Constructing secret, verifiable auction schemes from election schemes

Elizabeth A. Quaglia and Ben Smyth

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

December 16, 2015

Abstract

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

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

1 Introduction

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

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

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

Bidders and voters should be able to freely participate in auctions and elections [UN48, OAS69, OSC90], without fear of repercussions; this is known as *privacy*. Formulations of privacy depend on the environment [DKR06, DKR09, BHM08]. The following properties provide privacy in collusion-free environments [Smy15, MSQ14a].

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

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

Bidders and voters should be able to check that auctions and elections are run correctly [JCJ02, CRS05, Adi06, Dag07, Adi08, DJL13]; this is known as *verifiability*. We sometimes write *auction verifiability* and *election verifiability* to distinguish verifiability in each field. Verifiability includes the following two properties [KRS10, SFC15].

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

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

1.1 Constructing auction schemes from election schemes

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

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

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

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

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

1.2 Motivation and related work

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

Magkos, Alexandris & Chrissikopoulos [MAC02] and Her, Imamot & Sakurai [HIS05] have studied the relation between auction and election schemes. Magkos, Alexandris & Chrissikopoulos remark that auction and election schemes have a similar structure and share similar security properties. And Her, Imamot & Sakurai contrast privacy properties of auction and election schemes, and compare the use of homomorphic encryption and mixnets between fields. More concretely, McCarthy, Smyth & Quaglia [MSQ14a] derive auction schemes from the Helios and Civitas election schemes. Lipmaa, Asokan & Niemi study the converse: they propose an auction scheme and claim that their scheme could be used to construct an election scheme [LAN02, §9].

1.3 Contribution

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

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

[2]Selecting a winning bid from a set of winning bids – i.e., having a strategy to handle tie-breaks – is beyond the scope of this paper.

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

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

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

2 Auction schemes

2.1 Syntax

We formulate syntax for *auction schemes.*

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

Setup*, denoted*[3] $(pk, sk, mb, mp) \leftarrow \mathsf{Setup}(\kappa)$*, is run by the auctioneer.* 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 voters.* Bid *takes as input a public key* pk*, an upper-bound* np *on the range of biddable prices, a bidder's chosen price* p*, and a security parameter* κ*. A bidder's price should be selected from the range* $1, \dots, np$ *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.*

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

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 ballots* $\mathfrak{b}$*, a proof* pf*, and a security parameter* κ*. It outputs a bit* s*, which is* 1 *if the auction verifies successfully or* 0 *otherwise.*

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

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

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

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

Completeness stipulates that outputs of algorithm Open will be accepted by algorithm Verify. This prevents *biasing attacks* [SFC15, §6].

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

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

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

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

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

2.2 Bid secrecy

We formalise *bid secrecy* as an indistinguishability game between an adversary and a challenger.[4] Our game captures a setting where the challenger generates a key pair using the scheme's Setup algorithm, publishes the public key, and only uses the private key for opening.

The adversary has access to a left-right oracle [BDJR97, BR05] which can compute bids on the adversary's behalf. Bids can be computed by the left-right oracle in two ways, corresponding to a randomly chosen bit β. If $\beta = 0$, then, given a pair of prices p_0, p_1, the oracle outputs a bid for p_0. Otherwise ($\beta = 1$), the oracle outputs a bid for p_1. The left-right oracle essentially allows the adversary to control the distribution of prices in bids, but bids computed by the oracle are always computed using the prescribed Bid algorithm.

The adversary outputs a bulletin board (the bulletin board may contain bids output by the oracle and bids generated by the adversary), which is opened by the challenger to reveal winning 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 p, 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 *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 winning price p has been correctly computed, that is, when there exists a bid for price p on the bulletin board and there is no bid for a higher price. Moreover, injectivity ensures that the bid was created for that price.[5]

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

[4] Games are algorithms that output 0 or 1. An adversary *wins* a game by causing it to output 1. We denote an adversary's *success* $\mathsf{Succ}(\mathsf{Exp}(\cdot))$ in a game $\mathsf{Exp}(\cdot)$ as the probability that the adversary wins, that is, $\mathsf{Succ}(\mathsf{Exp}(\cdot)) = \Pr[\mathsf{Exp}(\cdot) = 1]$. Adversaries are assumed to be *stateful*, that is, information persists across invocations of the adversary in a single game, in particular, the adversary can access earlier assignments.

[5] 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.

$\mathfrak{bb} \wedge (b, p_0, p) \in L\}|$, where L is the set of oracle call inputs and outputs.

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

Our formalisation is as follows.

