'16 [SFC17])**.** *Election scheme* Helios'16 *is* $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$*, where Π is additively homomorphic El Gamal [CGS97, §2], Σ_1 is the sigma protocol for proving knowledge of discrete logarithms by Chaum et al. [CEGP87, Protocol 2], Σ_2 is the sigma protocol for proving knowledge of* *disjunctive equality between discrete logarithms by Cramer et al. [CFSY96, Figure 1], Σ_3 is the sigma protocol for proving knowledge of equality between discrete logarithms by Chaum & Pedersen [CP93, §3.2], and $\mathcal{H}$ is a random oracle.*

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

Bernhard et al. [BPW12a, §4] remark that the sigma protocols used by Helios'16 to prove discrete logarithms and equality between discrete logarithms both satisfy special soundness and special honest verifier zero-knowledge, hence, Theorem 22 is applicable. Bernhard et al. also remark that the sigma protocol for proving knowledge of disjunctive equality between discrete logarithms satisfies special soundness and "almost special honest verifier zero-knowledge" and argue that "we could fix this[, but] it is easy to see that ... all relevant theorems [including Theorem 22] still hold." We adopt the same position and assume that Theorem 22 is applicable.

G Auction schemes from Helios with tallying by homomorphic combinations

In this appendix, let $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ be Helios'16. And let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be additively homomorphic El Gamal [CGS97, §2]. Moreover, let $(\mathsf{ProveKey}, \mathsf{VerKey})$, respectively $(\mathsf{ProveDec}, \mathsf{VerifyDec})$ 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].

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

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

Lemma 41. *Helios'16 ensures honest key generation.*

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

Lemma 42. *Helios'16 satisfies simulation sound private key extractibility.*

Proof. We have Helios'16 = $\mathsf{Helios}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$, where Π, Σ_1, Σ_2, Σ_3 and $\mathcal{H}$ satisfy the conditions given in Definition 44. Let $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$. Since Σ_1 satisfies special soundness and special honest verifier zero-knowledge, there exists an extractor for $(\mathsf{ProveKey}, \mathsf{VerKey})$ by Theorem 22. Let $\mathsf{ExtProve}$ be such an extractor. We define algorithm $\mathcal{K}$ using $\mathsf{ExtProve}$.

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

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

G.1 Reveal algorithm

Definition 45. *We define reveal algorithm* Helios-Reveal *as follows.*

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

Proof. Suppose κ is a security parameter, nb and nc are integers, and $v, v_1, \ldots, v_{nb} \in \{1, \ldots, nc\}$ 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 $\mathfrak{bb} = \{b_1, \ldots, b_{nb}\}$. Suppose $\mathfrak{b}$ is an output of $\mathsf{Helios\text{-}Reveal}(sk', nc, \mathfrak{bb}, v, \kappa)$. By definition of Helios'16, the largest subset of $\mathfrak{bb}$ satisfying the conditions given by algorithm Tally, hence, Helios-Reveal operates on $\mathfrak{bb}$, rather than a strict 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 $b[v]$ is an El Gamal ciphertext. Moreover, if $v_i = v$, then $b[v]$ enciphers 1, otherwise, $b[v]$ enciphers 0. By correctness of El Gamal, we have with overwhelming probability that $v_i = v$ implies $\mathsf{Dec}(sk, b[v]) = 1$. Hence, by definition of Helios-Reveal, we have $b \in \mathfrak{b}$.

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

Lemma 44. *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 conditions given by algorithm Tally, 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:

$$\{b \mid (b, \mathbf{v}) \in W \wedge \mathbf{v}[v] = 1\} = \{b \mid (b, \mathbf{v}) \in W' \wedge \mathbf{v}[v] = 1\} \tag{13}$$

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

$$\{b_1, \ldots, b_\ell\} = \{b \mid (b, \mathbf{v}) \in W'\} \tag{14}$$

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 (13), it suffices to prove $\mathfrak{b} = \{b \mid (b, \mathbf{v}) \in W' \wedge \mathsf{Dec}(sk, b[v]) = 1\}$. By definition of $\mathsf{Helios\text{-}Reveal}$, we have $\mathfrak{b} = \{b_i \mid 1 \leq i \leq \ell \wedge \mathsf{Dec}(sk, b_i[v]) = 1\}$. Hence, we conclude by (14).

