# Voting with coercion resistance and everlasting privacy using linkable ring signatures

Panagiotis Grontas[0000−0001−7584−0643], Aris Pagourtzis[0000−0002−6220−3722], and Marianna Spyrakou[0000−0001−5694−8405]

School of Electrical and Computer Engineering
National Technical University of Athens,
9 Iroon Polytechniou St, 157 80 Zografou, Greece
pgrontas@corelab.ntua.gr, pagour@cs.ntua.gr, mspyrakou@mail.ntua.gr

**Abstract.** We propose an e-voting protocol based on a novel linkable ring signature scheme with unconditional anonymity. In our system, all voters register create private credentials and register their public counterparts. To vote, they create a ring (anonymity set) consisting of public credentials together with a proof of knowledge of their secret credential via our signature. Its unconditional anonymity prevents an attacker, no matter how powerful, from deducing the identity of the voter, thus attaining everlasting privacy. Additionally, our protocol provides coercion resistance in the JCJ framework; when an adversary tries to coerce a voter, the attack can be evaded by creating a signature with a fake but indistinguishable credential. During a moment of privacy, they will cast their real vote. Our scheme also provides verifiability and ballot secrecy.

**Keywords:** e-voting, coercion resistance, everlasting privacy, linkable ring signatures, unconditional anonymity

## 1 Introduction

Voting is a distributed decision making process. Voters submit their opinions, talliers aggregate them and everyone is bound by the result. Electronic voting aims to improve the speed, cost and accessibility of this process. To do so, it must satisfy a set of conflicting security properties. Verifiability [14] removes the trust in the various components of the system, by allowing voters to check all parts of the process. Privacy [7] encourages voters to express their true opinions. It can be implemented using secrecy [17] and/or anonymity [9]. In its most basic form, it protects only against other (passive) voters and the talliers through cryptography under computational assumptions. Everlasting privacy [37] protects voter privacy even after such assumptions cease to hold. Privacy can also be 'extended' to protect against corrupted voters that want to sell their votes (receipt - freeness [6]) and active adversaries that seek to coerce a particular action using threats in case of non-compliance. Coercion resistance [31] is the strongest form of privacy that must be satisfied for remote electronic voting to be broadly adopted.

**Related work** Voting protocols have a close connection to digital signatures schemes, both as means to force untrusted players to behave correctly ([17, 1, 8, 14, 39]) and as proofs of knowledge of private credentials to authenticate the voters ([31, 12]). The versatility of signatures as a primitive also enables the provision of privacy. For example, the schemes in [21, 23] use blind signatures. The seminal work of [34] utilizes linkable ring signatures to provide strong levels of anonymity, while eliminating the registration phase and preventing double voting. Unfortunately, [34] does not provide neither coercion resistance nor everlasting privacy, since the linking tag which connects votes also reveals the identity of the voter to a future adversary and can be used to check compliance from a contemporary coercer. Our scheme provides improvements on both these areas, without sacrificing verifiability.

Regarding coercion resistance, the JCJ framework [31] has been the definitive paradigm for many schemes. Its attack model considers three cases: the randomization attack, where the voter is forced to vote randomly, the forced abstention attack, where the voter must not cast a vote at all, and the simulation attack, where the coercer essentially takes total control of the voter and votes as they wish. Coercion resistance is achieved by not allowing the adversary to be certain if their attack succeeded, i.e. if the voter followed their instructions. This is made possible in [31] using two methods: multiple votes per voter and anonymous credentials. The voter follows the coercer's instructions when they are present, but during *a moment of privacy* (a necessary assumption) they cast their real vote. The coercer cannot tell which ballot was counted because of the use of anonymous and indistinguishable credentials. The talliers must prove that they counted the correct vote, to maintain verifiability, without leaking the credential that accompanied the real vote. To do so, they use a *plaintext equivalence test* (PET) [29]. A detailed implementation of JCJ was given in Civitas [12]. One of the main drawbacks of both these works is the *quadratic* tallying time to weed out real and fake votes using the PET. Selections [11], another implementation of JCJ provides a detailed specification of the registration phase, and a usable way to generate anonymous credentials through panic passwords. A noteworthy characteristic of [11] is that it associates tallying complexity with the degree of coercion resistance and allows the trade-off to be adjusted.

Works that provide everlasting privacy can be distinguished in two main categories depending on whether an anonymous casting phase is used [21, 35, 25] or not [37, 38, 19, 2, 18, 39]. The latter schemes accompany the encrypted vote with a perfectly hiding commitment. Everlasting privacy is achieved by allowing only the part of the ballot box that contains the commitments to be revealed to the public. The election server maintains, but never discloses, a secret ballot box that contains voting information that can be broken by an unbounded adversary. It is *trusted* to delete these values after the election ends thus making them unavailable in the future. Whether this trust is justified or not, is an open problem [23]. The best known schemes that provide everlasting privacy based on anonymous casting are [35, 25]. Both these works also provide coercion resistance by using the observation that an anonymous channel during casting can be used to thwart the forced abstention attack of a coercer together with everlasting privacy. The scheme of [35] utilizes deniable vote updating, where the voters initially follow the coercer's instructions, but deniably change their ballot later to reflect their true choice. On the other hand, [25] works in the JCJ setting with anonymous credentials and performs tallying in linear time to the total number of votes. In comparison, deniable vote updating requires stronger assumptions on the timing of the coercion attack, as a last-minute adversary has an easier task, while anonymous credentials require the voters to handle cryptographic material. Specialized voting devices or more user-friendly techniques like panic passwords [11] may be used to make such schemes practical.

Regarding the degree of everlasting privacy achieved, we can distinguish two variations: *practical* (or *weak*) and *strong*. Practical everlasting privacy [2] protects against an adversary that can break cryptographic assumptions, but only has access to publicly available data (long) after the election has ended. Strong everlasting privacy [24, 23] on the other hand, also protects against adversaries that have access to insider data maintained by the election authorities or intermediaries. It is clear that schemes based on public commitments cannot achieve strong everlasting privacy because an insider can access the contents of the secret ballot box (e.g. decommitment values) and reveal voter preferences. A criticism of anonymous casting [27], on the other hand, considers it to be a very strong assumption and finds it difficult to combine with verifiability. However, there are some clear advantages as well [23]. Firstly, it can provide strong everlasting privacy since there is no need for a secret ballot box. Everything submitted by the voters can be made public. Secondly, it can be used to provide ballot secrecy without trusting the election talliers for this property (as is common in all schemes descending from [17, 1]), because a future computationally unbounded adversary is equivalent to an untrusted contemporary tallier.

**Our contribution** This work can be classified in the fake-credentials and anonymous-casting paradigm. Our novel voting protocol provides coercion resistance (à la JCJ) together with everlasting privacy and ballot secrecy without sacrificing verifiability. To this end, we augment the design of [34] with a new ring signature scheme that achieves unconditional anonymity and admits fake credentials. Each vote in our protocol is accompanied by such a signature. The ring in our case consists of a list of credentials together with an extra one supplied by the voter themselves. If they are under coercion, the latter is invalid causing the ballot to be discarded. During a moment of privacy, they utilize their registered (genuine) credential. The construction of our credentials prevents the coercer from distinguishing these cases. The unconditional anonymity provided by our novel linkable ring signature creates an anonymous channel at the endpoints of the system and achieves everlasting privacy. Tallying complexity is quadratic as in JCJ [31], but in our scheme vote casting is a simple one-move operation by the voter, an improvement over [25] which requires voters to send two messages in different protocol phases (both over an anonymous channel). This balances the quadratic tallying time of our scheme against the linear time of [25]. Furthermore, anonymous casting is an inherent feature of our protocol. Thus, in our case, an external anonymous channel is required only to hide networks addresses and defend against side-channel attacks.

## 2 Preliminaries

**Notation** Let $\lambda$ denote the security parameter. The number of voters is $n$ and each voter is indexed by $i \in [n] = \{1, \dots, n\}$. We use the term *vote* to refer to the choice of the voter (in plaintext form), while the term *ballot* stands for the encrypted vote together with credentials and proofs of validity. In our scheme each ballot contains exactly $\beta$ credentials indexed by $j \in [\beta]$. Sampling uniformly at random from a set is denoted by $\leftarrow\!\!\$$, appending by $\Leftarrow$, assignment by $\leftarrow$, equality by $=$, and concatenation by $|$. We denote by $\mathsf{params}$ the cryptographic groups on which our schemes operate and the related sets (e.g. message, signature, pseudoidentity and event spaces). It is an input to all our algorithms and it will be omitted for brevity. It includes a group $\mathbb{G}$ of prime order $q$ where the DDH assumption holds, a hash function $\mathsf{H}_{\mathbb{G}}$ that maps binary strings to group elements, and a hash function $\mathsf{H}_q$ that maps binary strings to elements of $\mathbb{Z}_q$. We denote a list as $L$, its length as $|L|$ and a list item as $L_j$. Since our protocol utilizes both an encryption and a signature scheme, we denote the encryption keys by $(\mathsf{pk}, \mathsf{sk})$ and the signature keys (which also double as voter credentials) by $(\mathsf{pc}, \mathsf{sc})$. We also use the dot notation to refer to items of a tuple.

**Encryption** We require an IND-CPA-secure public key cryptosystem with homomorphic properties that can support distributed key generation and threshold decryption, like the ElGamal scheme [22]. We denote the ElGamal encryption of a group element $\mathtt{m} \in \mathbb{G}$ with randomness $r \in \mathbb{Z}_q$ under the public key $\mathsf{pk}$ as $\mathsf{Enc}_{\mathsf{pk}}(\mathtt{m}; r) = (g^r, \mathtt{m} \cdot \mathsf{pk}^r)$. A variation of our scheme may also use the exponential (or additive) version of ElGamal where $\mathtt{m} \in \mathbb{Z}_q$ and $g^{\mathtt{m}}$ is encrypted instead of $\mathtt{m}$. Given an ElGamal ciphertext $c = (c_1, c_2)$ we denote its reencryption with randomness $r' \in \mathbb{Z}_q$ under the same public key $\mathsf{pk}$ as $\mathsf{ReEnc}_{\mathsf{pk}}(c, r') = c \odot \mathsf{Enc}_{\mathsf{pk}}(1; r') = (c_1 \cdot g^{r'}, c_2 \cdot \mathsf{pk}^{r'})$, where $\odot$ is the elementwise multiplication. Similarly, the notation $c^z$ is used for $(c_1^z, c_2^z)$.

**Non-Interactive Zero-Knowledge Proofs of Knowledge** In verifiable e-voting schemes the voters and the various election authorities must provide proofs that they correctly executed the actions prescribed in the protocol without revealing their private inputs (e.g. candidate choices, secret keys or private credentials). Any interested party can then check these proofs and be convinced that the protocol was executed correctly.

We employ $\Sigma$-protocols for this task, which are public-coin 3-move interactive protocols ($\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Vrfy}$) where $\mathsf{NIZK.Prove}$ consists of the following steps: commitment by the prover ($\mathsf{Com}$), challenge by the verifier and response by the prover ($\mathsf{Resp}$). These protocols have the security properties of completeness, special-soundness and special honest-verifier zero-knowledge. The latter property implies the existence of a simulator $\mathsf{Sim}$ that creates accepting proofs for any public input. They can be made non-interactive ($\mathsf{NIZK}$), and thus publicly verifiable, with the Fiat-Shamir transform [20]. We use its strong version by including all public parameters in the hash to avoid the attacks of [8, 36].

Most standard $\Sigma$-protocols in the literature originate from the proof of knowledge of a discrete logarithm $\pi_S$ of Schnorr [40] and the proof of equality of discrete logarithm $\pi_{CP}$ (or equivalently proof that a tuple of group elements is a Diffie-Hellman tuple) due to Chaum and Pedersen [10]. The $\mathsf{NIZK}$ proofs commonly utilized in e-voting schemes are:

- $\pi_{\mathsf{Enc}}$: proof that the selected vote is a valid candidate encoding, which is constructed as a disjunction [16] of proofs that a ciphertext $c$ corresponds to a known message $\mathtt{m}$. It is important to stress that the $\mathsf{Enc} + \mathsf{PoK}$ paradigm with an IND-CPA secure encryption scheme together with the strong Fiat-Shamir transform provides NM-CPA security [8].
- $\pi_{\mathsf{Dec}}$: proof that a ciphertext $\mathsf{Enc}(\mathtt{m})$ has been correctly decrypted to $\mathtt{m}$ [17].
- $\pi_{\mathsf{ReEnc}}$: proof that ciphertext $c'$ correctly reencrypts ciphertext $c$.
- $\pi_{\mathsf{sc}}$: proof that a user knows a secret credential $x, y$ such that $c = \mathsf{Enc}_{\mathsf{pk}}(g^x h^y; r)$, constructed using a variation of the idea from [26].

A designated verifier proof [30], denoted by $\delta$, receives as an additional input the public key of a 'legitimate' verifier. This has the effect that only this verifier can be convinced that the proof is valid, as they can simulate proofs using their private key. In our scheme we utilize a designated verifier proof $\delta_{\mathsf{ReEnc}}$ that $c'$ is a correct reencryption of $c$ due to [28].

We provide more details for the construction of these proofs in Appendix A.

**Plaintext Equivalence Test (Proof)** An essential component of all JCJ-related schemes is a proof that two ciphertexts, encrypted with the same public key, hide the same plaintext. This proof is generated by a group of players, each holding shares of the decryption key, using the $\mathsf{PET}$primitive from [29], i.e.

$$\mathsf{PET}(\mathsf{Enc}_{\mathsf{pk}}(\mathtt{m}_1); \mathsf{Enc}_{\mathsf{pk}}(\mathtt{m}_2)) = 1 \Leftrightarrow \mathtt{m}_1 \equiv \mathtt{m}_2$$

A $\mathsf{PET}$ can be instantiated in a distributed El Gamal setting using the techniques of [29, 12, 36]. At first, each player 'divides' the two ciphertexts $c_1, c_2$ elementwise and the result $c = (\frac{c_{11}}{c_{21}}, \frac{c_{12}}{c_{22}})$ is blinded by each, using a blinding factor chosen uniformly at random, thus producing $c^{z_i} = ((\frac{c_{11}}{c_{21}})^{z_i}, (\frac{c_{12}}{c_{22}})^{z_i})$ Then everyone commits to $c^{z_i}$ and provides proofs $\pi_{CP}$ of correct construction (i.e that the same $z_i$ was used in both components). After all commitments and proofs have been published and verified, every player multiplies all $c^{z_i}$ together. The result $c' = \prod_i c^{z_i}$ is threshold decrypted and a proof $\pi_{\mathsf{Dec}}$ is generated. If $c_1, c_2$ indeed hid the same plaintext, the decryption yields 1, otherwise a random group element. The proofs of decryption and correct construction shall be collectively denoted as $\pi_{\mathsf{PET}}$. We note that in a practical implementation all the provisions of [36] (i.e. strong Fiat-Shamir transform, checks for trivial cases) must be included in order for the $\mathsf{PET}$ to be a proof even if all parties are dishonest.