Definition 5 (Bid secrecy)**.** *Let* $\Sigma = (\mathsf{Setup}, \mathsf{Bid}, \mathsf{Open}, \mathsf{Verify})$ *be an auction scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{Bid\text{-}Secrecy}(\Sigma, \mathcal{A}, \kappa)$ *be the following game.*[6]

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

Oracle $\mathcal{O}$ *is defined as follows:*[7]

- $\mathcal{O}(p_0, p_1)$ *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*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such that for all security parameters* κ*, we have* $\mathsf{Succ}(\mathsf{Bid\text{-}Secrecy}(\Sigma, \mathcal{A}, \kappa)) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$*.*

Our definition of bid secrecy is based upon the notion of ballot secrecy proposed by Smyth [Smy15] (cf. the forthcoming companion technical report) and, roughly speaking, corresponds to a symbolic security definition proposed by Dreier, Lafourcade & Lakhnech [DLL13, Definition 15].

2.2.1 Example: Enc2Bid

We demonstrate the applicability of our definition with a construction (Enc2Bid) for auction schemes from asymmetric encryption schemes.[8]

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

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

[8]We present definitions of cryptographic primitives and relevant security definitions in the forthcoming companion technical report.

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

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

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

Lemma 1. *Suppose* Π *is an asymmetric encryption scheme with perfect correctness. We have* $\mathsf{Enc2Bid}(\Pi)$ *is an auction scheme (i.e., correctness, completeness and injectivity are satisfied).*

The proof of Lemma 1 and all further proofs, except where otherwise stated, appear in the forthcoming companion technical report.

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

2.3 Auction verifiability

We formalise individual and universal verifiability as games between an adversary and a challenger. Our definitions are based upon analogous definitions for election schemes by Smyth, Frink & Clarkson [SFC15] (cf. Section 5.1).

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 in an auction.

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

```
$\mathsf{Exp\text{-}IV}(\Sigma, \mathcal{A}, \kappa) =$
    $(pk, np, p, p') \leftarrow \mathcal{A}(\kappa);$
    $b \leftarrow \mathsf{Bid}(pk, np, p, \kappa);$
    $b' \leftarrow \mathsf{Bid}(pk, np, p', \kappa);$
    if $b = b' \wedge b \neq \bot \wedge b' \neq \bot$ then
        return 1
    else
        return 0
```

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

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

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

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

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

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

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

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

3 Auction schemes from election schemes

3.1 Election scheme syntax

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

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

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

Vote*, denoted* $b \leftarrow \mathsf{Vote}(pk, nc, v, \kappa)$*, is run by voters.* Vote *takes as input a public key pk, some number of candidates nc, a voter's vote v, and a security parameter κ. A voter's vote should be selected from a sequence $1, \ldots, nc$ of candidates.* 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 $\mathfrak{bb}$, 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*[9] *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 or* 0 *otherwise.*

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

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

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

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

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

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

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

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

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

Comparing auction and election schemes. Auction schemes are distinguished from election schemes in the final step of their execution: auction schemes open the bulletin board to recover the winning price and winning bids, whereas, election schemes tally the bulletin board to recover the distribution of votes. Our goal is to bridge this gulf; we do so by introducing *reveal algorithms.*

3.2 Reveal algorithm

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

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

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

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

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

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

3.3 Construction

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

3.3.1 Non-verifiable auction schemes

Our first construction follows intuitively from our informal description (Section 1.1). Algorithm Bid is derived from Vote, simply by representing prices as candidates. Algorithm Open uses algorithm Tally to derive the distribution of prices and the winning price is determined from this distribution. Moreover, we exploit a reveal algorithm Reveal to disclose the set of winning bids.

Definition 15. *Given an election scheme* $\Gamma = (\mathsf{Setup}_\Gamma, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify}_\Gamma)$ *and a reveal algorithm* Reveal*, we define* $\Lambda(\Gamma, \mathsf{Reveal}) = (\mathsf{Setup}_\Lambda, \mathsf{Bid}, \mathsf{Open}, \mathsf{Verify}_\Lambda)$ *as follows.*

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

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

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

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

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

Lemma 2. *Let* Γ *be an election scheme and* Reveal *be a reveal algorithm. Suppose* Reveal *is correct with respect to* Γ*. We have* $\Lambda(\Gamma, \mathsf{Reveal})$ *is an auction scheme.*

3.3.2 Verifiable auction schemes

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

Definition 16. *Given an election scheme* $\Gamma = (\mathsf{Setup}_\Gamma, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify}_\Gamma)$*, a reveal algorithm* Reveal*, and a non-interactive proof system* $\Delta = (\mathsf{Prove}, \mathsf{Verify})$*, we define* $\Lambda(\Gamma, \mathsf{Reveal}, \Delta) = (\mathsf{Setup}_\Lambda, \mathsf{Bid}, \mathsf{Open}, \mathsf{Verify}_\Lambda)$ *as follows.*

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

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

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

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

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

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

Algorithms Tally *and* Verify *require* ϵ *to be a constant symbol.*

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

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

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

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

4 Privacy results

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

4.1 Ballot secrecy

Our definition of ballot secrecy strengthens an earlier definition by Smyth [Smy15].

