Athena: A verifiable, coercion-resistant voting system with linear complexity

Ben Smyth

December 17, 2019

Abstract

Seminal work by Juels, Catalano & Jakobsson delivered a verifiable, coercion-resistant voting system with quadratic complexity. This manuscript attempts to advance the state-of-the-art by delivering a voting system with equivalent security and linear complexity.

1 Introduction

Voting systems must ensure free-choice [57, 58, 71]. A notion of which is formalised by *ballot secrecy* (i.e., a voter's vote is not revealed to anyone) [8, 9, 11, 17, 65]. This notion can be satisfied by voting systems that simply instruct voters to encrypt their vote. But, free-choice may be compromised by an adversary that is able to communicate with voters, since the coins used for encryption serve as proof of how voters voted and voters may communicate those coins to the adversary. Hence, formulations of free-choice must be accompanied by operational assumptions and limitations on the adversary's capabilities. Indeed, ballot secrecy assumes that voters' ballots are constructed and tallied in the prescribed manner, and that the adversary's capabilities are limited to controlling ballot collection.

Receipt-freeness (i.e., a voter cannot collaborate with a conspirator to gain information which can be used to prove how they voted) formalises a notion of free-choice in the presence of an adversary that can communicate with voters [10, 14, 27, 31, 46, 56]. Yet free-choice may be compromised if voters deviate from the prescribed voting procedure. *Coercion-resistance* (i.e., a voter can deviate from a coercer's instructions, to cast their own vote, without detection) formalises a stronger notion of free-choice assuming that not only can voters deviate, but the adversary can instruct voters how to deviate [40, 51, 68, 72]. The distinction between receipt-freeness and coercion-resistance is subtle: "receipt-freeness deals with a [conspirator] who is only concerned with deducing information about how someone voted from receipts and public information, but who does not give detailed instructions on how to cast the vote. Coercion resistance, on the other hand, includes dealing with a coercer who gives details not just on which candidate to vote for but also on how to cast the vote" [37, §1.1]. Both receipt-freeness and coercion-resistance retain the assumption that voters' ballots are tallied in the prescribed manner, and receipt-freeness additionally assumes voters' ballots are constructed in the prescribed manner.

Beyond free-choice, voting systems must ensure that only voters vote [57, 58, 71], which can be achieved by issuing credentials to voters and using cryptography to ensure that authorised ballots are *unforgeability* (i.e., only voters can construct authorised ballots) [66,69]. (Unforgeability is sometimes known as *eligibility verifiability.*) Moreover, voting systems must ensure that voters have equal influence in the decision [57, 58, 71], which can be achieved by *universal verifiability* (i.e., anyone can check whether an outcome corresponds to votes expressed in collected ballots that are authorised, except for votes expressed in ballots from the same voter, which are all discarded, except for the voter's last vote) and *individual verifiability* (i.e., a voter can check whether their ballot is collected) [22, 46, 47, 50, 69].

Seminal work by Juels, Catalano & Jakobsson [39–41] made significant progress towards a voting system satisfying the aforementioned properties, moreover, Clarkson, Chong & Myers [20, 21] implemented their results as Civitas, albeit, complexity is $\mathcal{O}(|\mathfrak{bb}|^2)$, i.e., quadratic in the length of the bulletin board ($\mathfrak{bb}$). Quadratic complexity arises from the use of pairwise plaintext equality tests on the bulletin board's ballots to discard all but the last vote cast using a private credential. Pairwise plaintext equality tests are also used on mixed ballots and mixed public credentials to discard mixed ballots that are unauthorised, with complexity $\mathcal{O}(|L| \cdot |\mathfrak{bb}|)$, where L is the electoral roll.

Contribution. We advance the state-of-the-art with Athena: A verifiable, coercion-resistance voting system with linear complexity $\mathcal{O}(|\mathfrak{bb}|)$. Our system reveals anonymised credentials to discard ballots cast using the same private credential (with linear complexity) and uses plaintext equality tests on each individual mixed ballot – which includes a mixed public credential – to discard any mixed ballot that is unauthorised (with linear complexity). Athena works as follows.

Voting. Voters are issued with credential pairs, wherein the private credential is a nonce and the public credential is an encryption of that nonce. Each voter encrypts the negation of their private credential and their vote, and publishes the two resulting ciphertexts prepended with their public credential and appended with a counter (to the bulletin board). A voter computes ballots for any re-votes similarly, using an incremented counter. It follows that the bulletin board will contain voters' ballots, plus any adversarial ballots.

Tallying. Any ballots not containing a public credential are discarded, the second ciphertext of each remaining ballot is homomorphically combined with itself n-times (for some nonce n), and the resulting combination is decrypted to reveal an anonymised credential. Entries that share an anonymised credential are

Sidebar 1 Notation

We let $A(x_1, \ldots, x_n; r)$ denote the output of probabilistic algorithm A on inputs $x_1, \ldots, x_n$ and coins r, and we let $A(x_1, \ldots, x_n)$ denote $A(x_1, \ldots, x_n; r)$, where coins r are chosen uniformly at random from the coin space of algorithm A. Moreover, we let $x \leftarrow T$ denote assignment of T to x; $x \overset{r}{\leftarrow} A(x_1, \ldots, x_n)$ denote assignment of $A(x_1, \ldots, x_n; r)$ to x, where coins r are chosen uniformly at random from the coin space of algorithm A; and $x \leftarrow_R S$ denote assignment to x of an element chosen uniformly at random from set S (we use the same notation when S is a distribution). Furthermore, we let $x[i]$ denote component i of vector x and let $|x|$ denote the length of vector x. Finally, we write $(x_1, \ldots, x_{|T|}) \leftarrow T$ for $x \leftarrow T; x_1 \leftarrow x[1]; \ldots; x_{|T|} \leftarrow x[|T|]$, when T is a vector, and $x, x' \leftarrow_R S$ for $x \leftarrow_R S; x' \leftarrow_R S$.

discarded, except for the one with the highest counter, thus, only the last vote associated with each anonymised credential is retained. The first two ciphertexts of each retained ballot are homomorphically combined, deriving either: 1) the combination of a private credential and the negation of that credential, or 2) the combination of a private credential and some other message (excluding the credential's negation). The resulting homomorphic combinations and corresponding encrypted votes are mixed (using the same permutation), plaintext equality tests are used to determine whether the mixed homomorphic combinations where constructed using private credentials, and the corresponding mixed encrypted votes are decrypted if they were.

Intuitively, Athena achieves coercion-resistance, because a well-formed ballot that encrypts the negation of a voter's private credential is indistinguishable from an ill-formed ballot that encrypts some other message, hence, a voter cannot prove whether they cast a well-formed ballot (that will be counted, as opposed to an ill-formed ballot that will not), during the voting phase. Moreover, mixing ensures that ballots cannot be mapped to votes during tallying. Thus, coercion-resistance is achieved. (Unlike the voting system by Juels, Catalano & Jakobsson, Athena reveals the number of ballots cast using a voter's public credential, which requires voters to deny casting ballots when instructed by the coercer to abstain. This is a reasonable strategy, since no voter can prove whether they even cast a well-formed ballot.) Moreover, verifiability is achieved too, because only voters have access to private credentials, hence, only voters can construct authorised ballots (unforgeability), tallying produces evidence (specified in Definition 1) demonstrating that election outcomes correspond to the votes expressed in collected ballots that are authorised (universal verifiability), and ballots are recorded on a bulletin board, hence, voters can check whether their ballot is collected (individual verifiability).

2 Our voting system: Athena

Our voting system $(\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ is used as follows: The tallier initiates an election using algorithm Setup to compute a key pair, which includes a public key pk for an underlying multiplicative-homomorphic asymmetric encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, for which there exists a generator g of the scheme's message space. Next, the registrar uses algorithm $\mathsf{Register}$ to compute a credential pair, wherein the private credential is a nonce d and the public credential is an encryption $\mathsf{Enc}(pk, g^d; r)$ of that nonce, for some coins r. The registrar repeats the process to create further credential pairs and these pairs are issued to voters. Each voter uses algorithm Vote to compute their ballot, which includes: their public credential; an encryption $\mathsf{Enc}(pk, g^{-d}; s)$ of their negated private credential, for some coins s; an encryption $\mathsf{Enc}(pk, v; t)$ of their vote v, for some coins t; and a counter cnt. A voter similarly computes ballots for re-votes, using an incremented counter. It follows that the bulletin board will contain a ballot for a voter's first vote

$$\mathsf{Enc}(pk, g^d; r) \quad \mathsf{Enc}(pk, g^{-d}; s_1) \quad \mathsf{Enc}(pk, v_1; t_1) \quad cnt_1,$$

ballots for any of the voter's re-votes

$$\begin{array}{llll} \mathsf{Enc}(pk, g^d; r) & \mathsf{Enc}(pk, g^{-d}; s_2) & \mathsf{Enc}(pk, v_2; t_2) & cnt_2, \ \ldots, \\ \mathsf{Enc}(pk, g^d; r) & \mathsf{Enc}(pk, g^{-d}; s_k) & \mathsf{Enc}(pk, v_k; t_k) & cnt_k, \end{array}$$

such that $cnt_1 < \cdots < cnt_k$, and any other ballots cast using the voter's public credential, without the private credential (including those cast by the adversary or even the voter themselves), namely,

$$\begin{array}{llll} \mathsf{Enc}(pk, g^d; r) & \mathsf{Enc}(pk, g^{D_1}; \overline{s}_1) & \mathsf{Enc}(pk, \overline{v}_1; \overline{t}_1) & \overline{cnt}_1, \ \ldots, \\ \mathsf{Enc}(pk, g^d; r) & \mathsf{Enc}(pk, g^{D_l}; \overline{s}_l) & \mathsf{Enc}(pk, \overline{v}_l; \overline{t}_l) & \overline{cnt}_l. \end{array}$$

(Ballots also prove correct ciphertext construction, moreover, they prove that the second ciphertext of each ballot encrypts a message of the form g^m. Hence, we restrict ourselves to well-defined ballots above.) Furthermore, the bulletin board will contain ballots cast using other public credentials.

The tallier uses algorithm Tally to compute the election outcome as follows: The tallier generates a nonce n, homomorphically combines the second ciphertext of each entry on the bulletin board with itself n-times, decrypts the resulting homomorphic combinations to reveal anonymised credentials, and prepends entries with anonymised credentials, thereby producing output including

$$\begin{array}{lllll} g^{-d \cdot n} & \mathsf{Enc}(pk, g^d; r) & \mathsf{Enc}(pk, g^{-d}; s_1) & \mathsf{Enc}(pk, v_1; t_1) & cnt_1, \ \ldots, \\ g^{-d \cdot n} & \mathsf{Enc}(pk, g^d; r) & \mathsf{Enc}(pk, g^{-d}; s_k) & \mathsf{Enc}(pk, v_k; t_k) & cnt_k, \\ g^{D_1 \cdot n} & \mathsf{Enc}(pk, g^d; r) & \mathsf{Enc}(pk, g^{D_1}; \overline{s}_1) & \mathsf{Enc}(pk, \overline{v}_1; \overline{t}_1) & \overline{cnt}_1, \ \ldots, \\ g^{D_l \cdot n} & \mathsf{Enc}(pk, g^d; r) & \mathsf{Enc}(pk, g^{D_l}; \overline{s}_l) & \mathsf{Enc}(pk, \overline{v}_l; \overline{t}_l) & \overline{cnt}_l. \end{array}$$

Entries with the same public credential that are prepended with the same value are discarded, except for the one with the highest counter. Hence, the first $k-1$ entries (above) are discarded, whilst the kth entry is preserved. The remaining entries are similarly processed, therefore, the last will be kept if any entries sharing the prepended value ($g^{D_l \cdot n}$) have counter values lower than counter $\overline{cnt}_l$. (For example, suppose only the penultimate entry shares prepended value $g^{D_l \cdot n}$, i.e., $D_l = D_{l-1}$, and further suppose $\overline{cnt}_l > \overline{cnt}_{l-1}$. Hence, the last entry will be preserved and the penultimate entry will be discarded. By comparison, the penultimate entry will be kept if $\overline{cnt}_l < \overline{cnt}_{l-1}$.) The first two ciphertexts of preserved entries are homomorphically combined and paired with encrypted votes, producing

$$\mathsf{Enc}(pk, g^d \odot g^{-d}; r \oplus s_k) \qquad \mathsf{Enc}(pk, v_k; t_k),$$

for the kth entry (above), and

$$\mathsf{Enc}(pk, g^d \odot g^{D_l}; r \oplus \overline{s}_l) \qquad \mathsf{Enc}(pk, \overline{v}_l; \overline{t}_l),$$

for the last (assuming it is preserved). The homomorphic combinations and encrypted votes are mixed (using the same permutation). The tallier performs (optimised) plaintext equality tests on each of the mixed homomorphic combinations to determine whether they contain plaintext one, and decrypts the corresponding mixed encrypted votes when the test holds. Thus, the voter's vote v_k (above) is revealed, because mixed ciphertext $\mathsf{Enc}(pk, g^d \odot g^{-d}; r \oplus s_k \oplus w)$ encrypts 1, whereas vote $\overline{v}_l$ is not revealed, because mixed ciphertext $\mathsf{Enc}(pk, g^d \odot g^{D_l}; r \oplus \overline{s}_l \oplus \overline{w})$ does not (recall g^{D_l} was constructed without private credential d), where w and $\overline{w}$ are coins introduced during mixing. The election outcome is the tally of revealed votes.

Athena is formally specified by Definition 1, using cryptographic primitives introduced in Appendix A. Those primitives include sigma protocols for proving correct key generation, ciphertext construction, and decryption, which behave as one might expect. They also include a sigma protocol for proving iterative homomorphic combination, that is, proving that a ciphertext c is computed from another ciphertext c' such that $c = \bigotimes_1^n c'$, for some nonce n. We apply the Fiat-Shamir transformation to sigma protocols to derive non-interactive proof systems, which we use to achieve verifiability.

