Efficient Cleansing in Coercion-Resistant Voting

Rosario Giustolisi[1] and Maryam Sheikhi Garjan[2]

[1] IT University of Copenhagen, Copenhagen, Denmark
rosg@itu.dk
[2] Brandenburg University of Technology, Cottbus, Germany
sheikhig@b-tu.de

Abstract. Coercion resistance is a strong security property of electronic voting that prevents adversaries from forcing voters to vote in a specific way by using threats or rewards. There exist clever techniques aimed at preventing voter coercion based on fake credentials, but they are either inefficient or cannot support features such as revoting without leaking more information than necessary to coercers. One of the reasons is that invalid ballots cast due to revoting or coercion need to be removed before the tallying. In this paper, we propose a coercion-resistant Internet voting scheme that does not require the removal of invalid ballots, hence avoids the leakage of information, but still supports revoting. The scheme is very efficient and achieves linear tallying.

1 Introduction

The utilization of ICT solutions is becoming more prevalent, particularly within electoral processes. The EU Commission has recently acknowledged [12] that Internet voting facilitates elections and encourages the digitalization of many sectors and activities in society. However, Internet voting is risky. A shift towards Internet voting would introduce unprecedented challenges to election correctness. The most obvious one is voter coercion, in which a coercer forces a voter to cast a ballot in a particular way.

Most cryptographic schemes achieve coercion resistance by either deniable revoting or fake credentials. In deniable revoting, voters update or nullify previously cast votes while being under coercion. Schemes based on deniable revoting normally assume over-the-shoulder coercion and that the voter has always a chance to revote after being coerced. These assumptions are not needed in fake credentials as coercers cast ballots using fake credentials. Such ballots are removed from the tally during the so-called cleansing phase. Cleansing is a critical process that traditionally follows voting and precedes tallying, in which the talliers verifiably remove the ballots that should not be counted due to revoting or coercion. Efficient coercion-resistant voting schemes based on fake credentials have been proposed in the literature, but they publicly leak more information than necessary to coercers during cleansing. For example, information about ballots with the same credentials and those with invalid credentials is leaked out in JCJ [18]. A recent attempt [8] aimed at preventing information leakage during

cleansing achieves quasi-linear tallying, yet it is inefficient. Inefficiency originates from the fact that the the scheme rely on the multiparty computation (MPC) technique called CGate [27] and on mixnets [2] to prevent information leakage. CGate introduces heavy costs due to computations on bit-wise encryptions in the cleansing phase.

In this paper, we propose an Internet voting scheme that offers a new trade-off between coercion-resistance and efficiency. Assuming that the tally servers are trusted for coercion-resistance, we can avoid the leaking of any information during cleansing very efficiently and achieve linear tallying. Our scheme is based on noise ballots that obfuscate the ballots cast by voters, and on a cleansing procedure that excludes invalid and noise ballots without the need to remove ballots, therefore without leaking any information during cleansing publicly. The scheme guarantees a version of coercion-resistance that accounts for revoting and noise ballots. Our definition is based on a recent one by Cortier et al. [8]. The scheme is also very efficient as it provides linear tallying, and does not require MPC or mixnets. It relies on exponential ElGamal [11] and (disjunctive) non-interactive zero-knowledge proofs (NIZKP) of knowledge [10].

Contributions. The main contribution of this paper is a new Internet voting scheme that provides coercion-resistance without leaking any information during cleansing efficiently. We prove that our scheme satisfies coercion resistance under the DDH assumption in the random oracle model. Cleansing in our scheme is particularly efficient, and the complexity of tallying is linear. We provide a prototype implementation of our scheme in Python to demonstrate that the scheme provides fast cleansing and tallying.

2 Related work

The concept of the fake credential paradigm, introduced in JCJ [19], has been widely acknowledged as an effective method to achieve coercion resistance. In JCJ, tallying has a quadratic complexity in the number of votes. This is due to the cleansing steps required to eliminate invalid and revote ballots. Efforts have been made to enhance the efficiency of JCJ. Civitas [7] groups voters into blocks to reduce the tallying time. Weber et al. [31] use hash tables instead of plaintext equivalence tests (PET). Other approaches [16, 28–30] use a mix of hash tables and PET to remove ballots due to revoting and invalid credentials in linear time. Rønne et al. [26] advance a version of JCJ with linear-time tallying based on fully homomorphic encryption. Araújo et al. [3] use different cryptographic primitives than JCJ to achive tallying in linear time. However, all the schemes outlined above have the same cleansing leakage as JCJ [8].

To avoid information leakage at cleansing, Cortier et al. [8] propose CHide, a *cleansing-hiding* scheme that uses MPC and mixnets. Tallying is quasi-linear but MPC introduces several exponentiations and computations with bit-wise encryptions inducing a heavy cost for CHide. Differently from CHide, our work requires noise ballots and that the tally servers are trusted for coercion-resistance.

However, even considering a large number of noise ballots, our scheme is faster than CHide since it has no MPC or mixnets and it is fully parallelizable while guaranteeing *publicly* cleansing-hiding.

Schemes based on deniable vote updating [1,4,6,13,17,20–22,24,25] add noise ballots to mitigate information leakage to coercers. They require either that the voter can cast a ballot after being coerced or inalienable authentication at voting (i.e. over-the-shoulder coercion). Our scheme is based on fake credentials and does not have such assumptions.