**Verifiable shuffles** We require a functionality $\mathsf{Shuffle}$ that receives a list of items and aims to anonymize them, i.e. to stop an attacker from tracing the processing of an item in a particular position. In our protocol the items are the ballots and credentials of the voters. The $\mathsf{Shuffle}$ functionality is usually instantiated using permutations and reencryptions to alter both the position and the form of the ciphertexts. For verifiability purposes a $\mathsf{NIZK}$ proof of correct processing $\pi_{\mathsf{Shuffle}}$ is returned as an output as well. Verifiable shuffles are a very well studied topic in the literature and many such schemes exist. An implementation of our scheme could use a shuffle similar to [12].

## 3 A linkable ring signature with unconditional anonymity

The proposed voting scheme is based on a new linkable ring signature (LRS) inspired from [34, 33].

**Definition 1.** *A* $\mathsf{LRS}$ *scheme is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Link})$*:*

- $\mathsf{params} \leftarrow \mathsf{LRS.Setup}(\lambda)$. *Generates the system parameters.*
- $(\mathsf{sc}, \mathsf{pc}) \leftarrow \mathsf{LRS.KGen}()$. *Produces the secret and public credentials.*
- $\sigma \leftarrow \mathsf{LRS.Sign}(L, \mathtt{ev}, \mathsf{sc}, \mathtt{m})$. $\mathsf{Sign}$ *is the algorithm that is used to sign a message* $\mathtt{m}$ *by some* $\mathsf{sc}$ *with public counterpart in the ring* $L$ *for event* $\mathtt{ev}$.
- $0\backslash 1 \leftarrow \mathsf{LRS.Vrfy}(L, \mathtt{ev}, \mathtt{m}, \sigma)$ *is the public verification algorithm which outputs* $1$ *if the signature is valid or* $0$ *if it is not.*
- $0\backslash 1 \leftarrow \mathsf{LRS.Link}(\sigma_1, \sigma_2)$ *is the public linking algorithm which outputs* $1$ *if valid signatures* $\sigma_1$ *and* $\sigma_2$ *originate from the same signer for the same event.*

In our proposal, the $\mathtt{ev}$ variable will be produced using a hash function on the public election parameters (such as voting issue, candidate list, date etc.) It will serve as a unique election identifier.

Linkable ring signatures were first used for electronic voting in [34]. The security properties of our signature and their relation to the security of our voting scheme are:

- **Unforgeability**: Only the holder of the signing key can produce valid signatures. This property will be used to ensure verifiability and prevent double voting (in concert with linkability).
- **Anonymity**: The identity of the voter is hidden inside the ring, which serves as an anonymity set. This will help achieve (everlasting) privacy.
- **Linkability**: Given two signatures created by the same ring member for the same event, the $\mathsf{Link}$ algorithm will always return 1. As the anonymity property can allow malicious voters to vote twice, the linkability property can make double voting detectable and thus avoidable.
- **Non-slanderability**: Given a signature created by a ring member no one can create a valid signature that is linked to it. This ensures that no one can update the vote of a voter except for the voter themselves.

In Figure 1, we present our linkable ring signature for our voting scheme (cf. section 4). In order to participate, each player invokes the $\mathsf{LRS.KGen}$ function, which generates the pair of public and secret credentials. Then each player publishes $\mathsf{pc}$. As a result, the ring in our construction comprises a public list of ElGamal encrypted values [1] of the form:

$$L = \big(\mathsf{pc}_1, \ldots, \mathsf{pc}_n\big) = \big(\mathsf{Enc}_{\mathsf{pk}}(g^{x_1} h^{y_1}; r_1), \ldots,\ \mathsf{Enc}_{\mathsf{pk}}(g^{x_n} h^{y_n}; r_n)\big)$$

To create a signature, a signer that knows the secret credential corresponding to an item $L_i$ of $L$ invokes the signing algorithm $\mathsf{LRS.Sign}$. The secret credential is a tuple consisting of the secrets $x_i, y_i$ and the randomness $r_i$ used to create $L_i$. In essence, the signature is a $\mathsf{NIZK}$ proof of knowledge of this secret credential. Note that the ring creation is ad-hoc, as each signer can select a subset of the published credentials at will. To verify the signature, everyone can invoke the algorithm $\mathsf{LRS.Vrfy}$. Finally, to check if two signatures are linked, everyone can invoke the $\mathsf{LRS.Link}$, which checks if both signatures verify correctly and contain the same linking tag.

## 3.1 Security Analysis

**Theorem 1 (Unforgeability).** *Our* $\mathsf{LRS}$ *has the property of unforgeability in the random oracle model, according to the definition of [34], given that* $\mathsf{DLOG}$ *is hard.*

*Proof Sketch.* Assume a $\mathsf{PPT}$ adversary $\mathcal{A}$ that with non-negligible probability can forge a signature $\sigma$. We construct an algorithm $\mathcal{B}$, that given $n$ $\mathsf{DLOG}$ instances $\{X_i\}_{i=1}^n$ uses $\mathcal{A}$ to obtain a forgery $\sigma_0^*$. $\mathcal{B}$ rewinds $\mathcal{A}$ and by the rewind-on-success lemma [34], $\mathcal{A}$ will produce another forgery $\sigma_1^*$ with non-negligible probability. Then, $\mathcal{B}$ uses $\sigma_0^*$ and $\sigma_1^*$ to solve the $\mathsf{DLOG}$ problem on at least one of the given challenges $\{X_i\}_{i=1}^n$. The complete proof is in Appendix B.1.

**Theorem 2 (Anonymity).** *Our* $\mathsf{LRS}$ *has unconditional anonymity, as defined in [3].*

*Proof Sketch.* Assume a computationally unbounded adversary $\mathcal{A}$ and let $\sigma$ be a challenge signature given to $\mathcal{A}$. The proof is based on the idea that the linking tag of the signature cannot reveal the signer. Therefore, even if $\mathcal{A}$ can solve $\mathsf{DLOG}$ for the linking tag $\mathtt{t} = e^x$ and decrypt all the public credentials, they cannot distinguish to which signer $\mathtt{t}$ belongs to, since it is equally likely to have been produced by each one. More details are presented in Appendix B.2.

**Theorem 3 (Linkability).** *Our* $\mathsf{LRS}$ *has the property of strong linkability in the random oracle model, according to the definition of linkability of [4], if* $\mathsf{DLOG}$ *is hard.*

[1] Our signature can be implemented absent an encryption scheme without sacrificing any of its security properties. Encryption in our construction is used to easily integrate the $\mathsf{PET}$ functionality later in the voting scheme (cf. section 4)

```
$\mathsf{LRS.Setup}(1^\lambda)$
Generate a group $\mathbb{G}$ with prime order $q$
Select generators $g, h \leftarrow\!\!\$\ \mathbb{G}$.
Choose $\mathsf{H}_{\mathbb{G}} : \{0,1\}^* \rightarrow \mathbb{G}$ and $\mathsf{H}_q : \{0,1\}^* \rightarrow \mathbb{Z}_q$
Select $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{Z}_q$ and set $\mathsf{pk} \leftarrow g^{\mathsf{sk}}$
  as an ElGamal key pair
return $\mathsf{params} = (\mathbb{G}, q, g, h, \mathsf{H}_{\mathbb{G}}, \mathsf{H}_q, \mathsf{pk})$
```

```
$\mathsf{LRS.KGen}(\mathsf{params})$
Each participant:
  Samples $x, y, r \leftarrow\!\!\$\ \mathbb{Z}_q$
  Sets $\mathsf{sc} \leftarrow (x, y, r)$
  Computes $g^x \cdot h^y$
  $\mathsf{pc} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(g^x h^y; r)$
return $(\mathsf{sc}, \mathsf{pc})$
```

```
$\mathsf{LRS.Sign}(L, \mathtt{ev}, \mathsf{sc}_i, \mathtt{m})$
Parse $\mathsf{sc}_i = (x_i, y_i, r_i)$
Compute $e \leftarrow \mathsf{H}_{\mathbb{G}}(\mathtt{ev})$
Compute linking tag $\mathtt{t} \leftarrow e^{x_i}$
Sample $\alpha, \beta, \gamma \leftarrow\!\!\$\ \mathbb{Z}_q$
Compute
  $K_i \leftarrow g^\alpha h^\gamma \mathsf{pk}^\beta$
  $K'_i \leftarrow g^\beta$
  $K''_i \leftarrow e^\alpha$
  $c_{i+1} \leftarrow \mathsf{H}_q(\mathsf{params}, L, \mathtt{t}, K_i, K'_i, K''_i, \mathtt{m})$
for $j = i+1, \ldots, n, 1, \ldots, i-1$ do
  Parse $(L_{j1}, L_{j2}) = L_j$
  $s_j, t_j, p_j \leftarrow\!\!\$\ \mathbb{Z}_q$
  $K_j \leftarrow g^{t_j} \cdot h^{p_j} \cdot \mathsf{pk}^{s_j} \cdot (L_{j2})^{c_j}$
  $K'_j \leftarrow g^{s_j} \cdot (L_{j1})^{c_j}$
  $K''_j \leftarrow e^{t_j} \mathtt{t}^{c_j}$
  $c_{j+1} \leftarrow \mathsf{H}_q(\mathsf{params}, L, \mathtt{t}, K_j, K'_j, K''_j, \mathtt{m})$
The signer sets
  $s_i \leftarrow \beta - c_i r_i$
  $t_i \leftarrow \alpha - c_i x_i$
  $p_i \leftarrow \gamma - c_i y_i$
return $\sigma = (c_1, \{s_j\}_{j=1}^n, \{t_j\}_{j=1}^n, \{p_j\}_{j=1}^n, \mathtt{t})$
```

```
$\mathsf{LRS.Vrfy}(L, \mathtt{ev}, \mathtt{m}, \sigma)$
Parse $\sigma = (c_1, \{s_j\}_{j=1}^n, \{t_j\}_{j=1}^n, \{p_j\}_{j=1}^n, \mathtt{t})$
Compute $e \leftarrow \mathsf{H}_{\mathbb{G}}(\mathtt{ev})$
for $j = 1 \ldots n$ do
  Parse $(L_{j1}, L_{j2}) = L_j$
  $K_j \leftarrow g^{t_j} \cdot h^{p_j} \cdot \mathsf{pk}^{s_j} \cdot L_{j2}^{c_j}$
  $K'_j \leftarrow g^{s_j} L_{j1}^{c_j}$
  $K''_j \leftarrow e^{t_i} \mathtt{t}^{c_j}$
  $c_{j+1} \leftarrow \mathsf{H}_q(\mathsf{params}, L, \mathtt{t}, K_j, K'_j, K''_j, v)$
return 1 if and only if
$c_1 = \mathsf{H}_q(\mathsf{params}, L, \mathtt{t}, K'_n, K''_n, K'''_n, \mathtt{m})$
```

```
$\mathsf{LRS.Link}(\sigma_1, \sigma_2)$
Parse $\sigma_1 = (\cdot, \cdot, \cdot, \cdot, \mathtt{t}_1)$
Parse $\sigma_2 = (\cdot, \cdot, \cdot, \cdot, \mathtt{t}_2)$
if $\mathtt{t}_1 = \mathtt{t}_2$
   and both signatures verify correctly then
return 1
else return 0
```

**Fig. 1.** Our LRS construction

*Proof Sketch.* Assume a PPT adversary $\mathcal{A}$ that owns $k-1$ secret credentials and can produce $k$ pairwise unlinkable signatures with non-negligible probability. We construct an algorithm $\mathcal{B}$, that given $n$ DLOG instances $\{X_i\}_{i=1}^n$, obtains through $\mathcal{A}$, $k$ pairwise unlinkable signatures $\{\sigma_i^*\}_{i=1}^k$. $\mathcal{B}$ rewinds $\mathcal{A}$ $k$ times, and by the rewind-on-success lemma [34], $\mathcal{A}$ will produce $k$ forgeries $\{\sigma'_i\}_{i=1}^k$. Then, since $\mathcal{A}$ knows only $k-1$ secret credentials, $\mathcal{B}$ uses the forgeries $\{\sigma_i^*\}_{i=1}^k$ and $\{\sigma'_i\}_{i=1}^k$ to solve the DLOG problem on one of the given challenges $\{X_i\}_{i=1}^n$. The full proof can be found in Appendix B.3.

**Theorem 4 (Non-slanderability).** *Our* LRS *has the property of non-slanderability, in the random oracle model as defined in [33], given that* DLOG *is hard.*

The proof is implied by the unforgeability and strong linkability of our $\mathsf{LRS}$ construction.

## 4 A Voting Scheme using Linkable Ring Signatures

**Syntax** A voting scheme consists of the following entities:

- a set of $n$ eligible voters $\mathsf{V} = \{\mathsf{V}_1, \ldots, \mathsf{V}_n\}$ identified by their index in $\mathsf{V}$.
- a set of $k$ candidates $\mathsf{CS} = \{\mathsf{cnd}_1, \cdots \mathsf{cnd}_k\}$,
- the registration authority ($\mathsf{RA}$), distributed among a set of $n_{\mathsf{RA}}$ registrars,
- the tallying authority ($\mathsf{TA}$) consisting of $n_{\mathsf{TA}}$ talliers,
- A bulletin board ($\mathsf{BB}$) which can be considered as an authenticated append-only ledger. It contains all public election data. For clarity, we split the $\mathsf{BB}$ into sections named after the corresponding phases of the voting protocol.

In an implementation, the voters would be represented by the *voting clients* which are combinations of software and hardware components to handle cryptographic operations - they can even be realized in a web browser like in [1]. Clients for JCJ-related schemes require the extra functionality of generating fake credentials, which can be either included as a component of the client or by using the more user friendly mechanism of panic passwords [11]. We also assume an election supervisor ($\mathsf{EA}$) to invoke the various server-side functionalities. The $\mathsf{EA}$ is involved with maintenance procedures so it is considered honest.

