Enhancing E-Voting with Multiparty Class Group Encryption

Michele Battagliola[1][0000−0002−8269−2148], Giuseppe D'Alconzo[2][0000−0001−7377−6617], Andrea Gangemi[2][0000−0001−9689−8473], and Chiara Spadafora[3][0000−0003−3352−9210]

[1] Department Information Engineering, Marche Polytechnic University
m.battagliola@staff.univpm.it
[2] Department of Mathematical Sciences, Politecnico di Torino
{giuseppe.dalconzo,andrea.gangemi}@polito.it
[3] Department of Mathematics, University of Trento
chiara.spadafora@unitn.it

Abstract. CHide is one of the most prominent e-voting protocols, which, while combining security and efficiency, suffers from having very long encrypted credentials. In this paper, starting from CHide, we propose a new protocol, based on multiparty Class Group Encryption (CGE) instead of discrete logarithm cryptography over known order groups. We achieve a computational complexity of $O(nr)$, for n votes and r voters, while calling the MixNet algorithm one time. The homomorphic properties of CGE allow for credentials that are shorter by a factor of 20 while maintaining the same level of security, at the cost of a small slowdown in efficiency.

Keywords: Class Group Encryption · Threshold Encryption · Coercion Resistance · E-voting.

1 Introduction

E-voting protocols are usually based on discrete logarithm cryptography; in particular, the most widely used encryption scheme is ElGamal (e.g., [2,24,35,36]), due to its homomorphic properties. However, protocols often require both sum and multiplication to be computed efficiently, and ElGamal is not suitable for this purpose, leading to protocols that are often computationally [30] or memory inefficient [4,18].

A possible alternative to ElGamal is the multiparty Class Group Encryption (CGE) [11,10]. Introduced in 2015 by Castagnos and Laguillaumie [13], CGE is the first discrete logarithm-based scheme that allows for an unlimited number of linear operations on plaintexts, without losing the ability to decrypt. Recently, Braun et al. introduced the notion of threshold CGE [10,11], which allows efficient multiplication of plaintexts using a multiparty protocol. This feature is particularly helpful for designing e-voting protocols, as it allows a wider range of

computations. In particular, our attention is focused on the CHide voting protocol: proposed by Cortier et al. in [18], CHide is an efficient [4,18] and secure protocol, as it achieves the strongest notion of coercion resistance.

1.1 E-voting protocols

Internet voting, often referred to as remote electronic voting (e-voting), enables voters to cast their votes via the Internet, removing the need for physical polling stations. Since its introduction in the early 2000s in Estonia and in the United States, its popularity has grown and it is now implemented to various degrees in several countries, including Switzerland [25], Canada [12] and Australia [27].

To be considered secure, an election must ensure vote and voter privacy, vote verifiability and the correctness of the final results. Cryptographic protocols are well-suited for this task, and numerous protocols, as Helios [3], Selene [34] and Civitas [15], have been recently developed to secure Internet-based elections.

However, there is one additional property that is equally crucial to address in a fair and democratic electoral process: resistance to coercion. In other words, a voting protocol must protect voters from being forced into voting in a specific way, either through threats or rewards. This threat becomes even greater in the context of e-voting, due to its remote nature. This greatly expands the range of possible coercion-attacks compared to in-person voting at polling stations.

1.2 Related Work

One of the first e-voting protocols that counters coercion is [30] by Juels, Catalano and Jakobsson (JCJ). They proposed the first mathematical formalization of coercion resistance, which is still the benchmark for research in the field. However, the real security of JCJ was recently questioned: the paper [18] critically examines the definition of coercion resistance given in [30] and identifies a weakness that stems from the procedure preceding the tally, in which trustees remove ballots that should not be counted.

Thus, the authors proposed a new definition of coercion resistance which overcomes the weaknesses of the one proposed by JCJ and designed CHide, a new protocol that achieves it. In addition of being more secure than JCJ, CHide is also more efficient, having a computational complexity of $O((n+r)\log(n+r)^2)$, where n denotes the number of votes and r is the number of voters, instead of the quadratic complexity of the JCJ protocol. Parallel to CHide, Aranha et al. [4] used a very similar approach to solve the security issue of JCJ, achieving a better computational complexity of $O((n+r)\log(n+r))$, at the price of an additional mixnet. However, both of them suffer from the fact that the encrypted credentials are very long: in fact, instead of the "standard" ElGamal encryption, the credentials are encrypted bit-by-bit, so that a k bits credential is expanded in k ElGamal ciphertexts. This negatively impacts performance, resulting in very space-consuming protocols and causing big constant factor in the computational complexity.

Protocol	Tally compl.	MixNets	Enc. cred.	Security
JCJ [30]	$O(n^2 + nr)$	2	95.6 B	JCJ
CHide [18]	$O\left((n+r)\log_2(n+r)^2\right)$	1	8160 B	CHide
Aranha et al. [4]	$O\left((n+r)\log_2(n+r)\right)$	2	8160 B	CHide
This work	$O(nr)$	1	390.6 B	CHide

Table 1. Comparison between our proposal and the JCJ and CHide voting schemes. **Tally compl.** is the asymptotic complexity of the Tally Phase, where n denotes the number of cast votes and r the number of credentials; **MixNets** is the number of calls to the MixNet subroutine; **Enc. cred.** is the size of the encryption of a credential for a security parameter $\lambda = 128$ and **Security** is the type of security provided by the scheme: with *CHide* we mean that the scheme achieves the Coercion Resistance property of Definition 2, while *JCJ* denotes the weaker notion from [30].

Other notions of coercion resistance. Besides the definition proposed by JCJ in [30], other definitions have been used, such as the one utilized by NetVote and ReVote [1,39] and by VoteAgain [31]. Comparing these definitions is often hard, but in general the (patched) JCJ one is regarded as the strongest [26]. For this reason we focus our analysis on protocols that achieve it.

1.3 Our Contribution

This paper combines the advantages of the JCJ protocol [30] and the CHide variants [4,18]. Our contribution is twofold. First, we propose an alternative version of [4,18] with shorter credentials, modifying the encryption scheme from ElGamal to CGE. The greater flexibility offered by CGE allows us to avoid encrypting credentials bit-by-bit, resulting in a protocol that requires much less memory: as shown in Table 1 our protocols require about 20 times less memory than [4,18], while achieving the same security level. However, by doing so, we lose the advantage of bit-by-bit encryption, which is the possibility to sort credentials, so our protocol suffers a slowdown compared to CHide. As a second contribution, to limiting this slowdown we propose an optimized version of our proposal, obtaining a tally that, while not as efficient as [4], requires less than quadratic time. Specifically, the idea is to "amortize" the time required to compute the final count by performing some precomputation during the voting phase, instead of waiting until the end of the election. While the overall computational complexity is the same, the time interval between the end of the election and the publication of results is significantly shorter.

Table 1 shows a comparison between the various protocols: for JCJ and CHide (both versions), to compute the length of an encrypted credential we assume that ElGamal is performed on Curve25519 [8], while we use [9] for CGE.

Organization. The paper is organized as follows: Section 2 recaps the cryptographic tools that are later used in Section 3, where our new voting protocol is described. This section also presents all the zero-knowledge protocols that are necessary. Section 4 is devoted to security proofs, while Section 5 describes two optimizations of the base protocol.

2 Preliminaries

In the course of this paper, we denote with λ the security parameter. In the pseudocode "$\leftarrow\!\!\$$" denotes the random sampling, "$\leftarrow$" is a variable assignment and "$=$" is an equality check.

2.1 Proofs and Arguments of Knowledge

An interactive protocol for a relation $\mathbb{L} \subseteq \mathbb{Y} \times \mathbb{W}$ is a protocol between a prover, who holds a statement-witness pair $(y, w) \in \mathbb{L}$, and a verifier, who knows only the statement y. At the end of the interaction, we want an honest prover to be able to convince the verifier, possibly without leaking any additional information. On the other hand, we want a dishonest prover (not knowing w) to be unable to convince the verifier. The sequence of exchanged messages is said *transcript*.

Definition 1. *[37] An interactive protocol Π between a Prover P and a Verifier V proving a relation $\mathbb{L}$ can satisfy the following properties.*

- Completeness*: if P follows the protocol on input $(y, w) \in \mathbb{L}$, the verifier accepts with overwhelming probability.*
- Soundness*: for every $y \in \mathbb{Y}$ such that does not exist $w \in \mathbb{W}$ for which $(y, w) \in \mathbb{L}$, a Prover P having input y is accepted by a Verifier V with negligible probability.*
- Special Soundness*: there exists an efficient deterministic algorithm $\mathcal{E}$, called* extractor*, with the following property: whenever $\mathcal{E}$ is given as input the statement $y \in \mathbb{Y}$ and two accepting transcripts* $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ *and* $(\mathsf{com}, \mathsf{ch}', \mathsf{resp}')$*, with* $\mathsf{ch} \neq \mathsf{ch}'$*, $\mathcal{E}$ outputs $w \in \mathbb{W}$ such that $(y, w) \in \mathbb{L}$.*
- Honest-Verifier Zero Knowledge *(HVZK): there exists a polynomial-time algorithm $\mathcal{S}$, called* simulator*, which on input $y \in \mathbb{Y}$ and a random challenge* ch*, outputs an accepting transcript* $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ *with the same probability distribution as the transcripts between an honest P and V on input y.*

An interactive protocol having completeness, special soundness and HVZK is called zero-knowledge proof of knowledge*. An interactive protocol satisfying completeness, soundness and HVZK is said* zero-knowledge argument of knowledge*.*

To ease the notation, in the rest of the paper we will refer both to proofs and arguments using the term "proofs". However, it will be specified when an interactive protocol is an argument and not a proof. An interactive protocol is said to be *public coin* if the message of the verifier is taken from a random source. For such class of protocols, the Fiat-Shamir transform [23] allows to remove the interaction between the prover and the verifier. We will use this method to produce non-interactive zero-knowledge proofs and arguments from the interactive ones in Section 3.1.