Definition 1 (Athena). *Suppose* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a multiplicative-homomorphic asymmetric encryption scheme with a message space that is super-polynomial in the security parameter and for which a generator exists;* $\mathcal{M}$ *is a verifiable pairwise mixnet;* Σ_1*,* Σ_2*,* Σ_3 *and* Σ_4 *are sigma protocols that prove key generation, ciphertext construction, decryption and iterative homomorphic combination, respectively; and* $\mathcal{H}$ *is a hash function. Let* $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$*,* $\mathsf{FS}(\Sigma_2, \mathcal{H}) = (\mathsf{ProveCiph}, \mathsf{VerCiph})$*,* $\mathsf{FS}(\Sigma_3, \mathcal{H}) = (\mathsf{ProveDec}, \mathsf{VerDec})$*, and* $\mathsf{FS}(\Sigma_4, \mathcal{H}) = (\mathsf{ProveComb}, \mathsf{VerComb})$*.* Athena*, denoted* $\mathsf{Athena}(\Pi, \mathcal{M}, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$*, is defined by the following algorithms.*

- $\mathsf{Setup}(\kappa)$. *Compute*

  $$\begin{aligned}&(pk, sk, \mathfrak{m}) \xleftarrow{r} \mathsf{Gen}(\kappa);\\ &\rho \leftarrow \mathsf{ProveKey}((\kappa, pk, \mathfrak{m}), (sk, r), \kappa);\\ &\mathbf{pk} \leftarrow (pk, \mathfrak{m}, \rho);\\ &\mathbf{sk} \leftarrow (pk, sk),\end{aligned}$$

  let mb be the largest integer upper-bound by a polynomial in the security parameter, let mc be the largest integer such that $\{0, \ldots, mc\} \subseteq \{0\} \cup \mathfrak{m}$ and mc is upper-bound by a polynomial in the security parameter, and output $(\mathbf{pk}, \mathbf{sk}, mb, mc)$.

- $\mathsf{Register}(\vec{pk}, k)$. *Parse $\vec{pk}$ as $(pk, \mathfrak{m}, \rho)$, outputting $(\perp, \perp)$ if parsing fails or $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) = \perp$, generate nonce d, compute*

  $$\begin{aligned}&pd \leftarrow \mathsf{Enc}(pk, g^{d});\\ &\mathbf{d} \leftarrow (pd, d),\end{aligned}$$

  and output $(pd, \mathbf{d})$, where g is a generator of message space $\mathfrak{m}$.

- $\mathsf{Vote}(\vec{d}, \vec{pk}, v, cnt, nc, \kappa)$. *Parse $\vec{d}$ as a vector (pd, d) and $\vec{pk}$ as a vector $(pk, \mathfrak{m}, \rho)$, outputting $\perp$ if parsing fails or $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) = \perp \vee v \notin \{1, \ldots, nc\} \vee \{1, \ldots, nc\} \not\subseteq \mathfrak{m}$, compute*

  $$\begin{aligned}&c_1 \xleftarrow{s} \mathsf{Enc}(pk, g^{-d});\\ &c_2 \xleftarrow{t} \mathsf{Enc}(pk, v);\\ &\sigma_1 \leftarrow \mathsf{ProveCiph}((pk, g, c_1, \mathfrak{m}), (-d, s), m, \kappa);\\ &\sigma_2 \leftarrow \mathsf{ProveCiph}((pk, c_2, \{1, \ldots, nc\}), (v, t), m, \kappa),\end{aligned}$$

  and output $(pd, c_1, c_2, \sigma_1, \sigma_2, cnt)$, where message $m = (pd, c_1, c_2, cnt)$ and g is the aforementioned generator of message space $\mathfrak{m}$.

- $\mathsf{Tally}(\vec{sk}, \mathfrak{bb}, nc, L, \kappa)$. *Parse $\vec{sk}$ as vector (pk, sk), initialise $\mathfrak{v}$ as a zero-filled vector of length nc, and proceed as follows.*

  1. *Remove invalid ballots: Let $\{b_1, \ldots, b_\ell\}$ be the largest subset of senary vectors in $\mathfrak{bb}$ such that $b_1[1] \leq \cdots \leq b_\ell[1]$ and for each $(pd, c_1, c_2, \sigma_1, \sigma_2, cnt)$ in the subset we have $pd \in L \wedge \mathsf{VerCiph}((pk, g, c_1, \mathfrak{m}), \sigma_1, m, \kappa) \wedge \mathsf{VerCiph}((pk, c_2, \{1, \ldots, nc\}), \sigma_2, m, \kappa)$, where g is again the aforementioned generator of message space $\mathfrak{m}$ and message $m = (pd, c_1, c_2, cnt)$. If the subset is empty, then output $(\mathfrak{v}, \perp)$.*
  2. *Mix final votes: Initialise* $\mathbf{pfr}$ *as an empty vector and* $\mathbf{A}$ *as an empty map from pairs (comprising a ciphertext and a group element) to triples (comprising a counter and two ciphertexts), generate nonce n, compute*

```
for $1 \le i \le \ell$ do
    $c'_i \leftarrow \bigotimes_1^n b_i[2]$;
    $N \leftarrow \mathsf{Dec}(sk, c'_1)$;
    $\mathbf{t} \leftarrow \mathbf{A}[(b_i[1], N)]$;
    if $\mathbf{t} = \mathtt{null} \vee \mathbf{t}[1] < b_i[6]$ then
        // Update the map if $\mathbf{A}[(b_i[1], N)]$ is empty
        // or contains a lower counter
        $\mathbf{A}[(b_i[1], N)] \leftarrow (b_i[6], b_i[1] \otimes b_i[2], b_i[3])$;
    else if $\mathbf{t}[1] = b_i[6]$ then
        // Disregard duplicate counters
        $\mathbf{A}[(b_i[1], N)] \leftarrow (b_i[6], \perp, \perp)$;
    $\varsigma \leftarrow \mathsf{ProveDec}((pk, c'_i, N), sk, \kappa)$;
    if $|\mathbf{pfr}| > 0$ then
        // Prove $c'_{i-1}$ and $c'_i$ are derived by iterative
        // homomorphic combination wrt nonce $n$
        $\omega \leftarrow \mathsf{ProveComb}((pk, (c'_{i-1}, c'_i), (b_{i-1}[2], b_i[2])), n, \kappa)$;
        $\mathbf{pfr} \leftarrow \mathbf{pfr} \parallel (c'_i, N, \varsigma, \omega)$;
    else
        $\mathbf{pfr} \leftarrow \mathbf{pfr} \parallel (c'_i, N, \varsigma)$,
```

and apply (pairwise) mixnet $\mathcal{M}$ to the pairs of ciphertexts in map $\mathbf{A}$ to derive vector $\mathbf{B}$.

3. *Reveal eligible votes: Initialise* $\mathbf{pfd}$ *as an empty vector, generate nonces $n_1, \dots, n_{|\mathbf{B}|}$, and compute*

```
for $(c_1, c_2) \in \mathbf{B}$ do
    $c' \leftarrow \bigotimes_1^{n_{|\mathbf{pfd}|+1}} c_1$;
    $m \leftarrow \mathsf{Dec}(sk, c')$;
    $\omega \leftarrow \mathsf{ProveComb}((pk, c', c_1), n_{|\mathbf{pfd}|+1}, \kappa)$;
    $\varsigma_1 \leftarrow \mathsf{ProveDec}((pk, c', m), sk, \kappa)$;
    if $m = 1$ then
        // $c_1$ encrypts $g^0$, hence, is derived from homo
        // comb of pub cred and enc of neg priv cred
        $v \leftarrow \mathsf{Dec}(sk, c_2)$;
        $\mathfrak{v}[v] \leftarrow \mathfrak{v}[v] + 1$;
        $\varsigma_2 \leftarrow \mathsf{ProveDec}((pk, c_2, v), sk, \kappa)$;
        $\mathbf{pfd} \leftarrow \mathbf{pfd} \parallel (c', v, \omega, \varsigma_1, \varsigma_2)$;
    else
        $\mathbf{pfd} \leftarrow \mathbf{pfd} \parallel (c', m, \omega, \varsigma_1)$,
```

and output $(\mathfrak{v}, (\mathbf{pfr}, \mathbf{B}, \mathbf{pfd}))$, where g is the aforementioned generator of message space $\mathfrak{m}$.

- $\mathsf{Verify}(\vec{pk}, \mathfrak{bb}, nc, L, \mathfrak{v}, pf, \kappa)$. *Parse $\vec{pk}$ as vector $(pk, \mathfrak{m}, \rho)$ and $\mathfrak{v}$ as a vector of length nc, outputting 0 if parsing fails,* $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) = \perp$, *or $nc \not\leq mc$, where mc is computed as per algorithm* Setup. *Perform the* *following checks.*

1. *Check ballot removal. Compute* $\{b_1, \dots, b_\ell\}$ *as per Step 1 of algorithm* Tally *and check* $\{b_1, \dots, b_\ell\} = \emptyset$ *implies* $\mathfrak{v}$ *is a zero-filled vector.*
2. *Check mix. Check pf parses as a vector* $(\mathbf{pfr}, \mathbf{B}, \mathbf{pfd})$ *and* $\mathbf{pfr}$ *parses as a vector* $((c'_1, N_1, \varsigma_1), (c'_2, N_2, \varsigma_2, \omega_2), \dots, (c'_\ell, N_\ell, \varsigma_\ell, \omega_\ell))$ *such that* $\bigwedge_{1 \le i \le \ell} \mathsf{VerDec}((pk, c'_i, N_i), \varsigma_i, \kappa)$ *and* $\bigwedge_{1 < i \le \ell} \mathsf{VerComb}((pk, (c'_{i-1}, c'_i), (b_{i-1}[2], b_i[2])), \omega_i, \kappa)$*, initialise* $\mathbf{A}$ *as an empty map from pairs to triples, compute*

   ```
   for $1 \le i \le \ell$ do
       $\mathbf{t} \leftarrow \mathbf{A}[(b_i[1], N_i)]$;
       if $\mathbf{t} = \texttt{null} \vee \mathbf{t}[1] < b_i[6]$ then
           $\mathbf{A}[(b_i[1], N)] \leftarrow (b_i[6], b_i[1] \otimes b_i[2], b_i[3])$,
       else if $\mathbf{t}[1] = b_i[6]$ then
           $\mathbf{A}[(b_i[1], N)] \leftarrow (b_i[6], \perp, \perp)$;
   ```

   and check $\mathbf{B}$ *was output by the mix applied in Step 2 of algorithm* Tally *on input of the pairs of ciphertexts in* $\mathbf{A}$.
3. *Check revelation. Checks* $\mathbf{pfd}$ *parses as a vector of length* $|\mathbf{B}|$ *such that for each* $v \in \{1, \dots, nc\}$ *we have*

   $$\begin{aligned}\exists^{=\mathfrak{v}[v]} i \in \{1, \dots, |\mathbf{B}|\} : \exists c_1, c_2, c', \omega, \varsigma_1, \varsigma_2 : (c_1, c_2) = \mathbf{B}[i] \wedge \\ (c', v, \omega, \varsigma_1, \varsigma_2) = \mathbf{pfd}[i] \wedge \mathsf{VerComb}((pk, c', c_1), \omega, \kappa) \wedge \\ \mathsf{VerDec}((pk, c', 1), \varsigma_1, \kappa) \wedge \mathsf{VerDec}((pk, c_2, v), \varsigma_2, \kappa),\end{aligned}$$

   and for each remaining integer $i \in \{1, \dots, |\mathbf{B}|\}$ *we have* $\mathbf{B}[i]$ *parses as* (c_1, c_2)*,* $\mathbf{pfd}[i]$ *parses as* $(c', m, \omega, \varsigma_1)$*, and* $\mathsf{VerComb}((pk, c', c_1), \omega, \kappa) \wedge \mathsf{VerDec}((pk, c', m), \varsigma_1, \kappa) \wedge m \neq 1$.

Output 1 *if all the above checks hold.*

Athena is specified in terms of *election scheme* syntax by Smyth, Frink & Clarkson [69], which we extend to include re-voting (Appendix B). Election schemes must satisfy a correctness condition that ensures such schemes function, i.e., election outcomes correspond to votes expressed in ballots (except for votes expressed in ballots from the same voter, which are all discarded, except for the voter's last vote), when ballots are constructed and tallied in the prescribed manner, and we prove Athena satisfies the condition.

Lemma 1. $\mathsf{Athena}(\Pi, \mathcal{M}, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ *is an election scheme when cryptographic primitives satisfy the preconditions of Definition 1 and asymmetric encryption scheme* Π *is perfectly correct.*

A proof of Lemma 1 appears in Appendix C. Beyond correctness, verifiable election schemes should satisfy *completeness*, i.e., auditing should succeed for evidence produced by tallying, hence, algorithm Verify should accept outputs of algorithm Tally. We prove Athena satisfies completeness in Section 4.1.

Athena should be instantiated with an asymmetric encryption scheme satisfying IND-CPA and sigma protocols satisfying special soundness and special honest verifier zero-knowledge. This ensures the non-interactive proof systems derived by application of the Fiat-Shamir transformation satisfy zero-knowledge and simulation sound extractability [12], which help achieve both privacy and verifiability. Moreover, this ensures that ballots are non-malleable [12], which is necessary for privacy [65]. Furthermore, for linear complexity, we require computation $\bigotimes_1^n c$ to be linear in the length of c, which is possible for El Gamal, for instance. (Indeed, we have $\bigotimes_1^n (g^r, (g^x)^r \cdot M) \equiv (g^r, (g^x)^r \cdot M)^n \equiv (g^{r \cdot n}, (g^x)^{r \cdot n} \cdot M^n)$.)

Implementation. Athena is formally stated independently of the underlying cryptographic primitives (for generality, algorithm agility, and ease of proofs). In practice, $\mathsf{Athena}(\Pi, \mathcal{M}, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ can be instantiated with established cryptographic primitives. For instance, we might instantiate asymmetric encryption scheme Π as El Gamal [28] and we might instantiate sigma protocols as follows: Σ_1 as the protocol for proving knowledge of discrete logarithms by Chaum *et al.* [15, Protocol 2], Σ_2 as the protocol for proving knowledge of disjunctive equality between discrete logarithms by Cramer *et al.* [26, Figure 1], Σ_3 as the protocol for proving knowledge of equality between discrete logarithms by Chaum & Pedersen [16, §3.2], and Σ_4 as a slight variant of the protocol by Chaum & Pedersen.[1]

3 Privacy results

An Athena ballot contains a public credential, i.e., an encryption of the corresponding private credential, and an encryption of the negated private credential. Yet, no voter can prove that any ballot contains their private credential. Indeed, a well-formed Athena ballot that encrypts the negation of a voter's private credential is indistinguishable from an ill-formed ballot that encrypts some other value, rather than such a negation. Hence, during the voting phase, a voter cannot prove whether they cast a well-formed ballot (that will be counted, as opposed to an ill-formed ballot that will not), let alone prove how they voted, thereby assuring coercion-resistance during the voting phase.