- 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 (13), 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 $\mathsf{Helios\text{-}Reveal}$, 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 (14), 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 $\{0,1\}$. And, by definition of Setup, we have $\{0, \ldots, nc-1\} \subseteq \mathfrak{m}$. It follows that $\sum_{j=1}^{nc-1} \mathsf{Dec}(sk, b[j]) \in \mathfrak{m}$. Since the encryption scheme is additively homomorphic, we have $\sum_{j=1}^{nc-1} \mathsf{Dec}(sk, b[j]) = \mathsf{Dec}(sk, b[1]) \odot \cdots \odot \mathsf{Dec}(sk, b[nc-1])$, moreover, $b[1] \otimes \cdots \otimes b[nc-1]$ is a ciphertext, with overwhelming probability. And, by perfect correctness, we have $\mathsf{Dec}(sk, b[1]) \odot \cdots \odot \mathsf{Dec}(sk, b[nc-1]) = \mathsf{Dec}(sk, b[1] \otimes \cdots \otimes b[nc-1])$. 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, $\mathsf{Helios\text{-}Reveal}$ satisfies reveal soundness with respect to Helios'16. □

Lemma 45. *Relation $R(\text{Helios'16}, \mathsf{Helios\text{-}Reveal})$ 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 exist 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 $R(\text{Helios'16}, \mathsf{Helios\text{-}Reveal})$ 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 Helios-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 Helios-Reveal, we have that b satisfies the conditions given by algorithm Tally. Thus, b is a vector of length $2 \cdot nc - 1$ and $\bigwedge_{j=1}^{nc-1} \mathsf{VerifyCiph}((pk, b[j], \{0,1\}), (b[j+nc-1]), j) = 1 \wedge \mathsf{VerifyCiph}((pk, b[1] \otimes \cdots \otimes b[nc-1], \{0,1\}), (b[2 \cdot nc - 1]), nc) = 1$. In their proof that Helios'16 satisfies universal verifiability, Smyth, Frink & Clarkson [SFC16, SFC17] show:

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

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

Thus, there exists β, r such that

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

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

$$\forall j \in \{1, \ldots, nc-1\} \,.\, m_j = 1 \Longleftrightarrow \beta = j \tag{15}$$

$$\sum_{j=1}^{nc-1} m_j = 0 \Longleftrightarrow \beta = nc \tag{16}$$

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

By definition of Helios-Reveal, we have either: 1) $\mathsf{Dec}(sk, b[v]) = 1$, hence, $m_v = 1$ by perfect correctness of El Gamal, and $\beta = v$ by (15); or 2) $\mathsf{Dec}(sk, b_i[1] \otimes \cdots \otimes b_i[nc-1]) = 0$, hence, $m_1 \odot \cdots \odot m_{nc-1} = 0$ by the perfect correctness and perfectly homomorphic properties of El Gamal, 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 (16). 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 perfect correctness of Helios'16, we have b satisfies the conditions given by algorithm Tally. Moreover, by definition of algorithm Vote, if $1 \leq v < nc$, then there exist coins r such that $b[v] = \mathsf{Enc}(pk, 1; r)$, and by perfect correctness of El Gamal, 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, $\mathsf{Dec}(sk, b_i[1] \otimes \cdots \otimes b_i[nc-1]) = 0 \odot \cdots \odot 0 = 0$ since El Gamal is perfectly correct and perfectly homomorphic, thus, $b \in \mathfrak{b}$, concluding Case II, and our proof. $\square$

G.2 Alternative non-interactive proof system

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

Definition 46. *We define the tuple of algorithms* $(\mathsf{ProveReveal}, \mathsf{VerifyReveal})$ *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 algorithm* Tally*. 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 algorithm* Tally*. 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 46. *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 $\{0, ..., mc\} \subseteq \{0\} \cup \mathfrak{m}$.