Given a zero-knowledge proof Π for the relation $\mathbb{L} \subseteq \mathbb{Y} \times \mathbb{W}$, we denote the protocol performed on witness $w \in \mathbb{W}$ and statement $y \in \mathbb{Y}$ as $\Pi[y, w]$.

Proofs and arguments of knowledge can be composed to perform AND and OR proofs of the relations they prove. The AND proof of two protocols Π_1 and Π_2 is simply given by their parallel composition, and the AND proof is accepting if both are accepting; we write $\Pi_1 \wedge \Pi_2$. The standard construction of an OR proof can be found in [21, Sect. 4] and we denote it with $\Pi_1 \vee \Pi_2$.

2.2 Cryptography from Class Groups

Class Group Encryption (CGE) is a public key encryption scheme introduced in [13]. This construction is *linearly homomorphic*: given ciphertexts $c_1 = \mathsf{Enc}(m_1)$, $\ldots$, $c_k = \mathsf{Enc}(m_k)$ and public values $\mu_0, \mu_1, \ldots, \mu_k$, there exists a procedure Linear such that

$$\mathsf{Linear}(c_1, \ldots, c_k; \mu_0, \mu_1, \ldots, \mu_k) = \mathsf{Enc}\left(\mu_0 + \sum_{i=1}^{k} \mu_i m_i\right).$$

To build such a scheme, we follow the construction from [13]. For a security parameter λ, consider an abelian finite group $\hat{G}$ of unknown order having a cyclic subgroup $F = \langle f \rangle$ of order $p > 2^\lambda$, where computing the discrete logarithm of any element is easy. The order of $\hat{G}$ is then $\hat{s}p$ for some integer $\hat{s}$. Let $h = x^p$ for a random $x \in \hat{G}$, and define $g = hf$ and $G = \langle g \rangle$. We can see that $G \cong H \times F$ where $H = \{y^p \mid y \in G\}$ and $H = \langle h \rangle$. Then, let $\epsilon \in \mathbb{N}$ be a statistical distance parameter and let $\mathcal{D}_H$ be a distribution over the integers. We assume that the distributions $\{h^x \mid x \leftarrow\!\$\ \mathcal{D}_H\}$ and the uniform distribution in H have negligible statistical distance. Suppose to know a bound $\tilde{s} > \hat{s}$, then $\mathcal{D}_H$ can be instantiated as the uniform distribution on $[0, 2^{\epsilon-2}\tilde{s}[$. The public parameters of the encryption scheme are given by the tuple $\mathsf{pp} = (\hat{G}, \tilde{s}, g, h, f)$. The secret key is given by $\mathsf{sk} \leftarrow\!\$\ \mathcal{D}_H$, while the public key is $\mathsf{pk} = h^{\mathsf{sk}}$. Then, to encrypt a message m in $\mathbb{Z}/p\mathbb{Z}$, the sender samples a random r from $\mathcal{D}_H$ and computes

$$\mathsf{Enc}(\mathsf{pk}, m, r) = (h^r, \mathsf{pk}^r f^m) = (\mathsf{ct}_1, \mathsf{ct}_2) = \mathsf{ct}.$$

From now on, the ciphertexts will be denoted as pairs $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$, i.e., a list of ciphertexts will have the indexes as superscripts $\mathsf{ct}^{(1)}, \ldots, \mathsf{ct}^{(\ell)}$, and each entry is a pair $(\mathsf{ct}_1^{(i)}, \mathsf{ct}_2^{(i)})$. To decrypt, the receiver computes $\mathsf{ct}_2/\mathsf{ct}_1^{\mathsf{sk}} = f^m$ and then retrieves m since $f^m \in F$ and the discrete logarithm in F is easy. We denote $m = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_1, \mathsf{ct}_2)$. This encryption scheme is linearly homomorphic: given two ciphertexts $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ and $\mathsf{ct}' = (\mathsf{ct}'_1, \mathsf{ct}'_2)$ which encrypt $m, m' \in \mathbb{Z}/p\mathbb{Z}$ respectively, we have that $(\mathsf{ct}_1 \cdot \mathsf{ct}'_1, \mathsf{ct}_2 \cdot \mathsf{ct}'_2)$ is an encryption of $m + m'$, and $(\mathsf{ct}_1^a, \mathsf{ct}_2^a)$ is an encryption of am for any $a \in \mathbb{Z}$.

Threshold Encryption and Multiparty Computation from CGE. In [11], a threshold CGE and a multiparty computation protocol are presented. In particular, secret keys are shared with a t-out-of-n secret sharing scheme, meaning that every subset of $t-1$ users does not obtain any information on the secrets. Then, this threshold scheme is employed to allow an efficient algorithm for plaintext

multiplication that does not require any decryption. We will use the arithmetic functionalities presented in [11] as the basis of the operations used in the voting protocol. Namely, the functionalities $\mathsf{Init}, \mathsf{KeyGen}, \mathsf{Decrypt}$ from [11, Sect. 6.1] and the arithmetic black-box functionalities $\mathsf{Init}, \mathsf{Input}, \mathsf{Output}, \mathsf{Linear}$ and $\mathsf{Multiply}$ from [11, Sect. 7]. For the sake of readability, since CGE is already linearly homomorphic and linear operations do not require communication, in our protocols, instead of calling Linear, they are executed directly. Differently, when a homomorphic multiplication between two ciphertexts is needed, we will use the $\mathsf{Multiply}$ functionality. Recently, the paper [10] improves on [11], by presenting a new key generation algorithm with lower communication complexity and new methods to batch zero-knowledge proofs in unknown order groups.

Mixnet from CGE. A *Mixnet* is a protocol that takes as input an ordered set of k ciphertexts $(C_1, \dots, C_k)$ and returns another ordered set of ciphertexts $(C'_1, \dots, C'_k)$ such that C'_i is a re-encryption of the plaintext from $C_{\pi^{-1}(i)}$ for a random permutation π. In the e-voting scenario, mixnets are used to randomize the list of valid encrypted votes. In this way, the link between the voter and their preference is broken and the secrecy is ensured.

A well-known mixnet that works with ElGamal encryption has been defined in [5] by Bayer and Groth. Later, Beaugrand et al. [6] adapted the protocol from [5] to work with the class group framework, providing a mixnet with sublinear communication complexity. In our construction, we will use the proposal from [6]: the functionality Mixnet takes as input a list of class group ciphertexts and returns the permuted and re-encrypted list, with a proof π_{Mixnet} of its correctness.

Cryptographic assumptions. It has been shown that the CGE public key encryption scheme is $\mathsf{IND\text{-}CPA}$ secure in [14] under the *Hard Subgroup Membership assumption* (HSM) [14], stating that distinguishing between elements of the form h^a from elements of the form g^b is intractable. In this context, $a \leftarrow\!\!\$\ \mathcal{D}_H$ and b is sampled from a distribution on integers $\mathcal{D}_G$ such that the statistical distance between the uniform distribution on G and $\{g^a \mid a \leftarrow\!\!\$\ \mathcal{D}_G\}$ is negligible.

In groups of unknown order, to prove statements in zero-knowledge, it is not known how to use Schnorr-like proofs since, to show the soundness of the protocol, some exponents need to be inverted. Thus, to overcome the limitation of the unknown order of $\hat{G}$ in the soundness proofs, we rely on the *C-rough assumption* (RO_C) [11], which informally states that class groups $\hat{G}$ having order $q \nmid \mathrm{ord}(\hat{G})$ for each prime $q < C$ are indistinguishable from class groups not having this property.

Another assumption needed in our protocol is the *Unknown Order* (ORD) one [20], where it must be intractable to compute an element $h \in \hat{G} \setminus F$ and a non-zero integer e such that $h^e = 1$. In other words, it must be infeasible to find an element in $\hat{G} \setminus F$ and a multiple of its order.

The threshold CGE between N parts presented in [11] is secure and tolerates up to $t < N/2$ corruptions under the ORD and RO_{N+1} assumptions. Finally, the CGE-based mixnet presented in [6] is secure under the RO_C assumption.

All the assumptions presented here are formally defined in Appendix A.

2.3 Security Properties of E-Voting Protocols

In order to be considered secure, an e-voting protocol with algorithms ($\mathsf{Setup}, \mathsf{Reg}, \mathsf{Vote}, \mathsf{Tally}$) should enforce the following properties: *correctness* [30], *fairness* [19], *vote privacy* [38], individual verifiability (usually expressed in terms of *cast-as-intended* [22], *recorded-as-cast* [32], *tallied-as-recorded* [33]), *universal verifiability* [38], *eligibility verifiability* [17]) and coercion resistance [18,30].

Coercion resistance is one of the most challenging properties to achieve when designing e-voting protocols. In this paper, we use the definition proposed in [18], which is an improvement of [30]. In simple words, in the registration phase, every voter is provided with a voting credential and a way to produce *fake credentials* that are indistinguishable from real ones. Votes cast with fake credentials are subsequently discarded. To evade coercion, voters can handle to the coercer the fake credential, without losing the ability to cast a valid vote. In the following, we report its formal definition.

We consider a distribution $\mathbb{D}$ of sequence of pairs (j, ν) where j is a voter and ν is a voting option in $[1, n_V]$. Additionally, fake votes are modeled as pairs where $j \notin [1, n_V]$. The definition of coercion resistance follows the standard real-ideal world paradigm: in the real world, the adversary controls some authority and participates in the protocol, while in the ideal world, at the end of the voting phase, the tally is done by a trusted third party. We say that a voting protocol is secure if and only if for every adversary in the real world there is an equivalent adversary in the ideal world. With reference to [18], we have the following definition.