Associating each public credential with anonymised credentials (to discard early votes prior to mixing) reveals the number of ballots whose second ciphertext contains the same plaintext (be that a private credential or some other value). For instance, a voter that casts a specific number of ballots containing such a plaintext can check to see whether an anonymised credential appears

[1] To prove iterative homomorphic combination using equality between discrete logarithms, witness that $\bigwedge_{1 \leq i \leq n} (\alpha_i, \beta_i) = (\alpha'_i, \beta'_i)^n$ iff $\bigwedge_{1 \leq i \leq n} \log_{\alpha'_i} \alpha_i \equiv \log_{\beta'_i} \beta_i \wedge \bigwedge_{1 < i \leq n} \log_{\alpha_{i-1}} \alpha_{i-1} \equiv \log_{\alpha_i} \alpha_i$, where $(\alpha_1, \beta_1), (\alpha'_1, \beta'_1), \ldots, (\alpha_n, \beta_n), (\alpha'_n, \beta'_n)$ are El Gamal ciphertexts.

the specified number of times (in association with the voter's public credential). But, no voter can prove that those ballots are well-formed. Indeed, the voter may cast the expected number of ballots using a nonce in place of their private credential's negation, which will result in the expected relation, yet the ballots are ill-formed and will not be counted. Hence, coercion-resistance is preserved before mixing.

Finally, homomorphically combining ciphertexts, mixing those combinations and encrypted votes, and using plaintext equality tests to determine voters' votes (as opposed to adversarial votes) preserves coercion-resistance, as does decrypting mixed (voters') votes. (Ballots prove that votes are selected from the sequence of candidates, which provides protection against randomisation attacks [41, §1.1].) It follows that tallying preserves coercion-resistance. Thus, Athena is a coercion-resistant voting system.

The desire to formally prove that Athena satisfies coercion resistance initiated a study of definitions by Juels, Catalano & Jakobsson [39–41], Gardner, Garera & Rubin [32], Unruh & Müller-Quade [72], and Küsters, Truderung & Vogt [48, 51]. The study reveals that definitions by Gardner, Garera & Rubin and Unruh & Müller-Quade are satisfiable by voting systems that are not coercion resistant, and that the definition by Küsters, Truderung & Vogt is unsatisfiable by systems that are [68]. Hence, those definitions do not adequately formalise coercion resistance and are unsuitable for the analysis of Athena. It remains to study the definition by Juels, Catalano & Jakobsson, and a formal proof that Athena satisfies coercion resistance is deferred until the suitability of their definition (or another) is established.

Distributed tallying. Coercion-resistance does not provide assurances when deviations from the prescribed tallying procedure are possible. Indeed, such deviations include revealing the tallier's private key, which undermines privacy. Hence, the tallier must be trusted. Alternatively, we can design voting systems that distribute the tallier's role amongst several talliers and ensure free-choice assuming at least one tallier behaves. Extending Athena in this direction is straightforward, since distributed variants of the underlying primitives are well-known. Ultimately, we would prefer not to trust talliers; unfortunately, this is only known to be possible for decentralised voting systems, e.g., [33, 36, 43–45, 61], which do not scale.

4 Verifiability results

Athena records ballots on a (public) bulletin board, hence, voters can check whether their ballot is collected (individual verifiability). Moreover, tallying produces evidence demonstrating that the announced election outcome corresponds to the votes expressed in collected ballots that are authorised (universal verifiability). Furthermore, only voters can construct authorised ballots (unforgeability). It follows that Athena is a verifiable election scheme, as we shall prove using formal definitions from Smyth, Frink & Clarkson [69] that we extend to include re-voting (Appendix D).

4.1 Universal verifiability

Universal verifiability asserts that anyone must be able to check whether an election outcome corresponds to votes expressed in collected ballots that are authorised. Since checks can be performed by algorithm Verify, it suffices that the algorithm accept if and only if the outcome corresponds to votes expressed in collected ballots that are authorised. The *only if* requirement is formalised by Soundness (Definition 10), which requires algorithm Verify to only accept correct outcomes, and the *if* requirement is captured by Completeness (Definition 11), which requires election outcomes produced by algorithm Tally to be accepted by algorithm Verify.

Proposition 2 (Soundness). *Election scheme* Athena$(\Pi, \mathcal{M}, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ *satisfies* Soundness*, when asymmetric encryption scheme* Π *is perfectly correct, mixnet* $\mathcal{M}$ *is verifiable, sigma protocols* Σ_1*,* Σ_2*,* Σ_3 *and* Σ_4 *satisfy special soundness and special honest verifier zero-knowledge, and* $\mathcal{H}$ *is a random oracle, assuming* Injectivity *is satisfied.*

We defer consideration of Injectivity (Definition 9) to Section 4.3.

Proof sketch. We must establish that outcomes accepted by algorithm Verify correspond to votes expressed in collected ballots that are authorised. Step 1 of the algorithm ensures accepted outcomes are only influenced by bulletin board entries constructed by algorithm Vote, i.e., only (valid) ballots have influence (invalid ballots do not), and only when they contain a public credential. Step 2 ensures no influence from any mixed ballots that share a public credential (and a anonymised credential) with another mixed ballot, whilst being associated with a (strictly) lower counter value. Moreover, pairs of mixed ballots that share a public credential (and a anonymised credential) and a counter are ensured to have no influence either. It follows that only mixed ballots expressing the last vote associated with a public and an anonymised credential may have influence. Finally, Step 3 restricts influence to mixed ballots associated with a voter's public and private credential, i.e., only voters' last votes have influence, hence, accepted outcomes correspond to votes expressed in authorised collected ballots. □

A detailed proof of Proposition 2 and all other verifiability proofs appear in Appendix E.

Proposition 3 (Completeness). *Election scheme* Athena$(\Pi, \mathcal{M}, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ *satisfies* Completeness *when mixnet* $\mathcal{M}$ *is verifiable, sigma protocol* Σ_2 *satisfies special soundness and special honest verifier zero-knowledge, and* $\mathcal{H}$ *is a random oracle.*

Proof sketch. We must establish that outcomes produced by algorithm Tally are accepted by algorithm Verify. It is trivial to see that an outcome output by the first step of algorithm Tally will be accepted by the first step of algorithm Verify, and it remains to consider outcomes output by the last step of algorithm Tally. Simulation sound extractability (of sigma protocol Σ_2) assures us that such outcomes are derived from ballots containing well-formed ciphertexts. It is straightforward to see that computations performed by the second step of algorithm Tally can be successfully checked in the second step of algorithm Verify, in particular, proofs can be verified, because proof systems are complete. Moreover, since map $\mathbf{A}$ is equivalently computed (from well-formed ciphertexts) by both algorithms and since the mixnet is verifiable, it follows that checks performed on the mixnet's output succeed. Finally, the checks performed by the third step of algorithm Verify succeed, because proof systems are complete, thus, outcomes produced by algorithm Tally are accepted. □

4.2 Unforgeability

Unforgeability asserts that only voters can construct authorised ballots. Since ballots are authenticated by private credentials, it suffices to ensure that knowledge of a private credential is necessary to construct an authentic ballot, which is formalised by Unforgeability (Definition 12). We defer a formal proof to later work.

Comparison with the voting system by Juels, Catalano & Jakobsson. Smyth, Frink & Clarkson [69, §6] show that the voting system by Juels, Catalano & Jakobsson only achieves unforgeability assuming the tallier is honest, because the tallier's private key can be used to discover private credentials (by decrypting public credentials), which enables adversarial construction of authorised ballots. By comparison, Athena achieves unforgeability even if the tallier is dishonest, since the tallier's private key can only be used to recover g^d or g^{-d}, neither of which can be used to construct an authorised ballot, because ballots must prove knowledge of private credential d. Thus, Athena improves upon the security of the voting system by Juels, Catalano & Jakobsson. (Their voting system can probably be improved using a similar idea.)

4.3 Individual verifiability

Individual verifiability asserts that voters must be able to check whether their ballot is amongst those collected. Since ballots should be collected and recorded on a bulletin board, and since the board must be available to everyone, it suffices for voters to check that their ballot (i.e., the ballot they constructed) is on the bulletin board. Hence, it is necessary for voters to check that their ballot has not been omitted from the bulletin board. Yet, this is insufficient, because the presence of a ballot identical to a voter's ballot, does not imply the presence of the ballot constructed by the voter. Indeed, such a ballot might have been constructed by another voter. Thus, individual verifiability requires that voters must be able to uniquely identify their ballot, i.e., ballots do not collide, which is formalised by Individual-Verifiability (Definition 13).

To ensure Athena satisfies individual verifiability, it suffices to require that the underlying encryption scheme produces distinct ciphertexts with overwhelming probability. Smyth explains that "[s]ecurity properties of asymmetric encryption schemes ensure [distinct] ciphertexts...But, such security properties assume public keys are generated (by key generation algorithms) using coins chosen uniformly at random. By comparison, individual verifiability and injectivity assume public keys are constructed by the adversary. Thus, security properties are insufficient to ensure...individual verifiability and injectivity" [67]. Nonetheless, given that Athena checks correct key generation, it suffices that ciphertexts are distinct for correctly generated keys.

Proposition 4 (Individual-Verifiability). *Election scheme* $\mathsf{Athena}(\Pi, \mathcal{M}, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ *satisfies* Individual-Verifiability *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$ *and security parameters* κ *we have* $\Pr[(pk, \mathfrak{m}, \rho, m, m') \leftarrow \mathcal{A}(\kappa); c \leftarrow \mathsf{Enc}(pk, m); c' \leftarrow \mathsf{Enc}(pk, m') : \mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) = 1 \wedge m, m' \in \mathfrak{m} \Rightarrow c \neq c'] > 1 - \mathsf{negl}(\kappa)$*, where* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *and* $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$. *Moreover,* Injectivity *is satisfied if the probability is* 1 *when plaintexts* m *and* m' *are distinct.*

The preconditions used by Proposition 4 are due to Smyth [67, §3], and our proof is structurally similar to his proof of individual verifiability and injectivity for a class of encryption-based voting systems.

5 Complexity analysis

Analysing the complexity of algorithms Setup, Register, and Vote is straightforward (given the simplicity of those algorithm): Setup generates a key pair and proof of correct generation, Register computes a ciphertext, and Vote computes two ciphertexts along with proofs of correct construction. Hence, complexity of the registration phase is linear in the number of voters and complexity of the voting phase is linear in the number of ballots cast. Algorithms Tally and Verify are more elaborate and analysis is more involved. We proceed by a detailed inspection of each algorithm and find that complexity remains linear.

Tally. We consider each step of algorithm Tally: It is trivial to see that complexity is upper-bound by the bulletin board's length in Step 1. For Step 2, we have assumed computation $\bigotimes_1^n c$ is linear in the length of c (§2), hence, it is straightforward to see that complexity is upper-bound by the number of for-loop iterations, which is constrained by the number of valid ballots on the bulletin board, therefore, complexity is again upper-bound by the bulletin board's length, because the number of valid ballots is at most the number of ballots on the bulletin board. Complexity of Step 3 is similarly upper-bound by the number of for-loop iterations, which is constrained by the number of pairs output by the mix and at most the number of ballots on the bulletin board, therefore, complexity is upper-bound by the bulletin board's length.

Verify. We consider the steps of algorithm Verify: Complexity of Step 1 is trivially linear in the bulletin board's length; Step 2 is straightforwardly linear in the number of for-loop iterations, which is linear in the bulletin board's length; and Step 3 is straightforwardly linear in the number pairs output by the mix, which is again linear in the bulletin board's length.

Thus, Athena has linear complexity $\mathcal{O}(|\mathfrak{bb}|)$ in the length of the bulletin board ($\mathfrak{bb}$), assuming linear complexity of iterative homomorphic combinations (§2), which is possible for El Gamal, for instance.

Comparison with the voting system by Juels, Catalano & Jakobsson. Complexity of the voting system by Juels, Catalano & Jakobsson is quadratic, due to pairwise plaintext equality tests performed on ballots to ensure that only the last choice of each voter has influence, which is needed for universal verifiability. By comparison, Athena uses anonymised credentials in a manner that achieves the same property, whilst reducing complexity. (The voting system by Juels, Catalano & Jakobsson also performs pairwise plaintext equality tests on mixed ballots and mixed credentials, to identify authorised ballots. The complexity of those tests is upper-bound by $\mathcal{O}(|L| \cdot |\mathfrak{bb}|)$, where L is the electoral roll and $|\mathfrak{bb}|$ is the bulletin board's length. By comparison, the plaintext equality tests performed by Athena are upper-bound by $\mathcal{O}(|\mathfrak{bb}|)$.)

6 General design principles

General design principles were identified and embraced during the development of Athena. This section shares these principles to aid the development of future voting systems, especially those with linear complexity. Our first design principle is guided by definitions of correctness and universal verifiability:

1. Algorithm Tally must map votes expressed in authorised ballots to the outcome corresponding to those votes, except for any early votes.

It follows that:

2. Algorithm Vote must authenticate ballots.

The next two design principles follow from our informal definitions of ballot secrecy and coercion resistance, respectively:

3. Algorithm Vote must ensure votes cannot be revealed from ballots; and

4. Algorithm Tally must not reveal any (meaningful) mapping between ballots and the outcome.

Forgoing coercion resistance (in favour of ballot secrecy), the previous design principle can be generalised: Algorithm Tally must not reveal any (meaningful) mapping between *authorised* ballots and the outcome. But this permits revealing authorised ballots, which allows *simulation attacks* [41, §1.1], whereby a coercer instructs a voter to reveal their private credential, uses that private credential to cast a ballot, and determines whether the voter followed instruction by checking whether the cast ballot is authorised. By comparison, (4) gives way to the following design principle:

5. Algorithm Tally must not reveal authorised ballots.

Similarly, authorised ballots must not be revealed during casting and collection:

6. Algorithm Vote must not reveal authorised ballots.

Since ballots must be authenticated (1) without revealing authorised ballots (5 & 6), the following principle emerges:

7. Algorithm Tally should anonymise ballots prior to authentication.

Given that ballots should be anonymised (7) and that bulletin boards may contain more than just ballots, it is proposed that:

8. Algorithm Vote should prove correct ballot construction; and

9. Algorithm Tally should discard garbage, i.e., non-ballots.

Revealing re-votes after anonymisation can be problematic, for instance, a voter that casts a specific number of ballots can deanonymise their anonymised ballots. Thus, our final design principle is suggested:

10. Algorithm Tally should discard ballots representing early votes prior to anonymisation.

For compatibility between (7 & 10) and (4), our notion of *meaningful* should exclude garbage and early votes, i.e., algorithm Tally is permitted to reveal mappings between the outcome and garbage, ballots representing early votes, or both.