We have $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) = 1$, by perfect 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 perfect correctness of El Gamal and by perfect 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 perfect correctness of El Gamal and by perfect 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 perfect correctness of El Gamal and by perfect 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 perfect correctness of El Gamal and by perfect 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. □

Lemma 47. *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 Succ(ZK((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 $\mathcal{A}$. 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 $\mathsf{ProveDec}$. $\square$

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

Proof sketch. Suppose $(\mathsf{ProveReveal}, \mathsf{VerifyReveal})$ 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 $\mathsf{VerifyReveal}$, we have s parses as $(pk', nc, \mathfrak{bb}, v, \mathfrak{b}, \kappa)$ and pk' as $(pk, \mathfrak{m}, \rho)$. Moreover, $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) = 1 \wedge v \in \{1, \dots, nc\} \wedge \{1, \dots, nc\} \subseteq \mathfrak{m}$. Bernhard *et al.* [BPW12a, §4] remark that $(\mathsf{ProveKey}, \mathsf{VerKey})$ satisfies their notion of simulation sound extractability, hence, $(\mathsf{ProveKey}, \mathsf{VerKey})$ 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 $\{0, \dots, m\} \subseteq \{0\} \cup \mathfrak{m}$. By definition of Setup, there exists r' such that $(pk, sk, m, m) = \mathsf{Setup}(\kappa; r')$. We have $\{1, \dots, nc\} \subseteq \mathfrak{m}$ and $\{0, \dots, m\} \subseteq \mathfrak{m}$, 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 $\mathsf{VerifyReveal}$ 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 $\mathsf{Helios\text{-}Reveal}$, we have $\{b_1, \dots, b_\ell\} = \emptyset \wedge \mathfrak{b} = \emptyset$ implies $\mathfrak{b}$ is an output of $\mathsf{Helios\text{-}Reveal}(sk, nc, \mathfrak{bb}, v, \kappa)$.

Let $\{b_1, \dots, b_\ell\}$ be the largest subset of $\mathfrak{bb}$ satisfying the tallying conditions of Helios'16. Hence, $b_1[v], \dots, b_\ell[v]$ are ciphertexts on plaintexts in $\{0, 1\}$. Suppose $\mathfrak{b} \subseteq \{b_1, \dots, b_\ell\}$ 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 $(\mathsf{ProveDec}, \mathsf{VerifyDec})$ satisfies their notion of simulation sound extractability, hence, $(\mathsf{ProveDec}, \mathsf{VerifyDec})$ 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 all $b \in \mathfrak{b}$ we have $b[v]$ decrypts to 1, and for all $b \in \mathfrak{b} \setminus \{b_1, \ldots, b_\ell\}$ 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 all $b \in \mathfrak{b}$ 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 $(\mathsf{ProveDec}, \mathsf{VerifyDec})$ satisfies their notion of simulation sound extractability, hence, $(\mathsf{ProveDec}, \mathsf{VerifyDec})$ 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 all $b \in \mathfrak{b}$ we have $b[1] \otimes \cdots \otimes b[nc-1]$ decrypts to 0, and for all $b \in \mathfrak{b} \setminus \{b_1, \ldots, b_\ell\}$ 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 all $b \in \mathfrak{b}$ 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. □

Theorem 49. *If Helios'16 satisfies ballot secrecy, then auction scheme* $\Lambda($*Helios'16*$, \mathsf{Helios\text{-}Reveal}, (\mathsf{ProveReveal}, \mathsf{VerifyReveal}))$ *satisfies bid secrecy.*[46]

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

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

Proof. Smyth, Frink & Clarkson have shown that Helios'16 satisfies individual and universal verifiability [SFC16, SFC17]. Hence, we have $\Lambda(\text{Helios'16}, \mathsf{Helios\text{-}Reveal}, \delta(\text{Helios'16}))$ satisfies individual verifiability (Theorem 13). We have $(\mathsf{ProveReveal}, \mathsf{VerifyReveal})$ is a non-interactive proof system for relation $R(\text{Helios'16}, \mathsf{Reveal})$ (Lemma 46) satisfying soundness (Lemma 48). And $R($

[46] Formally, $\Lambda(\text{Helios'16}, \mathsf{Helios\text{-}Reveal}, (\mathsf{ProveReveal}, \mathsf{VerifyReveal}))$ is an auction scheme by Lemmata 4, 43, & 46.