Definition 2 (Coercion Resistance [18]). *A voting protocol with algorithms* ($\mathsf{Setup}, \mathsf{Reg}, \mathsf{Vote}, \mathsf{Tally}$) *is* coercion resistant *if there exists an algorithm* Fake *that, on input a credential* σ *outputs a random credential* $\tilde{\sigma}$ *such that for every adversary* $\mathcal{A}$*, for all parameters* $n_T, t, n_V, n_\mathcal{A}, n_C$ *and for all distributions* $\mathbb{D}$*, there exists a simulator* $\mathcal{S}$ *such that*

$$\Pr(\mathsf{Ideal}^{\mathsf{CR}}_{\mathcal{S}}(\lambda, n_V, n_\mathcal{A}, n_C, \mathbb{D}) = 1) - \Pr(\mathsf{Real}^{\mathsf{CR}}_{\mathcal{A}}(\lambda, n_T, t, n_V, n_\mathcal{A}, n_C, \mathbb{D}) = 1)$$

is negligible, where $\mathsf{Ideal}^{\mathsf{CR}}$ *and* $\mathsf{Real}^{\mathsf{CR}}$ *are defined in Figure 1.*

3 The Voting protocol

The participants in the voting protocol are:

- The *public board* $\mathcal{BB}$, an honest append-only list of data, where all the participants can write and read.
- The election *trustees*, a set $\mathcal{T}$ of $n_\mathcal{T}$ authorities that performs the tally. We allow corruptions up to the threshold $t < n_\mathcal{T}$ of the encryption scheme.
- The set of *voters* $\mathcal{V}$. There are $n_\mathcal{V}$ voters and at least 2 of them are honest.

$\mathsf{Real}^{\mathsf{CR}}(\mathcal{A}, k, n_T, t, n_V, n_\mathcal{A}, n_C, \mathcal{BB})$	$\mathsf{Ideal}^{\mathsf{CR}}(\mathcal{A}, k, n_V, n_\mathcal{A}, n_C, \mathcal{D})$
$\mathcal{BB} \leftarrow \emptyset$	
$\mathsf{pk}, \mathsf{sk}_i, h_i \leftarrow \mathsf{Setup}^{\mathcal{A}}(k, n_T, t)$	
$\{\sigma^i\}_{i \in V}, R \leftarrow \mathsf{Reg}(k, \mathsf{pk}, n_V)$	
$V_\mathcal{A} \leftarrow A()$	$V_\mathcal{A} \leftarrow A()$
$(j, \beta) \leftarrow \mathcal{A}(\{\sigma^i\}_{i \in V}, R)$	$(j, \beta) \leftarrow \mathcal{A}()$
if $\lvert V_\mathcal{A}\rvert \neq n_A$ or $j \notin V \setminus V_\mathcal{A}$ or $\beta \notin [0, n_C]$	**if** $\lvert V_\mathcal{A}\rvert \neq n_A$ or $j \notin V \setminus V_\mathcal{A}$ or $\beta \notin [0, n_C]$
return 0	**return** 0
$B \leftarrow \mathcal{D}(n_V - n_\mathcal{A}, n_C)$	$B \leftarrow \mathcal{D}(n_V - n_\mathcal{A}, n_C)$
for $(i, *) \in B, i \notin [1, n_V]$ **do**	
$\sigma^i \leftarrow \mathsf{Fake}(\sigma^i)$	
$b \leftarrow\!\!\$\ \{0, 1\}$	$b \leftarrow\!\!\$\ \{0, 1\}$
$\tilde{\sigma} \leftarrow \sigma^j$	
if $b == 1$	**if** $b = 1$
remove all $(j, *)$ from B	Remove all $(j, *)$ from B
else	**else**
Remove all $(j, *)$ from B but the last	Remove all $(j, *)$ from B but the last
Replace it with (j, β)	Replace it with (j, β)
$\tilde{\sigma} \leftarrow \mathsf{Fake}(\sigma^j)$	
$\mathcal{A}(\tilde{\sigma})$	
for $(i, \alpha) \in B$ **do**	$(\nu_i)_{i \in V_\mathcal{A}}, \beta' \leftarrow \mathcal{A}(\lvert B\rvert)$
$M \leftarrow \mathcal{A}(\mathcal{BB})$	**if** $b = 1$ and $\beta \neq \emptyset$
$\mathcal{BB} \leftarrow \mathcal{BB} \cup \{m \in M \lvert m \text{ valid}\}$	$B \leftarrow B \cup \{(j, \beta')\}$
$\mathcal{BB} \leftarrow \{\mathsf{Vote}(\sigma^i, \alpha, \mathsf{pk})\}$	$B \leftarrow B \cup \{(i, \nu_i) \lvert i \in V_\mathcal{A}, \nu_i \in [1, n_C]\}$
$M \leftarrow \mathcal{A}(\mathcal{BB})$	
$\mathcal{BB} \leftarrow \mathcal{BB} \cup \{m \in M \lvert m \text{ valid}\}$	
$X, \Pi \leftarrow \mathsf{Tally}^{\mathcal{A}}(\mathcal{BB}, R, \mathsf{pk}, \{h_i, \mathsf{sk}_i\}, t)$	$X \leftarrow \mathsf{Result}(\mathsf{Cleanse}(B))$
$b' \leftarrow \mathcal{A}()$	$b' \leftarrow \mathcal{A}(X)$
return $b = b'$	**return** $b = b'$

Fig. 1. Security game of coercion resistance. λ is the security parameter, n_T the number of talliers, t the threshold, n_V the number of voters, $n_\mathcal{A}$ the number of corrupted voters, n_C the number of voting options and $\mathbb{D}$ the distribution. The algorithm Reg generates the public-private credential pair for each voter, while Result returns the tally and $\mathsf{Cleanse}$ removes votes from invalid voters and re-votes. For more details, refer to [18].

- The *auditors*, a set of parties that checks the consistency of the data published on the board and the validity of all the zero-knowledge proofs (ZKPs). Since every check involves only public data, any party could serve as an auditor.
- The *registrars*, a second set $\mathcal{R}$ of $n_{\mathcal{R}}$ authorities that issue credentials to voters. For coercion resistance, we require that all the registrars are honest.

The voting protocol is composed by four phases.

1. **Setup Phase:** the authorities generate the public data and the parameters for the election (e.g. the public key, the hash function used for the non-interactive ZKPs, etc.).
2. **Registration Phase:** voters authenticate themselves with the relevant authorities and receive voting materials, usually containing their voting credentials and a proof of their correctness. For coercion resistance we required that the voting material allows for an *Evasion Strategy*, to be used in case of coercion.
3. **Voting Phase:** voters vote using the obtained credentials. Voters can vote more than once (usually, re-voting invalidates previous votes). During this phase, voters should be able to verify the correctness of the protocol, checking that the vote was cast-as-intended and recorded-as-cast. This is usually done via a combination of ZKP, the usage of a public board and device auditing techniques like the Benaloh Challenge [7].
4. **Tally Phase:** the election result is computed, and published along with a ZKP about its correctness.

Setup Phase. A security parameter λ is chosen. The $n_{\mathcal{T}}$ election trustees jointly execute the distributed key generation protocol presented in [10]. They use the Init and the KeyGen functionalities, producing a public key pk and private shares sk_i for $i = 1, ..., n_{\mathcal{T}}$, one for each trustee. A commitment h_i for the private share of sk_i is published by the i-th trustee on the public board, in addition to pk.

Registration phase. Credentials are created by registrars and their encryptions are published on the public bulletin board $\mathcal{BB}$. Each credential is sent privately to the voter, with designated verifier zero-knowledge proofs (DVZKP) to ensure its validity.[4] We denote with σ a credential and with $\mathbb{R}_\sigma$ the list of all the authorized (encrypted) credentials. We refer to elements in $\mathbb{R}_\sigma$ as the "public credentials".

Evasion Strategy. To evade coercion a voter can simply lie about their credential σ, generate a random fake credential $\tilde{\sigma}$ and give it to the coercer, manipulating the DVZKP accordingly. In this way, voters are also able to vote with their correct credential.

[4] Voter authentication is out of the scope of this paper but, for example, it could be done via a digital signature by the user with a long-term key pair.

Voting Phase. To cast a vote for candidate ν, voter V computes an encryption of their voting choice $\mathsf{ct}^{\mathsf{V}} = \mathsf{Enc}(\mathsf{pk}, \nu, \rho_\nu)$ and an encryption of their credential $\mathsf{ct}^{\mathsf{C}} = \mathsf{Enc}(\mathsf{pk}, \sigma, \rho_\sigma)$. Additionally, the voter computes two proofs: π_{Vote}, the proof produced by the protocol Π_{Vote} to prove that ν is a valid voting option, and π_{Cred} produced by Π_{Cred} to prove the knowledge of σ in ct^{C}. These proofs are also used to link together ct^{V} and ct^{C}, making the pair $(\mathsf{ct}^{\mathsf{V}}, \mathsf{ct}^{\mathsf{C}})$[5] non-malleable. To link these two proofs we apply the Fiat-Shamir transform to the AND (i.e. concatenation) of them. We call the tuple $(\mathsf{ct}^{\mathsf{V}}, \mathsf{ct}^{\mathsf{C}}, \pi_{\mathsf{Vote}}, \pi_{\mathsf{Cred}})$ a "ballot". This ballot is then published by the voter V on $\mathcal{BB}$ using an anonymous channel.
During the Voting Phase, each voter V can vote multiple times. For simplicity, the policy we implement is that we count only the last vote cast with each credential. This is implicit in how the Update function in the tally phase is designed. During this step the auditors verify the uniqueness of each ballot and that every published proof is valid. Votes that fail these checks are discarded and are not processed further.