By combining our design principles, we observe that algorithm Tally should filter the bulletin board to remove garbage (9) and ballots representing early votes (10). Moreover, after anonymising any remaining ballots, the algorithm should authenticate anonymised ballots and remove any unauthorised anonymous ballots (7). Finally, votes expressed in any authenticated anonymous ballots should be mapped to the outcome corresponding to those votes (1).

7 Related work

Acquisti [1], Smith [64], and Weber, Araújo & Buchmann [73] reduce complexity to linear in variants of the voting system by Juels, Catalano & Jakobsson, but those reductions led to the lose of coercion-resistance [3–6, 21]. Araújo *et al.* [3–5] make better progress, albeit, without supporting audits for statistically determining whether non-voters are issued with credentials [6, 60, 70] and without supporting reuse of credentials between elections [2, 6]. Haghighat, Dousti & Jalili do not permit reuse either [35], whereas Araújo *et al.* do [2, 6], albeit, Araújo *et al.* do not achieve *strong non-reusability* (i.e., only the last choice of each voter has influence [54]) nor universal verifiability, in the presence of an adversary that can re-order ballots (e.g., a network adversary), because they are reliant on ballot order to discard early votes. (The voting system by Juels, Catalano & Jakobsson does not satisfy strong non-reusability nor universal verifiability against such an adversary either, whereas Civitas does [69, §4.2.2].) Schläpfer *et al.* and Spycher *et al.* also make progress, albeit, Schläpfer *et al.* only achieve linear complexity for a trade in the degree of coercion-resistance and they leak the number of ballots each voter casts [60] and Spycher *et al.* make an additional trust assumption, namely, they assume the tallier introduces a secret number of dummy votes for each voter (without any means for voters to confirm they did) [70]. Beyond variants of the voting system by Juels, Catalano & Jakobsson, distinct voting systems have also been introduced: The system by Schweisgut [63] achieves linear-complexity, but fails to achieve coercion-resistance [5]. Clark & Hengartner propose the *Selections* voting system, which makes better progress, albeit, some degree of coercion-resistance is traded to achieve linear complexity and the number of ballots each voter casts is leaked [18, 19]. (Athena leaks the number of ballots cast in association with each public credential, but not the number of ballots each voter casts.) Finally, Essex, Clark & Hengartner propose the *Cobra* voting system, which achieves remarkably fast tallying, albeit, registration has quadratic complexity in the number of voters [29]. With the exception of Juels, Catalano & Jakobsson, Haghighat, Dousti & Jalili, and Clark & Hengartner, none of these prior works present security proofs and proving their security remains an open problem. (Araújo *et al.* [2, 5] formally state theorems, but defer proofs to full versions of their papers, which do not appear to be public.)

8 Conclusion

For one and a half decades, researchers have strived to improve upon seminal work by Juels, Catalano & Jakobsson. This work attempts to deliver such an improvement: A verifiable, coercion-resistant voting system with linear complexity. We have seen how several of the ideas can help improve security of existing voting systems. Moreover, they generalise beyond voting to other systems that require strong forms of privacy, authentication, and verifiability, thereby advancing not just voting technology, but the science of security.

A Cryptographic primitives

A.1 Asymmetric encryption

Definition 2 (Asymmetric encryption scheme [42])**.** *An* asymmetric encryption scheme *is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, *such that:*

- **Gen***, denoted* $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa)$*, inputs a security parameter* κ *and outputs a key pair* (pk, sk) *and message space* $\mathfrak{m}$.

- **Enc***, denoted* $c \leftarrow \mathsf{Enc}(pk, m)$*, inputs a public key* pk *and message* $m \in \mathfrak{m}$*, and outputs a ciphertext* c.

- **Dec***, denoted* $m \leftarrow \mathsf{Dec}(sk, c)$*, inputs a private key* sk *and ciphertext* c*, and outputs a message* m *or an error symbol. We assume* Dec *is deterministic.*

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

Definition 3 (Homomorphic encryption [69])**.** *An asymmetric encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* homomorphic*, with respect to ternary operators* $\odot$*,* $\oplus$*, and* $\otimes$*,*[2] *if there exists a negligible function* negl*, such that for all security parameters* κ*, we have the following.*[3] *First, for all messages* m_1 *and* m_2 *we have* $\Pr[(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa); c_1 \leftarrow \mathsf{Enc}(pk, m_1); c_2 \leftarrow \mathsf{Enc}(pk, m_2) : m_1, m_2 \in \mathfrak{m} \Rightarrow \mathsf{Dec}(sk, c_1 \otimes_{pk} c_2) = \mathsf{Dec}(sk, c_1) \odot_{pk} \mathsf{Dec}(sk, c_2)] > 1 - \mathsf{negl}(\kappa)$*. Secondly, for all messages* m_1 *and* m_2*, and all coins* r_1 *and* r_2*, we have* $\Pr[(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa) : m_1, m_2 \in \mathfrak{m} \Rightarrow \mathsf{Enc}(pk, m_1; r_1) \otimes_{pk} \mathsf{Enc}(pk, m_2; r_2) = \mathsf{Enc}(pk, m_1 \odot_{pk} m_2; r_1 \oplus_{pk} r_2)] > 1 - \mathsf{negl}(\kappa)$*. We say* Π *is* multiplicative homomorphic*, if for all security parameters* κ*, key pairs* pk, sk*, and message spaces* $\mathfrak{m}$*, such that there exists coins* r *and* $(pk, sk, \mathfrak{m}) = \mathsf{Gen}(\kappa; r)$*, we have* $\odot_{pk}$ *is the multiplication operator in group* $(\mathfrak{m}, \odot_{pk})$.

A.2 Proof systems

Definition 4. *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a homomorphic asymmetric encryption scheme and* Σ *be a sigma protocol for a binary relation* R*.*[4]

[2] For brevity, 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* $\otimes$.

[3] 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.

[4] 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.

- Σ proves key generation [69] *if a* $((\kappa, pk, \mathfrak{m}), (sk, s)) \in R \Leftrightarrow (pk, sk, \mathfrak{m}) = \mathsf{Gen}(\kappa; s)$*.*

Further, suppose that $(pk, sk, \mathfrak{m})$ *is the output of* $\mathsf{Gen}(\kappa; s)$*, for some security parameter* κ *and coins* s*.*

- Σ proves ciphertext construction *if* $((pk, c, \mathfrak{m}'), (m, r)) \in R \Leftrightarrow c = \mathsf{Enc}(pk, m; r) \wedge m \in \mathfrak{m}' \wedge \mathfrak{m}' \subseteq \mathfrak{m}$ [69]*, or* $((pk, g, c, \mathfrak{m}'), (m, r)) \in R \Leftrightarrow c = \mathsf{Enc}(pk, g^m; r) \wedge m \in \mathfrak{m}' \wedge \mathfrak{m}' \subseteq \mathfrak{m}$*, where* g *is a generator of message space* $\mathfrak{m}$*.*

- Σ proves decryption [69] *if* $((pk, c, m), sk) \in R \Leftrightarrow m = \mathsf{Dec}(sk, c)$*.*

- Σ proves iterative homomorphic combination *if* $((pk, \mathbf{c}, \mathbf{c}'), n) \in R \Leftrightarrow \bigwedge_{1 \leq i \leq |\mathbf{c}|} \mathbf{c}[i] = \bigotimes_1^n \mathbf{c}'[i] \wedge |\mathbf{c}| = |\mathbf{c}'|$*.*[5]

Definition 5 (Non-interactive proof system [69])**.** *A* non-interactive proof system *for a relation* R *is a tuple of algorithms* $(\mathsf{Prove}, \mathsf{Verify})$*, such that:*

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

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

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

Definition 6 (Fiat-Shamir transformation [30])**.** *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** $\mathsf{Verify}_\Sigma(s, (\mathsf{comm}, \mathsf{chal}, \mathsf{resp}), \kappa);$
```

A string m *can be included in the hashes computed by algorithms* Prove *and* Verify*. That is, the hashes are computed in both algorithms as* $\mathsf{chal} \leftarrow \mathcal{H}(\mathsf{comm}, s, m)$*. 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 string* m*.*

[5] We write $\mathsf{ProveComb}((pk, c, c'), n, \kappa)$ for $\mathsf{ProveComb}((pk, (c), (c')), n, \kappa)$ when c and c' are ciphertexts (rather than vectors), where $(\mathsf{ProveComb}, \mathsf{VerComb}) = \mathsf{FS}(\Sigma, \mathcal{H})$ for a sigma protocol Σ that proves iterative homomorphic combination and a hash function $\mathcal{H}$.

Definition 7 (Simulation sound extractability [12, 34, 69]). *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:*[6]

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

where $\mathcal{A}(\text{—}; r)$ 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 $\mathbf{Q}' \leftarrow \mathcal{A}(\text{—}; r)$, forwarding any of $\mathcal{A}$'s oracle queries to $\mathcal{K}$, and outputs $\mathbf{Q}'$. By running $\mathcal{A}(\text{—}; r)$, $\mathcal{K}$ 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 5 (from [12]). *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 simulation sound extractability.*

The Fiat-Shamir transformation may include a string in the hashes computed by functions Prove and Verify. Simulators can be generalised to include such a string too. We write $\mathcal{S}(s, m, \kappa)$ for invocations of simulator $\mathcal{S}$ which include string m. And remark that Theorem 5 can be extended to this generalisation.

B Election scheme syntax

We extend syntax by Smyth, Frink & Clarkson [69] to include re-voting, thereby capturing voting systems that consist of the following four steps. First, a tallier generates a key pair and a registrar generates credentials for voters. Secondly, each voter constructs and casts a ballot for their vote, and similarly for any re-votes. These ballots are collected and recorded on a bulletin board. Thirdly, the tallier tallies the collected ballots and announces the outcome as a frequency distribution of votes. The chosen representative is derived from this

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

distribution, e.g., as the candidate with the most votes. Finally, voters and other interested parties check that the outcome corresponds to votes expressed in collected ballots.

Definition 8 (Election scheme)**.** *An* election scheme *is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *such that:*[7]

Setup*, denoted* $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$*, is run by the tallier. The algorithm 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*.*

$\mathsf{Register}$*, denoted* $(pd, d) \leftarrow \mathsf{Register}(pk, \kappa)$*, is run by the registrar. The algorithm takes as input a public key* pk *and a security parameter* κ*, and it outputs a* credential pair (pd, d)*, where* pd *is a public credential and* d *is a private credential.*

Vote*, denoted* $b \leftarrow \mathsf{Vote}(d, pk, v, cnt, nc, \kappa)$*, is run by voters. The algorithm takes as input a private credential* d*, a public key* pk*, a voter's vote* v*, a counter* cnt*, some number of candidates* nc*, and a security parameter* κ*. Vote* v *should be selected from a sequence* $1, \dots, nc$ *of candidates, and counter* cnt *should be incremented between a voter's runs. (The counter might be a timestamp which increments with time or an integer that is manually incremented, for instance.) The algorithm outputs a ballot* b *or error symbol* $\perp$*.*

Tally*, denoted* $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, L, \kappa)$*, is run by the tallier. The algorithm takes as input a private key* sk*, a bulletin board* $\mathfrak{bb}$*, some number of candidates* nc*, an electoral roll* L*, and a security parameter* κ*, where* $\mathfrak{bb}$ *is a set. The algorithm outputs an election outcome* $\mathfrak{v}$ *and a non-interactive tallying proof* pf*, where* $\mathfrak{v}$ *is a vector of length* nc *and each index* v *of that vector should indicate the number of votes for candidate* v*. Moreover, the tallying proof should demonstrate that the outcome corresponds to votes expressed in ballots on the bulletin board.*

Verify*, denoted* $s \leftarrow \mathsf{Verify}(pk, \mathfrak{bb}, nc, L, \mathfrak{v}, pf, \kappa)$*, is run to audit an election. The algorithm takes as input a public key* pk*, a bulletin board* $\mathfrak{bb}$*, some number of candidates* nc*, an electoral roll* L*, an election outcome* $\mathfrak{v}$*, a tallying proof* pf*, and a security parameter* κ*. The algorithm outputs a bit* s*, which is* 1 *if the outcome should be accepted and* 0 *otherwise. We require the algorithm to be deterministic.*

[7] The syntax bounds the number of ballots mb, respectively candidates mc, to broaden the correctness definition's scope (indeed, voting systems that encrypt votes typically require mc to be less than or equal to the size of the encryption scheme's message space and schemes that homomorphically combine votes require mb to be less than or equal to the size of that space). The syntax represents votes as integers, rather than alphanumeric strings, for brevity. Finally, the syntax employs sets, rather than multisets or lists, to preclude the construction of schemes vulnerable to attacks that arise due to duplicate ballots [13, §2.1 & §4.3] (systems vulnerable to such attacks cannot be modelled using the syntax).

Election schemes must satisfy correctness*: there exists a negligible function* negl*, such that for all security parameters* κ*, integers* nv *and* nc*, vectors of votes* $\mathbf{v}_1, \ldots, \mathbf{v}_{nv}$ *over* $\{1, \ldots, nc\}$*, and vectors of counters* $\mathbf{c}_1, \ldots, \mathbf{c}_{nv}$ *such that* $\bigwedge_{1 \leq i \leq nv} |\mathbf{v}_i| = |\mathbf{c}_i| \wedge \mathbf{c}_i[1] < \cdots < \mathbf{c}_i[|\mathbf{c}_i|]$*, it holds that, given a zero-filled vector* $\mathfrak{v}$ *of length* nc*, we have:*