Helios'16, $\mathsf{Helios\text{-}Reveal}$) is Λ-suitable (Lemmata 45). Hence, we have $\Lambda($ Helios'16, $\mathsf{Helios\text{-}Reveal}, (\mathsf{ProveReveal}, \mathsf{VerifyReveal}))$ satisfies universal verifiability (Theorem 14). □

H Helios with tallying by mixnet

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

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

- Σ proves plaintext knowledge *if* $((pk, c), (m, r)) \in R \Leftrightarrow c = \mathsf{Enc}(pk, m; r) \wedge m \in \mathfrak{m}$.
- Σ proves mixing *if* $((pk, \mathbf{c}, \mathbf{c}'), (\mathbf{r}, \chi)) \in R \Leftrightarrow \bigwedge_{1 \le i \le |\mathbf{c}|} \mathbf{c}'[\chi(i)] = \mathbf{c}[i] \otimes \mathsf{Enc}(pk, \mathfrak{e}; \mathbf{r}[i]) \wedge |\mathbf{c}| = |\mathbf{c}'| = |\mathbf{r}|$*, where* $\mathbf{c}$ *and* $\mathbf{c}'$ *are both vectors of ciphertexts encrypted under* pk*,* $\mathbf{r}$ *is a vector of coins,* χ *is a permutation on* $\{1, \ldots, |\mathbf{c}|\}$*, and* $\mathfrak{e}$ *is an identity element of the encryption scheme's message space with respect to* $\odot$*.*

Definition 48 (Generalised Helios with tallying by mixnet). *Suppose* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a multiplicative homomorphic asymmetric encryption scheme,* $\mathfrak{e}$ *is an identity element of* Π*'s message space with respect to* $\odot$*,* Σ_1 *proves correct key construction,* Σ_2 *proves plaintext knowledge,* Σ_3 *proves correct decryption,* Σ_4 *proves mixing, and* $\mathcal{H}$ *is a hash function. Let* $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$*,* $\mathsf{FS}(\Sigma_2, \mathcal{H}) = (\mathsf{ProveCiph}, \mathsf{VerifyCiph})$*,* $\mathsf{FS}(\Sigma_3, \mathcal{H}) = (\mathsf{ProveDec}, \mathsf{VerifyDec})$*,* $\mathsf{FS}(\Sigma_4, \mathcal{H}) = (\mathsf{ProveMix}, \mathsf{VerMix})$*, and* p *be some polynomial function. We define election scheme* generalised Helios with tallying by mixnet*, denoted* $\mathsf{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$*, as follows.*

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

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

[47] Some aspects of Definition 49 first appeared in [Smy18b].

$\mathsf{Tally}(sk', nc, \mathfrak{bb}, \kappa)$. *Initialise* $\mathbf{v}$ *as a zero-filled vector of length* nc. *Parse* sk' *as a vector* (pk, sk), *output* $(\mathbf{v}, \perp)$ *if parsing fails. Proceed as follows.*

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

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

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

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

   for $1 \leq i \leq \ell$ **do**
       $\mathbf{W}[i] \leftarrow \mathsf{Dec}(sk, \mathbf{bb}[i])$;
       $pf_2[i] \leftarrow \mathsf{ProveDec}((pk, \mathbf{bb}[i], \mathbf{W}[i]), sk, \kappa)$;
       if $1 \leq \mathbf{W}[i] \leq nc$ **then**
           $\mathbf{v}[\mathbf{W}[i]] \leftarrow \mathbf{v}[\mathbf{W}[i]] + 1$;

Output $(\mathbf{v}, (\mathbf{bb}, pf_1, \mathbf{W}, pf_2))$.