Tally Phase. The tally procedure is a multiparty protocol between the $n_{\mathcal{T}}$ trustees. In order to tally the votes, we need the equality Eq and update Update functionalities presented later. From now on, we refer to $\mathsf{Enc}(\sigma) = \mathsf{Enc}(\mathsf{pk}, \sigma, \rho)$. The equality functionality Eq is used to check whether two ciphertexts encrypt the same credential or not: it takes as input two ciphertexts $(\mathsf{Enc}(\sigma), \mathsf{Enc}(\bar{\sigma}))$ and returns the encryption of 1 if $\sigma = \bar{\sigma}$ and an encryption of 0 otherwise. It is presented in Figure 2 and uses the exponentiation functionality Exp, which, in turn, needs the multiparty multiplication protocol Multiply from [11]. The protocol for equality Eq is correct, since when $\sigma = \bar{\sigma}$ we have that $b = 0$, otherwise $b = 1$, due to $x^p = 1$ for all $x \in (\mathbb{Z}/p\mathbb{Z})^*$. Observe that the exponentiation procedure Exp, shown in Figure 2, is performed using the square-and-multiply procedure, i.e. using $O(\log_2(p))$ multiplications.

Exp($\mathsf{Enc}(x)$,s)

1 : $(s_0, \ldots, s_\ell) \leftarrow \mathsf{Binary}(s)$
2 : $Y \leftarrow \mathsf{Enc}(1)$
3 : **for** $i = \ell, \ldots, 0$ **do**
4 : $\quad Y \leftarrow \mathsf{Multiply}(Y, Y)$
5 : $\quad$ **if** $k_i = 1$
6 : $\quad\quad Y \leftarrow \mathsf{Multiply}(Y, \mathsf{Enc}(x))$
7 : **return** Y

Eq($\mathsf{Enc}(\sigma)$,$\mathsf{Enc}(\bar{\sigma})$)

1 : $y \leftarrow \mathsf{Enc}(\sigma) \cdot \mathsf{Enc}(\bar{\sigma})^{-1}$
2 : $b \leftarrow \mathsf{Exp}(y, p)$
3 : **return** $\mathsf{Enc}(1) \cdot b^{-1}$

Fig. 2. Exponentiation and Equality procedures. Here p is the order of F. Multiply is the multiplication functionality from in [11].

[5] Notice that each ciphertext of the couple is a couple itself, as per defined in Section 2.2.

Now, we present the Update functionality (Figure 3) for the update of the votes. It takes as input two pairs of vote-credential encryptions $(\mathsf{Enc}(\nu), \mathsf{Enc}(\sigma))$ and $(\mathsf{Enc}(\bar{\nu}), \mathsf{Enc}(\bar{\sigma}))$, and it returns a ciphertext $\mathsf{ReEnc}(\nu^*)$, where $\nu^* = \nu$ if $\sigma \neq \bar{\sigma}$ and $\nu^* = \bar{\nu}$ otherwise. Here, $\mathsf{ReEnc}(x)$ is the encryption of x using a different randomness.

Finally, we describe the Tally protocol (Figure 3). The idea is to start with a list R of null votes, encoded by the element 0, one for each credential in $\mathbb{R}_\sigma$ and update it with all the ballots in $\mathcal{BB}$, so that, at end, R contains the last vote made with each credential. Formally, at the beginning of Tally Phase, we have

$$R = \{(\mathsf{Enc}(0), \mathsf{Enc}(\sigma_i)) \mid \sigma_i \in \mathbb{R}_\sigma\}\,.$$

For each ballot in $\mathcal{BB}$, the whole list R is updated using Update, such that the entry corresponding to the credential in the considered ballot is changed, while the others are simply re-encrypted. Note that it is impossible to distinguish whether an entry is simply re-encrypted or changed. Also, note that when a vote is cast with a fake credential, it means that the entire R is re-encrypted, since no entry in R correspond to a fake credential. This is done to avoid leaking the number of fake votes. In order to do this, first, the $n_\mathcal{T}$ trustees create the list R as described above. Note that this is a copy and not a re-encryption of the credentials σ_i's. Then, for each entry $\mathcal{BB}_i$ of the public bulletin board $\mathcal{BB}$ and for each entry R_j of the list R the Update function is performed, so that, after this computation, the first entry of R_j is the encryption of the last vote made with the credential σ_j. Let $R[1]$ be the list of the first entries of every element in R: at the end of the procedure this is the list of valid votes. The mixnet is performed on this list using the functionality Mixnet from [6], obtaining a permuted list of valid votes, breaking the link between the voter and his preference. The Mixnet procedure additionally produces a proof π_{Mixnet} that has to be checked in order to verify the correctness of the shuffle. Finally, the trustees can decrypt the permuted list via the functionality Decrypt from [11], getting the valid preferences in clear.

3.1 Zero-Knowledge Proofs

Apart from the proofs of knowledge needed and produced by the multiparty protocol from [11,10] and the mixnet from [6], the voting system, in its voting phase, uses two different zero-knowledge proofs. Let $(\mathsf{ct}^\mathsf{V}, \mathsf{ct}^\mathsf{C})$ be the pair of ciphertexts where $\mathsf{ct}^\mathsf{V} = \mathsf{Enc}(\mathsf{pk}, \nu, \rho_\nu)$ is the encryption of a voting preference ν and $\mathsf{ct}^\mathsf{C} = \mathsf{Enc}(\mathsf{pk}, \sigma, \rho_\sigma)$ is an encryption of their credential σ. The first argument of knowledge, denoted with Π_{Vote} proves that ν is a valid voting option. The second proof Π_{Cred} is needed to prove the knowledge of σ in the ciphertext ct^C.

The zero-knowledge proofs will be presented as a 3-move interactive protocols which can be made non-interactive by the Fiat-Shamir transform [23] by computing the verifier message as a hash function of the concatenation of the statement and the first message. For all the upcoming protocols, $\mathsf{pp} = \left(\hat{G}, \widetilde{s}, g, h, f\right)$ is the tuple of public parameters, where $\hat{G}$ has unknown order and it has a subgroup

$\mathsf{Update}(B, \bar{B})$

1 : $(\mathsf{Enc}(v), \mathsf{Enc}(\sigma)) \leftarrow B$
2 : $(\mathsf{Enc}(\bar{v}), \mathsf{Enc}(\bar{\sigma})) \leftarrow \bar{B}$
3 : $b \leftarrow \mathsf{Eq}(\mathsf{Enc}(\bar{\sigma}), \mathsf{Enc}(\sigma))$
4 : $m_1 \leftarrow \mathsf{Multiply}(b, \mathsf{Enc}(\bar{v}))$
5 : $m_2 \leftarrow \mathsf{Multiply}(\mathsf{Enc}(1) \cdot b^{-1}, \mathsf{Enc}(v))$
6 : **return** $m_1 \cdot m_2$

$\mathsf{Tally}(\mathcal{BB}, \mathbb{R}_\sigma)$

1 : **for** $i = 1 \ldots, |\mathbb{R}_\sigma|$ **do**
2 : $\quad R_i \leftarrow (\mathsf{Enc}(\perp), \mathbb{R}_{\sigma\, i})$
3 : **for** $i = 1, \ldots, |\mathcal{BB}|$ **do**
4 : $\quad$ **for** $j = 1, \ldots, |R|$ **do**
5 : $\quad\quad \mathsf{Enc}(v^*) \leftarrow \mathsf{Update}(\mathcal{BB}_i, R_j)$
6 : $\quad\quad R_j[1] \leftarrow \mathsf{Enc}(v^*)$
7 : $(V, \pi_{\mathsf{Mixnet}}) \leftarrow \mathsf{Mixnet}(R[1])$
8 : **if** $\mathsf{Verify}(\pi_{\mathsf{Mixnet}}) = \perp$ **return** $\perp$
9 : $y \leftarrow \mathsf{Decrypt}(V)$
10 : **return** y

Fig. 3. Update and Tally procedures.

Π_{PoPK}

$\underline{\textbf{Prover}}(\mathsf{pp}, (\mathsf{ct}_1, \mathsf{ct}_2), m, \rho)$		$\underline{\textbf{Verifier}}(\mathsf{pp}, (\mathsf{ct}_1, \mathsf{ct}_2))$
$\tilde{\rho} \leftarrow\!\!\$\ [0, 2^{2\epsilon} C\tilde{s}[,\ \tilde{m} \leftarrow\!\!\$\ \mathbb{Z}/\mathbb{Z}p,$		
$\tilde{\mathsf{ct}} \leftarrow \mathsf{Enc}(\mathsf{pk}, \tilde{m}, \tilde{\rho})$	$\xrightarrow{\tilde{\mathsf{ct}}}$	
	$\xleftarrow{x}$	$x \leftarrow\!\!\$\ [0, C[$
$\hat{\rho} \leftarrow \tilde{\rho} + \rho x,\ \hat{m} \leftarrow \tilde{m} + mx$	$\xrightarrow{\hat{m}, \hat{\rho}}$	**accept if**
		$\tilde{\mathsf{ct}}_1 \cdot \mathsf{ct}_1^x = h^{\hat{\rho}},\ \tilde{\mathsf{ct}}_2 \cdot \mathsf{ct}_2^x = \mathsf{pk}^{\hat{\rho}} f^{\hat{m}}$

Fig. 4. Interactive protocol Π_{PoPK} to prove knowledge of m and ρ from [6].

$G \cong H \times F$, with H a cyclic group generated by h and F the unique subgroup of G of order p, which is generated by f. Furthermore, the public key used to encrypt is $\mathsf{pk} = h^{\mathsf{sk}}$, where sk is the corresponding secret key. More details can be found in Section 2.2.