```
$\Pr[(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa);$
    $\mathfrak{bb} \leftarrow \emptyset;$
    for $1 \leq i \leq nv$ do
        $(pd_i, d_i) \leftarrow \mathsf{Register}(pk, \kappa);$
        if $0 < |\mathbf{v}_i|$ then
            for $1 \leq j \leq |\mathbf{v}_i|$ do
                $b_j \leftarrow \mathsf{Vote}(d_i, pk, \mathbf{v}_i[j], \mathbf{c}_i[j], nc, \kappa);$
            $\mathfrak{bb} \leftarrow \mathfrak{bb} \cup \{b_1, \ldots, b_{|\mathbf{v}_i|}\};$
            $\mathfrak{v}[\mathbf{v}_i[|\mathbf{v}_i|]] \leftarrow \mathfrak{v}[\mathbf{v}_i[|\mathbf{v}_i|]] + 1;$
    $(\mathfrak{v}', pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \{pd_1, \ldots, pd_{nb}\}, \kappa) :$
     $|\mathfrak{bb}| \leq mb \wedge nc \leq mc \Rightarrow \mathfrak{v} = \mathfrak{v}'] > 1 - \mathsf{negl}(\kappa).$
```

The syntax provides a language to model voting systems and the correctness condition ensures such systems function. Athena is defined in terms of this syntax, moreover, we will adopt definitions of verifiability and privacy expressed in the syntax and prove they are satisfied.

C Proof of correctness (Lemma 1)

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$, $\mathsf{FS}(\Sigma_2, \mathcal{H}) = (\mathsf{ProveCiph}, \mathsf{VerCiph})$, and $\mathsf{Athena}(\Pi, \mathcal{M}, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$.

Suppose κ is a security parameter, nv and nc are integers, $\mathbf{v}_1, \ldots, \mathbf{v}_{nv}$ are vectors over $\{1, \ldots, nc\}$, $\mathbf{c}_1, \ldots, \mathbf{c}_{nv}$ are vectors such that $\bigwedge_{1 \leq i \leq nv} |\mathbf{v}_i| = |\mathbf{c}_i| \wedge \mathbf{c}_i[1] < \cdots < \mathbf{c}_i[|\mathbf{c}_i|]$, and $\mathfrak{v}$ is a zero-filled vector of length nc. Further suppose we compute:

```
$(\mathbf{pk}, \mathbf{sk}, mb, mc) \leftarrow \mathsf{Setup}(\kappa);$
$\mathfrak{bb} \leftarrow \emptyset;$
for $1 \leq i \leq nv$ do
    $(pd_i, \mathbf{d}_i) \leftarrow \mathsf{Register}(pk, \kappa);$
    if $0 < |\mathbf{v}_i|$ then
        for $1 \leq j \leq |\mathbf{v}_i|$ do
            $b_{i,j} \leftarrow \mathsf{Vote}(\mathbf{d}_i, pk, \mathbf{v}_i[j], \mathbf{c}_i[j], nc, \kappa);$
        $\mathfrak{bb} \leftarrow \mathfrak{bb} \cup \{b_{i,1}, \ldots, b_{i,|\mathbf{v}_i|}\};$
        $\mathfrak{v}[\mathbf{v}_i[|\mathbf{v}_i|]] \leftarrow \mathfrak{v}[\mathbf{v}_i[|\mathbf{v}_i|]] + 1;$
```

If $|\mathfrak{bb}| \not\leq mb \vee nc \not\leq mc$, then correctness is trivially satisfied, otherwise ($|\mathfrak{bb}| \leq mb \wedge nc \leq mc$), we proceed as follows.

By definition of algorithm Setup, we have $\mathbf{pk}$ parses as vector $(pk, \mathfrak{m}, \rho)$ and $\mathbf{sk}$ parses as (pk, sk), where $(pk, sk, \mathfrak{m}) = \mathsf{Gen}(\kappa; r)$ for some coins r and ρ is an output of $\mathsf{ProveKey}((\kappa, pk, \mathfrak{m}), (sk, r), \kappa)$. Moreover, by completeness, we have $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa)$ holds. Let g be a generator of message space $\mathfrak{m}$. By definition of algorithm $\mathsf{Register}$, we have for each $i \in \{1, \ldots, nv\}$ that $pd_i = \mathsf{Enc}(pk, g^{d_i}; r_i)$ and $\mathbf{d}_i = (pd_i, d_i)$, for some coins r_i chosen uniformly at random and nonce d_i. Moreover, by definition of algorithm Vote, we have for each $i \in \{1, \ldots, nv\}$ and $j \in \{1, \ldots, |\mathbf{v}_i|\}$ that $b_{i,j}$ is a vector of length six, $b_{i,j}[1] = pd_i$,

$$b_{i,j}[2] = \mathsf{Enc}(pk, g^{-d_i}; s_{i,j}),$$
$$b_{i,j}[3] = \mathsf{Enc}(pk, \mathbf{v}_i[j]; t_{i,j}),$$

$b_{i,j}[4]$ is an output of $\mathsf{ProveCiph}((pk, g, b_{i,j}[2], \mathfrak{m}), (-d_i, s_{i,j}), m, \kappa)$, $b_{i,j}[5]$ is an output of $\mathsf{ProveCiph}((pk, b_{i,j}[3], \{1, \ldots, nc\}), (\mathbf{v}_i[j], t_{i,j}), m, \kappa)$, and $b_{i,j}[6] = \mathbf{c}_i[j]$, where $s_{i,j}$ and $t_{i,j}$ are coins chosen uniformly at random and $m = (pd_i, b_{i,j}[2], b_{i,j}[3], b_{i,j}[6])$. Let us consider the computation of $(\mathfrak{v}', pf)$ by $\mathsf{Tally}(sk, \mathfrak{bb}, nc, \{pd_1, \ldots, pd_{nb}\}, \kappa)$.

We have $\mathfrak{bb} = \bigcup_{1 \leq i \leq nv \wedge |\mathbf{v}_i| > 0} \{b_{i,1}, \ldots, b_{i,|\mathbf{v}_i|}\}$. Suppose a subset of that set is computed as per Step 1 of algorithm Tally. By completeness and since for each $i \in \{1, \ldots, nv\}$ we have $pd_i = b_{i,1}[1] = \cdots = b_{i,|\mathbf{v}_i|}[1]$, that subset is $\{b_{\pi(1),\pi_1(1)}, \ldots, b_{\pi(1),\pi_1(|\mathbf{v}_1|)}, \ldots, b_{\pi(nv),\pi_{nv}(1)}, \ldots, b_{\pi(nv),\pi_{nv}(|\mathbf{v}_{nv}|)}\}$ for some permutation π on $\{1, \ldots, nv\}$ and for each $i \in \{1, \ldots, nv\}$ some permutation π_i on $\{1, \ldots, |\mathbf{v}_i|\}$ such that $b_{\pi(1),\pi_1(1)}[1] \leq \cdots \leq b_{\pi(1),\pi_1(|\mathbf{v}_1|)}[1] \leq \cdots \leq b_{\pi(\ell),\pi_\ell(1)}[1] \leq \cdots \leq b_{\pi(\ell),\pi_\ell(|\mathbf{v}_\ell|)}[1]$. If $nv = 0 \vee \bigwedge_{1 \leq i \leq nv} |\mathbf{v}_i| = 0$, then $\mathfrak{v}$ and $\mathfrak{v}'$ are both zero-filled vectors of length nc, and we conclude immediately, otherwise, we proceed as follows.

Suppose ciphertexts, plaintexts, and a map are computed as per Step 2 of algorithm Tally, with respect to nonce n. Since Π is a multiplicative-homomorphic asymmetric encryption scheme, we have for each $i \in \{1, \ldots, nv\}$ and $j \in \{1, \ldots, |\mathbf{v}_i|\}$ that

$$c'_{i,j} = \bigotimes_{1}^{n} b_{i,j}[2] = \mathsf{Enc}(pk, \odot_1^n g^{-d_i}; \oplus_1^n s_{i,j}) \equiv \mathsf{Enc}(pk, g^{-d_i \cdot n}; \oplus_1^n s_{i,j}),$$

hence, by (perfect) correctness, we have

$$N_{i,j} = \mathsf{Dec}(sk, c'_{i,j}) \equiv g^{-d_i \cdot n},$$

where ciphertext $c'_{i,j}$ and plaintext $N_{i,j}$ are computed by algorithm Tally. (We require perfect correctness, because the adopted definition of homomorphic encryption only considers combination of distinct ciphertexts constructed from distinct coins, whereas we consider iterative combination of a single ciphertext.) Hence, for each $i \in \{1, \ldots, nv\}$ we have

$$N_{i,1} = \cdots = N_{i,|\mathbf{v}_i|}.$$

Since $d_1, \dots, d_{nv}$ are nonces, g is a generator of message space $\mathfrak{m}$, and $|\mathfrak{m}|$ is super-polynomial in the security parameter, we have $N_{1,|\mathbf{v}_1|}, \dots, N_{nv,|\mathbf{v}_{nv}|}$ are pairwise distinct, moreover, since $|\mathbf{v}_i| = |\mathbf{c}_i| \wedge \mathbf{c}_i[1] < \cdots < \mathbf{c}_i[|\mathbf{c}_i|]$ and $b_{i,j}[6] = \mathbf{c}_i[j]$ for $j \in \{1, \dots, |\mathbf{c}_i|\}$, we have for each $i \in \{1, \dots, nv\}$ that

$$\mathbf{A}[(pd_i, N_{i,1})] = (b_{i,|\mathbf{v}_i|}[6], b_{i,|\mathbf{v}_i|}[1] \otimes b_{i,|\mathbf{v}_i|}[2], b_{i,|\mathbf{v}_i|}[3]),$$

where map $\mathbf{A}$ is computed by algorithm Tally. It follows that map $\mathbf{A}$ is defined over ciphertexts $b_{1,|\mathbf{v}_1|}[1] \otimes b_{1,|\mathbf{v}_1|}[2], b_{1,|\mathbf{v}_1|}[3], \dots, b_{nv,|\mathbf{v}_{nv}|}[1] \otimes b_{nv,|\mathbf{v}_{nv}|}[2], b_{nv,|\mathbf{v}_{nv}|}[3]$. Suppose mixnet $\mathcal{M}$ is applied to those pairs of ciphertexts to derive vector $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_{nv})$, as per Step 2 of algorithm Tally. Since Π is a multiplicative-homomorphic asymmetric encryption scheme, we have for each $i \in \{1, \dots, nv\}$ that

$$\begin{aligned}\mathbf{b}_i[1] &= \mathsf{Enc}(pk, g^{d_\iota} \odot g^{-d_\iota}; r_\iota \oplus s_{\iota,|\mathbf{v}_\iota|} \oplus w_i)\\ &= \mathsf{Enc}(pk, g^0; r_\iota \oplus s_{\iota,|\mathbf{v}_\iota|} \oplus w_i)\\ \mathbf{b}_i[2] &= \mathsf{Enc}(pk, \mathbf{v}_\iota[|\mathbf{v}_\iota|]; t_{\iota,|\mathbf{v}_\iota|} \oplus x_i),\end{aligned}$$

where ι denotes $\chi(i)$ and χ is a permutation over $\{1, \dots, nv\}$ and coins w_i and x_i were introduced during mixing.

Suppose for each $i \in \{1, \dots, nv\}$ that $c' = \bigotimes_1^{n_i} \mathbf{b}_i[1]$ and $m = \mathsf{Dec}(sk, c')$ are computed as per Step 3 of algorithm Tally. It follows by (perfect) correctness and homomorphic properties that $m = 1$. Moreover, $\mathsf{Dec}(sk, \mathbf{b}_i[2]) = \mathbf{v}_{\chi(i)}[|\mathbf{v}_{\chi(i)}|]$ for each $i \in \{1, \dots, nv\}$. Since χ is a permutation over $\{1, \dots, nv\}$, it follows that $\mathfrak{v}$ is equivalent to the outcome that would be computed by Step 3 of algorithm Tally, which concludes our proof. □

D Verifiability by Smyth, Frink & Clarkson

We cast the verifiability definitions by Smyth, Frink & Clarkson [69] into the context of our syntax, extend their definition of Soundness to include re-voting, strengthen definitions of Injectivity, Individual-Verifiability and Unforgeability, and incorporate some minor refinements by Smyth [66, 67]. The definition of Completeness remains unchanged (beyond syntax changes).

D.1 Universal verifiability

Universal verifiability requires algorithm Verify to accept if and only if the election outcome is correct. The *only if* requirement is captured by soundness and the *if* requirement is captured by completeness.

Soundness. Correct outcomes are formalised using function *correct-outcome*. The function uses a predicate $(\exists^{=\ell} x : P(x))$ that holds exactly when there are ℓ distinct values of x for which $P(x)$ is satisfied [62]. Using the predicate, function *correct-outcome* is defined such that

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

where $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, M, \kappa)$ is a vector of length nc, $1 \leq v \leq nc$, and

$$\begin{aligned}&\mathit{authorised}(pk, nc, \mathfrak{bb}, M, \kappa) =\\ &\quad\big\{b_k \mid \exists! b_1, \dots, b_k \in \mathfrak{bb} : \exists \mathit{cnt}_1, \dots, \mathit{cnt}_k : \mathit{cnt}_1 \leq \cdots \leq \mathit{cnt}_{k-1} < \mathit{cnt}_k\\ &\qquad \wedge \exists (pd, d) \in M : \bigwedge_{1 \leq j \leq k} \exists v, r : b_j = \mathsf{Vote}(d, pk, v, \mathit{cnt}_j, nc, \kappa; r)\\ &\qquad \wedge \neg \exists b \in \mathfrak{bb} \setminus \{b_1, \dots, b_k\}, v, \mathit{cnt}, r : b = \mathsf{Vote}(d, pk, v, \mathit{cnt}, nc, \kappa; r)\big\}.\end{aligned}$$

Function *authorised* discards all ballots submitted under the same credential, except for a ballot containing the last vote. Hence, component v of vector $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, M, \kappa)$ equals ℓ iff there exist ℓ authorised ballots for vote v on the bulletin board. Function *correct-outcome* requires that ballots be interpreted for only one candidate, which can be ensured by injectivity, i.e., a ballot for vote v can never be interpreted for a distinct vote v'.

Definition 9 (Injectivity). *An election scheme* (Setup, Register, Vote, Tally, Verify) *satisfies* Injectivity*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, security parameters* κ *and computations* $(pk, nc, d_1, v_1, \mathit{cnt}_1, d_2, v_2, \mathit{cnt}_2) \leftarrow \mathcal{A}(\kappa); b_1 \leftarrow \mathsf{Vote}(d_1, pk, v_1, \mathit{cnt}_1, nc, \kappa); b_2 \leftarrow \mathsf{Vote}(d_2, pk, v_2, \mathit{cnt}_2, nc, \kappa)$ *such that* $v_1 \neq v_2 \wedge b_1 \neq \perp \wedge b_2 \neq \perp$*, we have* $b_1 \neq b_2$*.*

Equipped with a notion of correct outcomes, we formalise soundness (Definition 10) as a game that tasks the adversary to compute inputs to algorithm Verify – including an election outcome and some ballots – that cause the algorithm to accept an incorrect outcome.

Definition 10 (Soundness). *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{Soundness}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