$\mathsf{Verify}(pk', nc, \mathfrak{bb}, \mathbf{v}, pf, \kappa)$. *Let* mc *be the largest integer such that* $\{0, \ldots, mc\} \subseteq \{0\} \cup \mathfrak{m}$. *Parse* pk' *as a vector* $(pk, \mathfrak{m}, \rho)$ *and* $\mathbf{v}$ *parses as a vector of length* nc, *output* 0 *if parsing fails,* $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) \neq 1$, $|\mathfrak{bb}| \not\leq p(\kappa)$, *or* $nc \not\leq mc$. *Proceed as follows.*

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

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

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

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

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

Lemma 51. *Suppose* Π, Σ_1, Σ_2, Σ_3, Σ_4 *and* $\mathcal{H}$ *satisfy the preconditions of Definition 43. We have* $\mathsf{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ *satisfies correctness.*

Lemma 52. *Suppose* Π, Σ_1, Σ_2, Σ_3, Σ_4 *and* $\mathcal{H}$ *satisfy the preconditions of Definition 48. Further suppose* Π *satisfies perfect correctness and is perfectly homomorphic. Moreover, suppose* Σ_1, Σ_2 *and* Σ_4 *satisfy perfect completeness. We have* $\Gamma = \mathsf{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ *satisfies perfect correctness.*

Proof sketch. Lemma 51 shows that $\mathsf{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ satisfies correctness. And that proof can be adapted to show perfect correctness. In particular, perfect completeness of Σ_1 ensures that algorithm Vote does not output $\perp$ and perfect completeness of Σ_2 ensures that algorithm Tally considers $\mathfrak{bb}$ as the largest subset of $\mathfrak{bb}$ satisfying the necessary conditions. Since Π is perfectly homomorphic, the mix outputs ciphertexts. And, since Π is perfectly correct, the outcome represents the votes. $\square$

Lemma 53. *Suppose* Π, Σ_1, Σ_2, Σ_3, Σ_4 *and* $\mathcal{H}$ *satisfy the preconditions of Definition 43. Further suppose* Σ_2 *satisfies special soundness and special honest verifier zero-knowledge, and* $\mathcal{H}$ *is a random oracle. We have* $\mathsf{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ *satisfies completeness.*

Lemma 54. *Suppose* Π, Σ_1, Σ_2, Σ_3, Σ_4 *and* $\mathcal{H}$ *satisfy the preconditions of Definition 48. Further suppose* Σ_1, Σ_2, Σ_3 *and* Σ_4 *satisfy perfect completeness, moreover,* Σ_2 *perfectly satisfies special soundness and special honest verifier zero-knowledge, and* $\mathcal{H}$ *is a random oracle. We have* $\mathsf{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ *satisfies perfect completeness.*

Proof sketch. Lemma 53 shows that $\mathsf{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ satisfies completeness. And that proof can be adapted to show perfect completeness (assuming the proof of Theorem 22 can be adapted to show $\mathsf{FS}(\Sigma_2, \mathcal{H})$ satisfies perfect simulation sound extractability), when Σ_1, Σ_3 and Σ_4 are perfectly complete. $\square$

Lemma 55. *Suppose* Π, Σ_1, Σ_2, Σ_3, Σ_4 *and* $\mathcal{H}$ *satisfy the preconditions of Definition 43. Further suppose* Π *is perfectly correct. We have* $\mathsf{HeliosM}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ *satisfies Injectivity.*

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

Definition 49. HeliosM'16 *is the class of election schemes that includes every scheme* $\Gamma = \mathsf{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ *such that:* Π, Σ_1, Σ_2, Σ_3 Σ_4 *satisfy the preconditions of Definition 48;* Π *is perfectly correct;* Σ_1, Σ_2, Σ_3 *and* Σ_4 *satisfy perfect completeness, moreover,* Σ_2, Σ_3 *and* Σ_4 *satisfy special soundness and special honest verifier zero-knowledge;* $\mathcal{H}$ *is a random oracle;* Γ *ensures honest key generation and satisfies* Smy-Ballot-Secrecy*,* Exp-IV-Ext*, and* Exp-UV-Ext*.*