The starting point to design both Π_{Vote} and Π_{Cred} is the *proof of plaintext knowledge* Π_{PoPK} from [6, Sect. 3], which demonstrates plaintext knowledge of a public ciphertext, in particular, given a public key pk, it is an interactive protocol for the following relation

$$\{(\mathsf{ct}, (m, \rho)) \mid \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m, \rho)\}\,.$$

For completeness, the protocol is described in Figure 4, and its security is based on the RO_C assumption. Note that the integer C that appears in the protocol is exactly the C used in the definition of this assumption. More details about how to prove the soundness and the special soundness properties of interactive protocols with unknown group order can be found in [6].

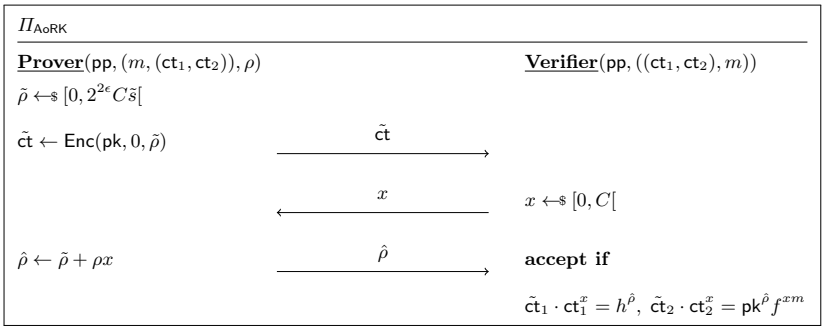

Fig. 5. Interactive protocol Π_{AoRK} to prove knowledge of ρ such that $(\mathsf{ct}_1, \mathsf{ct}_2) = \mathsf{Enc}(\mathsf{pk}, m, \rho)$, for a public message m.

The zero-knowledge proof Π_{Cred} is exactly the proof of plaintext knowledge given in Figure 4 for a ciphertext $\mathsf{ct}^{\mathsf{C}} = \mathsf{Enc}(\mathsf{pk}, \sigma, \rho_\sigma)$ encrypting a credential σ with randomness ρ_σ

$$\Pi_{\mathsf{Cred}}\left[\mathsf{ct}^{\mathsf{C}}, (\sigma, \rho_\sigma)\right] = \Pi_{\mathsf{PoPK}}\left[\mathsf{ct}^{\mathsf{C}}, (\sigma, \rho_\sigma)\right].$$

The proof Π_{Vote} proves that, given ct^{V}, the voter 1) knows the plaintext, i.e. the vote, and 2) the vote is an admissible vote. Indeed, Π_{Vote} can be viewed as an AND proof of the two statements above. The first one is directly given by the proof of plaintext knowledge Π_{PoPK} from Figure 4, while, for the second, a slightly involved proof is needed.

We want to design a protocol to prove that the vote ν given in $\mathsf{ct}^{\mathsf{V}} = \mathsf{Enc}(\mathsf{pk}, \nu, \rho_\nu)$ is a correct vote, i.e. it belongs to the set of valid votes $\{\nu_1, \ldots, \nu_k\}$. The building block of this proof is an *argument of randonmess knowledge* Π_{AoRK}, proving the relation

$$\{((m, \mathsf{ct}), \rho) \mid \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m, \rho)\}.$$

Observe that in this case the message m is part of the statement. We can design an argument of randonmess knowledge slightly modifying the proof of plaintext knowledge from Figure 4. The protocol is given in Figure 5.

Remark 1. There are some technical difficulties to design a *proof* of randonmess knowledge. As noted in [6], the proof of plaintext knowledge Π_{PoPK} from Figure 4 is *partial extractable*; this means that only a part of the witness (m, ρ) can be efficiently extracted, namely, the message m. However, even if we cannot extract the randomness ρ in the protocol of Figure 5, it is complete, sound, and honest-verifier zero-knowledge, showing that it is an *argument* of knowledge. This is enough for our purposes, as we show later in Section 4. Indeed, the protocol is only used during the voting phase to identify and discard invalid votes, and in the security proof we do not need to extract the randomness used.

Theorem 1. *The protocol Π_{AoRK} for randomness knowledge described in Figure 5 is a zero-knowledge argument, i.e., it is complete, sound and honest-verifier zero-knowledge under the RO_C assumption.*

The proof of Theorem 1 can be found in Appendix B and it follows the one of [6, Th. 2].

On top of the argument of randomness knowledge Π_{AoRK}, we can build an *argument of plaintext equality* Π_{AoPE} for the relation

$$\{((\mathsf{ct}, \mathsf{ct}'), (m, \rho, \rho')) \mid \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m, \rho) \wedge \mathsf{ct}' = \mathsf{Enc}(\mathsf{pk}, m, \rho')\}\,.$$

We can then show that two ciphertexts $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m, \rho)$ and $\mathsf{ct}' = \mathsf{Enc}(\mathsf{pk}, m, \rho')$ share the same public message m showing that, due to the homomorphic properties of the CGE, the ciphertext ct/ct' is an encryption of 0. This is done using Π_{AoRK}, setting ct/ct' as the ciphertext, 0 as the message and $\rho - \rho'$ as randomness. Following the notation from above, we set

$$\Pi_{\mathsf{AoPE}}\left[(m, \mathsf{ct}, \mathsf{ct}'), (\rho, \rho')\right] = \Pi_{\mathsf{AoRK}}\left[(0, \mathsf{ct}/\mathsf{ct}'), \rho - \rho'\right].$$

At this point, we have the ingredients to design the argument of knowledge Π_{Vote}. It is obtained by using the OR construction of k argument of plaintext equality, each of which proves that the plaintext in ct^{C} is ν_i for $i \in [k]$. Observe that the standard construction for OR proof given in [21, Sect. 4] has communication linear in the number of clauses, in our case k. Recall that the AND of two interactive protocols is simply given by their concatenation. Hence, the proof Π_{Vote}, proving the relation

$$\{(\mathsf{ct}, (\nu, \rho_\nu)) \mid \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, \nu, \rho_\nu) \wedge \nu \in \{\nu_1, \ldots, \nu_k\}\}\,,$$

for public ρ_i's, is given by

$$\Pi_{\mathsf{Vote}}\left[\mathsf{ct}^{\mathsf{V}}, (\nu, \rho_\nu)\right] = \Pi_{\mathsf{PoPK}}\left[\mathsf{ct}^{\mathsf{V}}, (\nu, \rho_\nu)\right] \bigwedge \left(\bigvee_{i=1}^{k} \Pi_{\mathsf{AoPE}}\left[\left(\nu_i, \mathsf{ct}^{\mathsf{V}}, \mathsf{Enc}(\mathsf{pk}, \nu_i, \rho_i)\right), (\rho_\nu, \rho_i)\right]\right).$$

The argument of knowledge Π_{Vote} and the proof of knowledge Π_{Cred} are then linked together to make them non-malleable: when using the Fiat-Shamir transform, for each proof the challenge is computed using the first messages and statements of the two proofs together.

3.2 Designated Verifier Re-Encryption Proof

For the security proof, a last zero-knowledge proof is needed. We present Π_{DVRE}, a Designated Verifier Re-Encryption proof [28] which can be faked by the Designated Verifier Ver in order to validate a fake credential. In short, it is an OR proof of 1) a proof of a Re-Encryption, i.e. given $\mathsf{ct}^{(0)}, \mathsf{ct}^{(1)}$, one proves that they are two encryptions of the same message, and 2) the proof of knowledge of a secret key $\mathsf{sk}_{\mathsf{Ver}}$ linked to a publicly known public key $\mathsf{pk}_{\mathsf{Ver}}$.

For the sake of clarity, we first show the proof Π_{RE} of Re-Encryption for the relation

$$\left\{\left((\mathsf{ct}^{(0)}, \mathsf{ct}^{(1)}), \tilde{\rho}\right) \mid \mathsf{ct}^{(1)} = (h^{\tilde{\rho}}\mathsf{ct}_1^{(0)}, \mathsf{pk}^{\tilde{\rho}}\mathsf{ct}_2^{(0)})\right\}$$

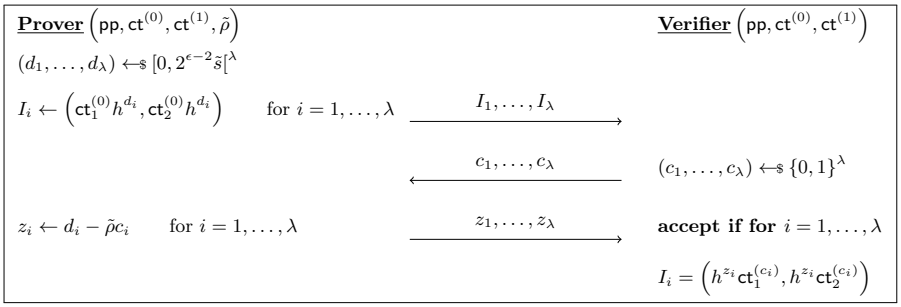

Fig. 6. Protocol to prove that $\mathsf{ct}^{(1)}$ is a re-encryption of $\mathsf{ct}^{(0)}$.

in Figure 6. Here, we use a standard parallel repetition with λ binary challenges, where λ is the security parameter.

Theorem 2. *The interactive protocol Π_{RE} described in Figure 6 is complete, special sound and honest-verifier zero-knowledge.*

The proof of Theorem 2 is a standard proof for a parallel repetition protocol with binary challenge and it can be found in Appendix B.

For the second part, observe that the key pair $(\mathsf{sk}_{\mathsf{Ver}}, \mathsf{pk}_{\mathsf{Ver}})$ does not have to be related to CGE. However, we can still use a discrete logarithm-based keypair. In particular, suppose that there exists a public element $\mathfrak{g}$ of order q such that $\mathsf{pk}_{\mathsf{Ver}} = \mathfrak{g}^{\mathsf{sk}_{\mathsf{Ver}}}$. This implies that we can use known order groups, allowing us to use more efficient protocols, such as Schnorr. However, since the two challenge spaces must be equal, for the proof of knowledge of $\mathsf{sk}_{\mathsf{Ver}}$, we use λ parallel repetitions with binary challenges for simplicity. The DVZKP is then an OR proof, obtained using standard techniques. For completeness, it is presented in Appendix C.