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

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

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

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

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

Our formalisation of ballot secrecy challenges an adversary to determine whether the left-right oracle produces ballots for "left" or "right" inputs. In addition to the oracle's outputs, the adversary is given the election outcome and tallying proof derived by tallying the adversary's board (intuitively, this captures a setting where the bulletin board is constructed by an adversary that casts ballots on behalf of a subset of voters and controls the distribution of votes cast by the remaining voters). The adversary is also given a mapping W from ballots to votes, for all ballots on the bulletin board which were not output by the oracle. To avoid trivial distinctions, we insist that oracle queries are balanced, i.e., predicate *balanced* must hold. Intuitively, if the adversary does not succeed, then ballots for different votes cannot be distinguished, hence, a voter cannot be linked to a vote, i.e., ballot secrecy is preserved. On the other hand, if the adversary does succeed, then ballots can be distinguished and ballot secrecy is not preserved.

Comparing notions of ballot secrecy. Our definition of ballot secrecy (Ballot-Secrecy) strengthens an earlier definition (SB-Ballot-Secrecy) by Smyth (the forthcoming companion technical report). In particular, in Ballot-Secrecy the adversary is given the vote corresponding to *any* ballot that was *not* computed by the oracle, whereas in SB-Ballot-Secrecy the adversary does not have this capability. It is trivial to see that Ballot-Secrecy strengthens SB-Ballot-Secrecy, because any adversary against SB-Ballot-Secrecy (without access to W) is also an adversary against Ballot-Secrecy (with access to W). In the forthcoming companion technical report, we show that Ballot-Secrecy is strictly stronger using a scheme that satisfies SB-Ballot-Secrecy but not Ballot-Secrecy, hence separating the two notions.

4.1.1 Example: Enc2Vote satisfies ballot secrecy

We demonstrate the applicability of our definition using a construction (Enc2Vote) for election schemes from non-malleable public-key encryption schemes.[10]

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

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

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

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

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

Intuitively, given an encryption scheme Π satisfying non-malleability, the election scheme $\mathsf{Enc2Vote}(\Pi)$ derives ballot secrecy from the encryption scheme until tallying and tallying maintains ballot secrecy by only disclosing the number of votes for each candidate. Formally, we have the following result.[11]

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

4.2 Ballot secrecy implies 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. However, by closer inspection of point 2, we observe a non-trivial distinction which falsifies our intuition in the general case. In particular, privacy of losing bidders must be preserved even if the winning price is incorrectly announced.

4. Our ballot secrecy game maps ballots to votes, except for those output by the oracle, whereas our bid secrecy game permits oracle bids to be mapped to non-winning prices.

This leads to a separation. Nevertheless, we can formulate soundness conditions which capture a class of election schemes for which our intuition holds.

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

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

concoct a scenario in which the election outcome does not include the votes of all ballots on the bulletin board that were produced by Vote.

Formally, we capture the correct election outcome using function *correct-outcome*, which is defined such that for all pk, nc, $\mathfrak{bb}$, κ, ℓ, and $v \in \{1, \ldots, nc\}$, we have[12]

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

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

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

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

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

Reveal soundness. Correctness for reveal algorithms ensures that algorithm Reveal produces the set of ballots for a particular vote under ideal conditions. A similar property, which we call *reveal soundness*, can hold in the presence of an adversary. Our formulation of reveal soundness challenges the adversary to concoct a scenario in which the set of ballots for a particular vote is not correct, i.e., the set does not contain all the ballots for the specified vote.

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

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

[12]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.

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

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

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

4.2.1 Bid secrecy for non-verifiable auction schemes

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

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

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

Example: Enc2Bid satisfies bid secrecy

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

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

4.2.2 Bid secrecy for verifiable auction schemes

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

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

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

5 Verifiability results

We recall definitions of individual and universal verifiability for election schemes by Smyth, Frink & Clarkson [SFC15]. 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.

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

```
$\mathsf{Exp\text{-}IV\text{-}Ext}(\Gamma, \mathcal{A}, \kappa) =$
    $(pk, nc, v, v') \leftarrow \mathcal{A}(\kappa)$;
    $b \leftarrow \mathsf{Vote}(pk, nc, v, \kappa)$;
    $b' \leftarrow \mathsf{Vote}(pk, nc, v', \kappa)$;
    if $b = b' \wedge b \neq \perp \wedge b' \neq \perp$ then
        return 1
    else
        return 0
```

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

5.1.2 Universal verifiability

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

Formally, we capture the correct election outcome using function *correct-outcome*. Since function *correct-outcome* will now be parameterised with a public key constructed by the adversary, rather than a public key constructed by algorithm

Setup (cf. Section 4.2), we must strengthen injectivity to hold for adversarial keys.