**Definition 2.** *A voting scheme is a tuple* $\mathsf{VS} = (\mathsf{Setup}, \mathsf{Register}, \mathsf{SetupElection}, \mathsf{Vote}, \mathsf{IsValid}, \mathsf{Shuffle}, \mathsf{Tally}, \mathsf{Vrfy})$ *where:*

- $(\mathsf{params}, \mathsf{pk}, (\mathsf{pk}_i)_{i=1}^{n_{\mathsf{TA}}}, (\mathsf{sk}_{\mathsf{TA}_i})_{i=1}^{n_{\mathsf{TA}}}) \leftarrow \mathsf{VS.Setup}(\lambda)$, *is a PPT algorithm that generates the cryptographic parameters of the voting scheme.*
- $(\mathsf{pc}_i, \mathsf{sc}_i) \leftarrow \mathsf{VS.Register}(i)$, *allows voter $i$ to generate their credentials.*
- $(L, \mathsf{CS}, \mathsf{ev}) \leftarrow \mathsf{SetupElection}()$ *generates configuration data for a specific election and returns the public credential list for eligible voters.*
- $b_i \leftarrow \mathsf{VS.Vote}(\mathsf{cnd}, \mathsf{sc}_i, i)$, *generates the ballot $b_i$ for voter $\mathsf{V}_i$ given the candidate choice* $\mathsf{cnd}$ *and the secret credential of* $\mathsf{V}_i$.
- $0\backslash 1 \leftarrow \mathsf{VS.IsValid}(b, \mathsf{BB})$ *is an algorithm executed by the* $\mathsf{BB}$. *It returns* $1$ *if the ballot $b$ can be added to the* $\mathsf{BB}$.
- $\pi_{\mathsf{Shuffle}} \leftarrow \mathsf{VS.Shuffle}(\mathsf{BB})$ *is an algorithm that shuffles the ballots in the* $\mathsf{BB}$ *and returns a proof of correct operation.*
- $(T, \pi) \leftarrow \mathsf{VS.Tally}(\mathsf{sk}_{\mathsf{TA}}, \mathsf{BB})$, *outputs the result of the election $T$, together with proof(s) $\pi$ of the correctness of $T$.*
- $0\backslash 1 \leftarrow \mathsf{VS.Vrfy}(\mathsf{BB}, T, \pi)$ *allows anyone to verify the election result.*

### Our construction

Our proposed voting scheme is depicted in Figure 2 and described next.

**Setup** In the setup phase the $\mathsf{EA}$ initiates a variation of the $\mathsf{LRS.Setup}$ algorithm. The difference from the one in Figure 1 is that a distributed key generation phase is executed. Consequently the $\mathsf{TA}$ secret key $\mathsf{sk}_{\mathsf{TA}}$ is split into $n_{\mathsf{TA}}$ parts $\mathsf{sk}_{\mathsf{TA}_i}$ with public counterparts $\mathsf{pk}_{\mathsf{TA}_i}$. The joint $\mathsf{TA}$ public key is $\mathsf{pk}$. The returned values $\mathsf{params}, \mathsf{pk}_{\mathsf{TA}_i}, \mathsf{pk}$ are posted on the $\mathsf{BB}$.

**Registration** The registration phase utilizes the $\mathsf{VS.Register}$ and $\mathsf{VS.SetupElection}$ algorithms. The former is executed by each voter in concert with the $\mathsf{RA}$ and its objective is to enable the voters to create and enroll their real credentials. Each voter $\mathsf{V}_i$ executes $\mathsf{LRS.KGen}$ to compute $(\mathsf{pc}_i, \mathsf{sc}_i)$. The secret credential is the real credential that will be used to indicate that the vote is the input of the voter. Any other value for $\mathsf{sc}_i$ indicates coercion. The public credential is encrypted using $\mathsf{pk}$. The voter sends to the $\mathsf{RA}$ their index[2] $i$ along with their public credential $\mathsf{pc}_i$ and a proof of knowledge $\pi_{\mathsf{sc}}$ of its secret counterpart to avoid replay attacks. The $\mathsf{RA}$ reencrypts the credential $\mathsf{pc}'_i \leftarrow \mathsf{ReEnc}(\mathsf{pc}_i, r'_i)$, and provides a designated-verifier proof of correct reencryption ($\delta_{\mathsf{ReEnc}}$) that does not reveal the randomness used. To create this proof, every voter must have a public key $\mathsf{pk}_i$ which they generate themselves and send to the $\mathsf{RA}$ during this registration phase. Using its secret counterpart $\mathsf{sk}_i$ will allow the voter to fake the proof $\delta_{\mathsf{ReEnc}}$ for the coercer. After all the voters have registered and a specific election begins, the $\mathsf{RA}$ filters only the eligible ones and executes the $\mathsf{SetupElection}$ algorithm to produce the list of candidates $\mathsf{CS}$, and $\mathsf{ev} = \mathsf{H}_{\mathbb{G}}(\mathsf{CS}, \mathsf{V}, \mathsf{issue}, \mathsf{params}, \mathsf{pk}, \mathsf{pk}_{\mathsf{TA}_i})$ as the election identifier. Note that $\mathsf{issue}$ denotes a description of the question that the particular election aims to settle. The election authority also selects the global size of the anonymity set, denoted by $\beta$. This will be the size of the ring for each signature that comes with the ballot. If a ballot is accompanied with a ring of different size it will be rejected as it could be used as a tag to enable coercion (cf. section 5.4) Then it posts to the $\mathsf{BB}$ the public credential $\mathsf{pc}'_i$ along with the identities of the voters. Thus, after the registration phase, the $\mathsf{BB}$ contains the list of (reencrypted) public credentials of the $n$ eligible voters $L^{(0)} = (\mathsf{pc}'_1, \ldots, \mathsf{pc}'_n)$, where each voter only knows their own secret credential $\mathsf{sc}_i$ and can infer its position/index $i$ in the list. Since the $\mathsf{RA}$ has reencrypted the credentials, $\mathsf{V}_i$ does not know the final randomness used in encrypting $\mathsf{pc}_i$.

**Voting** In order to generate a vote for candidate $\mathsf{cnd}$, the voter invokes the $\mathsf{VS.Vote}(\mathsf{cnd}, \mathsf{sc}^*_i, i)$ algorithm, where $\mathsf{sc}^*_i$ is their secret credential, real or fake.

Cast-as-intended verifiability can be provided through a cut-and-choose mechanism, like the Benaloh-challenge [5]; the voter selects their candidate of choice and then the voting client prepares the ballot. The voter is given the option to audit or cast it. In the former case, the voting client reveals the randomness and other parameters used to create the ballot, so that the voter can externally replicate the process to verify its correctness. Of course an audited ballot is never

[2] In an implementation of our scheme the index $i$ can be replaced by real-world identification

cast. In order to avoid clash attacks, our scheme follows the recommendation of [32] by generating the randomness used to encrypt the candidate choice in cooperation with the voter. As suggested in [32], the voter may simply type a random string consisting of a specified number of characters which is combined with randomness generated by the voter device to create the final value that will be used when encrypting $\mathsf{cnd}$.

If the voter is under coercion, the value of $\mathsf{sc}_i^*$ is generated on the fly by the voting client. During the moment of privacy, the voter uses the credential created during the registration phase, i.e. $\mathsf{sc}_i^* = \mathsf{sc}_i$ and disregards all instructions of the coercer. In both cases, the voter encrypts $\mathsf{sc}_i^*$ to obtain a credential $\mathsf{pc}_i^*$. Afterwards they obtain the list of encrypted credentials $L^{(0)}$ from the $\mathsf{BB}$ and select the rest $\beta - 1$ credentials in a random order which they reencrypt. In order to check that the voting client correctly reencrypted the credentials, the Benaloh challenge mechanism is employed again. For usability, the audit or cast challenge applies to all $\beta - 1$ credentials and not to each individual one. These credentials will serve as the anonymity set. Finally, the voter embeds $\mathsf{pc}_i^*$ in a random position among the $\beta - 1$ decoys. As a result, a new list $L^{(i)}$ containing $\beta$ credentials is produced by every voter. Then they encrypt their candidate choice as $\mathsf{Enc}(\mathsf{cnd}; r)$ and produce a proof $\pi_{\mathsf{Enc}}^{L^{(i)}}$ that $\mathsf{cnd}$ is a valid candidate choice from $\mathsf{CS}$ and that it is encrypted correctly. This proof is made non-interactive using the strong Fiat-Shamir heuristic, where the input to the hash function also contains the actual ring $L^{(i)}$ used during signature generation. Thus the hash call contains the full statement and as a result the ballot is not malleable ([8]).

In the end, the voters invoke the $\mathsf{LRS.Sign}$ algorithm to sign $\mathtt{m}_i = \mathsf{Enc}(\mathsf{cnd}; r) \mid \pi_{\mathsf{Enc}}^{L^{(i)}}$. As the signature is a proof of knowledge of the secret credential $\mathsf{sc}_i$, there is no need to include an extra proof in this stage. The ballot returned from the voting algorithm consists of $(\mathtt{m}_i, L^{(i)}, \sigma)$ and appended to the $\mathsf{BB}$.

We emphasize the use of the Benaloh challenge to allow the voter to check that the decoy credentials have been properly reencrypted. This protects individual verifiability. Note that this procedure does not interfere with coercion resistance. The voter will follow the same steps, albeit with different credentials - the real ones during the moment of privacy and the fakes when the coercer is present, making the proof valid in both cases. Universal verifiability, is also maintained since the signature makes any corruption in the ballot during tallying detectable, and successful PETs guarantee that the decoy credentials correspond to the ones in $L^{(0)}$. Another possible option would have been to equip all ballots with proofs of correct reencryption ($\pi_{\mathsf{ReEnc}}$) for the credentials in the anonymity set. This however would break ballot 'symmetry', as it would contain $\beta - 1$ proofs of correct reencryptions *only* for the decoy credentials, thus giving away which element of $L^{(i)}$ is the encryption of the real one. Thus, the adversary against everlasting privacy could use its advanced power to decrypt it and then compare it against the decrypted contents of the $\mathsf{BB}$ during the election setup phase. This would allow them to find out if a credential is valid or not and in the former case to uncover both the identity and the preference of a targeted voter.

**Ballot weeding** Each ballot undergoes some checks to validate its well-formedness and to protect against replay attacks by $\mathsf{IsValid}$ algorithm which makes sure that:

- the format of the ballot is the one specified from $\mathsf{VS.Vote}$
- $L^{(i)}$ contains exactly $\beta$ items.
- there does not exist an exact copy of the ballot currently in the $\mathsf{BB}$.
- there does not exist a duplicate of $\mathtt{m}_i$ in $\mathsf{BB}$
- the commitments (inputs to the hash function) for $\pi_{\mathsf{Enc}}^{L^{(i)}}$ cannot be found in any other such proof in the $\mathsf{BB}$.

The $\mathsf{IsValid}$ algorithm can be executed by any interested party (either a voter, election administrator or election observer).

**Mixing** When the voting phase has concluded, the $\mathsf{EA}$ discards ballots with invalid signatures and invalid proofs of well-formedness. Then it invokes the $\mathsf{LRS.Link}$ algorithm for all ballots to see if there are votes that were cast using the same credential. If such are found, only one is kept according to a pre-specified policy (e.g. the one that was cast later). After these tests, the $\mathsf{EA}$ initiates shuffling to anonymize the ballots so that the coercer loses track of the ballot they cast and the position of the credential in the credential list. Initially the list $L^{(0)}$ is shuffled. For each ballot $b_i$, only the encrypted voter choice $\mathsf{Enc}(\mathsf{cnd})$ and the encrypted credential list $L^{(i)}$ are kept. The latter is shuffled to prevent the use of the credential list as a tag (horizontal shuffle). Subsequently, the list of all the ballots is shuffled again (vertical shuffle).

**Tally** The objective of the tally phase is to verifiably remove coerced ballots and to count only the ones that were cast using the real credentials of the voters. To this end, the members of the $\mathsf{TA}$ utilize the $\mathsf{PET}$ functionality. In particular, for each ballot $b_i$, each element of the list $L^{(i)}$ is compared with all the elements of the initial list $L^{(0)}$. If each of the credentials in $L^{(i)}$ reencrypt some element in $L^{(0)}$, it means that $b_i$ was cast with the correct credentials and the choice of the voter must be included in the tally. If, on the other hand, there exists an element of $L^{(i)}$ that does not correspond to an element of $L^{(0)}$, then the ballot is considered a product of coercion and thus discarded. Since during the mixing phase both the order of the ballots and the order of the credential list was shuffled the coercer cannot tell if their own ballot was discarded or not.

From this stage on, two types of tallying on the encrypted candidate choice may be applied. Firstly, all the acceptable ciphertexts may be homomorphically combined - assuming that the encryption scheme is the exponential ElGamal. Then the talliers will jointly decrypt the 'aggregate ballot' and announce the election result along with a proof of correct computation $\pi_{\mathsf{Dec}}$. Alternatively, each $\mathsf{Enc}(\mathsf{cnd})$ corresponding to an accepted ballot may be decrypted along with a proof of correct decryption for verifiability - for conciseness we depict only this case in Figure 2. After all ballots have been decrypted the result calculation function can be applied to the plaintexts corresponding to each accepted ballot.

Then the election result is announced. The second option is of more general use, as it can be employed in protocols with elaborate vote counting functions.

Everyone can verify the result by checking the proof $\pi_{\mathsf{Enc}}^{L^{(i)}}$ and verifying the signature in each ballot contained in the $\mathsf{BB}_{\mathsf{Vote}}$ section of the bulletin board. Afterwards, they can check the proofs for all $\mathsf{PET}$ in the $\mathsf{BB}_{\mathsf{discard}}$ and $\mathsf{BB}_{\mathsf{Tally}}$ section of the $\mathsf{BB}$ and all the proofs of correct ballot decryption $\pi_{\mathsf{Dec}}$. Finally, any interested party can reapply the result calculating function to the voters' candidate choices. These actions are part of $\mathsf{VS.Vrfy}$.

**Performance** We express the performance of our scheme by counting the operations executed by the voter during the voting phase and by the various authorities during the mixing and tallying phase.

The voters perform $\beta - 1$ reencryptions, 1 encryption and 1 sign operation which depends on the size of $L^{(i)}$. Thus the complexity of $\mathsf{VS.Vote}$ is $\mathcal{O}(\beta)$.

After voting has finished, assume that there are $m > n$ ballots in the $\mathsf{BB}$, since voters may vote multiple times to evade coercion or because they are following the instructions of the coercer. Before shuffling, the check of validity for all ballots requires $\mathcal{O}(nm)$ time, while discovering duplicates requires $\mathcal{O}\left(m^2\right)$ time. This can be reduced to $\mathcal{O}(m)$ by passing all the linking tags through a hash table. Assuming shuffling is implemented by reencryption and sorting the results as binary strings, it takes $\mathcal{O}(m \log m)$ time. During tallying, the $n$ credentials in $L^{(0)}$ participate in $\mathsf{PET}$ with the $\beta$ credentials contained in each of the $m$ ballots. Consequently, tallying takes $\mathcal{O}(\beta mn)$ time.