The first formal definition of coercion-resistance [19] sets the coercer's advantage to distinguish between a real and an ideal game that simulates the voting scheme. Later, various definitions based on a real-ideal games have been proposed [18]. A general approach that defines quantitative coercion-resistance has been proposed in [21] as δ-coercion resistance. In this approach, the coerced voter has a specific strategy to evade coercion. Coercion resistance is ensured if the adversary cannot distinguish whether the coerced voter evades coercion with an advantage greater than δ. Grewal et al. [15] introduced a relaxed version of coercion-resistance in which voters can signal coercion attempts. We aim to a non-relaxed versions of coercion-resistance instead. More recently, Cortier et al. [8] proposed a definition of coercion resistance based on the one introduced in JCJ that captures revoting and the addition of noise ballots. We use this definition to prove that our scheme is coercion-resistant.

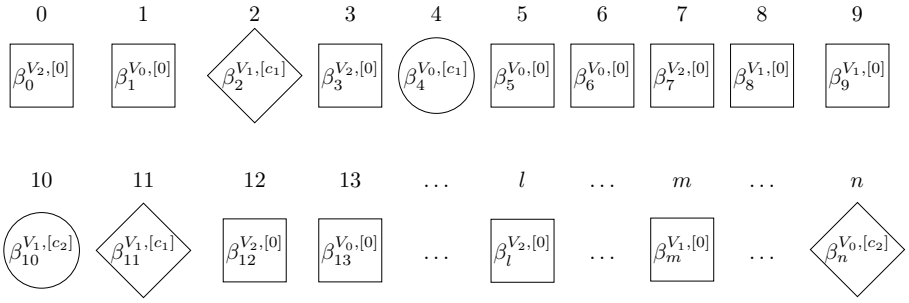

Fig. 1: In the voting phase, the bulletin board is filled with ballots. Ballots cast by a voter under coercion are in diamond, while coercion-free cast ballots are in circle. Noise ballots are in square. Inside each square bracket is the chosen candidate. In this example, the last ballot on the bulletin board is a coerced one and is for candidate $[c_2]$.

3 Overview

In the voting phase, the bulletin board receives ballots from voters and coercers, as well as, noise ballots from voting authorities. It is important that the distributions used to sample the number of noise ballots and to determine the time to

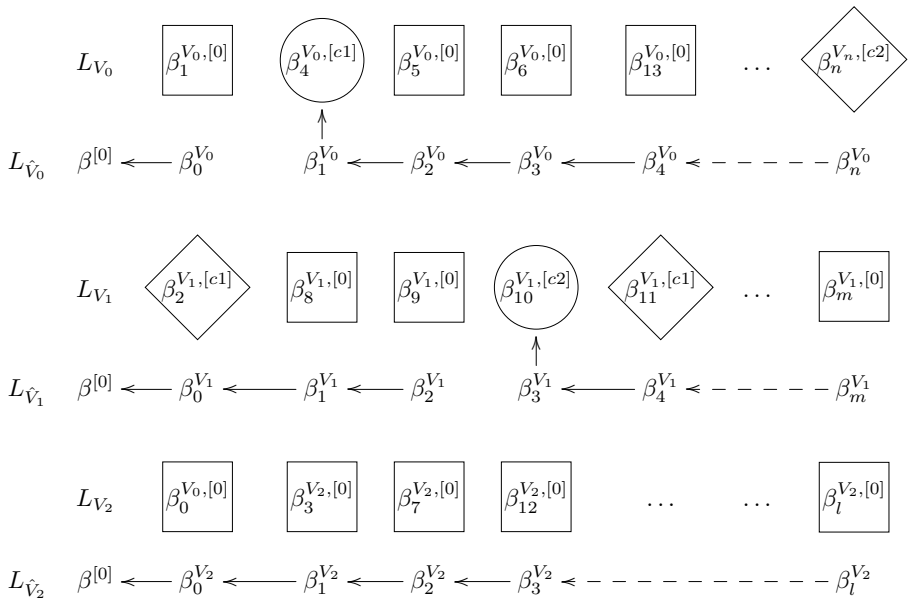

Fig. 2: Our cleansing technique in practice. Each voter list contains the ballots with the public identity of voter V_i. Arrows indicate which ballots are re-randomized to generate the cleansing lists. V_2 captures a voter who abstains from voting. The last ballots in the cleansing lists are those considered for tallying.

cast each of them are unpredictable [21] otherwise a coercer can learn the voting cast behaviour of voters and significantly distinguish the amout of noise ballots from the real ones. Voters can also generate and cast noise ballots to mitigate forced-abstention attacks and the extent to which the effectiveness of coercion resistance is dependent on authorities controlling noise ballot generation. In each ballot, it is indicated in clear to which voter the ballot should be assigned. Figure 1 shows an example of a bulletin board filled with some ballots.