4 Security Analysis

The security proof for our voting protocol is very similar to the ones presented in [4,18]. Hence, here we just sketch the proof.

Theorem 3. *Under the* HSM*, the* RO_C *and the* ORD *assumptions, the voting system presented in Section 3 is coercion-resistant.*

Proof. Let $\mathcal{A}$ be an adversary for the real game. We give to $\mathcal{A}$ the power to impersonate t among $n_{\mathcal{T}}$ election trustees and up to $n_{\mathcal{A}}$ voters. Our goal is to build an adversary $\mathcal{S}$ for the ideal game using $\mathcal{A}$ as a subroutine. In particular, $\mathcal{S}$ controls the remaining $n_{\mathcal{T}} - t$ trustees and needs to simulate both the Setup and the Tally.

First of all, $\mathcal{S}$ and $\mathcal{A}$ run the Setup algorithm to generate a common public key pk, secret shares of the private key $\mathsf{sk}_1, ..., \mathsf{sk}_{n_{\mathcal{T}}}$ and the public commitments $h_1, ..., h_{n_{\mathcal{T}}}$. During this step, $\mathcal{S}$ is also able to reconstruct the secret key sk by

extracting $\mathcal{A}$'s secrets. The simulation of the key generation is shown in Section 6.2 of [11]. Then, $\mathcal{S}$ follows the real game normally, getting the set of corrupted voters $V_{\mathcal{A}}$, the coerced voter j and the voting choice β from the adversary. In the ideal game, $\mathcal{S}$ sends the same choices for $V_{\mathcal{A}}, j, \beta$.

When asking for the credential of voter j, $\mathcal{S}$ provides to $\mathcal{A}$ the real credential σ^j. From the ideal game, $\mathcal{S}$ learns the size $|B|$ of the ideal board and uses it to simulate the voting process. For $|B|$ times:

- $\mathcal{S}$ calls $\mathcal{A}$ with input B getting $\mathbb{M}$, a list of ballots the adversary wants to make.
- $\mathcal{S}$ decrypts all the valid votes and credentials in $\mathbb{M}$. For every authorized credential σ^i, $\mathcal{S}$ saves the tuple (σ^i, ν) or updates a previously saved (σ^i, ν').
- $\mathcal{S}$ adds all valid ballots in $\mathbb{M}$ to B.
- $\mathcal{S}$ chooses a random voter and a valid voting option and casts a valid vote, adding it to B.

At the end of the voting process, $\mathcal{S}$ adds the same votes in the ideal game.

$\mathcal{S}$ learns the result of the election X at the end of the ideal game and uses it to simulate the tallying process in the real game:

- $\mathcal{S}$ simulates all the Update procedure, due to the UC security of threshold CGE.
- $\mathcal{S}$ simulates the MixNet controlling the honest authorities, while $\mathcal{A}$ uses the dishonest ones.
- $\mathcal{S}$ chooses $|X|$ entries at random and simulates its partial decryption: every entry not chosen is decrypted to 0, while such $|X|$ entries are decrypted such that the result is exactly X.

At this point $\mathcal{A}$ makes its guess b and $\mathcal{S}$ forwards the same guess in the ideal game. The differences between a real execution and the simulation are:

- In the real game $\mathcal{A}$ can get either the real credential σ^j or a fake one. In the simulation $\mathcal{A}$ always receives σ^j. Since in both the real and ideal worlds fake credentials have uniformly random distribution and the DVZKP could be simulated, $\mathcal{A}$ can only distinguish a real execution from a simulated one if and only if it is able to distinguish whether the received credential is a plaintext of one of the encrypted credentials in $\mathbb{R}_\sigma$ or not. Since CGE is IND-CPA due to the HSM assumption, this is impossible.
- During the simulation of the voting loop $\mathcal{S}$ adds random ballots, while in the real game ballots are drawn according to $\mathbb{D}$. As before, since the ballots are encrypted, the simulation is indistinguishable from the real game.
- During the tally, $\mathcal{S}$ simulates the execution of the Update procedure. Due to the UC security of the MPC class group encryption, the simulation is indistinguishable from a real execution, as shown in [11].
- In the simulation, the result always includes all the last valid ballots cast by honest voters. In a real execution, the adversary may change it by casting ballots on behalf of an honest voter. However, to do so, the adversary must be able to create a valid proof about the credential used, and this is unfeasible.
- $\mathcal{S}$ simulates the decryption protocol at the end. This simulation is indistinguishable under the ORD and the RO_C assumptions, as shown in [11]. □

4.1 Notes about Privacy and Verifiability

Here, we provide an informal discussion about privacy and verifiability, with a sketch of the proof. Before that, however, we need to define IND-PA0 security, that is indistinguishability under parallel chosen plaintext attack. It is a stronger property than security under chosen plaintext attack (IND-CPA), where the adversary has also access to a list of decrypted ciphertexts. For a formal definition, see [18]. It is possible to show that the voting map of Section 3

$$(\nu, \sigma) \to (\mathsf{Enc}(\mathsf{pk}, \nu, \rho_\nu), \mathsf{Enc}(\mathsf{pk}, \sigma, \rho_\sigma), \pi_{\mathsf{Vote}}, \pi_{\mathsf{Cred}})$$

is an IND-PA0 encryption scheme, where π_{Vote} and π_{Cred} are the proofs produced by Π_{Vote} and Π_{Cred}, the protocols described in Section 3.1. In particular, π_{Vote} is a proof that the vote ν is known and correct and π_{Cred} proves the plaintext knowledge of σ.

Privacy. Informally, privacy means that it is impossible to guess which option a voter chose. Formally, in the privacy game, the adversary $\mathcal{A}$ chooses two voting options ν_0, ν_1 and an "observed voter" v_o, who picks a random bit b and votes ν_b. The adversary, controlling $t-1$ trustees, wins if they guess b. See Appendix C of [18] for a formal definition. To prove it, we make a reduction to the IND-PA0 security of the encryption protocol used.

Sketch. Suppose that $\mathcal{A}$ is an adversary that wins the privacy game with non-negligible advantage. We show how to build an adversary $\mathcal{S}$ for the IND-PA0 that wins with non-negligible advantage. Indeed, $\mathcal{S}$ chooses ν_0, ν_1 as the plaintext for the IND-PA0 game. Before the tally, $\mathcal{S}$ uses the whole board $\mathcal{BB}$ as a query in the IND-PA0 game, except for the vote cast by v_o, for which $\mathcal{S}$ chooses randomly one of the two voting options. At this point, $\mathcal{S}$ can simulate the whole tally knowing the end result and guesses whatever bit b $\mathcal{A}$ guesses. When $\mathcal{S}$ chooses correctly the vote for v_o, the simulation is perfect and $\mathcal{S}$ wins every time $\mathcal{A}$ wins, which happens with probability $\frac{1}{2} + \mathsf{negl}$. Instead, if $\mathcal{S}$ picks the wrong choice, which happens $\frac{1}{2}$ of the times, $\mathcal{S}$ wins with probability $\frac{1}{2}$. Overall, the winning probability of $\mathcal{S}$ is $\frac{1}{2} + \mathsf{negl}$ which is non negligible. □

Verifiability. Universal verifiability is granted by the proofs produced by the trustees and by the honesty of the bulletin board. Cast-as-intended instead is more tricky, since it should require to design algorithms for voters to inquire about their own devices. Many protocols, such as the Benaloh challenge [7], are suitable, however, they usually rely more on "responsible behaviour" from the user, without having a solid security proof. Moreover, they often fall short when analyzed from a game theory standpoint, like in [29] where the authors suggest that the optimal strategy is to (almost) always ignore the audit step and cast the vote immediately. For this reason, we left cast-as-intended out of our scope and we suggest to employ one of the various established solutions since our protocol can support many of them ([16]).

5 Performance and Optimizations

The main advantage of our protocol is the compactness of the encrypted credentials, which are k times shorter than in [4,18], since we only need a single ciphertext instead of k for a k-bit credential. The complexity of Tally Phase is $O(nr)$, where n is the number of votes and r is the number of registered credentials. Although not optimal and higher than the complexity of the protocols presented in [4,18], our complexity is still better than that of many coercion resistance protocols, whose execution typically takes place in $O(n^2 + nr)$. Furthermore, it is worth noting that having shorter credentials has a large impact on the actual execution time of the protocol, since the execution time of [4,18] is linear with respect to the number of ciphertexts.

We propose two improvements in order to achieve a better execution time while, unfortunately, maintaining the same asymptotic complexity $O(nr)$.

Online tally. To further optimize the Tally procedure (see Figure 3), a possibility would be to perform the Update procedure every time a new vote is added to $\mathcal{BB}$. While the overall complexity would remain the same, this would allow for distibute the workload throughout the entire voting process, rather than only at the end, resulting in an overall reduction of the delay between the end of the voting phase and the publication of the results.

Removing the MixNet. If the number N of candidates is small enough, an optimization can be implemented to avoid the mixnet. The principle is the following. Suppose that the votes are cast in $\mathbb{Z}/p\mathbb{Z}$, where $p \simeq 2^M$. A preference for the i-th candidate is encoded in the vote $2^{(i-1)\frac{M}{N}}$. In the tally phase, instead of decrypting all the votes (after the MixNet), we can simply compute $T = \Pi_{i=1}^{|R|} R_i[1]$ and then decrypt T. Notice that $T = \sum_{i=0}^{N-1} 2^{(i-1)\frac{M}{N}} x_i$, where x_i is the number of votes for the i-th candidate. From the knowledge of T, one can easily retrieve all the x_i's using its representation in base $2^{\frac{M}{N}}$. To avoid an overflow of votes, an upper bound to the number of voters is $2^{\frac{M}{N}}$.