As a result, and assuming $\beta$ is constant, our scheme is of quadratic complexity, similar to most JCJ-related schemes. However, our scheme is also 'vote-and-go' and provides everlasting privacy, in contrast to [25] which provides similar security properties but requires two messages from the voters at different phases of the protocol. These benefits justify in our view the quadratic cost, which can be further managed by partitioning a large voting population to $\beta$-sized groups, in a manner similar to precincts in physical elections. Such organizational measures will be explored in future works.

## 5 Security Analysis

### 5.1 Assumptions

For all security properties we assume a $\mathsf{BB}$ that will not delete or reject ballots. The voting client is trusted for all security properties *except* verifiability.

For verifiability, the $\mathsf{TA}$ is not trusted and the adversary may corrupt some voters. To prove strong verifiability [13] we require that the $\mathsf{BB}$ and the $\mathsf{RA}$ are not simultaneously corrupted, in the sense that the $\mathsf{BB}$ will not stuff ballots *and* the $\mathsf{RA}$ will not handle credentials in a malicious way.

For (everlasting) privacy and coercion resistance, we trust the client not to reveal the voting choice and randomness used, to the (everlasting) privacy adversary and to the coercer. We also trust it to generate fake credentials during

```
$\mathsf{VS.Setup}(\lambda)$
$\mathsf{params} \leftarrow \mathsf{LRS.Setup}(\lambda)$
$\mathsf{TA}_i$ : Select $\mathsf{sk}_{\mathsf{TA}_i} \leftarrow\!\!\$\ \mathbb{Z}_q$
Compute joint $\mathsf{pk}$
$\mathsf{BB} \Leftarrow (\mathsf{params}, \mathsf{pk}, \mathsf{pk}_{\mathsf{TA}_i})$
```

```
$\mathsf{VS.SetupElection}(\mathsf{params})$
$\mathtt{ev} \leftarrow$
    $\mathsf{H}_{\mathbb{G}}(\mathsf{CS}, \mathsf{V}, \mathsf{issue}, \mathsf{params}, \mathsf{pk}, \mathsf{pk}_{\mathsf{TA}_i})$
$L^{(0)} = \left(\mathsf{pc}'_1, \cdots, \mathsf{pc}'_n\right)$
Select global ring size $\beta$
$\mathsf{BB}_{\mathsf{SetupElection}} \Leftarrow$
    $(L^{(0)}, \mathsf{V}, \mathsf{CS}, \mathtt{ev}, \mathsf{issue})$
for all eligible voters $\mathsf{V}_i$ $i \in [n]$ do
    $\mathsf{BB}_{\mathsf{SetupElection}} \Leftarrow (i, \mathsf{pc}'_i)$
```

```
$\mathsf{VS.Register}(i)$
$\mathsf{V}_i$ executes:
    $(\mathsf{pc}_i, \mathsf{sc}_i) \leftarrow \mathsf{LRS.KGen}(\mathsf{params})$
    $\mathsf{pk}_i \leftarrow g^{\mathsf{sk}_i}$ where $\mathsf{sk}_i \leftarrow\!\!\$\ \mathbb{Z}_q$
    Sends to the $\mathsf{RA}$ : $(i, \mathsf{pk}_i, \mathsf{pc}_i)$
$\mathsf{RA}$ executes
    $\mathsf{pc}'_i \leftarrow \mathsf{ReEnc}(\mathsf{pc}_i)$
    Sends to $\mathsf{V}_i$ : $(\mathsf{pc}'_i, \delta_{\mathsf{ReEnc}})$
```

```
$\mathsf{VS.Vote}(\mathsf{cnd}, \mathsf{sc}^*_i, i)$
Sample $r^*_i, r, \vec{r} \leftarrow\!\!\$\ \mathbb{Z}_q$
Parse $\mathsf{sc}^*_i = (x^*_i, y^*_i)$
Compute $\mathsf{pc}^*_i \leftarrow \mathsf{Enc}(g^{x^*_i} h^{y^*_i}, r^*_i)$
for $j = 1 \ldots \beta - 1$, $j \neq i$ do
    $L^{(i)}_j \leftarrow\!\!\$\ L^{(0)} \setminus \{L^{(0)}_i\}$
    $L^{(i)}_j \leftarrow \mathsf{ReEnc}(L^{(i)}_j, \vec{r}_j)$
Check reencryptions using Benaloh-challenge
Embed $\mathsf{pc}^*_i$ in a random position in $L^{(i)}$
Receive randomness $r_v$ by $\mathsf{V}_i$
Compute $\mathsf{Enc}(\mathsf{cnd}; \mathsf{H}_q(r||r_v)), \pi^{L^{(i)}}_{\mathsf{Enc}}$
Check encryption using Benaloh-challenge
Set $\mathtt{m}_i = \mathsf{Enc}(\mathsf{cnd}; r)|\pi^{L^{(i)}}_{\mathsf{Enc}}$
$\sigma_i \leftarrow \mathsf{LRS.Sign}(L^{(i)}, \mathtt{ev}, \mathsf{sc}^*_i, \mathtt{m}_i)$
$b_i = (\mathtt{m}_i, L^{(i)}, \sigma_i)$
$\mathsf{BB}_{\mathsf{Vote}} \Leftarrow b_i$
```

```
$\mathsf{VS.Shuffle}(\mathsf{BB})$
for each $b_i \in \mathsf{BB}_{\mathsf{Vote}}$ do
    if $\mathsf{NIZK.Vrfy}(b_i.\pi_{\mathsf{Enc}}) = 0$ or
        $\mathsf{LRS.Vrfy}(L^{(i)}, \mathtt{ev}, \mathtt{m}, b_i.\sigma_i) = 0$
            $\mathsf{BB}_{\mathsf{discarded}} \Leftarrow b_i$
    else
            $\mathsf{BB}_{\mathsf{valid}} \Leftarrow b_i$
for each $b_i, b_j \in \mathsf{BB}_{\mathsf{valid}}$ do
    if $\mathsf{LRS.Link}(b_i.\sigma, b_j.\sigma) = 1$
        $b \leftarrow$ RemoveDuplicate $(b_i, b_j)$
        $\mathsf{BB}_{\mathsf{unique}} \Leftarrow b$
$L^{(0)\prime} \leftarrow \mathsf{Shuffle}(L^{(0)})$
for each $b_i \in \mathsf{BB}_{\mathsf{unique}}$ do
    $L^{(i)\prime} \leftarrow \mathsf{Shuffle}(L^{(i)})$
    Remove $\sigma_i$
    From $\mathtt{m}_i$ keep $\mathsf{Enc}(\mathsf{cnd})_i$
    Set $b'_i \leftarrow (\mathsf{Enc}(\mathsf{cnd})_i, L^{(i)\prime})$
    $\mathsf{BB}_{\mathsf{clean}} \Leftarrow b'_i$
$\mathsf{BB}_{\mathsf{Shuffle}} \Leftarrow \mathsf{Shuffle}(\mathsf{BB}_{\mathsf{clean}}), \pi_{\mathsf{Shuffle}}$
```

```
$\mathsf{VS.IsValid}(b_i, \mathsf{BB})$
Parse $b_i = (\mathtt{m}_i, L^{(i)}, \sigma_i)$
if parse failed   return 0
if $|L^{(i)}| \neq \beta$   return 0
if $b_i \in \mathsf{BB}$   return 0
if $\exists b_j = (\mathtt{m}_j, \cdot, \cdot) \in \mathsf{BB} : \mathtt{m}_i = \mathtt{m}_j$   return 0
Let $(\mathsf{Com}_i, \mathsf{Resp}_i) = b_i.\mathtt{m}_i.\pi^{L^{(i)}}_{\mathsf{Enc}}$
if $\exists b_j = (\mathtt{m}_j, \cdot, \cdot) \in \mathsf{BB} : \mathtt{m}_j.\pi^{L^{(j)}}_{\mathsf{Enc}}.\mathsf{Com}_j = \mathsf{Com}_i$
    return 0
return 1
```

```
$\mathsf{VS.Tally}(\mathsf{sk}_{\mathsf{TA}}, \mathsf{BB})$
for each $b_i \in \mathsf{BB}_{\mathsf{Shuffle}}$ do
    for each $\mathsf{pc}_i \in L^{(i)}$ do
        if $\forall\, \mathsf{pc}_j \in L^{(0)}$ :   $\mathsf{PET}(\mathsf{pc}_i, \mathsf{pc}_j) = 0$
            $\mathsf{BB}_{\mathsf{discard}} \Leftarrow b, \pi_{\mathsf{PET}}$
            Continue with next ballot
    $(\mathsf{cnd}, \pi_{\mathsf{Dec}}) \leftarrow \mathsf{Dec}(\mathsf{sk}_{\mathsf{TA}}, \mathsf{Enc}(\mathsf{cnd})_i)$
    $\mathsf{BB}_{\mathsf{Tally}} \Leftarrow (\mathsf{cnd}, \pi_{\mathsf{Dec}})$
Apply tallying algorithm to $\mathsf{BB}_{\mathsf{Tally}}$
```

**Fig. 2.** Our VS construction

the coercion attack that are indistinguishable from the registered ones. For both these properties we allow the adversary to corrupt some voters and use them to cast arbitrary ballots (e.g. for replay attacks). For (everlasting) privacy we assume that the adversary has access to all messages posted by the voters in all phases of the protocol and that all the tallying authorities might be corrupted.

Regarding coercion resistance, we assume that the coercer does not have full control over a voter during the entirety of the election. If such were the case, they essentially *would become the voter* [12]. In particular, we assume that the voter has a moment of privacy to cast their ballot using their real credentials during elections and cannot be impersonated by the coercer during the registration phase. In practice this is implemented by using well-known techniques, e.g. through an untappable channel using a physical polling station like in [11]. While this option contradicts the idea of remote voting, the 'inconvenience' can be mitigated by reusing the credentials in many elections through a 'rebasing' mechanism (e.g. by changing the group generator from $g$ to $g^a$, where $a$ is randomly selected for each new election). Alternatively, as mentioned in [31], the transcript of the registration phase can be securely deleted or the voter would learn which member of the $\mathsf{RA}$ is corrupted. In this case, the voter could fake the transcript with the honest $\mathsf{RA}$ members using designated verifier proofs in order to prove the validity of *any* secret credential. This presumes that the voter has a registration key pair which is only used for this proof, but nowhere else in our scheme. Our protocol is compatible with all these scenarios, but for conciseness we described only the latter in Figure 2. We also assume (as in JCJ [31]) that the adversary does not have complete knowledge of the honest voters' behavior. This uncertainty can be implemented in practice by allowing some external entities (e.g. non-governmental organizations) to cast decoy ballots. Finally we require at least one honest $\mathsf{TA}$ for coercion resistance.

### 5.2 Verifiability

Our scheme satisfies election verifiability as formalized in [13]. This notion incorporates the varieties of individual and universal verifiability. It is achieved mainly through the $\mathsf{NIZK}$ proofs provided by the voters, the registration and tallying authorities and the use of the Benaloh challenge mechanism. Since these proofs are sound, even if these entities are corrupted they cannot force an honest vote to be ignored. Unfortunately, our scheme does not satisfy eligibility verifiability, a related variation (not covered by the definition of [13]). If this property were satisfied, everyone would be able to check that each ballot that was successfully tallied was cast by a voter that had the right to vote [41]. But this would be incompatible with coercion resistance, as the coercer would be able to discover if a *particular* ballot belonging to a known voter was *successfully* tallied or not, or equivalently if the credentials in the ballot were fake or not. In our scheme these properties hold globally, i.e. everyone can verify that *all* tallied ballots were cast by voters with the right to participate, without being able to isolate specific voters' ballots.

To intuitively see why our scheme satisfies individual and universal verifiability we examine how it fares against some related attacks.

*Individual Verifiability* An adversary cannot create a clash attack nor can they invalidate a ballot without the voter finding out assuming that they contribute to the randomness required by the cryptographic functionalities [41]. Firstly, the RA cannot assign the same credential to two distinct *honest* voters as they generate it on their own using LRS.KGen. Also, the credentials cannot be invalidated by a dishonest RA. The soundness of the designated verifier proof ($\delta_{\mathsf{ReEnc}}$) proves that their reencryption was correctly applied. During voting, the Benaloh challenge along with the fact that the voter contributes to the encryption randomness prevent the system from substituting the preferred candidate and from casting duplicate and invalid ballots. We must note here that the coercer cannot use the receipt $(r, \vec{r})$ generated by VS.Vote to find out if their attack succeeded or not. When a coercer is present, the voter will use $(r, \vec{r})$ along with the fake credentials, so the ballot will verify, but will not be counted, since contrary to Helios ([1]) not every ballot in the BB is counted in JCJ-compliant schemes.

*Universal Verifiability* The adversarial goal for universal verifiability is to either alter or drop ballots belonging to honest voters or to add new ballots to the tally (ballot stuffing) apart from the ones that correspond to corrupt voters. The architecture of our scheme does not allow such attacks to succeed.

- Ballot altering: The adversary cannot alter a ballot after vote casting (by changing either the candidate selection or the credential list), since they would have to produce an honest signature for the altered ballot. This is prevented by the unforgeability of the LRS scheme. Additionally, the adversary may try to alter a ballot during the Shuffle functionality, but this is averted by the soundness of the proofs used to verify mixing. Finally, during tallying, the adversary might try to decrypt a ballot to a different candidate, but this would violate the proof of correct decryption.
- Ballot removal: Since the BB cannot delete ballots they can only be removed during the shuffling and tallying phase. As before, the proof of correct shuffle prevents this attack. During tallying, the adversary might wrongly mark a ballot cast with real credentials, as being a product of coercion. This is avoided by the soundness of the PET proof.
- Ballot injection: The adversary will need to associate a credential with an identity. Assuming that the voter roll provided to the RA is trustworthy (a necessary condition for all elections), the attacker must associate the stuffed ballot with an existing identity, which can be detected because the correspondence of credentials to identities is public and our assumption that the RA and the BB are not simultaneously corrupted.

More formally, to prove that our scheme satisfies the notion of verifiability of [13], we note that our scheme satisfies the properties of correctness, accuracy and tally uniqueness. Therefore, it is weakly verifiable (i.e verifiable under the assumption that both the RA and the BB are honest). Furthermore, since our

LRS has the property of unforgeability it also satisfies the notion of strong verifiability of [13], i.e. verifiability when the $\mathsf{BB}$ and the $\mathsf{RA}$ are not simultaneously corrupted. The full proof can be found in Appendix C.1.