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

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

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

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

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

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

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

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

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

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

5.2 Election verifiability implies auction verifiability

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

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

The proof of Theorem 12 follows from Definitions 7, 16 & 22 and we omit a formal proof.

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

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

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

6 Case study: Helios

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

6.1 Helios

Helios is an open-source, web-based electronic voting system, which has been deployed in the real-world: the International Association of Cryptologic Research (IACR) has used Helios annually since 2010 to elect board members [I13, BVQ10, HBH10], 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 [Adi09, P12].

Informally, Helios can be modelled as an election scheme ($\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify}$) such that:

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

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

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

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

The original Helios scheme [AMPQ09] is known to be vulnerable to ballot secrecy attacks, and defences against those attacks have been proposed [CS11, CS13, SC11, Smy12, BCP$^+$11, BPW12a]. We adopt the formal definition of Helios proposed by Smyth, Frink & Clarkson [SFC15], which adopts non-malleable ballots [SHM15] to defend against those attacks. Henceforth, we write *Helios 4.0* to refer to that formalization.

6.2 An auction scheme from Helios 4.0

We derive an auction scheme from Helios 4.0 using our construction parametrised with a reveal algorithm and a non-interactive proof system. We formally describe that reveal algorithm and proof system in the forthcoming companion technical report, and refer to the resulting scheme as *the auction scheme from Helios 4.0*. Our privacy and verifiability results allow us to prove security of that scheme:

Theorem 14. *If Helios 4.0 satisfies ballot secrecy, then the auction scheme from Helios 4.0 satisfies bid secrecy.*

Proof. Smyth, Frink & Clarkson have shown that Helios 4.0 satisfies universal verifiability [SFC15]. It follows from Lemma 10 that Helios 4.0 satisfies tally soundness. Hence, by Theorem 9, it suffices to prove that the reveal algorithm satisfies reveal soundness and that the non-interactive proof system is zero-knowledge. We defer those proofs to the forthcoming companion technical report. □

Proving that Helios 4.0 satisfies ballot secrecy would advance the state-of-the-art in a manner that is beyond the scope of this case study. Indeed, the only privacy results [BPW12a, Ber14, BCG$^+$15] for Helios consider variants of Helios 4.0 and depend upon undesirable trust assumptions [Smy15].

Theorem 15. *The auction scheme from Helios 4.0 satisfies individual and universal verifiability.*

Proof. Smyth, Frink & Clarkson have shown that Helios 4.0 satisfies individual and universal verifiability [SFC15]. Hence, by Theorem 12, the auction scheme from Helios 4.0 satisfies individual verifiability. To show universal verifiability, it suffices (Theorem 13) to prove that the non-interactive proof system satisfies soundness and the associated relation is verifiable. We defer those proofs to the forthcoming companion technical report. □

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

7 Conclusion

We demonstrate that the seemingly disjoint research fields of auctions and elections are actually related. In particular, we present a generic construction for auction schemes from election schemes. And we formulate precise conditions under which auction schemes produced by our construction are secure. The value of our results is two-fold: we enable advances in the auction research field from existing election literature, as well as capitalise upon future progress in election research.

Acknowledgements

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

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.

[Adi09] Ben Adida. Helios deployed at Princeton. `http://heliosvoting.org/2009/10/13/helios-deployed-at-princeton/` (accessed 8 February 2013), 2009.

[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. A comprehensive analysis of game-based ballot privacy definitions. Cryptology ePrint Archive, Report 2015/255 (version 20150319:100626), 2015.

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

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

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

[BHM08] Michael Backes, Cătălin Hriţcu, and Matteo Maffei. Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-calculus. In *CSF'08: 21st Computer Security Foundations Symposium*, pages 195–209. IEEE Computer Society, 2008.

[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.

[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.

[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.

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

[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.

[DKR06] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Coercion-Resistance and Receipt-Freeness in Electronic Voting. In *CSFW'06: 19th Computer Security Foundations Workshop*, pages 28–42. IEEE Computer Society, 2006.

[DKR09] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.

[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.

[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.

[I13] IACR Elections. `http://www.iacr.org/elections/` (accessed 3 April 2013), 2013.

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

[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.

[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.

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

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

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

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

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

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

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

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

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

[P12] Helios Princeton Elections. `https://princeton.heliosvoting.org/` (accessed 8 February 2013), 2012.

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

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

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

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

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

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

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

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

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

[Smy15] Ben Smyth. Secrecy and independence for election schemes. Cryptology ePrint Archive, Report 2015/942, 2015.

[UN48] Universal Declaration of Human Rights, 1948.