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

I Auction schemes from Helios with tallying by mixnet

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

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

Proof sketch. Let $\mathsf{FS}(\Sigma_3, \mathcal{H}) = (\mathsf{ProveDec}, \mathsf{VerifyDec})$ and $\mathsf{FS}(\Sigma_4, \mathcal{H}) = (\mathsf{ProveMix}, \mathsf{VerMix})$. Suppose κ is a security parameter, nc is an integer, $\mathfrak{bb}$ is a bulletin board, (pk, sk, mb, mc) is an output of $\mathsf{Setup}(\kappa)$, and $(\mathbf{v}, pf)$ is an output of $\mathsf{Tally}(sk, nc, \mathfrak{bb}, nc, \kappa)$. If $pf = \perp$, then there trivially exists a tallying proof system for Γ that satisfies zero-knowledge, otherwise, we proceed as follows. By inspection of Tally, we have pf is a vector $(\mathbf{bb}, pf_1, \mathbf{W}, pf_2)$ such that $\mathbf{bb}$ is a vector of mixed ciphertexts, pf_1 is a proof produced by $(\mathsf{ProveMix}, \mathsf{VerMix})$, $\mathbf{W}$ is a vector of plaintexts, and pf_1 is a proof produced by $(\mathsf{ProveDec}, \mathsf{VerifyDec})$. By Theorem 22, we have $(\mathsf{ProveMix}, \mathsf{VerMix})$ and $(\mathsf{ProveDec}, \mathsf{VerifyDec})$ satisfy zero-knowledge, hence, it suffices to show that $\mathbf{bb}$ and $\mathbf{W}$ can be constructed by a simulator. (Formally, a single simulator is needed, but it is straightforward to see how simulators can be combined, so we omit these details for brevity.)

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

$\mathbf{Q} \leftarrow (((pk, b_1[1]), b_1[2]), \dots, ((pk, b_\ell[1]), b_\ell[2]))$;
$\mathbf{W} \leftarrow \mathsf{ExtProve}(\mathbf{H}, \mathbf{P}, \mathbf{Q})$;
for $1 \le i \le \ell$ **do** $\mathbf{W}[i] \leftarrow \mathbf{W}[i][1]$;

Since $\mathsf{ExtProve}$ is an extractor, we have $\mathsf{ExtProve}(\mathbf{H}, \mathbf{P}, \mathbf{Q})$ outputs a vector of witnesses associated with statements $(pk, b_1[1]), \ldots, (pk, b_\ell[1])$, i.e., a vector of pairs such that the first element is the plaintext corresponding to the ciphertext in the associated statement. Thus, the simulation is valid. □

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

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

I.1 Reveal algorithm

Definition 50. *We define reveal algorithm $\mathsf{HeliosM\text{-}Reveal}$ as follows.*

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

Lemma 58. *Reveal algorithm $\mathsf{HeliosM\text{-}Reveal}$ is correct with respect to Γ.*

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

Lemma 59. *Reveal algorithm $\mathsf{HeliosM\text{-}Reveal}$ satisfies reveal soundness with respect to Γ, assuming Π is perfectly correct and PET is perfectly correct.*

Proof sketch. 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 $1 \le v \le nc \le mc$ and $|\mathfrak{bb}| \le mb$. 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 $\mathsf{HeliosM\text{-}Reveal}$. To prove that $\mathsf{HeliosM\text{-}Reveal}$ satisfies reveal soundness with respect to Γ, 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** $(\mathbf{v}, pf) \leftarrow \mathsf{Tally}(sk', nc, \{b\}, \kappa)$; $W \leftarrow W \cup \{(b, \mathbf{v})\}$. We have for all $(b, \mathbf{v}) \in W$ that $b \in \mathfrak{bb}$ and, by definition of algorithm Tally, either $\emptyset$ or $\{b\}$ is the largest subset of $\{b\}$ such that b is a pair and $\mathsf{VerifyCiph}((pk,$