### 5.3 Privacy

**Ballot secrecy** We note that our scheme is impervious to some well known attacks in the literature. First of all, we avoid the attacks of [8] by the use of the strong Fiat-Shamir transform. Secondly, ballot weeding via the $\mathsf{VS.IsValid}$ functionality of the $\mathsf{BB}$ and later through the tallying process, provides ballot independence and prevents replay attacks that seek to break privacy [15]. In particular, the checks of during ballot weeding (c.f. section 4) together with the proof $\pi_{\mathsf{Enc}}^{L^{(i)}}$ prevent an adversary from replaying entire ballots or only their individual $\mathtt{m}_i$ components, verbatim or through malleability, to protect privacy [15]. Importantly, this is done without leaking the identity of the voter. We stress that even if the ballots were malleable, their binding with the ring $L^{(i)}$ would force a replay adversary to know a secret credential belonging to the particular $L^{(i)}$ in order to sign the copied ballot.

More formally, our scheme satisfies the BPRIV definition of [7] which essentially states that the cryptographic components of a voting system do not leak information that could help an attacker uncover the vote of an honest voter beyond what is deducible from the election result. In the BRPIV definition game the adversary must distinguish between two bulletin boards $\mathsf{BB}_0, \mathsf{BB}_1$ which are built by the ballots cast by honest and corrupt voters. The former may cast different choices to each of $\mathsf{BB}_0, \mathsf{BB}_1$ (selected by the adversary), while the latter casts the same ballot to both. The tally algorithm always executes on $\mathsf{BB}_0$ while the adversary observes one of $\mathsf{BB}_0, \mathsf{BB}_1$ chosen by the challenger uniformly at random. The adversary must guess which of the two bulletin boards they viewed.

In our scheme, $\mathcal{A}$ succeeds in winning this game only with negligible probability. To argue about this, it suffices to prove that the challenger can successfully swap the honest ballots between $\mathsf{BB}_0, \mathsf{BB}_1$ without the adversary noticing. Since our scheme supports two tallying methods this must be proved for both. The case of homomorphic tallying is easier. If the attacker observes $\mathsf{BB}_1$ the challenger must simulate the proof of correct decryption $\pi_{\mathsf{Dec}}$ only for the aggregate result. On the other hand, if each individual ballot is decrypted, then the challenger must simulate the proof of correct decryption $\pi_{\mathsf{Dec}}$ for each ballot. This is not a problem since in both cases $\pi_{\mathsf{Dec}}$ possesses the (special honest-verifier) zero-knowledge property. As a result the challenger can always decrypt each ballot in $\mathsf{BB}_0$ to the corresponding honest vote in $\mathsf{BB}_1$ and provide a (simulated in the case of different honest votes) proof that the adversary will distinguish with negligible probability.

There is a subtle issue with the framework of [7] in case each ballot is decrypted. Since the adversary dictates the honest choices, in election rules which carry a lot of information, the adversary may dictate a particular and very rare choice for one of the bulletin boards and easily win the game by checking in which of $\mathsf{BB}_0, \mathsf{BB}_1$ it appears after decryption. For instance, if the voters must rank a lot of candidates, then the adversary may require that a particular permutation of unpopular candidates appears on the last places of the ranking so that they can distinguish on which of $\mathsf{BB}_0, \mathsf{BB}_1$ it appears (i.e. the well known Italian attack [12]). This attack applies to all schemes where individual ballots are decrypted and not only to ours and allows the adversary to defeat the definition of ballot secrecy. As this is a general characteristic of the model and not of particular systems, in order to overcome it, we assume that all voter choices dictated by the adversary in $\mathsf{BB}_0, \mathsf{BB}_1$ are equal as multisets. More details are included in Appendix C.2.

**Everlasting privacy** Our scheme provides practical everlasting privacy [2]. Given only the publicly available election data in the $\mathsf{BB}$ a powerful attacker that can break the encryption of the published credentials and the ballots, still cannot map honest ballots to honest voters. This is due to the unconditional anonymity of our $\mathsf{LRS}$ construction. Indeed, assume that $\mathcal{A}$ retrieves all $\{g^{x_i}h^{y_i}\}_{i=1}^{n}$ from to $\{\mathsf{pc}'_i = \mathsf{Enc}(g^{x_i}h^{y_i})\}_{i=1}^{n}$ in $\mathsf{BB}_{\mathsf{SetupElection}}$ and all $\{g^{x_i}h^{y_i}\}_{i=1}^{\beta}$ for each ballot in $\mathsf{BB}_{\mathsf{Vote}}$. Then $\mathcal{A}$ knows which identities comprise the anonymity set selected by $\mathsf{V}_i$. However the unconditional anonymity (c.f. Theorem 2) of our signature does not allow to pinpoint exactly which of these identities actually signed and submitted the ballot. Even if a credential is used in fewer than $\beta$ ballots, the adversary doesn't have a significantly better probability of pinpointing the ballot of this voter, since they cannot be sure whether the voter that owns this credential actually voted or abstained. The expected number of times that each credential is used, given that $N$ valid votes are in the final tally is $\frac{\beta \cdot N}{n}$, since each credential has probability $\frac{\beta}{n}$ to be in $L^{(j)}, j \in \{1, \ldots N\}$.

It must be also noted that the same analysis also applies to the election talliers. Indeed, the $\mathsf{TA}$ might not be computationally powerful but is in possession of the private decryption key $\mathsf{sk}_{\mathsf{TA}}$. As a result, it is conceptually equivalent to the powerful future attacker of everlasting privacy. The $\mathsf{TA}$, thus, may legitimately decrypt the individual ballots in order to compute the result, but they cannot deduce the identity of the voter, but only whether the ballot is valid or not. As a result, the common trust assumption in most voting schemes that the $\mathsf{TA}$ is trusted for privacy does not need to apply for our scheme. If voters combine this unconditionally anonymous 'channel' implemented at the endpoints with countermeasures such as a Tor client or even a public computer to hide networking information (e.g. IP addresses), they erase all traces of insider data that is possible to obtain.

## 5.4 Coercion Resistance

Coercion resistance is a property designed to protect from an active adversary that can perform impersonation, random voting and forced abstention attacks and has receipt-freeness as a prerequisite. Our scheme provides coercion resistance according to the framework of JCJ [31, 11], assuming that the coercer hasn't corrupted a majority of the $\mathsf{RA}$, $\mathsf{TA}$ and that the $\mathsf{DDH}$ problem is hard.

The definition of coercion resistance of [31, 11] is comprised of two games, the real and the ideal. In the real game a coin is flipped to determine whether the voter will provide the real or a fake credential to the adversary. If a fake credential is provided, the voter may cast a ballot using their real credential. The adversary can cast a ballot using the provided credential, as well as ballots using corrupted credentials. Honest voters cast their ballots at an order programmed by the adversary. The goal of the adversary is to guess whether their attack succeeded, namely, whether the vote cast with the coerced credential was counted in the final tally. The ideal game is similar to the real game with the difference that the adversary always gets the real credential, cannot cast ballots using corrupted credentials, doesn't have access to any cryptographic material nor $\mathsf{BB}$ and an idealized tally is produced that includes only the valid votes. The adversary wins if they can distinguish the real from the ideal game with non-negligible probability.

Intuitively, the fake but indistinguishable credentials allow the voter to seemingly obey the instructions of the coercer, but undo them by casting their ballot during their moment of privacy. The anonymization offered by the mixing and our signature prevent the attacker from discovering if their ballot was counted or not. Linking is of no use to the coercer, since the ballots ordered by them and the ones during the moment of privacy are cast with different credentials. Additionally, our scheme avoids tagging attacks caused by allowing each voter to select their own size of anonymity set. Indeed, if each voter could select the number of decoy credentials, then the coercer could tag the ballot by forcing the use of a particular size for the anonymity set (e.g. 1009 [42]) and then with very high probability find out if it was discarded or not. This is the reason for which the anonymity set has globally a constant size $\beta$.

More formally, when voter $j$ is under coercion, the voting device generates a fake credential $\mathsf{sc}_j^* = (x^*, y^*)$. The adversary cannot distinguish whether the credential is real, since the tuple $(g,\quad L_{i1}^{(j)} = g^r,\quad \mathsf{pk},\quad \frac{L_{i2}^{(j)}}{g^{x^*}h^{y^*}} = \frac{g^x h^y}{g^{x^*}h^{y^*}}\mathsf{pk}^r)$ is a $\mathsf{DDH}$ tuple and the coercer hasn't corrupted a majority of the $\mathsf{RA}$ nor of $\mathsf{TA}$, and thus neither the coercer nor the voter know the randomness of the encryption of $\mathsf{pc}_j$. The ballot created by $\mathsf{VS.Vote}$ and then posted on the $\mathsf{BB}$ is indistinguishable from a ballot created with real credentials, since it contains a valid signature, valid proofs and the list $L^{(j)}$ that contains an encryption of $\mathsf{pc}_j^*$. We note that, a signature produced by the voter during the moment of privacy cannot be detected by the coercer, since the signature contained in each ballot has unconditional anonymity, as described in Theorem 2. After $\mathsf{VS.Shuffle}$ the coercer loses track of their ballot and in the $\mathsf{VS.Tally}$ the proofs of the tally do not reveal anything more to the coercer, since the ballots and the credential list where shuffled.

As far as receipt-freeness is concerned, a voter may claim the ownership of a ballot, by disclosing $\mathsf{pc}$ and the randomness used to create it, but cannot prove whether the ballot was counted in the final tally. More details are provided in Appendix C.3.

## 6 Conclusion and future work

In this paper, we proposed a voting scheme that supports coercion resistance and everlasting privacy without sacrificing verifiability. Our work is based on the JCJ framework and does not require trust in the election talliers to provide ballot secrecy. These guarantees are achieved through a new linkable ring signature scheme with unconditional anonymity. Our construction has favorable security properties but suffers from the increased complexity of the tallying phase which is in part inherent in the JCJ architecture and in part for achieving everlasting privacy. In future work, we aim to improve the efficiency of the tallying phase. One mechanism we will explore in this direction, is the batching of $\mathsf{PET}$s during the identification of coerced votes. Instead of checking each credential with the ones in $L^{(0)}$ all credentials in a ballot will be batched and the result will be compared against a batch of $L^{(0)}$ credentials. This will reduce the complexity by a factor of $n$. However, a naive implementation might negatively affect universal verifiability. Additionally, reducing the complexity of the vote casting phase from linear to logarithmic will allow the voters to increase the size of their anonymity set. We also aim to explore the optimal value for $\beta$ by taking into consideration the trade-off between performance and privacy for particular election types. Finally, we plan to improve the usability of the casting phase by integrating a mechanism similar to panic passwords of [11] to allow the voters to easily generate their secret credentials. This must be done in a way that does not affect everlasting privacy.

*Acknowledgements* The authors would like to thank the anonymous reviewers of previous versions of this paper for their comments and suggestions which greatly improved this work.

## References

[1] Ben Adida. "Helios: web-based open-audit voting". In: *Proceedings of the 17th conference on Security symposium.* USENIX, 2008, pp. 335–348.

[2] Myrto Arapinis, Véronique Cortier, Steve Kremer, and Mark Ryan. "Practical everlasting privacy". In: *LNCS.* Vol. 7796 LNCS. 2013, pp. 21–40. DOI: 10.1007/978-3-642-36830-1_2.

[3] Danai Balla, Pourandokht Behrouz, Panagiotis Grontas, Aris Pagourtzis, Marianna Spyrakou, and Giannis Vrettos. "Designated-Verifier Linkable Ring Signatures with Unconditional Anonymity". In: *9th International Conference on Algebraic Informatics, CAI 2022.* Vol. 13706. LNCS. 2022, pp. 55–68. DOI: 10.1007/978-3-031-19685-0_5.

[4] Pourandokht Behrouz, Panagiotis Grontas, Vangelis Konstantakatos, Aris Pagourtzis, and Marianna Spyrakou. "Designated-Verifier Linkable Ring Signatures". In: *24th International Conference on Information Security and Cryptology - ICISC 2021.* Vol. 13218. LNCS. 2022, pp. 51–70. DOI: https://doi.org/10.1007/978-3-031-08896-4_3.

[5] Josh Benaloh. "Simple Verifiable Elections". In: *2006 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT 06)*. Vancouver, B.C.: USENIX Association, Aug. 2006. URL: `https://www.usenix.org/conference/evt-06/simple-verifiable-elections`.

[6] Josh Benaloh and Dwight Tuinstra. "Receipt-free secret-ballot elections (extended abstract)". In: *STOC '94*. ACM, 1994, pp. 544–553. DOI: `10.1145/195058.195407`.

[7] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. "SoK: A Comprehensive Analysis of Game-Based Ballot Privacy Definitions". In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2015, pp. 499–516. DOI: `10.1109/SP.2015.37`.

[8] David Bernhard, Olivier Pereira, and Bogdan Warinschi. "How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios". In: *ASIACRYPT 2012*. Vol. 7658. LNCS. 2012, pp. 626–643. DOI: `10.1007/978-3-642-34961-4_38`.

[9] David Chaum. "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms". In: *Commun. ACM* (1981), pp. 84–88.

[10] David Chaum and Torben P. Pedersen. "Wallet Databases with Observers". In: *CRYPTO '92*. Vol. 740. LNCS. 1992, pp. 89–105. DOI: `10.1007/3-540-48071-4_7`.

[11] Jeremy Clark and Urs Hengartner. "Selections: Internet Voting with Over-the-Shoulder Coercion-Resistance". In: *Financial Cryptography and Data Security, FC 2011*. Vol. 7035. LNCS. 2011, pp. 47–61. DOI: `10.1007/978-3-642-27576-0_4`.

[12] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. "Civitas: Toward a Secure Voting System." In: *IEEE Security and Privacy Symposium*. May 19, 2008.

[13] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. "Election Verifiability for Helios under Weaker Trust Assumptions". In: *ESORICS 2014*. Cham, 2014, pp. 327–344.

[14] Véronique Cortier, David Galindo, Ralf Küsters, Johannes Müller, and Tomasz Truderung. "SoK: Verifiability Notions for E-Voting Protocols". In: *IEEE Symposium on Security and Privacy, SP 2016*. IEEE Computer Society, 2016, pp. 779–798. DOI: `10.1109/SP.2016.52`.

[15] Veronique Cortier and Ben Smyth. "Attacking and Fixing Helios: An Analysis of Ballot Secrecy". In: *24th Computer Security Foundations Symposium*. 2011, pp. 297–311. DOI: `10.1109/CSF.2011.27`.