In the cleansing phase, the tally servers associate each ballot to the assigned voter according to their cast time, generating public *voter lists* of ballots. For example, in Figure 2, the voter list for voter V_0 is L_{V_0}. The goal is that at the end of cleansing, the last ballot of each voter list encrypts the last vote cast with the valid credential, if any, or an encryption of zero. To do so, for each voter, the tally servers generates a *cleansed list* that will contain the same number of ballots of the voter list. In Figure 2 the cleansed list for voter V_0 is $L_{\hat{V}_0}$. The tally servers populate the cleansed list by checking, in order, the credential encrypted in each ballot from the corresponding voter list. If the ballot has the correct credential, the tally servers add to the cleansed list a new ballot that is the re-randomization of the ballot with the correct credential. Otherwise, the new ballot is the re-randomization of the previous ballot in the cleansed list. The tally servers re-randomize tha ballots using ElGamal re-encryption and

prove in zero-knowledge (i.e. using disjunctive NIZKP) the correctness of the re-randomization. The last ballots in the cleansing lists can homomorphically added to obtain the final tally.

3.1 Threat model

The list of participants is the same as JCJ. We consider voters, a registration authority, a bulletin board, and tally servers. Differently from JCJ, we require an authority (e.g. the tally servers) generating noise ballots. Voters may be dishonest and collude with the attacker. The attacker may attempt to coerce honest voters. The registration authority provides credentials to voters therefore is assumed to be honest. The bulletin board is assumed to present the same content to all readers therefore is an honest, append-only list of data that is publicly accessible. Therefore, it is susceptible to denial of service attacks as it receives anonymous ballots. The tally servers are responsible for tallying and publishing the final election results on the bulletin board. These servers form an honest majority t-out-of-n threshold encryption system and are trusted for coercion resistance. The communication channels between the voters and the registrar are untappable, and the voters send their ballots to the bulletin board via anonymous channels. Civitas [7] is an example of techniques that implement distributed participants with the related trust assumptions outlined above. Finally, we consider a computationally bounded adversary whose aim is to break ballot privacy, verifiability, deniable revoting, and coercion resistance.

3.2 Cryptographic primitives

The only two cryptographic primitives required in our scheme are ElGamal encryption and NIZKP.

Let λ and κ be the security parameters. Let $\mathbb{G}$ be a cyclic group of prime order p and generators $g, g_1, g_2 \in \mathbb{G}$. We denote the integers modulo p with $\mathbb{Z}_p$ and write $r \xleftarrow{\$} \mathbb{Z}_p$ for r being chosen uniformly from $\mathbb{Z}_p$. The encryption scheme is the modified ElGamal encryption scheme [19] for group $\mathbb{G}$, with generators g_1, g_2 of order p and message space $\mathbb{M} = {g_1}^b$, where $b = \{0, 1\}$ consisting of the following algorithms:

- $\mathsf{TKeyGen}(1^\lambda)$, which, on input of security parameter 1^λ, outputs a pair of ElGamal decryption and encryption keys (sk, pk) where $sk = (x_1, x_2), x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$, and $pk = g_1^{x_1} g_2^{x_2}$.
- $\mathsf{Enc}(pk, m; r)$, which, given a public key pk, a message $m \in \mathbb{M}$, and some randomness $r \xleftarrow{\$} \mathbb{Z}_p$, outputs a ciphertext $(c_1, c_2, c_3) = (g_1^r, g_2^r, m \cdot pk^r)$.
- $\mathsf{Dec}(sk, ct = (c_1, c_2, c_3))$, which outputs $m = (c_1)^{-x_1} \cdot (c_2)^{-x_2} \cdot c_3$.
- $\mathsf{ReEnc}(pk, ct = (c_1, c_2, c_3); r)$, which, using randomness $r \xleftarrow{\$} \mathbb{Z}_p$, outputs the reencryption of ct namely $(c_1 \cdot g_1^r, c_2 \cdot g_2^r, c_3 \cdot pk^r)$.
- $\mathsf{CKeyGen}(1^\kappa)$ which, on input security parameter 1^κ, outputs the credential σ, where $\sigma \xleftarrow{\$} \mathbb{G}$.

For verifiability, we use NIZKP of knowledge based on the Fiat-Shamir transforma to prove relations. We define the following relations to verify the proper construction of a voter's ballot and the computation of the tally.

The *proof of well-formed encryption* assures the verifier that ct is an accurate encryption of a message m and randomness r known to the prover, using the public encryption key pk. The corresponding relation is defined as $R_{enc} = \{((ct, pk), (r, m)) \in R_{enc}$ iff $ct = \mathsf{Enc}(pk, m; r)\}$ to compute NIZKP of knowledge π_{enc}. We also prove that $m \in \mathbb{M}$, where $\mathbb{M}$ denotes the range of the messages.

The *proof of correct decryption* assures the verifier that ct is decrypted to m by applying the knowledge of secret encryption key sk on ciphertext ct. The decryption relation is defined as $R_{dec} = \{((pk, m, ct), sk) \in R_{dec}$ iff $m = \mathsf{Dec}(sk, ct) \wedge pk = g^{sk}\}$ to compute the NIZKP of knowledge π_{dec}.

The *proof of correct re-encryption* assures the verifier that ct' is a valid re-encryption of ciphertext ct using randomness r with respect to a public encryption key pk. The corresponding relation $R_{r_{enc}}$ is defined as $R_{r_{enc}} = \{((pk, ct, ct'), r) \in R_{r_{enc}}$ iff $ct' = \mathsf{ReEnc}(pk, ct; r)\}$ to compute a NIZKP of knowledge $\pi_{r_{enc}}$.