$b[1]), b[2], \kappa) = 1$. In the former case, we have $\mathbf{v}$ is a zero-filled vector of length nc. In the latter case, we have $b[1]$ is a ciphertext, with overwhelming probability. And, if $\mathsf{Dec}(sk, b[1] \otimes \mathsf{Enc}(pk, \mathfrak{e}; r)) \notin \{1, \ldots, nc\}$, then $\mathbf{v}$ is a zero-filled vector of length nc, otherwise, $\mathbf{v}$ is a zero-filled vector of length nc, except for index $\mathsf{Dec}(sk, b[1] \otimes \mathsf{Enc}(pk, \mathfrak{e}; r))$ which contains 1, where r is chosen uniformly at random by algorithm Tally. Since Γ is homomorphic, we have $b[1] \otimes \mathsf{Enc}(pk, \mathfrak{e}; r)$ is a ciphertext with overwhelming probability. And, by perfect correctness, we have $\mathsf{Dec}(sk, b[1] \otimes \mathsf{Enc}(pk, \mathfrak{e}; r)) = \mathsf{Dec}(sk, b[1]) \odot_{pk} \mathsf{Dec}(sk, \mathsf{Enc}(pk, \mathfrak{e}; r))$ and $\mathsf{Dec}(sk, b[1]) \odot_{pk} \mathsf{Dec}(sk, \mathsf{Enc}(pk, \mathfrak{e}; r)) = \mathsf{Dec}(sk, b[1]) \odot_{pk} \mathfrak{e}$, with overwhelming probability. And, since $\mathfrak{e}$ is an identity element, we have $\mathsf{Dec}(sk, b[1]) \odot_{pk} \mathfrak{e} = \mathsf{Dec}(sk, b[1])$, with overwhelming probability. It follows that

$$\{b \mid (b, \mathbf{v}) \in W \wedge \mathbf{v}[v] = 1\} = \{b \mid b \in \{b_1, \ldots, b_\ell\} \wedge \mathsf{Dec}(sk, b[1]) = v\}$$

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

Lemma 60. *Relation* $R(\Gamma, \text{HeliosM-Reveal})$ *is* Λ*-suitable.*

Proof. Suppose $((pk', nc, \mathfrak{bb}, v, \mathfrak{b}, \kappa), sk') \in R(\Gamma, \text{HeliosM-Reveal})$. By definition of $R(\Gamma, \text{HeliosM-Reveal})$, there exist mb, mc, r, r' such that $(pk', sk', mb, mc) = \mathsf{Setup}(\kappa; r')$, $\mathfrak{b} = \text{HeliosM-Reveal}(sk', nc, \mathfrak{bb}, v, \kappa; r)$, $1 \leq v \leq nc \leq mc$, and $|\mathfrak{bb}| \leq mb$. Let $\mathfrak{b}' = \mathfrak{bb} \cap \{b \mid b = \mathsf{Vote}(pk', nc, v, \kappa; r'')\}$. To prove relation $R(\Gamma, \text{HeliosM-Reveal})$ 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 HeliosM-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 HeliosM-Reveal, we have that b is a pair such that $\mathsf{VerifyCiph}((pk, b[1]), b[2], \kappa) = 1$. By Theorem 22, we have ($\mathsf{ProveCiph}$, $\mathsf{VerifyCiph}$) satisfies simulation sound extractability, hence, there exists coins r, r' and a plaintext m from the message space such that $b[2] = \mathsf{ProveCiph}((pk, b[1], \{0, 1\}), (m, r), \kappa; r')$ and $b[1] = \mathsf{Enc}(pk, m; r)$ with overwhelming probability. It follows that b is an output of Vote. Moreover, by perfect correctness of PET, we have $m = v$, with overwhelming probability. Hence, we conclude Case I.

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

I.2 Proof of Theorem 21

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

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

References

[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+15] 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.

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

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

[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.

[BT94a] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *STOC '94: Twenty-sixth Annual ACM Symposium on Theory of Computing*, pages 544–553, New York, NY, USA, 1994. ACM Press.

[BT94b] Josh Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections. In *STOC'94: 26th Theory of computing Symposium*, pages 544–553. ACM Press, 1994.

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

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

[CCC+08] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman. Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In *EVT'08: Electronic Voting Technology Workshop*. USENIX Association, 2008.