```
$\mathsf{Soundness}(\Gamma, \mathcal{A}, \kappa) =$
    $(pk, nv) \leftarrow \mathcal{A}(\kappa);$
    for $1 \leq i \leq nv$ do $(pd_i, d_i) \leftarrow \mathsf{Register}(pk, \kappa);$
    $L \leftarrow \{pd_1, \dots, pd_{nv}\};$
    $M \leftarrow \{(pd_1, d_1), \dots, (pd_{nv}, d_{nv})\};$
    $(\mathfrak{bb}, nc, \mathfrak{v}, pf) \leftarrow \mathcal{A}(M);$
    return $\mathsf{Verify}(pk, \mathfrak{bb}, nc, L, \mathfrak{v}, pf, \kappa) = 1$
        $\wedge\ \mathfrak{v} \neq \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, M, \kappa);$
```

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

Completeness. We formalise completeness (Definition 11) as a game that tasks the adversary to compute a bulletin board and some number of candidates such that the corresponding election outcome computed by algorithm Tally is rejected by algorithm Verify, when the key pair is computed by algorithm Setup and voter credentials are computed by algorithm $\mathsf{Register}$.

Definition 11 (Completeness)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{Completeness}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

```
$\mathsf{Completeness}(\Gamma, \mathcal{A}, \kappa) =$
    $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa);$
    $nv \leftarrow \mathcal{A}(pk, \kappa);$
    $\textbf{for } 1 \leq i \leq nv \textbf{ do } (pd_i, d_i) \leftarrow \mathsf{Register}(pk, \kappa);$
    $L \leftarrow \{pd_1, \ldots, pd_{nv}\};$
    $M \leftarrow \{(pd_1, d_1), \ldots, (pd_{nv}, d_{nv})\};$
    $(\mathfrak{bb}, nc) \leftarrow \mathcal{A}(M);$
    $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, L, \kappa);$
    $\textbf{return } \mathsf{Verify}(pk, \mathfrak{bb}, nc, L, \mathfrak{v}, pf, \kappa) \neq 1 \land |\mathfrak{bb}| \leq mb \land nc \leq mc;$
```

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

D.2 Unforgeability

We formalise unforgeability (Definition 12) as a game that tasks the adversary to compute a ballot containing a private credential.

Definition 12 (Unforgeability)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{Unforgeability}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

```
$\mathsf{Unforgeability}(\Gamma, \mathcal{A}, \kappa) =$
    $(pk, nv) \leftarrow \mathcal{A}(\kappa);$
    $\textbf{for } 1 \leq i \leq nv \textbf{ do } (pd_i, d_i) \leftarrow \mathsf{Register}(pk, \kappa);$
    $L \leftarrow \{pd_1, \ldots, pd_{nv}\};$
    $Crpt \leftarrow \emptyset;\ Rvld \leftarrow \emptyset;$
    $b \leftarrow \mathcal{A}^{C,R}(L);$
    $\textbf{return } \exists i \in \{1, \ldots, nv\}, v, cnt, nc, r : b = \mathsf{Vote}(d_i, pk, v, cnt, nc, \kappa; r)$
        $\land\ b \neq \perp \land b \notin Rvld \land d_i \notin Crpt;$
```

Oracles C *and* R *are defined such that:*

- $C(i)$ *computes* $Crpt \leftarrow Crpt \cup \{d_i\}$ *and outputs* d_i*, where* $1 \leq i \leq nv$*, and*
- $R(i, v, cnt, nc)$ *computes* $b \leftarrow \mathsf{Vote}(d_i, pk, v, cnt, nc, \kappa); Rvld \leftarrow Rvld \cup \{b\}$ *and outputs* b*.*

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

D.3 Individual verifiability

We formalise individual verifiability (Definition 13) as a game that tasks the adversary to compute inputs to algorithm Vote that cause the algorithm to output ballots that collide.[8]

Definition 13 (Individual verifiability)**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{Individual\text{-}Verifiability}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