We use disjunctive NIZKP of knowledge as introduced by Cramer et al. [9] for verifiable cleansing in the tally phase. Let $R_d = R_1 \vee R_2$ and $x = (x_1, x_2)$, a disjunctive NIZKP relation R is defined as follows:

$$\{((x_1, x_2), \omega) \in R_d \text{ iff } (x_1, \omega) \in R_1 \vee (x_2, \omega) \in R_2\}$$

To generate a proof for a defined relation, we use the function $\mathsf{Proof}(x, \omega)$, which takes a public statement x and a secret witness ω of the defined relation, and outputs the corresponding proof. We assume that the function Proof takes the corresponding relation as implicit input. For disjunctive NIZKP of knowledge, we use the function $\mathsf{DisjProof}(x, \omega)$ to compute the related proof.

4 Formal description

The algorithms defining the schemes are as follow.

- $\mathsf{Setup}(1^\lambda, (t, n), \mathbb{I}, \mathbb{C}) \rightarrow ((pk_T, sk_T), (pk_R, sk_R))$: on input of the security parameter 1^λ, threshold parameter (t, n) electoral roll $\mathbb{I}$, and candidate list $\mathbb{C}$ computes $(pk_T, sk_T) \xleftarrow{\$} \mathsf{TKeyGen}(1^\lambda)$ and $(pk_R, sk_R) \xleftarrow{\$} \mathsf{SKeyGen}(1^\lambda)$.
- $\mathsf{Register}(1^\kappa, \mathbb{I}, (sk_R, pk_R), pk_T) \rightarrow (\mathbb{L}, \{(id, \sigma, \hat{ct})\}_{id \in \mathbb{I}}$: on input of the security parameter 1^κ, sk_R, pk_R, pk_T, and $\mathbb{I}$ do the following.
  - Compute $\sigma \xleftarrow{\$} \mathsf{CKeyGen}(1^\kappa)$ to create a voting credential for voter id.
  - Compute $\hat{ct} \xleftarrow{\$} \mathsf{Enc}(pk_T, \sigma; r_{id})$
  - Add the tuple $(id, \hat{ct})$ to the registered voter roll $\mathbb{L}$.
  - Append the signed voter roll $\mathbb{L}$, to the public bulletin board, $\mathcal{BB}$.
  - Return $(id, \sigma, \hat{ct})$ to the voter id.
- $\mathsf{Vote}(\hat{ct}, \sigma, c) \rightarrow \beta$: on implicit input the tallier public key pk_T, secret credential σ, candidate option $c \in \mathbb{C}$, do the following.

- Compute $ct_\sigma \xleftarrow{\$} \mathsf{Enc}(pk_T, \sigma; r)$ and $ct_c \xleftarrow{\$} \mathsf{Enc}(pk_T, c; r_c)$.
- Run $\pi \xleftarrow{\$} \mathsf{Proof}(x, \omega)$, where $x=(pk_T, \hat{ct}, ct_\sigma, ct_c)$ and $\omega=(r_c, c, r, \sigma)$:

$$(x, \omega) \in R_\beta \text{ iff } ct_c = \mathsf{Enc}(pk_T, c; r_c) \wedge ct_\sigma = \mathsf{Enc}(pk_T, \sigma; r).$$

- Return the ballot $\beta = (ct_c, ct_\sigma, \hat{ct}, \pi)$ on $\mathcal{BB}$.

– $\mathsf{Validate}(\mathcal{BB}, \beta) \rightarrow \top/\bot$: on input a ballot $\beta = (ct_v, ct_\sigma, \hat{ct}, \pi)$ and implicit input (pk_T, L) checks that i) $\hat{ct} \in L$, ii) β does not already appear in $\mathcal{BB}$, and iii) $\top \leftarrow \mathsf{Verify}(x, \pi)$. If any of the checks fail, it returns $\bot$ otherwise $\top$.

– $\mathsf{Append}(\mathcal{BB}, \beta) \rightarrow \mathcal{BB}$: on input a ballot $\beta = (ct_c, ct_\sigma, \hat{ct}, \pi)$ updates $\mathcal{BB}$ by appending the ballot β.

– $\mathsf{VerifyVote}(\mathcal{BB}, \hat{ct}, \sigma, c, \beta) \rightarrow \bot/\top$: on input a ballot $\beta = (ct_c, ct_\sigma, \hat{ct}, \pi)$, secret credential σ, public credential $\hat{ct}$, and vote option c checks that β is on $\mathcal{BB}$ and that $\mathsf{Validate}(\mathcal{BB}, \beta) = \top$. If any of the checks fail return $\bot$ otherwise $\top$.

– $\mathsf{Tally}(\mathcal{BB}, sk_T) \rightarrow (R, \Pi)$: on input $\mathcal{BB}$ and the decryption key sk_T apply cleansing and compute the election result as follows: Let $N = |\mathbb{L}|$, where $\mathbb{L}$ is the set of public credentials of registered voters on $\mathcal{BB}$. Let $L_{\hat{ct}}$ be a voter list of ordered ballots based on the submission time such that $\hat{ct} \in \mathbb{L}$, where $\beta_i = (ct_{c_i}, ct_{\sigma_i}, \hat{ct}, \pi_i)$ and $\beta_i \in L_{\hat{ct}}$. Filter the ballots as follows:

- Arrange the ballots with public credential $\hat{ct}$ in the order they appear on $\mathcal{BB}$ and store them in $L_{\hat{ct}}$.
- Initialise the cleansed list $L_{\hat{ct}_T} = [\hat{ct}, ct_0]$ for each $\hat{ct} \in \mathbb{L}$, where $ct_0 = \mathsf{Enc}(pk_T, 0; 0)$ denotes a null vote ballot.
- Run $\mathsf{Append}(\mathcal{BB}, L_{\hat{ct}}) \rightarrow \mathcal{BB}$ and $\mathsf{Append}(\mathcal{BB}, L_{\hat{ct}_T}) \rightarrow \mathcal{BB}$.
- If $\mathsf{Dec}(sk_T, \frac{ct_{\sigma_i}}{\hat{ct}}) = 1$, given $ct_{\sigma_i} \in \beta_i$ and $\hat{ct} \in L_{\hat{ct}_T}$, then compute $ct_{T_i} \xleftarrow{\$} \mathsf{ReEnc}(pk_T, ct_{c_i}; r_{T_i})$ and run $\pi_i \xleftarrow{\$} \mathsf{DisjProof}(x, \omega)$, where $x = (L_{\hat{ct}}, L_{\hat{ct}_T}, pk_T, ct_{T_i})$ and $\omega = (sk_T, r_{T_i})$

$$(x, \omega) \in R_{\text{eq}} \text{ iff } ct_{T_i} = \mathsf{ReEnc}(pk_T, ct_{c_i}; r_{T_i}) \wedge \mathsf{Dec}(sk_T, \frac{ct_{\sigma_i}}{\hat{ct}}) = 1$$

- Else compute $ct_{T_i} \xleftarrow{\$} \mathsf{ReEnc}(pk_T, ct_{T_{i-1}}; r_{T_i})$ and run $\pi \xleftarrow{\$} \mathsf{DisjProof}(x, \omega)$, where $x = (L_{\hat{ct}}, L_{\hat{ct}_T}, pk_T, ct_{T_i})$ and $\omega = (sk_T, r_{T_i})$

$$(x, \omega) \in R^{\text{Uneq}} \text{ iff } ct_{T_i} = \mathsf{ReEnc}(pk, ct_{T_{i-1}}; r_{T_i}) \wedge \mathsf{Dec}(sk_T, \frac{ct_{\sigma_i}}{\hat{ct}}) \neq 1$$

where $R_T = R^{\text{eq}} \vee R^{\text{Uneq}}$ and $i \geq 1$.

- Set (ct_{T_i}, π_i) as a last vote ballot in $L_{\hat{ct}_T}$ and run $\mathsf{Append}(\mathcal{BB}, L_{\hat{ct}_T}) \rightarrow \mathcal{BB}$.

Compute $T_i = \prod_{k=1}^{N} ct_k^i$, where $ct_k \in L_{\hat{ct}_T}$ denotes the last vote ciphertext. The tally t_i for candidate c_i is produced by decrypting T_i with the key sk_T. Compute the result $R = (t_1, \ldots, t_{|\mathbb{C}|})$ and Π, i.e. all Fiat-Shamir proofs including the proof for correct decryption of the result. Output (R, Π).

– $\mathsf{VerifyTally}(\mathcal{BB}, (R, \Pi)) \rightarrow \bot/\top$: on input $\mathcal{BB}$, result (R, Π), verifies the correctness of (R, Π) on $\mathcal{BB}$. If any of the checks fail return $\bot$ otherwise $\top$.

The scheme is organized in the following four phases.

Setup phase: The algorithm $\mathsf{Setup}(1^\lambda, (t,n), \mathbb{I}, \mathbb{C}) \rightarrow ((pk_T, sk_T), (pk_R, sk_R))$ allows, respectively, tallying servers and registrars to generate the key pairs (pk_T, sk_T)[3] and (pk_R, sk_R). The bulletin board $\mathcal{BB}$ is initialized with the lists of candidates $\mathbb{C}$, eligible voters identities $\mathbb{I}$, pk_R, and pk_T.

Registration phase: The registration authority registers the voter with $id \in \mathbb{I}$ to the election by running $\mathsf{Register}(1^\kappa, \mathbb{I}, sk_R, pk_R, pk_T) \rightarrow (\mathbb{L}, \{(\sigma_{id}, \hat{ct}_{\sigma_{id}})\}_{id \in \mathbb{I}}))$, which returns to each voter id, a secret credential $\sigma_{id} \in \mathbb{G}$, and a public credential $\hat{ct}_{\sigma_{id}}$. Then it sets a voting roll $\mathbb{L}$ of the voters' public credential (e.g. $(id, \hat{ct}_{\sigma_{id}})$). Finally, it executes $\mathsf{Append}(\mathcal{BB}, \mathbb{L}))$, which appends the signed $\mathbb{L}$ on $\mathcal{BB}$.

Voting phase: A voter makes a choice of a candidate from $\mathbb{C}$, selects a public credential $\hat{ct}_{\sigma_{id}}$ from $\mathbb{L}$ on $\mathcal{BB}$, encrypts and generates the proof for their ballots. The voter then submits their ballot to $\mathcal{BB}$ through an anonymous channel. The voters use NIZKP to prove the relation R_β. The voter proves in zero-knowledge that they know the vote and their choice is well-formed. The voter runs $\mathsf{VerifyVote}(\mathcal{BB}, \hat{ct}, \sigma, c, \beta) \rightarrow \bot/\top$ to verify their ballot and check that it is included in the bulletin board.