[16] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. "Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols". In: *CRYPTO '94*. Vol. 839. LNCS. 1994, pp. 174–187. DOI: `10.1007/3-540-48658-5_19`.

[17] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. "A Secure and Optimally Efficient Multi-Authority Election Scheme". In: *EUROCRYPT '97*. 1997, pp. 103–118.

[18] Edouard Cuvelier, Olivier Pereira, and Thomas Peters. “Election Verifiability or Ballot Privacy: Do We Need to Choose?” In: *ESORICS 2013*. 2013, pp. 481–498.

[19] Denise Demirel, J Van De Graaf, and R Araújo. “Improving Helios with Everlasting Privacy Towards the Public”. In: *EVT/WOTE'12* (2012).

[20] Amos Fiat and Adi Shamir. “How to Prove Yourself: Practical Solutions to Identification and Signature Problems”. In: *CRYPTO '86*. Vol. 263. LNCS. 1986, pp. 186–194. DOI: 10.1007/3-540-47721-7_12.

[21] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. “A Practical Secret Voting Scheme for Large Scale Elections”. In: *AUSCRYPT '92*. Vol. 718. LNCS. 1992, pp. 244–251. DOI: 10.1007/3-540-57220-1_66.

[22] Taher El Gamal. “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms”. In: *CRYPTO '84*. Vol. 196. LNCS. 1984, pp. 10–18. DOI: 10.1007/3-540-39568-7_2.

[23] Panagiotis Grontas and Aris Pagourtzis. “Anonymity and everlasting privacy in electronic voting”. In: *Int. J. Inf. Sec.* 22.4 (2023), pp. 819–832. DOI: 10.1007/S10207-023-00666-2.

[24] Panagiotis Grontas, Aris Pagourtzis, and Alexandros Zacharakis. “Security models for everlasting privacy”. In: *E-Vote-ID* (2019), p. 140.

[25] Panagiotis Grontas, Aris Pagourtzis, Alexandros Zacharakis, and Bingsheng Zhang. “Towards Everlasting Privacy and Efficient Coercion Resistance in Remote Electronic Voting”. In: *Financial Cryptography Workshops*. Vol. 10958. LNCS. 2018, pp. 210–231.

[26] Jens Groth. “Non-interactive Zero-Knowledge Arguments for Voting”. In: *ACNS 2005*. LNCS. 2005, pp. 467–482. DOI: 10.1007/11496137_32.

[27] Thomas Haines, Rafieh Mosaheb, Johannes Müller, and Ivan Pryvalov. “SoK: Secure E-Voting with Everlasting Privacy”. In: *Proc. Priv. Enhancing Technol.* 2023.1 (), pp. 279–293. DOI: 10.56553/POPETS-2023-0017.

[28] Martin Hirt and Kazue Sako. “Efficient Receipt-Free Voting Based on Homomorphic Encryption”. In: *EUROCRYPT 2000*. Vol. 1807. LNCS. 2000, pp. 539–556. DOI: 10.1007/3-540-45539-6_38.

[29] Markus Jakobsson and Ari Juels. “Mix and Match: Secure Function Evaluation via Ciphertexts”. In: *ASIACRYPT 2000*. Vol. 1976. LNCS. 2000, pp. 162–177. DOI: 10.1007/3-540-44448-3_13.

[30] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. “Designated Verifier Proofs and Their Applications”. In: *EUROCRYPT '96*. Vol. 1070. LNCS. 1996, pp. 143–154. DOI: 10.1007/3-540-68339-9_13.

[31] Ari Juels, Dario Catalano, and Markus Jakobsson. “Coercion-resistant electronic elections”. In: *WPES 2005*. ACM, 2005, pp. 61–70. DOI: 10.1145/1102199.1102213.

[32] Ralf Kusters, Tomasz Truderung, and Andreas Vogt. “Clash Attacks on the Verifiability of E-Voting Systems”. In: *2012 IEEE Symposium on Security and Privacy*. 2012, pp. 395–409. DOI: 10.1109/SP.2012.32.

[33] Joseph K. Liu, Man Ho Au, Willy Susilo, and Jianying Zhou. "Linkable Ring Signature with Unconditional Anonymity". In: *IEEE Trans. Knowl. Data Eng.* 26.1 (2014), pp. 157–165.

[34] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. "Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups (Extended Abstract)". In: *ACISP 2004*. Vol. 3108. LNCS. 2004, pp. 325–335. DOI: `10.1007/978-3-540-27800-9_28`.

[35] Philipp Locher, Rolf Haenni, and Reto E. Koenig. "Coercion-Resistant Internet Voting with Everlasting Privacy". In: *FC'16 Workshops, BITCOIN, VOTING, WAHC.* 2016. DOI: `10.1007/978-3-662-53357-4_11`.

[36] Eleanor McMurtry, Olivier Pereira, and Vanessa Teague. "When Is a Test Not a Proof?" In: *ESORICS 2020*. Vol. 12309. LNCS. 2020, pp. 23–41. DOI: `10.1007/978-3-030-59013-0_2`.

[37] Tal Moran and Moni Naor. "Receipt-Free Universally-Verifiable Voting with Everlasting Privacy". In: *CRYPTO 2006*. Vol. 4117. LNCS. 2006, pp. 373–392. DOI: `10.1007/11818175_22`.

[38] Tal Moran and Moni Naor. "Split-ballot voting". In: *ACM Transactions on Information and System Security* 13.2 (2010), pp. 1–43. ISSN: 10949224. DOI: `10.1145/1698750.1698756`.

[39] David Pointcheval. "Efficient Universally-Verifiable Electronic Voting with Everlasting Privacy". In: *Security and Cryptography for Networks*. Cham: Springer Nature Switzerland, 2024, pp. 323–344. ISBN: 978-3-031-71070-4. DOI: `10.1007/978-3-031-71070-4_15`.

[40] Claus-Peter Schnorr. "Efficient Identification and Signatures for Smart Cards (Abstract)". In: *EUROCRYPT '89*. Vol. 434. LNCS. 1989, pp. 688–689. DOI: `10.1007/3-540-46885-4_68`.

[41] Ben Smyth, Steven Frink, and Michael R. Clarkson. *Election Verifiability: Cryptographic Definitions and an Analysis of Helios, Helios-C, and JCJ.* Cryptology ePrint Archive, Paper 2015/233. 2015.

[42] Stefan G. Weber, Roberto Araujo, and Johannes Buchmann. "On Coercion-Resistant Electronic Elections with Linear Work." In: *ARES*. IEEE, 2007, pp. 908–916. URL: `http://dblp.uni-trier.de/db/conf/IEEEares/ares2007.html#WeberAB07`.

## A Details of Proofs of Knowledge

Given ciphertexts $c = (c_1, c_2) = \mathsf{Enc}_{\mathsf{pk}}(\mathtt{m}; r) = (g^r, \mathtt{m} \cdot \mathsf{pk}^r)$ and $c' = \mathsf{ReEnc}(c) = (g^{r+r'}, \mathtt{m} \cdot \mathsf{pk}^{r+r'})$ we construct the proofs $\pi_{\mathsf{Enc}}, \pi_{\mathsf{Dec}}, \pi_{\mathsf{ReEnc}}$ as follows:

– If $c$ is a correct encryption of $\mathtt{m}$ then the tuple $(g, \mathsf{pk}, c_1, c_2\mathtt{m}^{-1}) = (g, \mathsf{pk}, g^r, \mathsf{pk}^r)$ is a valid Diffie - Hellman tuple. This can be proved using $\pi_{CP}$ of [10]. $\pi_{\mathsf{Enc}}$ can then be constructed as an OR-proof where the various messages are the encodings of the candidates in $\mathsf{CS}$.
– If $c$ correctly decrypts to $\mathtt{m}$ using $\mathsf{sk}$ then $(g, c_1, \mathsf{pk}, c_2\mathtt{m}^{-1}) = (g, g^r, g^{\mathsf{sk}}, c_1^{\mathsf{sk}})$ is a valid Diffie - Hellman tuple which can be also proved using $\pi_{CP}$ of [10].
– If $c'$ is a reencryption of $c$ then $c' \odot c^{-1} = (g^{r'}, \mathsf{pk}^{r'})$ which can be proved using $\pi_{CP}$ of [10].

The proof of knowledge of secret credential corresponding to public credential ($\pi_{\mathsf{sc}}$) and the designated verifier proof of correct reencryption ($\delta_{\mathsf{ReEnc}}$) can be found in Figure 3. A designated verifier with knowledge of $\mathsf{sk}_V$ such that $\mathsf{pk}_{\mathsf{V}} = g^{\mathsf{sk}_{\mathsf{V}}}$ can simulate the proof $\delta_{\mathsf{ReEnc}} = (e, s, t_2, t_3)$ by selecting $s', a, b \leftarrow\$ \mathbb{Z}_q$ and computing: $T_1 = g^{s'}(c_1'/c_1)^{-a}, T_2 = \mathsf{pk}^{s'}(c_2'/c_2)^{-a}, T_3 = g^b$ and $t_2 = a - e, t_3 = (b - t_2)\mathsf{sk}_{\mathsf{V}}^{-1}$ which verifies correctly.

## B Security properties of our $\mathsf{LRS}$ scheme

### B.1 Proof of unforgeability for our LRS (Theorem 1)

*Proof.* Assume a $\mathsf{PPT}$ adversary $\mathcal{A}$ that with non-negligible probability can forge a signature $\sigma$ that passes the verification without knowledge of any of the secret credentials. We will construct an algorithm $\mathcal{B}$ that given $n$ $\mathsf{DLOG}$ instances $\{X_i\}_{i=1}^n$ and by using $\mathcal{A}$ outputs the discrete logarithm of at least one of them with non-negligible probability.

The input of $\mathcal{B}$ is $\mathbb{G}, g, q, \{X_i\}_{i=1}^n$ and $\mathcal{B}$ simulates the environment for $\mathcal{A}$. $\mathcal{A}$ may query the random oracle $\mathcal{RO}$, to receive a random element of $\mathbb{Z}_q$ or of $\mathbb{G}$, the joining oracle $\mathcal{JO}$, to add users and their public credentials to the system, the corruption oracle $\mathcal{CO}$, where $\mathcal{A}$ may ask for the secret credential that corresponds to a public credential and lastly the signing oracle $\mathcal{SO}$, where $\mathcal{A}$ receives a signature on behalf of a specific signer. $\mathcal{B}$ generates the system parameters and simulates the oracles that $\mathcal{A}$ has access to. Upon query of the $\mathcal{RO}$, $\mathcal{B}$ returns a random value and replies consistently to all the queries. Upon query of the $\mathcal{JO}$, $\mathcal{B}$ adds public credentials to the system, either by adding $\mathsf{pc}_i \leftarrow (g^{r_i}, X_i h^{y_i}\mathsf{pk}^{r_i})$ using the given $\mathsf{DLOG}$ challenges and $y_i, r_i \leftarrow\$ \mathbb{Z}_q$ or by adding $\mathsf{pc}_i \leftarrow (g^{r_i}, g^{x_i} h^{y_i}\mathsf{pk}^{r_i})$, using random values $x_i, y_i, r_i \leftarrow\$ \mathbb{Z}_q$. Furthermore, $\mathcal{A}$ may query the $\mathcal{CO}$ for a public credential, where $\mathcal{B}$ replies with the corresponding private credential and lastly $\mathcal{A}$ may query the $\mathcal{SO}$, where $\mathcal{B}$ programs the random oracle and replies with a signature created on behalf of the queried signer on the queried message.

$\pi_{\mathsf{sc}} = \mathsf{NIZK}\big\{(\mathbb{G}, q, g, h, \mathsf{pk}, c), (x, y, r) : c = \mathsf{Enc}_{\mathsf{pk}}(g^x h^y; r)\big\}$

| **Prover** | | **Verifier** |
|---|---|---|
| $x', y', r' \leftarrow\!\!\$\ \mathbb{Z}_q$ | | |
| $c' \leftarrow \mathsf{Enc}_{\mathsf{pk}}(g^{x'} h^{y'}; r')$ | | |
| | $\xrightarrow{c'}$ | |
| | | $e \leftarrow\!\!\$\ \mathbb{Z}_q$ |
| | $\xleftarrow{e}$ | |
| $x'' \leftarrow x' + ex$ | | |
| $y'' \leftarrow y' + ey$ | | |
| $r'' \leftarrow r' + er$ | | |
| | $\xrightarrow{x'', y'', r''}$ | |
| | | Accept if and only if: |
| | | $\mathsf{Enc}_{\mathsf{pk}}(g^{x''} h^{y''}; r'') = c' \odot c^e$ |

$\delta_{\mathsf{ReEnc}} = \mathsf{NIZKpk}_{\mathsf{V}}\big\{(\mathbb{G}, q, c, c', \mathsf{pk}_{\mathsf{V}}, \mathsf{pk}), (r) : c' = (c_1', c_2') = (g^r c_1, \mathsf{pk}^r c_2) = \mathsf{ReEnc}(c)\big\}$

| **Prover** $(r)$ | | **Verifier** $(\mathsf{sk}_{\mathsf{V}})$ |
|---|---|---|
| $t_1, t_2, t_3 \leftarrow\!\!\$\ \mathbb{Z}_q$ | | |
| $T_1 \leftarrow g^{t_1}, T_2 \leftarrow \mathsf{pk}^{t_1}, T_3 \leftarrow g^{t_2} \mathsf{pk}_{\mathsf{V}}^{t_3}$ | | |
| | $\xrightarrow{T_1, T_2, T_3}$ | |
| | | $e \leftarrow\!\!\$\ \mathbb{Z}_q$ |
| | $\xleftarrow{e}$ | |
| $s \leftarrow t_1 + r(e + t_2)$ | | |
| | $\xrightarrow{s, t_2, t_3}$ | |
| | | Accept if and only if: |
| | | $T_3 = g^{t_2} \mathsf{pk}_{\mathsf{V}}^{t_3}$ |
| | | $g^s = (c_1'/c_1)^{e+t_2} T_1$ |
| | | $\mathsf{pk}^s = (c_2'/c_2)^{e+t_2} T_2$ |