Acknowledgments. The first author is supported by the Italian Ministry of University's PRIN 2022 program under the "Mathematical Primitives for Post Quantum Digital Signatures" (P2022J4HRR) and by the FISA project "Quantum-sate cryptographic tools for the protection of national data and intormation technology assets" (QSAFEIT). The second and the third authors are members of the INdAM Research group GNSAGA and CrypTO, the group of Cryptography and Number Theory of the Politecnico di Torino. The work of the fourth author has been supported by a joint laboratory between Fondazione Bruno Kessler and the Italian State Mint and Polygraphic Institute.

References

1. Achenbach, D., Kempka, C., Löwe, B., Müler-Quade, J.: Improved Coercion-Resistant electronic elections through deniable Re-Voting. USENIX Journal of

Election Technology and Systems (JETS) **3**(2), 26–45 (Aug 2015), https://www.usenix.org/jets/issues/0302/achenbach
2. Adida, B.: Helios: Web-based open-audit voting. In: USENIX security symposium. vol. 17, pp. 335–348 (2008)
3. Adida, B.: Helios: Web-based Open-Audit Voting. In: USENIX Security Symposium. pp. 335–348. USENIX Association (2008)
4. Aranha, D.F., Battagliola, M., Roy, L.: Faster coercion-resistant e-voting by encrypted sorting. E-Vote-ID 2023 (2023). https://doi.org/10.18420/e-vote-id2023_03
5. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31. pp. 263–280. Springer (2012)
6. Beaugrand, A., Castagnos, G., Laguillaumie, F.: Efficient succinct zero-knowledge arguments in the cl framework. Journal of Cryptology **38**(1) (Jan 2025). https://doi.org/10.1007/s00145-024-09534-1
7. Benaloh, J.: Simple verifiable elections. In: Workshop on Accurate Electronic Voting Technology. p. 5. EVT'06, USENIX Association (2006)
8. Bernstein, D.J.: Curve25519: New diffie-hellman speed records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) Public Key Cryptography - PKC 2006. pp. 207–228. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
9. Bouvier, C., Castagnos, G., Imbert, L., Laguillaumie, F.: I want to ride my bicycl: Bicycl implements cryptography in class groups. J. Cryptol. **36**(3) (Apr 2023). https://doi.org/10.1007/s00145-023-09459-1, https://doi.org/10.1007/s00145-023-09459-1
10. Braun, L., Castagnos, G., Damgård, I., Laguillaumie, F., Melissaris, K., Orlandi, C., Tucker, I.: An improved threshold homomorphic cryptosystem based on class groups. In: Galdi, C., Phan, D.H. (eds.) Security and Cryptography for Networks. pp. 24–46. Springer Nature Switzerland, Cham (2024)
11. Braun, L., Damgård, I., Orlandi, C.: Secure Multiparty Computation from Threshold Encryption Based on Class Groups. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023. pp. 613–645. Springer Nature Switzerland, Cham (2023)
12. Cardillo, A., Akinyokun, N., Essex, A.: Online Voting in Ontario Municipal Elections: A Conflict of Legal Principles and Technology? In: E-VOTE-ID. LNCS, vol. 11759, pp. 67–82. Springer (2019)
13. Castagnos, G., Laguillaumie, F.: Linearly Homomorphic Encryption from DDH. In: Nyberg, K. (ed.) Topics in Cryptology — CT-RSA 2015. pp. 487–505. Springer International Publishing, Cham (2015)
14. Castagnos, G., Laguillaumie, F., Tucker, I.: Practical fully secure unrestricted inner product functional encryption modulo p. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 733–764. Springer (2018)
15. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a Secure Voting System. In: IEEE Symposium on Security and Privacy. pp. 354–368. IEEE Computer Society (2008)
16. Cortier, V., Dreier, J., Gaudry, P., Turuani, M.: A simple alternative to Benaloh challenge for the cast-as-intended property in Helios/Belenios. In: HAL (2019)
17. Cortier, V., Gaudry, P., Glondu, S.: Belenios: a simple private and verifiable electronic voting system. In: Foundations of Security, Protocols, and Equational Reasoning, pp. 214–238. Springer (2019)

18. Cortier, V., Gaudry, P., Yang, Q.: Is the JCJ voting system really coercion-resistant? (2024). https://doi.org/10.1109/CSF61375.2024.00003
19. Recommendation CM/Rec(2017)5 of the Committee of Ministers to member States on standards for e-voting, https://search.coe.int/cm/Pages/result_details.aspx?ObjectID=0900001680726f6f
20. Couteau, G., Klooß, M., Lin, H., Reichle, M.: Efficient range proofs with transparent setup from bounded integer commitments. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 247–277. Springer (2021)
21. Damgård, I.: On σ-protocols. Lecture Notes, University of Aarhus, Department for Computer Science **84** (2002)
22. Escala, A., Guasch, S., Herranz, J., Morillo, P.: Universal cast-as-intended verifiability. In: International Conference on Financial Cryptography and Data Security. pp. 233–250. Springer (2016). https://doi.org/10.1007/978-3-662-53357-4_16
23. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the theory and application of cryptographic techniques. pp. 186–194. Springer (1986)
24. Glondu, S.: Belenios specification. Tech. rep., Belenios Project (October 2024), https://www.belenios.org/specification.pdf, accessed on 30 May 2025
25. Haines, T., Pereira, O., Teague, V.: Running the Race: A Swiss Voting Story. In: E-Vote-ID. LNCS, vol. 13553, pp. 53–69. Springer (2022)
26. Haines, T., Smyth, B.: Surveying definitions of coercion resistance. IACR ePrint Arch. p. 822 (2019), https://eprint.iacr.org/2019/822
27. Halderman, J.A., Teague, V.: The New South Wales iVote System: Security Failures and Verification flaws in a live online election. In: VoteID. LNCS, vol. 9269, pp. 35–53. Springer (2015)
28. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 539–556. Springer (2000)
29. Jamroga, W.: Pretty Good Strategies for Benaloh Challenge. In: Volkamer, M., Duenas-Cid, D., Rønne, P., Ryan, P.Y.A., Budurushi, J., Kulyk, O., Rodriguez Pérez, A., Spycher-Krivonosova, I. (eds.) Electronic Voting. pp. 106–122. Springer Nature Switzerland, Cham (2023)
30. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Proceedings of the 2005 ACM workshop on Privacy in the electronic society. pp. 61–70 (2005)
31. Lueks, W., Querejeta-Azurmendi, I.n., Troncoso, C.: Voteagain: a scalable coercion-resistant voting system. In: Proceedings of the 29th USENIX Conference on Security Symposium. SEC'20, USENIX Association, USA (2020)
32. Müller, J., Truderung, T.: Caised: A protocol for cast-as-intended verifiability with a second device. In: International Joint Conference on Electronic Voting. pp. 123–139. Springer (2023)
33. Popoveniuc, S., Kelsey, J., Regenscheid, A., Vora, P.: Performance requirements for End-to-End verifiable elections. In: 2010 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 10). USENIX Association, Washington, DC (Aug 2010), https://www.usenix.org/conference/evtwote-10/performance-requirements-end-end-verifiable-elections
34. Ryan, P.Y.A., Rønne, P.B., Iovino, V.: Selene: Voting with Transparent Verifiability and Coercion-Mitigation. In: Financial Cryptography Workshops. LNCS, vol. 9604, pp. 176–192. Springer (2016)

35. State Electoral Office of Estonia: Ivxv protocols: Specification. Tech. Rep. Dok IVXV-PR-EN-1.8.0, State Electoral Office of Estonia (December 2022), https://www.valimised.ee/sites/default/files/2023-02/IVXV-protocols.pdf, accessed on 30 May 2025
36. Swiss Post: Swiss post e-voting system documentation. https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation/-/tree/master/System (May 2025), accessed on 30 May 2025
37. Thaler, J., et al.: Proofs, arguments, and zero-knowledge. Foundations and Trends® in Privacy and Security **4**(2–4), 117–660 (2022)
38. U.S. Election Assistance Commission: Voluntary Voting System Guidelines (VVSG) version 2.0 (02 2021), https://www.eac.gov/voting-equipment/voluntary-voting-system-guidelines
39. Weber, S.G., Araujo, R., Buchmann, J.: On coercion-resistant electronic elections with linear work. In: The Second International Conference on Availability, Reliability and Security (ARES'07). pp. 908–916 (2007). https://doi.org/10.1109/ARES.2007.108

A Cryptographic Assumptions

In this section, we report the cryptographic assumptions on which the proposed e-voting scheme relies.

Definition 3 (Hard Subgroup Membership assumption (HSM). *Let $\mathcal{D}_G$ and $\mathcal{D}_H$ two distributions over the integers such that*

1. *$\{g^x \mid x \leftarrow\$ \mathcal{D}_G\}$ is at distance less than $2^{-\lambda}$ from the uniform distribution on G and*
2. *$\{h^x \mid x \leftarrow\$ \mathcal{D}_H\}$ is at distance less than $2^{-\lambda}$ from the uniform distribution on G.*

Let $\mathsf{pp} = (\hat{G}, \tilde{s}, g, h, f)$ be the public parameters, then the Hard Subgroup Membership *assumption states that*

$$\left| \Pr\left[b = b' : x \leftarrow\$ \mathcal{D}_G,\, x' \leftarrow\$ \mathcal{D}_H,\, b \leftarrow\$ \{0,1\},\, Z_0 = g^x,\, Z_1 = h^{x'},\, b' \leftarrow\$ \mathcal{A}\left(\mathsf{pp}, Z_b\right)\right] - \frac{1}{2} \right|$$

is negligible in λ for every PPT adversary $\mathcal{A}$.