[CE16] Nicholas Chang-Fong and Aleksander Essex. The Cloudier Side of Cryptographic End-to-end Verifiable Voting: A Security Analysis of Helios. In *ACSAC'16: 32nd Annual Conference on Computer Security Applications*, pages 324–335. ACM Press, 2016.

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

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

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

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

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

[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 Küsters, Johannes Mueller, and Tomasz Truderung. Verifiability Notions for E-Voting Protocols. Cryptology ePrint Archive, Report 2016/287 (version 20160317:161048), 2016.

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

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

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

[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.

[FR96] M.K. Franklin and M.K. Reiter. The design and implementation of a secure auction service. *IEEE Transactions on Software Engineering*, 22(5):302–312, 1996.

[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.

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

[Gro06] Jens Groth. Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In *ASIACRYPT'02: 12th International Conference on the Theory and Application of Cryptology and Information Security*, volume 4284 of *LNCS*, pages 444–459. Springer, 2006.

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

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

[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.

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

[HTK98] Michael Harkavy, J. Doug Tygar, and Hiroaki Kikuchi. Electronic auctions with private bids. In *Proceedings of the 3rd USENIX Workshop on Electronic Commerce*, 1998.

[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.

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

[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.

[KTV12] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Clash Attacks on the Verifiability of E-Voting Systems. In *S&P'12: 33rd IEEE Symposium on Security and Privacy*, pages 395–409. IEEE Computer Society, 2012.

[KTV15] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and relationship to verifiability. Cryptology ePrint Archive, Report 2010/236 (version 20150202:163211), 2015.

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

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

[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.

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

[MSQ14] Adam McCarthy, Ben Smyth, and Elizabeth A. Quaglia. Hawk and Aucitas: e-auction schemes from the Helios and Civitas e-voting schemes. In *FC'14: 18th International Conference on Financial Cryptography and Data Security*, volume 8437 of *LNCS*, pages 51–63. Springer, 2014.

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

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

[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] Organization for Security and Co-operation in Europe. *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.

[Per16] Olivier Pereira. Internet Voting with Helios. In *Real-World Electronic Voting: Design, Analysis and Deployment*, volume 8604, chapter 11. CRC Press, 2016.

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

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

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

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

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

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

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

[SFC16] Ben Smyth, Steven Frink, and Michael R. Clarkson. Election Verifiability: Definitions and an Analysis of Helios and JCJ. Cryptology ePrint Archive, Report 2015/233 (version 20161018:111117), 2016.

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

[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.

[SK95] Kazue Sako and Joe Kilian. Receipt-Free Mix-Type Voting Scheme - A Practical Solution to the Implementation of a Voting Booth. In *EUROCRYPT'95: 12th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 921 of *LNCS*, pages 393–403. Springer, 1995.

[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.

[Smy16] Ben Smyth. Secrecy and independence for election schemes. Cryptology ePrint Archive, Report 2015/942 (version 20161228:181001), 2016.

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

[Smy18a] Ben Smyth. A foundation for secret, verifiable elections. Cryptology ePrint Archive, Report 2018/225, 2018.

[Smy18b] Ben Smyth. Verifiability of Helios Mixnet. In *Voting'18: 3rd Workshop on Advances in Secure Electronic Voting*, LNCS. Springer, 2018.

[SP13] Ben Smyth and Alfredo Pironti. Truncating TLS Connections to Violate Beliefs in Web Applications. In *WOOT'13: 7th USENIX Workshop on Offensive Technologies*. USENIX Association, 2013. (First appeared at Black Hat USA 2013.).

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

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

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

[US90] U.S. Congress. *Sherman Antitrust Act*, 1890.

[ZCC+13] Filip Zagórski, Richard T. Carback, David Chaum, Jeremy Clark, Aleksander Essex, and Poorvi L. Vora. Remotegrity: Design and Use of an End-to-End Verifiable Remote Voting System. In *ACNS'13: 11th International Conference on Applied Cryptography and Network Security*, volume 7954 of *LNCS*, pages 441–457. Springer, 2013.