```
$\mathsf{Individual\text{-}Verifiability}(\Gamma, \mathcal{A}, \kappa) =$
    $(pk, nc, d_1, v_1, cnt_1, d_2, v_2, cnt_2) \leftarrow \mathcal{A}(\kappa)$;
    $b_1 \leftarrow \mathsf{Vote}(d_1, pk, v_1, cnt_1, nc, \kappa)$;
    $b_2 \leftarrow \mathsf{Vote}(d_2, pk, v_2, cnt_2, nc, \kappa)$;
    return $b_1 = b_2 \wedge b_1 \neq \perp \wedge b_2 \neq \perp$;
```

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* $\mathsf{Succ}(\mathsf{Individual\text{-}Verifiability}(\Gamma, \mathcal{A}, \kappa)) \leq \mathsf{negl}(\kappa)$*.*

E Proof of Propositions 2–4 (verifiability)

E.1 Proof of Proposition 2 (Soundness)

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$, $\mathsf{FS}(\Sigma_2, \mathcal{H}) = (\mathsf{ProveCiph}, \mathsf{VerCiph})$, $\mathsf{FS}(\Sigma_3, \mathcal{H}) = (\mathsf{ProveDec}, \mathsf{VerDec})$, $\mathsf{FS}(\Sigma_4, \mathcal{H}) = (\mathsf{ProveComb}, \mathsf{VerComb})$, and $\mathsf{Athena}(\Pi, \mathcal{M}, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$.

Suppose $\mathcal{A}$ is a probabilistic polynomial-time adversary, κ is a security parameter, $(\vec{pk}, nv)$ is an output of $\mathcal{A}(\kappa)$, and $(pd_1, \mathbf{d}_1), \dots, (pd_{nv}, \mathbf{d}_{nv})$ are outputs of $\mathsf{Register}(pk, \kappa)$. Let $L = \{pd_1, \dots, pd_{nv}\}$ and $M = \{(pd_1, \mathbf{d}_1), \dots, (pd_{nv}, \mathbf{d}_{nv})\}$. Suppose $(\mathfrak{bb}, nc, \mathfrak{v}, pf)$ is an output of $\mathcal{A}(M)$ such that $\mathsf{Verify}(\vec{pk}, \mathfrak{bb}, nc, L, \mathfrak{v}, pf, \kappa) = 1$. By definition of algorithm Verify, public key $\vec{pk}$ parses as a vector $(pk, \mathfrak{m}, \rho)$ and outcome $\mathfrak{v}$ parses as a vector of length nc such that $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) \wedge nc \leq mc$, where mc is computed as per algorithm Setup. Moreover, by simulation sound extractability, public key pk is an output of algorithm Gen. Furthermore, by definition of algorithm Register, we have for each $i \in \{1, \dots, nv\}$ that public credential $pd_i = \mathsf{Enc}(pk, g^{d_i}; r_i)$ and private credential $\mathbf{d}_i = (pd_i, d_i)$, for some nonce d_i and coins r_i chosen uniformly at random.

Let set $\{b_1, \dots, b_\ell\}$ be computed as per Step 1 of algorithm Verify. It follows that there exists an function $\lambda : \{1, \dots, \ell\} \to \{1, \dots, nv\}$ such that $b_i[1] = pd_{\lambda(i)}$

[8] Correctness, individual verifiability and injectivity all require that ballots do not collide, albeit under different assumptions.

for each $i \in \{1, \ldots, \ell\}$. Moreover, for all credentials $(pd, \mathbf{d}) \in M$, counters cnt, votes $v \in \{1, \ldots, nc\}$, and outputs b of algorithm $\mathsf{Vote}(\vec{d}, \vec{pk}, v, cnt, nc, \kappa)$, we have $b \notin \{b_1, \ldots, b_\ell\}$, since such an occurrence would imply a contradiction: $\{b_1, \ldots, b_\ell\}$ is not the largest subset of $\mathfrak{bb}$ satisfying the conditions in Step 1 of algorithm Tally, because b parses as a senary vector $(pd, c_1, c_2, \sigma_1, \sigma_2, cnt)$ such that $pd \in L \wedge \mathsf{VerCiph}((pk, g, c_1, \mathfrak{m}), \sigma_1, m, \kappa) \wedge \mathsf{VerCiph}((pk, c_2, \{1, \ldots, nc\}), \sigma_2, m, \kappa)$, where $m = (pd, c_1, c_2, cnt)$, yet $b \notin \{b_1, \ldots, b_\ell\}$. Thus,

$$\begin{aligned} &\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, M, \kappa) \\ &\qquad\qquad = \mathit{correct\text{-}outcome}(pk, nc, \{b_1, \ldots, b_\ell\}, M, \kappa) \quad (1) \end{aligned}$$

A proof of (1) follows from the definition of *correct-outcome*. If $\{b_1, \ldots, b_\ell\} = \emptyset$, then outcome $\mathfrak{v}$ and $\mathit{correct\text{-}outcome}(pk, nc, \{b_1, \ldots, b_\ell\}, M, \kappa)$ are zero-filled vectors of length nc, hence, $\mathsf{Soundness}$ is satisfied. Otherwise, we proceed as follows.

By simulation sound extractability, we have for each $i \in \{1, \ldots, \ell\}$ that there exists messages $d'_i \in \mathfrak{m}$ and $v_i \in \{1, \ldots, nc\}$ and coins s_i and t_i such that

$$\begin{aligned} b_i[2] &= \mathsf{Enc}(pk, g^{d'_i}; s_i), \\ b_i[3] &= \mathsf{Enc}(pk, v_i; t_i), \end{aligned}$$

$b_i[4]$ is an output of $\mathsf{ProveCiph}((pk, g, b_i[2], \mathfrak{m}), (d'_i, s_i), m, \kappa)$, and $b_i[5]$ is an output of $\mathsf{ProveCiph}((pk, b_i[3], \{1, \ldots, nc\}), (v_i, t_i), m, \kappa)$, where $m = (b_1[1], b_i[2], b_i[3], b_i[6])$. It follows by inspection of algorithm Vote that $\forall i \in \{1, \ldots, \ell\}, \exists r : b_i = \mathsf{Vote}(-d'_i, \vec{pk}, v_i, b_i[6], nc, \kappa)$, hence, $\{b_1, \ldots, b_\ell\}$ is a set of ballots.

By Step 2 of algorithm Verify, we have that pf parses as a vector $(\mathbf{pfr}, \mathbf{B}, \mathbf{pfd})$ and $\mathbf{pfr}$ parses as a vector $((c'_1, N_1, \varsigma_1), (c'_2, N_2, \varsigma_2, \omega_2), \ldots, (c'_\ell, N_\ell, \varsigma_\ell, \omega_\ell))$ such that $\bigwedge_{1 \leq i \leq \ell} \mathsf{VerDec}((pk, c'_i, N_i), \varsigma_i, \kappa)$ and $\bigwedge_{1 < i \leq \ell} \mathsf{VerComb}((pk, (c'_{i-1}, c'_i), (b_{i-1}[2], b_i[2])), \omega_i, \kappa)$. By simulation sound extractability, there exists a nonce n such that for all $i \in \{1, \ldots, \ell\}$ we have

$$c'_i = \bigotimes_1^n b_i[2] = \mathsf{Enc}(pk, \odot_1^n g^{d'_i}; \oplus_1^n s_i) \equiv \mathsf{Enc}(pk, g^{d'_i \cdot n}; \oplus_1^n s_i)$$

and $\mathsf{Dec}(sk, c'_i) = N_i$, moreover, by (perfect) correctness, we have

$$N_i \equiv g^{d'_i \cdot n}.$$

Let map $\mathbf{A}$ be computed as per Step 2 of algorithm Verify. It follows for each $i \in \{1, \ldots, \ell\}$ that

$$\begin{aligned} &\mathbf{A}[(b_i[1], N_i)] = (b_i[6], b_i[1] \otimes b_i[2], b_i[3]) \Leftrightarrow \\ &\qquad\qquad \neg\exists j\{1, \ldots, \ell\} \setminus \{i\} : b_i[1] = b_j[1] \wedge N_i = N_j \wedge b_i[6] \leq b_j[6] \end{aligned}$$

i.e., public credential $b_i[1]$ and anonymised credential N_i are mapped to a triple derived from ballot b_i iff there is no other ballot b_j with the same public credential and the same anonymised credential that has a greater-than or equal-to counter value. Hence, for each anonymised credential, map $\mathbf{A}$ contains the encrypted vote associated with the highest counter, that is, the last vote related to the credential. Since (pairwise) mixnet $\mathcal{M}$ is verifiable and since Step 2 of algorithm Verify checks that $\mathbf{B}$ was output by the mixnet, there exists an injective function $\chi : \{1, \ldots, |\mathbf{B}|\} \rightarrow \{1, \ldots, \ell\}$ such that for each $i \in \{1, \ldots, |\mathbf{B}|\}$ we have $\mathbf{B}[i]$ is a pair (c_1, c_2),

$$\begin{aligned} c_1 &= \mathsf{Enc}(pk, g^{d_{\chi(\lambda(i))}} \odot g^{d'_{\chi(i)}}; r_{\chi(\lambda(i))} \oplus s_{\chi(i)} \oplus w_i), \text{ and} \\ c_2 &= \mathsf{Enc}(pk, v_{\chi(i)}; t_{\chi(i)} \oplus x_i), \end{aligned}$$

where coins w_i and x_i were introduced during mixing. It follows that

$$\begin{aligned} &\mathit{authorised}(pk, nc, \{b_1, \ldots, b_\ell\}, M, \kappa) \\ &\qquad\qquad = \mathit{authorised}(pk, nc, \{b_{\chi(i)} \mid 1 \leq i \leq |\mathbf{B}|\}, M, \kappa) \quad (2) \end{aligned}$$

because any ballot that shares a public credential (and a anonymised credential) with another ballot, whilst being associated with a (strictly) lower counter value can be discarded, as can any pair of ballots that share a public credential (and a anonymised credential) and a counter.

By Step 3 of algorithm Verify, we have for each $i \in \{1, \ldots, |\mathbf{B}|\}$ that $\mathbf{B}[i]$ parses as a vector (c_1, c_2) and $\mathbf{pfd}[i]$ parses a vector $(c', m, \omega, \varsigma_1)$ or $(c', v, \omega, \varsigma_1, \varsigma_2)$, such that $\mathsf{VerComb}((pk, c', c_1), \omega, \kappa)$, hence, by simulation sound extractability, there exists a nonce n such that $c' = \bigotimes_1^n c_1$, moreover, we have $\mathsf{VerDec}((pk, c', 1), \varsigma_1, \kappa)$ when $|\mathbf{pfd}[i]| = 5$ and $\mathsf{VerDec}((pk, c', m), \varsigma_1, \kappa) \wedge m \neq 1$ when $|\mathbf{pfd}[i]| = 4$, hence, by simulation sound extractability, (perfect) correctness, and multiplicatively-homomorphic properties, we have

$$|\mathbf{pfd}[i]| = 5 \Leftrightarrow \mathsf{Dec}(sk, c') = 1 \Leftrightarrow 1 \equiv g^{d_{\chi(\lambda(i))}} \odot g^{d'_{\chi(i)}} \Leftrightarrow d'_{\chi(i)} \equiv -d_{\chi(\lambda(i))}.$$

It follows that $b_{\chi(i)}$ is constructed from $(pd_{\chi(\lambda(i))}, \mathbf{d}_{\chi(\lambda(i))}) \in M$ iff $|\mathbf{pfd}[i]| = 5$, where $i \in \{1, \ldots, |\mathbf{B}|\}$, hence,

$$\begin{aligned} &\mathit{authorised}(pk, nc, \{b_{\chi(i)} \mid 1 \leq i \leq |\mathbf{B}|\}, M, \kappa) \\ &\qquad = \mathit{authorised}(pk, nc, \{b_{\chi(i)} \mid 1 \leq i \leq |\mathbf{B}| \wedge |\mathbf{pfd}[i]| = 5\}, M, \kappa) \\ &\qquad\qquad\qquad = \{b_{\chi(i)} \mid 1 \leq i \leq |\mathbf{B}| \wedge |\mathbf{pfd}[i]| = 5\} \quad (3) \end{aligned}$$

because any ballot not constructed from $(pd, \mathbf{d}) \in M$ can be discarded and no further ballots can. Moreover, it follows from the remainder of Step 3 that for each $v \in \{1, \ldots, nc\}$ we have $\exists^{=\mathfrak{v}[v]} i \in \{1, \ldots, |\mathbf{B}|\} : \exists c_1, c_2, c', \omega, \varsigma_1, \varsigma_2 : (c_1, c_2) = \mathbf{B}[i] \wedge (c', v, \omega, \varsigma_1, \varsigma_2) = \mathbf{pfd}[i] \wedge \mathsf{VerDec}((pk, c_2, v), \varsigma_2, \kappa)$, hence, by simulation sound extractability, we have

$$\exists^{=\mathfrak{v}[v]} i \in \{1, \ldots, |\mathbf{B}| \wedge |\mathbf{pfd}[i]| = 5\} : \exists c_1, c_2 : (c_1, c_2) = \mathbf{B}[i] \wedge \mathsf{Dec}(sk, c_2) = v,$$

furthermore, by (perfect) correctness, we have

$$\exists^{=\mathfrak{v}[v]} i \in \{1, \ldots, |\mathbf{B}| \wedge |\mathbf{pfd}[i]| = 5\} : v = v_{\chi(i)}.$$

If follows for each $v \in \{1, \ldots, nc\}$ that

$$\exists^{=\mathfrak{v}[v]} b \in \{b_{\chi(i)} \mid 1 \leq i \leq |\mathbf{B}| \wedge |\mathbf{pfd}[i]| = 5\} : \\ \exists d, cnt, r : b = \mathsf{Vote}(d, pk, v, cnt, nc, \kappa; r).$$

Finally, by (1)–(3) and since error symbol $\perp$ is not a vector, we have $\mathfrak{v} = \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, M, \kappa)$, concluding our proof. □

E.2 Proof of Proposition 3 (Completeness)

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$, $\mathsf{FS}(\Sigma_2, \mathcal{H}) = (\mathsf{ProveCiph}, \mathsf{VerCiph})$, $\mathsf{FS}(\Sigma_3, \mathcal{H}) = (\mathsf{ProveDec}, \mathsf{VerDec})$, $\mathsf{FS}(\Sigma_4, \mathcal{H}) = (\mathsf{ProveComb}, \mathsf{VerComb})$, and $\mathsf{Athena}(\Pi, \mathcal{M}, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$.

Suppose κ is a security parameter and $\mathcal{A}$ is a probabilistic polynomial-time adversary. Further suppose $(\vec{pk}, \vec{sk}, mb, mc)$ is an output of $\mathsf{Setup}(\kappa)$, nv is an output of $\mathcal{A}(pk, \kappa)$, and $(pd_1, \mathbf{d}_1), \ldots, (pd_{nv}, \mathbf{d}_{nv})$ are outputs of $\mathsf{Register}(pk, \kappa)$. Let $L = \{pd_1, \ldots, pd_{nv}\}$ and $M = \{(pd_1, \mathbf{d}_1), \ldots, (pd_{nv}, \mathbf{d}_{nv})\}$. Suppose $(\mathfrak{bb}, nc)$ is an output of $\mathcal{A}(M)$ and $(\mathfrak{v}, pf)$ is an output of $\mathsf{Tally}(sk, \mathfrak{bb}, nc, L, \kappa)$. If $|\mathfrak{bb}| \not\leq mb \vee nc \not\leq mc$, then we conclude immediately, otherwise $(|\mathfrak{bb}| \leq mb \wedge nc \leq mc)$, we proceed as follows. By definition of algorithm Setup, we have $\vec{pk}$ parses as $(pk, \mathfrak{m}, \rho)$ and $\vec{sk}$ parses as (pk, sk), where $(pk, sk, \mathfrak{m}) = \mathsf{Gen}(\kappa; r)$ and ρ is an output of $\mathsf{ProveKey}((\kappa, pk, \mathfrak{m}), (sk, r), \kappa)$, for some coins r chosen uniformly at random. Moreover, by definition of algorithm Tally, we have $\mathfrak{v}$ is a vector of length nc. It follows that algorithm Verify can parse inputs correctly. Moreover, by completeness, we have $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) = 1$.

Suppose subset $\{b_1, \ldots, b_\ell\}$ is computed as per Step 1 of algorithm Tally. If that set is empty, then $\mathfrak{v}$ is a zero-filled vector, because $\mathfrak{v}$ is initialised as a zero-filled vector by algorithm Tally. Thus, the check holds in Step 1 of algorithm Verify.

By Step 1 of algorithm Tally, we have for each $i \in \{1, \ldots, \ell\}$ that b_i parses as $(pd, c_1, c_2, \sigma_1, \sigma_2, cnt)$ such that $pd \in L \wedge \mathsf{VerCiph}((pk, g, c_1, \mathfrak{m}), \sigma_1, m, \kappa) \wedge \mathsf{VerCiph}((pk, c_2, \{1, \ldots, nc\}), \sigma_2, m, \kappa)$, where $m = (pd, c_1, c_2, cnt)$. Hence, there exists an integer $j \in \{1, \ldots, nv\}$ such that $pd = pd_j$. It follows by definition of algorithm $\mathsf{Register}$ that $b_i[1] = \mathsf{Enc}(pk, g^{d_j}; r_j)$, for some coins r_j chosen uniformly at random and nonce d_j such that private credential $\mathbf{d}_j = (pd_j, d_j)$. Moreover, since Σ_2 satisfies special soundness and special honest verifier zero-knowledge, we have by simulation sound extractability that $b_i[2] = \mathsf{Enc}(pk, g^{\overline{d}_j}; s_j)$ and $b_i[3] = \mathsf{Enc}(pk, v_j; t_j)$, for some coins s_j and t_j, plaintext $\overline{d}_j \in \mathfrak{m}$, and vote $v_j \in \{1, \ldots, nc\}$. It follows that the map $(\mathbf{A})$ computed in Step 2 of algorithm Tally is populated with pairs of ciphertexts. Thus, vector $\mathbf{B}$ – derived by application of (pairwise) mixnet $\mathcal{M}$ to map $\mathbf{A}$ in Step 2 of algorithm Tally – passes the check in Step 2 of algorithm Verify, because $\mathcal{M}$ is verifiable. The preceding checks also pass. Indeed, by definition of algorithm Tally, it is trivial to see that pf parses as a vector $(\mathbf{pfr}, \mathbf{B}, \mathbf{pfd})$. Moreover, the vector $(\mathbf{pfr})$ computed in Step 2 of algorithm Tally parses as $((c'_1, N_1, \varsigma_1), (c'_2,$

$N_2, \varsigma_2, \omega_2), \ldots, (c'_\ell, N_\ell, \varsigma_\ell, \omega_\ell))$ and, by completeness, the proofs in that vector pass the checks in Step 2 of algorithm Verify. Thus, checks hold in Step 2 of algorithm Verify.

By Step 3 of algorithm Tally, we have **pfd** parses as a vector of length $|\mathbf{B}|$, hence, Step 3 of algorithm Verify successfully parses that vector. Since $\mathfrak{v}$ is initialised as a zero-filled vector of length nc and $\mathfrak{v}[v]$ is incremented by one for each $i \in \{1, \ldots, |\mathbf{B}|\}$ such that $\mathsf{Dec}(sk, c') = 1$, where $c' = \bigotimes_1^{n_i} c_1$, $v = \mathsf{Dec}(sk, c_2)$, $\mathbf{B}[i] = (c_1, c_2)$, and n_i is a nonce, and since $\mathbf{pfd}[i] = (c', v, \omega, \varsigma_1, \varsigma_2)$, where ω is an output of $\mathsf{ProveComb}((pk, c', c_1), n_i, \kappa)$, ς_1 is an output of $\mathsf{ProveDec}((pk, c', 1), sk, \kappa)$, and ς_2 is an output of $\mathsf{ProveDec}((pk, c_2, v), sk, \kappa)$, we have for each $v \in \{1, \ldots, nc\}$ that $\exists^{=\mathfrak{v}[v]} i \in \{1, \ldots, |\mathbf{B}|\} : \exists c_1, c_2, c', \omega, \varsigma_1, \varsigma_2 : (c_1, c_2) = \mathbf{B}[i] \wedge (c', v, \omega, \varsigma_1, \varsigma_2) = \mathbf{pfd}[i] \wedge \mathsf{VerComb}((pk, c', c_1), \omega, \kappa) \wedge \mathsf{VerDec}((pk, c', 1), \varsigma_1, \kappa) \wedge \mathsf{VerDec}((pk, c_2, v), \varsigma_2, \kappa)$ by completeness, moreover, for each remaining integer $i \in \{1, \ldots, |\mathbf{B}|\}$ we have $\mathbf{pfd}[i]$ parses as $(c', m, \omega, \varsigma_1)$, and $\mathsf{VerComb}((pk, c', c_1), \omega, \kappa) \wedge \mathsf{VerDec}((pk, c', m), \varsigma_1, \kappa) \wedge m \neq 1$. Thus, checks hold in Step 3 of algorithm Verify.