Trustees generate noise ballots identical to the voters' ballots to provide re-voting deniability and participation privacy. To generate a noise ballot, the trustee computes $\mathsf{Vote}(pk_T, \sigma', \hat{ct}, c') \rightarrow \beta'$ using a fake credential σ' generated by the trustees and a random candidate $c' \in \mathbb{C}$. Both voters and trustees can generate these noise ballots.

Tallying phase: The tallying servers execute $\mathsf{Tally}(\mathcal{BB}, sk_T) \rightarrow (R, \Pi)$. Anyone can verify the process of tallying and result R of tallying by executing $\mathsf{VerifyTally}(\mathcal{BB}, R, \Pi)$, which checks Π w.r.t. $\mathcal{BB}$ and R. The tally servers use disjunctive NIZKP of knowledge π to prove that $(x, \omega) \in R_T$, where $R_T = R^{\text{eq}} \vee R^{\text{Uneq}}$. The tally phase proceeds as follows.

- The tallying servers eliminate ballots that contain invalid proofs or unregistered public credentials.
- The tallying servers arrange the ballots based on their public credentials. The ballots with credential $\hat{ct}$ are stored in the list $L_{\hat{ct}}$.
- The tallying servers initiate a list called $L_{\hat{ct}_T} = [ct_0, \hat{ct}]$ corresponds to the original list $L_{\hat{ct}}$.
- The tallying servers generate a new vote ballot ct_T corresponding with $\beta_{\hat{ct}} = (ct_v, ct_\sigma, \pi) \in L_{\hat{ct}}$. If $\mathsf{Dec}(sk_T, \frac{ct_{\sigma_i}}{\hat{ct}}) = 1$, then ct_T is a re-randomiztion of $ct_c \in \beta$. The tally servers prove $(x, \omega) \in R^{\text{eq}}$ and simulate the relation R^{Uneq}. Otherwise, they re-randomize the last vote ciphertext in $L_{\hat{ct}_T}$. In this case, the they prove $(x, \omega) \in R^{\text{Uneq}}$ and simulate the R^{eq}.

[3] The secret key is generated in a distributed way, thus no single server learns the key.

- In homomorphic tallying, the final ballots of each tally list $L_{\hat{ct}_T}$ are multiplied, and the resulting ciphertext is decrypted. The tally servers provide proof of correct decryption. Furthermore, the steps of tallying with corresponding proofs are added to $\mathcal{BB}$ to ensure universal verifiability.

Algorithm 1 RealCR

Require: $\mathcal{A}, 1^\lambda, 1^\kappa, n_A, \mathbb{I}, \mathbb{C}, \mathcal{B}$

$\mathcal{BB} \leftarrow \emptyset$
$((pk_T, sk_T), (pk_R, sk_R)) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa, \mathbb{I}, \mathbb{C})$
$\{\sigma_i; i \in \mathbb{I}\}, \mathbb{L} \leftarrow \mathsf{Register}(1^\kappa, \mathbb{I}, (sk_R, pk_R), pk_T)$
$A \leftarrow \mathcal{A}(\mathbb{L})$ ▷ corrupt voters
$(j, c_\alpha) \leftarrow \mathcal{A}(\{\sigma_i; i \in A\})$ ▷ coerce a voter j who has vote intention c_α
if $|A| \neq n_A \vee j \notin \mathbb{I} \backslash A \vee c_\alpha \notin \mathbb{C} \cup \{\phi\}$ **then** Return 0
end if
$B \leftarrow \mathcal{B}(\mathbb{I} \backslash A, \mathbb{C})$
▷ samples a sequence of pairs (i, c_i) with $i \in \mathbb{I} \backslash A) \cup \{-i \in \mathbb{Z} | i > 0\}$ and $c_i \in \mathbb{C}$
for $(-i, *) \in B, i \in \mathbb{I}$ **do** $\sigma_i^f \leftarrow$ Fakecred() ▷ ballots sent for the voter i with invalid creds
end for
$b \xleftarrow{\$} \{0, 1\}$
$\sigma_j^f \leftarrow \sigma_j$
if $b == 1$ **then**
    Remove all $(j, *) \in B$
else
    Remove all $(j, *) \in B$ but the last, which is replaced by (j, c_α) if $c_\alpha \neq \phi$ and removed otherwise
    $\sigma_j^f \leftarrow$ Fakecred (σ_j)
end if
$\mathcal{A}(\sigma_j^f)$ ▷ A learns σ_j^f
for $(i, c_i) \in B$ (in this order) **do**
    $M \leftarrow \mathcal{A}(\mathcal{BB})$ ▷ cast ballots
    $\mathcal{BB} \leftarrow \mathcal{BB} \cup \{m \in M | \mathsf{Validate}(m, BB) = 1\}$
    $\mathcal{BB} \leftarrow \mathcal{BB} \cup \{\mathsf{Vote}(i, \sigma_i, c_i)\} \cup \{\mathsf{Vote}(i, \sigma_i^f, c_i)\}$
end for
$M \leftarrow \mathcal{A}(\mathcal{BB}$,"last honest ballot sent")
$\mathcal{BB} \leftarrow BB \cup \{m \in M | \mathsf{Validate}(m, BB) = 1\}$
$(R, \Pi) \leftarrow \mathsf{Tally}(BB, sk_T)$
$b' \leftarrow \mathcal{A}()$

5 Security

We prove that our scheme ensures coercion resistance under the DDH assumption in the random oracle model. We informally argue that our scheme provides ballot privacy and universal verifiability. Our scheme satisfies ballot privacy if the underlying ballot encryption scheme is non-malleable under chosen plaintext attack (NM-CPA) secure under the DDH assumption in the random oracle model [5]. Universal verifiability means that anyone can refer to the public bulletin board to verify the correctness of the tally result produced by tally servers.