Definition 4 (C-rough assumption (RO_C)). *Let C be a natural number. Define CLGen as the algorithm taking as input the security parameter λ, a prime p and some randomness $\rho \in \{0,1\}^\lambda$ and returning the public parameters $\mathsf{pp} = (\hat{G}, \tilde{s}, g, h, f)$. Let $\mathcal{D}_C^{\mathsf{rough}}$ be the uniform distribution over the set*

$$\{\rho \in \{0,1\}^\lambda \mid \mathsf{pp} = \mathsf{CLGen}(1^\lambda, p, \rho) \vee \forall q < C,\, q \nmid ord(\hat{G})\}.$$

*The C-*rough *assumption states that*

$$\left| \Pr\left[1 \leftarrow\$ \mathcal{A}\left(1^\lambda, \rho_0\right) : \rho_0 \leftarrow\$ \{0,1\}^\lambda\right] - \Pr\left[1 \leftarrow\$ \mathcal{A}\left(1^\lambda, \rho_1\right) : \rho_1 \leftarrow\$ \mathcal{D}_C^{\mathsf{rough}}\right] \right|$$

is negligible in λ for every PPT adversary $\mathcal{A}$.

Definition 5 (Unknown Order assumption ORD). *Let* $\mathsf{pp} = (\hat{G}, \tilde{s}, g, h, f)$ *be the public parameters, then the* Unknown Order *assumption states that*

$$\Pr\left[(h, e) \leftarrow\!\$\ \mathcal{A}(1^\lambda, \mathsf{pp}) : h \in \hat{G} \setminus F,\ e \neq 0,\ h^e = 1\right]$$

is negligible in λ *for every PPT adversary* $\mathcal{A}$.

B Missing proofs

Proof of Theorem 1. The correctness follows from direct inspection.

The soundness property is proven following the same arguments from [6]. Assume that one can extract two accepting transcripts $(\tilde{\mathsf{ct}}, x, \hat{\rho})$ and $(\tilde{\mathsf{ct}}, x', \hat{\rho}')$ from a prover P^* which is accepted with non-negligible probability, with $x \neq x'$. Then, since they are accepting, we have that

$$(\tilde{\mathsf{ct}_1} \cdot \mathsf{ct}_1^x, \tilde{\mathsf{ct}_2} \cdot \mathsf{ct}_2^x) = (h^{\hat{\rho}}, \mathsf{pk}^{\hat{\rho}} f^{xm}) \text{ and } (\tilde{\mathsf{ct}_1} \cdot \mathsf{ct}_1^{x'}, \tilde{\mathsf{ct}_2} \cdot \mathsf{ct}_2^{x'}) = (h^{\hat{\rho}'}, \mathsf{pk}^{\hat{\rho}'} f^{x'm}).$$

Hence, dividing the two equations, we have

$$(\mathsf{ct}_1^{x-x'}, \mathsf{ct}_2^{x-x'}) = (h^{\hat{\rho}-\hat{\rho}'}, \mathsf{pk}^{\hat{\rho}-\hat{\rho}'} f^{(x-x')m}) = \mathsf{Enc}(\mathsf{pk}, (x-x')m, \hat{\rho} - \hat{\rho}').$$

By the C-rough assumption, since $0 < |x - x'| < C$, the element $x - x'$ is invertible modulo $q\hat{s}$, the order of $\hat{G}$ and denote its inverse with z. Then, we have

$$(\mathsf{ct}_1, \mathsf{ct}_2) = (\mathsf{ct}_1^{(x-x')z}, \mathsf{ct}_2^{(x-x')z}) = \mathsf{Enc}(\mathsf{pk}, m, z(\hat{\rho} - \hat{\rho}'))$$

and hence, there exists the randomness $z(\hat{\rho} - \hat{\rho}')$ for the ciphertext $(\mathsf{ct}_1, \mathsf{ct}_2)$ with plaintext m.

For the honest-verifier zero-knowledge property, we refer to [6, Th. 2]. □

Proof of Theorem 2. The correctness follows from direct inspection since, if $\mathsf{ct}^{(1)}$ is a re-encryption of $\mathsf{ct}^{(0)}$ using randomness $\tilde{\rho}$, then we have that

$$\mathsf{ct}^{(1)} = (\mathsf{ct}_1^{(1)}, \mathsf{ct}_2^{(1)}) = (h^{\tilde{\rho}} \mathsf{ct}_1^{(0)}, h^{\tilde{\rho}} \mathsf{ct}_2^{(0)}).$$

To prove special soundness, consider two accepting transcripts $(\{I_i\}, \{c_i\}, \{z_i\})_i$ and $(\{I_i\}, \{c_i'\}, \{z_i'\})_i$ with different challenges such that there exist an index j with $c_j \neq c_j'$. For instance, suppose that $c_j = 0$ and $c_j' = 1$. Then, the extractor computes the witness $\tilde{\rho}$ as $z_j' - z_j$.

The following simulator $\mathcal{S}_{\mathsf{RE}}$ provides the honest-verifier zero-knowledge property. For each i from 1 to λ, the simulator samples a random bit challenge c_i, then it picks a random z_i from $[0, 2^{\epsilon-2}\tilde{s}[$ and computes $I_i = (h^{z_i} \mathsf{ct}_1^{(c_i)}, h^{z_i} \mathsf{ct}_2^{(c_i)})$. The produced transcript has the same distribution of a honest-generated one. □

C Designated Verifier ZKP

Here we show the explicit Designated Verifier zero-knowledge proof from Section 3.1. Recall that it is designed as an OR proof of the proof of re-encryption from Figure 6 and a proof of knowledge of $\mathsf{sk}_{\mathsf{Ver}}$ given the public pair $(\mathfrak{g}, \mathfrak{g}^{\mathsf{sk}_{\mathsf{Ver}}} = \mathsf{pk}_{\mathsf{Ver}})$. For the latter, a standard proof using λ repetition and binary challenges can be used, having simulator $\mathcal{S}_{\mathsf{Ver}}$.

The OR proof is described in Figure 7. It uses the witness w, which can be $\tilde{\rho}$ or $\mathsf{sk}_{\mathsf{Ver}}$. If the parameter `MODE` is equal to 0, then the prover knows the randomness $\tilde{\rho}$ and calls $\mathcal{S}_{\mathsf{RE}}$, the simulator of the protocol in Figure 6; otherwise, if `MODE` is equal to 1, the prover knows the secret key $\mathsf{sk}_{\mathsf{Ver}}$ and calls the simulator $\mathcal{S}_{\mathsf{Ver}}$.

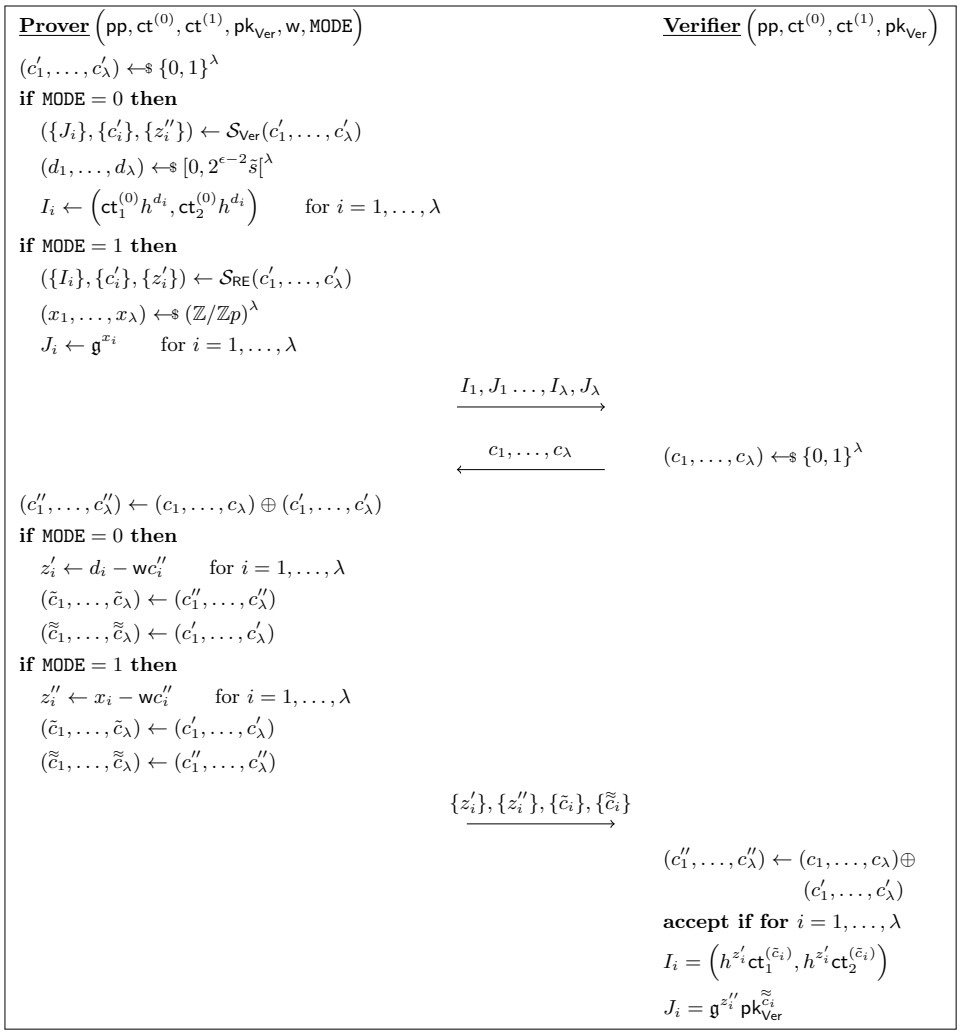

Fig. 7. Interactive protocol for the Designed Verifier ZKP.