Since all the above checks succeed, algorithm Verify outputs 1, concluding our proof. □

E.3 Proof of Proposition 4 (Individual-Verifiability & Injectivity)

Let $\mathsf{Athena}(\Pi, \mathcal{M}, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$. Suppose $\mathcal{A}$ is a probabilistic polynomial-time adversary and κ is a security. Further suppose $(\vec{pk}, nc, \vec{d}_1, v_1, cnt_1, \vec{d}_2, v_2, cnt_2)$ is an output of $\mathcal{A}(\kappa)$, $\mathbf{b}_1$ is an output of $\mathsf{Vote}(\vec{d}_1, \vec{pk}, v_1, cnt_1, nc, \kappa)$, and $\mathbf{b}_2$ is an output of $\mathsf{Vote}(\vec{d}_2, \vec{pk}, v_2, cnt_2, nc, \kappa)$, such that $\mathbf{b}_1 \neq \perp$ and $\mathbf{b}_2 \neq \perp$. By definition of algorithm Vote, public key $\vec{pk}$ is a vector $(pk, \mathfrak{m}, \rho)$ such that $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) = 1$ and $v_1, v_2 \in \{1, \ldots, nc\} \subseteq \mathfrak{m}$. Moreover, $\mathbf{b}_1$ and $\mathbf{b}_2$ are vectors such that $\mathbf{b}_1[2]$ is an output of $\mathsf{Enc}(pk, v_1)$ and $\mathbf{b}_2[2]$ is an output of $\mathsf{Enc}(pk, v_2)$. Thus, $\mathbf{b}_1 \neq \mathbf{b}_2$ by our precondition, with overwhelming probability, therefore, Individual-Verifiability is satisfied. For Injectivity, we further suppose $v_1 \neq v_2$, hence, $\mathbf{b}_1 \neq \mathbf{b}_2$ by our precondition, which concludes our proof. □

References

[1] Acquisti, A.: Receipt-Free Homomorphic Elections and Write-in Ballots. Cryptology ePrint Archive, Report 2004/105 (2004), `https://eprint.iacr.org/2004/105`

[2] Araújo, R., Barki, A., Brunet, S., Traoré, J.: Remote electronic voting can be efficient, verifiable and coercion-resistant. In: FC'16: 20th International Conference on Financial Cryptography and Data Security. LNCS, vol. 9604, pp. 224–232. Springer (2016)

[3] Araújo, R., Foulle, S., Traoré, J.: A practical and secure coercion-resistant scheme for remote elections. Tech. Rep. 07311, Schloss Dagstuhl, Germany (2008)

[4] Araújo, R., Foulle, S., Traoré, J.: A practical and secure coercion-resistant scheme for remote elections. In: Towards Trustworthy Elections: New Directions in Electronic Voting, LNCS, vol. 6000, pp. 330–342. Springer (2010)

[5] Araújo, R., Rajeb, N.B., Robbana, R., Traoré, J., Youssfi, S.: Towards Practical and Secure Coercion-Resistant Electronic Elections. In: CANS'10: International Conference on Cryptology and Network Security. pp. 278–297. No. 6467 in LNCS, Springer (2010)

[6] Araújo, R., Traoré, J.: A Practical Coercion Resistant Voting Scheme Revisited. In: VoteID'13: International Conference on E-Voting and Identity. LNCS, vol. 7985, pp. 193–209. Springer (2013)

[7] Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations Among Notions of Security for Public-Key Encryption Schemes. In: CRYPTO'98: 18th International Cryptology Conference. LNCS, vol. 1462, pp. 26–45. Springer (1998)

[8] Benaloh, J.: Verifiable Secret-Ballot Elections. Ph.D. thesis, Department of Computer Science, Yale University (1996)

[9] Benaloh, J., Yung, M.: Distributing the Power of a Government to Enhance the Privacy of Voters. In: PODC'86: 5th Principles of Distributed Computing Symposium. pp. 52–62. ACM Press (1986)

[10] Benaloh, J.C., Tuinstra, D.: Receipt-free secret-ballot elections. In: STOC'94: 26th Theory of computing Symposium. pp. 544–553. ACM Press (1994)

[11] Bernhard, D., Cortier, V., Galindo, D., Pereira, O., Warinschi, B.: SoK: A comprehensive analysis of game-based ballot privacy definitions. In: S&P'15: 36th Security and Privacy Symposium. pp. 499–516. IEEE Computer Society (2015)

[12] Bernhard, D., Pereira, O., Warinschi, B.: 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. LNCS, vol. 7658, pp. 626–643. Springer (2012)

[13] Bernhard, D., Smyth, B.: Ballot secrecy with malicious bulletin boards. Cryptology ePrint Archive, Report 2014/822 (version 20150413:170300) (2015)

[14] Chaidos, P., Cortier, V., Fuschbauer, G., Galindo, D.: BeleniosRF: A Non-interactive Receipt-Free Electronic Voting Scheme. In: CCS'16: 23rd ACM Conference on Computer and Communications Security. pp. 1614–1625. ACM Press (2016)

[15] Chaum, D., Evertse, J., van de Graaf, J., Peralta, R.: Demonstrating Possession of a Discrete Logarithm Without Revealing It. In: CRYPTO'86: 6th International Cryptology Conference. LNCS, vol. 263, pp. 200–212. Springer (1987)

[16] Chaum, D., Pedersen, T.P.: Wallet Databases with Observers. In: CRYPTO'92: 12th International Cryptology Conference. LNCS, vol. 740, pp. 89–105. Springer (1993)

[17] Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM 24, 84–90 (1981)

[18] Clark, J.: Democracy Enhancing Technologies: Toward deployable and incoercible E2E elections. Ph.D. thesis, University of Waterloo (2011)

[19] Clark, J., Hengartner, U.: Selections: Internet voting with over-the-shoulder coercion-resistance. In: FC'11: 15th International Conference on Financial Cryptography. LNCS, vol. 7035, pp. 47–61. Springer (2011)

[20] Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a Secure Voting System. Tech. Rep. 2007-2081, Cornell University (May 2007), revised March 2008

[21] Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a Secure Voting System. In: S&P'08: 29th Security and Privacy Symposium. pp. 354–368. IEEE Computer Society (2008)

[22] Cohen, J.D., Fischer, M.J.: A Robust and Verifiable Cryptographically Secure Election Scheme. In: FOCS'85: 26th Symposium on Foundations of Computer Science. pp. 372–382. IEEE Computer Society (1985)

[23] Cortier, V., Galindo, D., Glondu, S., Izabachène, M.: Election Verifiability for Helios under Weaker Trust Assumptions. In: ESORICS'14: 19th European Symposium on Research in Computer Security. LNCS, vol. 8713, pp. 327–344. Springer (2014)

[24] Cortier, V., Galindo, D., Küsters, R., Mueller, J., Truderung, T.: SoK: Verifiability Notions for E-Voting Protocols. In: S&P'16: 37th IEEE Symposium on Security and Privacy. pp. 779–798. IEEE Computer Society (2016)

[25] Cortier, V., Smyth, B.: Attacking and fixing Helios: An analysis of ballot secrecy. Journal of Computer Security 21(1), 89–148 (2013)

[26] Cramer, R., Franklin, M.K., Schoenmakers, B., Yung, M.: Multi-Autority Secret-Ballot Elections with Linear Work. In: EUROCRYPT'96: 15th International Conference on the Theory and Applications of Cryptographic Techniques. LNCS, vol. 1070, pp. 72–83. Springer (1996)

[27] Delaune, S., Kremer, S., Ryan, M.: Coercion-Resistance and Receipt-Freeness in Electronic Voting. In: CSFW'06: 19th Computer Security Foundations Workshop. pp. 28–42. IEEE Computer Society (2006)

[28] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)

[29] Essex, A., Clark, J., Hengartner, U.: Cobra: Toward Concurrent Ballot Authorization for Internet Voting. In: EVT/WOTE'12: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections. USENIX Association (2012)

[30] Fiat, A., Shamir, A.: How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In: CRYPTO'86: 6th International Cryptology Conference. LNCS, vol. 263, pp. 186–194. Springer (1987)

[31] Fraser, A., Quaglia, E.A., Smyth, B.: A critique of game-based definitions of receipt-freeness for voting. In: ProveSec'19: 13th International Conference on Provable and Practical Security. LNCS, Springer (2019)

[32] Gardner, R.W., Garera, S., Rubin, A.D.: Coercion Resistant End-to-end Voting. In: FC'09: 13th International Conference on Financial Cryptography and Data Security. LNCS, vol. 5628, pp. 344–361. Springer (2009)

[33] Groth, J.: Efficient maximal privacy in boardroom voting and anonymous broadcast. In: FC'04: 8th International Conference on Financial Cryptography. LNCS, vol. 3110, pp. 90–104. Springer (2004)

[34] Groth, J.: 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. LNCS, vol. 4284, pp. 444–459. Springer (2006)

[35] Haghighat, A.T., Dousti, M.S., Jalili, R.: An Efficient and Provably-Secure Coercion-Resistant E-Voting Protocol. In: PST'13: 11th International Conference on Privacy, Security and Trust. pp. 161–168. IEEE Computer Society (2013)

[36] Hao, F., Ryan, P.Y.A., Zieliński, P.: Anonymous voting by two-round public discussion. Journal of Information Security 4(2), 62 – 67 (2010)

[37] Heather, J., Schneider, S.: A formal framework for modelling coercion resistanc and receipt freeness. In: FM'12: 18th International Symposium on Formal Methods. pp. 217–231. No. 7436 in LNCS, Springer (2012)

[38] Jakobsson, M., Juels, A.: Mix and Match: Secure Function Evaluation via Ciphertexts. In: ASIACRYPT'00: 6th International Conference on the Theory and Application of Cryptology and Information Security. LNCS, vol. 1976, pp. 162–177. Springer (2000)

[39] Juels, A., Catalano, D., Jakobsson, M.: Coercion-Resistant Electronic Elections. Cryptology ePrint Archive, Report 2002/165 (2002)

[40] Juels, A., Catalano, D., Jakobsson, M.: Coercion-Resistant Electronic Elections. In: WPES'05: 4th Workshop on Privacy in the Electronic Society. pp. 61–70. ACM Press (2005)

[41] Juels, A., Catalano, D., Jakobsson, M.: Coercion-Resistant Electronic Elections. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y. (eds.) Towards Trustworthy Elections: New Directions in Electronic Voting, LNCS, vol. 6000, pp. 37–63. Springer (2010)

[42] Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman & Hall/CRC (2007)

[43] Khader, D., Smyth, B., Ryan, P.Y.A., Hao, F.: A Fair and Robust Voting System by Broadcast. In: EVOTE'12: 5th International Conference on Electronic Voting. Lecture Notes in Informatics, vol. 205, pp. 285–299. Gesellschaft für Informatik (2012)

[44] Khazaei, S., Rezaei-Aliabadi, M.: A rigorous security analysis of a decentralized electronic voting protocol in the universal composability framework. Journal of Information Security and Applications 43, 99–109 (2018)

[45] Kiayias, A., Yung, M.: Self-tallying elections and perfect ballot secrecy. In: PKC'01: 3rd International Workshop on Practice and Theory in Public Key Cryptography. LNCS, vol. 2274, pp. 141–158. Springer (2002)

[46] Kiayias, A., Zacharias, T., Zhang, B.: End-to-end verifiable elections in the standard model. In: EUROCRYPT'15: 34th International Conference on the Theory and Applications of Cryptographic Techniques. LNCS, vol. 9057, pp. 468–498. Springer (2015)

[47] Kremer, S., Ryan, M.D., Smyth, B.: Election verifiability in electronic voting protocols. In: ESORICS'10: 15th European Symposium on Research in Computer Security. LNCS, vol. 6345, pp. 389–404. Springer (2010)

[48] Küsters, R., Truderung, T., Vogt, A.: A Game-Based Definition of Coercion-Resistance and its Applications. In: CSF'10: 23rd IEEE Computer Security Foundations Symposium. pp. 122–136. IEEE Computer Society (2010)

[49] Küsters, R., Truderung, T., Vogt, A.: Accountability: Definition and relationship to verifiability. In: CCS'10: 17th ACM Conference on Computer and Communications Security. pp. 526–535. ACM Press (2010)

[50] Küsters, R., Truderung, T., Vogt, A.: Verifiability, Privacy, and Coercion-Resistance: New Insights from a Case Study. In: S&P'11: 32nd IEEE Symposium on Security and Privacy. pp. 538–553. IEEE Computer Society (2011)

[51] Küsters, R., Truderung, T., Vogt, A.: A Game-Based Definition of Coercion-Resistance and its Applications. Journal of Computer Security 20(6), 709–764 (2012)

[52] Küsters, R., Truderung, T., Vogt, A.: Clash Attacks on the Verifiability of E-Voting Systems. In: S&P'12: 33rd IEEE Symposium on Security and Privacy. pp. 395–409. IEEE Computer Society (2012)

[53] Küsters, R., Truderung, T., Vogt, A.: Accountability: Definition and relationship to verifiability. Cryptology ePrint Archive, Report 2010/236 (version 20150202:163211) (2015)

[54] Meyer, M., Smyth, B.: Exploiting re-voting in the helios election system. Information Processing Letters (143), 14–19 (2019)

[55] Michels, M., Horster, P.: Some Remarks on a Receipt-Free and Universally Verifiable Mix-Type Voting Scheme. In: ASIACRYPT'96: International Conference on the Theory and Application of Cryptology and Information Security. LNCS, vol. 1163, pp. 125–132. Springer (1996)

[56] Moran, T., Naor, M.: Receipt-Free Universally-Verifiable Voting with Everlasting Privacy. In: CRYPTO'06: 26th International Cryptology Conference. LNCS, vol. 4117, pp. 373–392. Springer (2006)

[57] Organization for Security and Co-operation in Europe: Document of the Copenhagen Meeting of the Conference on the Human Dimension of the CSCE (1990)

[58] Organization of American States: American Convention on Human Rights, "Pact of San Jose, Costa Rica" (1969)

[59] Sako, K., Kilian, J.: 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. LNCS, vol. 921, pp. 393–403. Springer (1995)

[60] Schläpfer, M., Haenni, R., Koenig, R., Spycher, O.: Efficient Vote Authorization in Coercion-Resistant Internet Voting. In: VoteID'11: International Conference on E-Voting and Identity. LNCS, vol. 7187, pp. 71–88. Springer (2011)

[61] Schoenmakers, B.: A simple publicly verifiable secret sharing scheme and its application to electronic voting. In: CRYPTO'99: 19th International Cryptology Conference. LNCS, vol. 1666, pp. 148–164. Springer (1999)

[62] Schweikardt, N.: Arithmetic, first-order logic, and counting quantifiers. ACM Transactions on Computational Logic 6(3), 634–671 (Jul 2005)

[63] Schweisgut, J.: Coercion-Resistant Electronic Elections with Observer. In: Electronic Voting. Lecture Notes in Informatics, vol. 86, pp. 171–177. Gesellschaft für Informatik (2006)

[64] Smith, W.D.: New cryptographic election protocol with best-known theoretical properties. In: Workshop on Frontiers in Electronic Elections. pp. 1–14 (2005)

[65] Smyth, B.: Ballot secrecy: Security definition, sufficient conditions, and analysis of Helios. Cryptology ePrint Archive, Report 2015/942 (2018)

[66] Smyth, B.: A foundation for secret, verifiable elections. Cryptology ePrint Archive, Report 2018/225 (version 20180301:164045) (2018)

[67] Smyth, B.: Verifiability of Helios Mixnet. In: Voting'18: 3rd Workshop on Advances in Secure Electronic Voting. LNCS, Springer (2018)

[68] Smyth, B.: Surveying definitions of coercion resistance. Cryptology ePrint Archive, Report 2019/822 (2019)

[69] Smyth, B., Frink, S., Clarkson, M.R.: Election Verifiability: Cryptographic Definitions and an Analysis of Helios and JCJ. Cryptology ePrint Archive, Report 2015/233 (version 20170213:132559) (2017)

[70] Spycher, O., Koenig, R., Haenni, R., Schläpfer, M.: A New Approach Towards Coercion-Resistant Remote E-Voting in Linear Time. In: FC'11: 15th International Conference on Financial Cryptography. LNCS, vol. 7035, pp. 182–189. Springer (2011)

[71] United Nations: Universal Declaration of Human Rights (1948)

[72] Unruh, D., Müller-Quade, J.: Universally Composable Incoercibility. In: CRYPTO'10: 30th International Cryptology Conference. LNCS, vol. 6223, pp. 411–428. Springer (2010)

[73] Weber, S.G., Araújo, R., Buchmann, J.: On Coercion-Resistant Electronic Elections with Linear Work. In: ARES'07: 2nd Internation Conference on Availability, Reliability and Security. pp. 908–916. IEEE (2007)

[74] Wikström, D.: Simplified Submission of Inputs to Protocols. In: SCN'08: 6th International Conference on Security and Cryptography for Networks. LNCS, vol. 5229, pp. 293–308. Springer (2008)