Algorithm 2 IdealCR

Require: $\mathcal{A}, 1^\lambda, 1^\kappa, n_A, \mathbb{I}, \mathbb{C}, \mathcal{B}$

$A \leftarrow \mathcal{A}(1^\lambda, 1^\kappa)$ ▷ corrupt voters
$(j, c_\alpha) \leftarrow \mathcal{A}()$ ▷ coerce a voter j who has vote intention c_α
if $|A| \neq n_A \vee j \notin \mathbb{I} \backslash A \vee c_\alpha \notin \mathbb{C} \cup \{\phi\}$ **then** Return 0
end if
$B \leftarrow \mathcal{B}(\mathbb{I} \backslash A, \mathbb{C})$ ▷ sample a sequence of pairs (i, c_i) with $i \in \mathbb{I} \backslash A) \cup \{i \in \mathbb{Z} | i < 0\}$ and $c_i \in \mathbb{C}$
$b \xleftarrow{\$} \{0, 1\}$
if $b == 1$ **then**
    Remove all $(j, *) \in B$
else
    Remove all $(j, *) \in B$ but the last, which is replaced by (j, c_α) if $c_\alpha \neq \phi$ and removed otherwise
end if
$(c_i)_{i \in A}, c_\beta \leftarrow \mathcal{A}(|B_i|)_{i \in \mathbb{I}})$ ▷ $|B_i|$ is a number of pairs $(i, c_i) \in B$ and $(-i, c_i) \in B$ for voter i.
if $(b == 1) \wedge (c_\beta \in \mathbb{C})$ **then** ▷ c_β is the coercer vote for the voter j
    $B \leftarrow B \cup \{(j, c_\beta)\}$
    $B \leftarrow B \cup \{(i, c_i) | i \in A, c_i \in \mathbb{C}\}$
end if
$X \leftarrow result(cleanse(B))$
$b' \leftarrow \mathcal{A}(X)$
Return 1 if $b' == b$ else 0

5.1 Coercion Resistance

Our scheme ensures coercion resistance, meaning that a coercer should not be able to determine the validity of a voter's credential based on the election result. Additionally, the data published on the bulletin board during the voting and tally phases should not reveal whether a registered voter abstained from voting or revoted. It is assumed that the bulletin board is honest and that the communication channels between the voters and the public board are anonymous. The

registration is untappable and the registration authority and the tally servers are trusted for coercion resistance. We adapt the definition of coercion-resistance by Cortier et al. [8] that takes into account revoting and the addition of noise ballots by tally servers. The main modification is in the ideal experiment in which, instead of giving the total number of ballots, we give to the adversary $\mathcal{A}^I$ in the ideal experiment each voter's number of ballots (including noise ballots). In doing so, the adversary $\mathcal{A}^R$ in the real experiment and $\mathcal{A}^I$ have the same knowledge about the number of ballots regarding a public credential, and not about the number of ballots cast by a voter. For simplification, we consider a single honest tallier. For a quantitative analysis of the effectiveness of coercion resistance depending on noise generation we refer to the one done in [25].

Theorem 1. *Our scheme provides coercion resistance under the DDH assumption in the random oracle model.*

Proof. We construct a probabilistic polynomial time algorithm $\mathcal{S}$ which is given a DDH test instance to simulate the election protocol process for $\mathcal{A}^R$ (Algorithm 1). Our goal is to prove that the advantage of $\mathcal{A}^R$ in the real experiment is only negligibly higher than that of $\mathcal{A}^I$ (Algorithm 2). This is important because if there is a non-negligible advantage $\mathcal{A}^R$ over $\mathcal{A}^I$, then the simulator $\mathcal{S}$ can solve the DDH problem with a non-negligible probability.

The group of voters who have been corrupted is denoted by A, while the voters who have valid credentials are denoted by $S = \mathcal{I} \setminus A$. $\mathcal{B}$ denotes a distribution of pairs (i, c) where $c \in \mathbb{C}$ and i represents a voter with valid credentials. A voter with fake credentials or noise ballots, is represented by $(-i, c)$. The distribution $\mathcal{B}$ models a voter's abstention with $(i, *)$, revoting with i appears in several pairs, and $(-i, c)$ as a ballot with a fake credential for the voter i. The ballots with fake credentials can be added either by any participant. Note that both the real and ideal experiments assume that there are noise ballots in $\mathcal{B}$ regardless of how many ballots a voter casts.