**Fig. 3.** Proofs $\pi_{\mathsf{sc}}$ and $\delta_{\mathsf{ReEnc}}$

Let $\sigma_0^*$ be a forgery that $\mathcal{A}$ produced on the list of public credentials $\{(g^{r_i}, X_i h^{y_i} \mathsf{pk}^{r_i})\}_{i=1}^n$. We can assume that $\mathcal{A}$ queried all $n$ queries used in the $\mathsf{Vrfy}$ algorithm. $\mathcal{B}$ rewinds $\mathcal{A}$, replies consistently to all hash queries but at the hash query where they replied $c_{j0}$ they reply $c_{j1} \neq c_{j0}$. From the rewind-on-success lemma [34] $\mathcal{A}$ will produce another forgery $\sigma_1^*$ with non-negligible probability, for which $c_{i0} = c_{i1}$, $\forall i \in \{1, \dots n\} \setminus \{j\}$. Therefore for the two forgeries, since they both verify correctly, it holds that:

$$K_{j0} = K_{j1} \Rightarrow g^{t_{j0}} \cdot h^{p_{j0}} \cdot \mathsf{pk}^{s_{j0}} \cdot (L_{j2})^{c_{j0}} = g^{t_{j1}} \cdot h^{p_{j1}} \cdot \mathsf{pk}^{s_{j1}} \cdot (L_{j2})^{c_{j1}} \tag{1}$$

There exist $x'_j$, $y'_j$, $r'_j \in \mathbb{Z}_q$ s.t. $L_{j2} = X_j h^{y'_j} \mathsf{pk}^{r'_j} = g^{x'_j} \cdot h^{y'_j} \cdot \mathsf{pk}^{r'_j}$. Then by Eq. 1:

$$g^{t_{j0}} \cdot h^{p_{j0}} \cdot \mathsf{pk}^{s_{j0}} (g^{x'_j} \cdot h^{y'_j} \cdot \mathsf{pk}^{r'_j})^{c_{j0}} = g^{t_{j1}} \cdot h^{p_{j1}} \cdot \mathsf{pk}^{s_{j1}} (g^{x'_j} \cdot h^{y'_j} \cdot \mathsf{pk}^{r'_j})^{c_{j1}}$$

$$g^{t_{j0} + c_{j1} x'_j} \cdot h^{p_{j0} + c_{j1} y'_j} \cdot \mathsf{pk}^{s_{j0} + c_{j1} r'_j} = g^{t_{j1} + c_{j1} x'_j} \cdot h^{p_{j0} + c_{j1} y'_j} \cdot \mathsf{pk}^{s_{j1} + c_{j1} r'_j}$$

$$x'_j = \frac{t_{j0} - t_{j1}}{c_{j1} - c_{j0}}, \quad y'_j = \frac{p_{j0} - p_{j1}}{c_{j1} - c_{j0}}, \quad r'_j = \frac{s_{j0} - s_{j1}}{c_{j1} - c_{j0}} \tag{2}$$

Thus $\mathcal{B}$ solved the $\mathsf{DLOG}$ problem for one of the given challenges with non-negligible probability, since $x'_j$ is the discrete logarithm of $X_j$.

### B.2 Proof of unconditional anonymity for our LRS construction (Theorem 2)

*Proof.* Assume a computationally unbounded adversary $\mathcal{A}$ and let a signature $\sigma = (c_1, \{s_i\}_{i=1}^n, \{t_i\}_{i=1}^n, \{p_i\}_{i=1}^n, \mathtt{t})$ be given as challenge to $\mathcal{A}$ to determine the signer. $\mathcal{A}$ has access to the $\mathcal{RO}, \mathcal{JO}, \mathcal{CO}, \mathcal{SO}$, as described in Appendix B.1. $\mathcal{B}$ simulates the oracles for $\mathcal{A}$ as in Appendix B.1, with the difference that upon query to the joining oracle $\mathcal{B}$ adds the public credential $\mathsf{pc}_i \leftarrow (g^{r_i}, g^{x_i} h^{y_i} \mathsf{pk}^{r_i})$ to the system, where $x_i, y_i, r_i \leftarrow\!\!\$\, \mathbb{Z}_q$.

Since $\mathcal{A}$ is computationally unbounded, they can compute $\{Z_i = g^{x_i} h^{y_i}\}_{i=1}^n$ and $\{r_i\}_{i=1}^n$ by decrypting $L = \big(\mathsf{Enc}_{\mathsf{pk}}(g^{x_1} h^{y_1}; r_1), \dots, \mathsf{Enc}_{\mathsf{pk}}(g^{x_n} h^{y_n}; r_n)\big)$. $\mathcal{A}$ can also compute $x \in \mathbb{Z}_q : \mathtt{t} = e^x$ as well as $\{w_i \in \mathbb{Z}_q : Z_i = g^x h^{w_i}\}_{i=1}^n$. Assume that $\mathcal{A}$ obtained $m_1$ secret credentials from the $\mathcal{CO}$ and $m_2$ signatures from the $\mathcal{SO}$. Then for the rest $n - m_1 - m_2$ public credentials that $\mathcal{A}$ doesn't know the secret credential they can compute for each $i \in [n - m_1 - m_2]$:

$$\alpha_i = s_i + c_i r_i, \quad \beta_i = t_i + c_i x, \quad \gamma_i = p_i + c_i w_i$$

The sign algorithm $\mathsf{LRS.Sign}(L, \mathsf{ev}, (x, w_i, r_i), m)$ initialized with $\alpha_i, \beta_i, \gamma_i$ instead of $\alpha, \beta, \gamma$ produces exactly the same signature $\sigma$ for each signer $i \in [n - m_1 - m_2]$ and hence $\mathcal{A}$ cannot distinguish which of the $n - m_1 - m_2$ signers is the signer that produced $\sigma$.

### B.3 Proof of linkability for our LRS (Theorem 3)

*Proof.* Assume a $\mathsf{PPT}$ adversary $\mathcal{A}$ that owns $k - 1$ secret credentials and with non-negligible probability can produce $k$ pairwise unlinkable signatures. We will construct an algorithm $\mathcal{B}$ that given $n$ $\mathsf{DLOG}$ instances $\{X_i\}_{i=1}^n$ outputs the discrete logarithm of at least one of them with non-negligible probability.

$\mathcal{B}$ simulates the environment and the oracles for $\mathcal{A}$ as described in the proof of unforgeability in Appendix B.1. Let $\{\sigma_i^*\}_{i=1}^k$ be the set of $k$ pairwise unlinkable signatures that $\mathcal{A}$ produced. We assume that $\mathcal{A}$ queried all queries used in the $\mathsf{Vrfy}$ algorithm.

*Case 1:* $\mathcal{A}$ produced at least 2 signatures $\sigma_a$ and $\sigma_b$ that had as last queries $l_a$ and $l_b$ respectively for the same index $j$. Then $\mathcal{B}$ rewinds $\mathcal{A}$ twice, on the $l_a$ and $l_b$ query respectively, replies consistently to all hash queries but at $l_a$ replies $c_{ja1}$ instead of $c_{ja0}$ and at $l_b$ replies $c_{jb1}$ instead of $c_{jb0}$. By the rewind on success lemma [34], $\mathcal{A}$ will produce a forgery $\sigma_{a1}^*$ on the first rewind and $\sigma_{b1}^*$ on the second rewind with non-negligible probability, for which similarly with the unforgeability proof it holds that:

$$K''_{ja0} = K''_{ja1} \Rightarrow e^{t_{ja0}} \cdot \mathtt{t}_a^{c_{ja0}} = e^{t_{ja1}} \cdot \mathtt{t}_a^{c_{ja1}} \overset{\mathtt{t}_a = e^{x'_a}}{\Rightarrow} x'_a = \frac{t_{ja0} - t_{ja1}}{c_{ja1} - c_{ja0}} \tag{3}$$

$$K''_{jb0} = K''_{jb1} \Rightarrow e^{t_{jb0}} \cdot \mathtt{t}_b^{c_{jb0}} = e^{t_{jb1}} \cdot \mathtt{t}_b^{c_{jb1}} \overset{\mathtt{t}_b = e^{x'_b}}{\Rightarrow} x'_b = \frac{t_{jb0} - t_{jb1}}{c_{jb1} - c_{jb0}} \tag{4}$$

By eqs. (2) and (3) we have that $x'_a = x'_j$ and by eqs. (2) and (4) we have that $x_b = x'_j$. Therefore $\mathsf{LRS.Link}(\sigma_a^*) = \mathsf{LRS.Link}(\sigma_b^*) = e^{x'_j}$, hence $\sigma_a$ and $\sigma_b$ are not unlinkable.

*Case 2:* All signatures produced by $\mathcal{A}$ had distinct index $j$ as the last query. $\mathcal{B}$ does $k$ rewind simulations on each of the last queries of each signature, similarly with the unforgeability proof. Since $\mathcal{A}$ knows only $k-1$ secret credentials, $\mathcal{B}$ solves the $\mathsf{DLOG}$ problem for at least one of the challenges $\{X_i\}_{i=1}^n$ with non-negligible probability.

## C Security properties of our voting scheme

### C.1 Verifiability

Firstly we will show that our scheme satisfies the notion of weak verifiability of [13]. According to [13, Theorem 4.1] it suffices to show that our protocol satisfies correctness, accuracy and tally uniqueness. Correctness is self-evident from our scheme specification in Figure 2.

Accuracy intuitively means that every ballot that is deemed valid corresponds to a correct vote and that the proof produced by the tally function will successfully pass $\mathsf{VS.Vrfy}$.

**Lemma 1 (Accuracy).** *Assuming that the* $\mathsf{DLOG}$ *problem is hard in* $\mathbb{G}$ *and that the soundness error of the* $\mathsf{NIZK}$ *scheme is negligible, our scheme provides accuracy.*

*Proof.* To prove the former claim consider a ballot $(\mathtt{m}, L^{(i)}, \sigma)$ where $\mathtt{m}$ contains $\mathsf{Enc}(\mathsf{cnd}; r)|\pi_{\mathsf{Enc}}^{L^{(i)}}$. Assuming that $\mathsf{VS.IsValid}(b, \mathsf{BB}) = 1$ we deduce that the ballot is syntactically correct, that it is unique in the $\mathsf{BB}$ and that the ring $L^{(i)}$ is correctly constructed. We also deduce that:

– $\mathsf{NIZK.Vrfy}(\pi^{L^{(i)}}_{\mathsf{Enc}}) = 1$. This means that $\mathsf{Enc}(\mathsf{cnd}; r)$ encrypts a real candidate $\mathsf{cnd} \in \mathsf{CS}$ with soundness error $\frac{1}{q}$.
– $\mathsf{LRS.Vrfy}(L^{(i)}, \mathtt{ev}, \mathtt{m}, \sigma) = 1$ which implies that a credential corresponding to $\mathsf{sc}^*_i$ has been included in the ring $L^{(i)}$ unless the signature has been forged. This credential might be real or fake. However the voter doing the verification, maybe using their own device, is aware if they have included in $L^{(i)}$ the public counterpart of the real or the fake credential.

Furthermore, if $\mathsf{VS.Vrfy}(\{b\}, \mathsf{Tally}(\mathsf{sk}_{\mathsf{TA}}, \{b\})) = 1$ then:

– $\mathsf{NIZK.Vrfy}(\pi_{\mathsf{Dec}}) = 1$. This means that $\mathsf{Enc}(\mathsf{cnd}; r)$ has been correctly decrypted with soundness error $\frac{1}{q}$.
– $\mathsf{NIZK.Vrfy}(\pi_{\mathsf{PET}}) = 1$. This means that the $\mathsf{PET}$ has been executed correctly on ballot $b$.

Combining the conclusions above, we can reach the conclusion that a ballot which passes validation and yields a tally that passes verification will lead to a vote that will be counted if the real credential has been used. □

**Lemma 2 (Tally uniqueness).** *Assuming that* $\mathsf{Enc}$ *is correct and the* $\mathsf{NIZK}$ *schemes* $\pi_{\mathsf{Dec}}, \pi_{\mathsf{PET}}$ *are sound, our scheme provides tally uniqueness.*

*Proof.* Assume that an adversary has managed to create a $\mathsf{BB}$ and $(T_1, \pi_1), (T_2, \pi_2)$ such that $T_1 \neq T_2$ and $\mathsf{Vrfy}(\mathsf{BB}, T_1, \pi_1) = \mathsf{Vrfy}(\mathsf{BB}, T_2, \pi_2) = 1$. This implies that there exists at least one ballot $b \in \mathsf{BB}$ for which $\mathsf{Enc}(\mathsf{cnd}; r)$ can be decrypted both as $\mathsf{cnd}_1, \mathsf{cnd}_2 \in \mathsf{CS}$ with $\mathsf{cnd}_1 \neq \mathsf{cnd}_2$ for $\pi_{\mathsf{Dec}}$ (which is the same in both cases as there is a single bulleting board) correctly verifies. But this goes against the correctness properties of the encryption and proof schemes. Alternatively there may be two ballots in the $\mathsf{BB}$ $b_1, b_2$ that encrypt different $\mathsf{cnd}_1, \mathsf{cnd}_2 \in \mathsf{CS}$ with $\mathsf{cnd}_1 \neq \mathsf{cnd}_2$ and the adversary can obtain $T_1$ by counting $\mathsf{cnd}_1$ and $T_2$ by counting $\mathsf{cnd}_2$ due to the validity of the credentials. This violates the soundness of the proof $\pi_{\mathsf{PET}}$. □

**Theorem 5 (Weak verifiability).** *Our protocol is weakly verifiable assuming that the* $\mathsf{DLOG}$ *problem is hard in* $\mathbb{G}$*, that the soundness error of the* $\mathsf{NIZK}$ *schemes is negligible and that* $\mathsf{Enc}$ *is correct*

*Proof.* Implied by Lemma 1 and Lemma 2 according to [13, Theorem 4.1]. □

**Theorem 6 (Strong verifiability).** *Our protocol is strongly verifiable.*

*Proof.* In Theorem 1 we proved that our LRS construction is unforgeable. Therefore, according to [13, Theorem 4.3] since our scheme is weakly verifiable (Theorem 5) it also satisfies the property of strong verifiability. □

## C.2 Privacy

To argue about ballot secrecy we adapt the BPRIV model [7] to our scheme.

**Theorem 7 (Ballot secrecy).** *Our scheme provides ballot secrecy according to BPRIV [7], assuming the soundness of the* $\mathsf{NIZK}$ *proofs and that the ElGamal encryption scheme together with the proof* $\pi_{\mathsf{Enc}}^{L^{(i)}}$ *satisfy NM-CPA security.*

*Proof.* We define a sequence of games that transition the view of the privacy adversary $\mathcal{A}$ from an election where a bulletin' board $\mathsf{BB}_0$ is both observed and tallied, to an election where $\mathsf{BB}_1$ is observed by $\mathcal{A}$ and $\mathsf{BB}_0$ is tallied.

The adversary may corrupt some voters and cast ballots on their behalf. This will be represented using an oracle $\mathcal{OC}(i, b)$ which casts the same ballot to both $\mathsf{BB}_0, \mathsf{BB}_1$. As a result, they do not need to be swapped and the challenger does not deal with them any further.

Regarding honest votes, the adversary selects their choices by using an oracle $\mathcal{OV}$. A call $\mathcal{OV}(i, \mathsf{cnd}_0, \mathsf{cnd}_1)$ dictates that an honest voter casts a ballot for $\mathsf{cnd}_0$ in $\mathsf{BB}_0$ and $\mathsf{cnd}_1$ in $\mathsf{BB}_1$. We assume that $\mathcal{OV}$ always uses a valid credential, as voting with a fake one would not help the privacy attacker.

In what follows, the challenger maintains a table containing tuples created by the calls to $\mathcal{OV}$ containing the values $(i, \mathsf{cnd}_0, b_0, \mathsf{cnd}_1, b_1)$. As explained in section 5.3, we assume that the set of choices $\{\mathsf{cnd}_0\}$ selected in $\mathsf{BB}_0$ is equal as a multiset to the set of choices $\{\mathsf{cnd}_1\}$ selected in $\mathsf{BB}_1$.

Assume that there are $m$ honest ballots cast in each of $\mathsf{BB}_0, \mathsf{BB}_1$. $\mathsf{Game}_0$ is the BPRIV game where $\mathsf{BB}_0$ is both observed and tallied. In $\{\mathsf{Game}_i\}_{i=1}^{m}$ the challenger swaps the $i$-th ballot of $\mathsf{BB}_0$ with the $i$-th ballot of $\mathsf{BB}_1$. As in the proof of section D.1 of [7], given an adversary that can distinguish between $\mathsf{Game}_i$ and $\mathsf{Game}_{i-1}$ with non-negligible probability, we can construct an adversary that can break the NM-CPA property of an ElGamal ciphertext $c$ accompanied by the proof $\pi_{\mathsf{Enc}}$ with the same non-negligible probability. It is easy to see that $\mathsf{Game}_m$ is the game where the adversary observes $\mathsf{BB}_1$. The challenger must now tally $\mathsf{BB}_0$ without the adversary knowing. The functionalities $\mathsf{Shuffle}, \mathsf{PET}$ do not aid $\mathcal{A}$ any further, since they all take as input any bulletin board and operate on the ballot ciphertexts. Furthermore, $\mathsf{PET}$ returns 1 for all honest ballots in both cases, since $\mathcal{OV}$ utilizes the correct credentials.

The only way the attacker could distinguish the bulletin boards is from the result, since they can calculate what the result is in $\mathsf{BB}_0$ from all the votes $\{\mathsf{cnd}_0\}$ they selected, and the corresponding result in $\mathsf{BB}_1$ from $\{\mathsf{cnd}_1\}$. The adversary expects to see the result from $\{\mathsf{cnd}_0\}$, but in $\mathsf{Game}_m$ the tally algorithm is applied to $\{\mathsf{cnd}_1\}$. To fool the adversary the challenger utilizes the existence of the simulator implied by the zero-knowledge property of $\pi_{\mathsf{Dec}}$. As mentioned in Appendix A this proof proves that $\mathsf{Dec}(c_1, c_2) = \mathtt{m}$ by showing $(g, c_1, \mathsf{pk}, c_2\mathtt{m}^{-1})$ is a valid Diffie - Hellman tuple using the Chaum-Pedersen protocol [10]. This proof can be simulated to show this for any value $v \in \mathbb{G}$ instead of $\mathtt{m}$.

As a result, in the case that all votes are homomorphically decrypted the challenger announces the tally $T_0$ of $\mathsf{BB}_0$ and creates a proof for $(g, c_1, \mathsf{pk}, c_2 T_0^{-1})$ where $c_1, c_2$ are the 'aggregated ballots' in $\mathsf{BB}_1$. In the case that each ballot is individually decrypted and then tallied in plaintext form, the challenger creates proofs $\left\{(g, c_{1i}, \mathsf{pk}, c_{2i}{\mathsf{cnd}_{0i}}^{-1})\right\}_{i=1}^{m}$ where $\{c_{1i}, c_{2i}\}$ comprise the individual ballots in $\mathsf{BB}_1$. Note that since $\{\mathsf{cnd}_0\}$ and $\{\mathsf{cnd}_1\}$ are equal as multiset the challenger may always find a $\mathsf{cnd}_0$ for all $\mathsf{cnd}_1$ selected by the adversary in their $\mathcal{OV}$ calls.

To conclude the proof, we also argue that our scheme satisfies the properties of strong consistency and strong correctness of [7].

Regarding strong consistency, since our ballots do not contain voter identities as in the version of Helios studied in [7] we define an $\mathsf{Extract}$ algorithm that is not required to output a voter identity. Instead, our $\mathsf{Extract}$ algorithm outputs all credentials in $L^{(i)}$. In more details, it accepts as input the tallier secret key $\mathsf{sk_{TA}}$ and parses a ballot $b$ as $(\mathsf{Enc}(\mathsf{cnd}; r) | \pi_{\mathsf{Enc}}^{L^{(i)}}, L^{(i)}, \sigma)$ and decrypts $\mathsf{Enc_{pk}}(\mathsf{cnd}; r), L^{(i)} = \{\mathsf{Enc_{pk}}(g^{x_i}h^{y_i})\}_{i=1}^{\beta}$ to obtain $\mathsf{cnd}, \{g^{x_i}h^{y_i}\}_{i=1}^{\beta}$. We also define an independent valid function $\mathsf{IndValid}$ that verifies the signature and the proof $\pi_{\mathsf{Enc}}^{L^{(i)}}$. It is easy to see that the first and second conditions of strong consistency are satisfied by construction. Furthermore, an adversarial ballot box will yield the same result both when the $\mathsf{Tally}$ algorithm is applied and when the $\mathsf{Extract}$ algorithm is applied on the ballots and the values $\mathsf{cnd}, \{g^{x_i}h^{y_i}\}$ are counted according to the voting rules. The reason for this is that in the latter case the credentials $g^{x_i}h^{y_i}$ will be compared in plaintext form and not through the $\mathsf{PET}$ but the output will be the same. Consequently, in both cases the same ballots will be counted.

Regarding strong correctness, it is easy to see that an honest ballot is going to be accepted even for an adversarially created bulletin board identically to [7]. According to our scheme's specification in Figure 2 an honestly generated ballot will only be discarded if the adversary has managed to output an identical one before. But this has negligible probability to occur since ElGamal and all other primitives that are used in the construction of our ballot are probabilistic. $\square$

## C.3 Coercion Resistance

In order to prove coercion resistance we adopt the model of [31, 11], where a non-adaptive adversary is assumed. Coercion resistance is defined with two games, the real and the ideal. The real game aims to model the behavior of the adversary, the honest voters and the coerced voter in a real coercion scenario, where the adversary tries to decide whether the attack succeeded or not. The goal is the same in the ideal game, except that $\mathcal{A}$ doesn't have access to any cryptographic material or the $\mathsf{BB}$, therefore they have no advantage in distinguishing the success or failure of the attack.

**Theorem 8 (Coercion resistance).** *If the DDH assumption holds in $\mathbb{G}$, then our scheme is coercion resistant according to the model of [31].*

*Proof.* In the real game, the challenger sets up the election and the adversary $\mathcal{A}$ chooses the set of voters they wish to corrupt. Then the challenger registers all the voters and yields to $\mathcal{A}$ the secret credentials of the corrupted voters. Next, $\mathcal{A}$ chooses a voter $j$ to coerce. A random coin $\mathfrak{b}$ is flipped, to model the behavior of the coerced voter. If $\mathfrak{b} = 0$ the voter provides a fake credential to the adversary, whereas if $\mathfrak{b} = 1$ the voter provides the real credential. Afterwards, the adversary programs the voting of the corrupted and the honest voters in whatever order it suits them. The honest voters vote according to a distribution $\mathcal{D}$ which aims to model uncertainty in their behavior and the inability of $\mathcal{A}$ to predict it. For instance, some voters may abstain from voting, or cast an invalid vote. If the attacker could fully predict the behavior of the honest voters then they would trivially win the game by checking the tally. If $\mathfrak{b} = 0$ the coercer casts a ballot with the fake credential, while in the moment of privacy the voter casts their ballot with their real credential. If $\mathfrak{b} = 1$ the voter hands control to the coercer who casts a ballot with the real credential. After the tally is counted, the adversary tries to guess $\mathfrak{b}$ to see if their coercion attempt succeeded.

The ideal game is similar to the real one, with the difference that the coercer always gets the real credential, does not have access to cryptographic material and the $\mathsf{BB}$, and an idealized tally is posted, containing only the valid votes maintained by the challenger in the reduction.

We note that in both the real and ideal game there is one more vote in the final tally if $\mathfrak{b} = 0$. This information cannot be used by $\mathcal{A}$ to distinguish whether $\mathfrak{b} = 0$ or $\mathfrak{b} = 1$, since the honest voters behave according to the distribution $\mathcal{D}$ and thus $\mathcal{A}$ cannot predict the total number of votes in the tally.

We can prove coercion resistance by a sequence of games, from the real to the ideal, where the advantage of the adversary to distinguish which game they are playing is negligible. The initial game $\mathsf{Game}_0$ is the real coercion resistance game. In $\mathsf{Game}_0$ if $\mathfrak{b} = 0$ the challenger casts a ballot using the real credential $\mathsf{sc}_j = (x, y)$, and constructs a fake credential, by choosing a random $\mathsf{sc}_j^* = (x^*, y^*) \leftarrow\!\!\$\ \mathbb{Z}_q^2$ and giving it to the coercer. If $\mathfrak{b} = 1$ the challenger hands the real credential $\mathsf{sc}_j = (x, y)$ to the coercer. Then the honest and the corrupted voters vote, as well as $\mathcal{A}$ using the credential (real or fake) of the coerced voter, according to the order $\mathcal{A}$ instructed. The tally is produced and the adversary must guess whether $\mathfrak{b} = 0$ or $\mathfrak{b} = 1$.

The intuition behind $\mathsf{Game}_1$ is that the real vote of the coerced voter doesn't give advantage to the coercer. $\mathsf{Game}_1$ is similar to $\mathsf{Game}_0$ with the difference that if $\mathfrak{b} = 0$ the challenger casts a ballot using a random value $(z_1, z_2) \leftarrow\!\!\$\ \mathbb{Z}_q^2$ as a credential. We note that $\mathcal{A}$ cannot distinguish whether he is playing $\mathsf{Game}_0$ or $\mathsf{Game}_1$, since if $\mathfrak{b} = 1$ the games are identical and if $\mathfrak{b} = 0$ then the only difference is the ballot $b_j = (\mathtt{m}, L^{(j)}, \sigma)$ of $\mathsf{Game}_0$ that is replaced by the ballot $\hat{b_j} = (\hat{\mathtt{m}}, \hat{L}^{(j)}, \hat{\sigma})$ in $\mathsf{Game}_1$. In this case, assuming that the credential of voter $j$ is in the $i$-th position of the list $L^{(j)}$ and in the $i'$-th position of $L^{(0)}$, the values

$$\left(g, \quad \frac{L_{i1}^{(j)}}{L_{i'1}^{(0)}} = g^{r_{ji} - r_{0i'}}, \quad \mathsf{pk}, \quad \frac{L_{i2}^{(j)}}{L_{i'2}^{(0)}} = \frac{g^x h^y \mathsf{pk}^{r_{ji}}}{g^x h^y \mathsf{pk}^{r_{0i'}}} = \mathsf{pk}^{r_{ji} - r_{0i'}}\right)$$

in $\mathsf{Game}_0$ form a Diffie-Hellman tuple, since the real credential was used in the $\mathsf{VS.Vote}$ algorithm, but in $\mathsf{Game}_1$ the last element of this tuple is $\frac{\hat{L}_{i2}^{(j)}}{L^{(0)}{}_{i'2}} =$

$\frac{g^{z_1} h^{z_2} \mathsf{pk}_1^r}{g^x h^y \mathsf{pk}^{r_{0i'}}}$, which is a random element of $\mathbb{G}$, since a random value was used as credential in the vote algorithm. Thus the adversary cannot distinguish them, assuming that the $\mathsf{DDH}$ problem is hard.

$\mathsf{Game}_2$ aims to model that the adversary cannot distinguish whether the coerced voter supplied the correct or a fake credential. It is similar to the $\mathsf{Game}_1$ with the difference that if $\mathfrak{b} = 0$ the challenger provides the real credential $\mathsf{sc}_j$ to the coercer, instead of a random value. We note that $\mathcal{A}$ cannot distinguish whether he is playing $\mathsf{Game}_1$ or $\mathsf{Game}_2$, since if $\mathfrak{b} = 1$ the games are identical and if $\mathfrak{b} = 0$ the adversary has to decide whether a tuple is a $\mathsf{DDH}$ tuple, similarly to the previous case. In this case, by using the credential provided to the adversary, the values

$$\left(g, \quad L_{i1}^{(j)} = g^r, \quad \mathsf{pk}, \quad \frac{L_{i2}^{(j)}}{g^x h^y} = \mathsf{pk}^r\right)$$

in $\mathsf{Game}_2$ form a Diffie-Hellman tuple, since the real credential was provided, but in $\mathsf{Game}_1$ the last element of this tuple is $\frac{L_{j2}}{g^{x^*} h^{y^*}}$, which is a random element of $\mathbb{G}$, since a fake credential was provided. Thus the adversary cannot distinguish them, assuming that the $\mathsf{DDH}$ problem is hard.

$\mathsf{Game}_3$ aims to model that the adversary cannot obtain any information by the actions of the honest voters. It is similar to $\mathsf{Game}_2$ with the difference that the challenger casts a vote for *each* honest voter using random credentials instead of their real credentials. The challenger produces a modified tally in $\mathsf{Game}_3$, containing only the result of the elections, without any proofs. In the final result the votes of coerced voters are counted and the votes of the honest voters are counted only if they were intended to, despite that random credentials were used. Therefore the final tally is the same as the final tally of $\mathsf{Game}_2$. Similarly to the previous case, $\mathcal{A}$ cannot distinguish whether they are playing $\mathsf{Game}_2$ or $\mathsf{Game}_3$, due to the hardness of the $\mathsf{DDH}$ problem.

We note that the ideal game is essentially $\mathsf{Game}_3$ since all the values included in each ballot not constructed by $\mathcal{A}$ are random elements of $\mathbb{Z}_q$ or $\mathbb{G}$. As a result, our scheme provides coercion resistance, since the adversary cannot distinguish between the real and the ideal coercion resistance games. $\square$