The challenger of DDH problem constructs the test quadruple (g_1, g_2, h_1, h_2) based on the coin d. If $d = 1$, the simulator $\mathcal{S}$ receives a DDH instance; otherwise, a random instance is given. The simulator $\mathcal{S}$, which is given (g_1, g_2, h_1, h_2) and a distribution $B \in \mathcal{B}$, simulates the election process for $\mathcal{A}^R$. If the test instance is DDH instance, $\mathcal{A}^R$'s view will be the same as their view in the real coercion-resistance experiment. Otherwise, $\mathcal{A}^R$'s view will be the same as $\mathcal{A}^I$'s in the ideal coercion-resistance experiment. The election process is simulated as follows:

1. **Setup.** Given the test quadruple (g_1, g_2, h_1, h_2), the simulator $\mathcal{S}$ who controls the tally servers and registrar simulates the setup phase for $\mathcal{A}^R$. The simulator $\mathcal{S}$, who knows the secret key of the tally server $sk_T = (x_1, x_2)$ and the secret key of the registrar sk_R outputs the electoral roll $\mathbb{I}$, the candidate list $\mathbb{C}$, the register keys (pk_R, sk_R), and tally server keys (pk_T, sk_T), where $pk_T = (g_1^{x_1} g_2^{x_2}, (g_1, g_2))$.
2. **Registration.** The simulator $\mathcal{S}$ generates credentials $\{\sigma_i \in \mathbb{G}\}_{i \in \mathcal{I}}$, encrypts them using pk_T, stores them in $\mathbb{L}$, and publishes the registration list $\mathbb{L}$.
3. **Adversarial corruption.** $\mathcal{A}^R$ selects a set A of n_A voters to corrupt.

4. **Adversarial coercion.** $\mathcal{A}^R$ selects the voter that they want to coerce and also chooses the vote c_α as the coerced voter's vote.
5. **Validity check** The simulation terminates if any of the following happens: $|A| \neq n_A$, $j \notin \mathcal{I} \setminus A$, or $c_\alpha \notin \mathbb{C} \cup \{\varnothing\}$ where $\varnothing$ denotes the choice to abstain.
6. **Bit flip.** Given a distribution B, $\mathcal{S}$ flips a random bit $b \in \{0,1\}$. If $b = 0$ and $c_\alpha \neq \varnothing$, $\mathcal{S}$ eliminates all valid pairs of voter j except the last one, replaces the last pair $(j,c) \in B$ with (j,c_α), and gives a fake credential σ^f to $\mathcal{A}^R$. On the other hand, if $b = 1$, $\mathcal{S}$ removes all valid pairs $(j,c) \in B$ and gives the coerced voter's credential σ to $\mathcal{A}^R$.
7. **Adversarial ballot casting.** $\mathcal{A}^R$ casts some of the ballots with credentials of the corrupted voters, as well as that of the coerced voter j.
8. **Honest voter simulation.** $\mathcal{S}$ generates a ballot for all pairs in B, namely the honest voters and their noise ballots as follows:
   - $\mathcal{S}$ selects the public credential $\hat{ct}$ from the registered voting roll $\mathbb{L}$ corresponding with voter i.
   - $\mathcal{S}$ computes $\mathsf{Dec}(sk_T, \hat{ct}) = \sigma_i$. For a noise ballot, it generates a random fake credential σ_i^f. Note that $\mathcal{S}$ knows the encryption secret key sk_T.
   - Given test quadruple (g_1, g_2, h_1, h_2) and the encryption secret key $sk_T = (x_1, x_2)$, $\mathcal{S}$ computes new public key $\bar{pk}_T = (h_1^{x_1} h_2^{x_2})$ where (h_1, h_2) denotes corresponding the new generators.
   - Given the vote ciphertext $\bar{ct}_{c_i} = (h_1^{r_{c_i}}, h_2^{r_{c_i}}, c_i(\bar{pk}_T)^{r_{c_i}})$ and credential ciphertext $\bar{ct} = (h_1^{r_i}, h_2^{r_i}, \sigma_i(\bar{pk}_T)^{r_i})$, $\mathcal{S}$ simulates the zero-knowledge proof π_i using programmable random oracle, and returns $\bar{\beta}_i = (\bar{ct}_{c_i}, \bar{ct}_{\sigma_i}, \hat{ct}, \pi_i)$.

   Note that the simulated honest voter ballot, denoted by $\bar{\beta}_i$, is different from the actual ballot β_i, which is generated by an honest voter i using a public key $pk_T = (g_1^{x_1} g_2^{x_2}, (g_1, g_2))$. We will demonstrate the advantage of $\mathcal{A}^R$ to distinguish this difference is equivalent to determining whether (g_1, g_2, h_1, h_2) is a Decisional Diffie-Hellman (DDH) instance (i.e., $d = 1$) or not.
9. **Adversarial last ballot casting** Adversary $\mathcal{A}^R$ casts the final set of ballots corresponding with the corrupted voters and the coerced voter j.
10. **Tallying** $\mathcal{S}$ simulates an honest tallier using the secret key sk_T. The correctness of each step in this phase can be verified publicly.
    a. **Proof checking** $\mathcal{S}$ verifies the proof of the ballot cast by $\mathcal{A}^R$. It then initialises the lists $(L_{\hat{ct}}, L_{\hat{ct}_T})$ for each $\hat{ct} \in L$ and for the all cast ballots.
    b. **Checking credentials and generating tally ballots** $\mathcal{S}$ generates cleansed ballots for $L_{\hat{ct}_T}$. It uses sk_T to decrypt the comparison result of credentials. Then $\mathcal{S}$ generates a voting ciphertext corresponding with each ballot $\beta \in L_{\hat{ct}}$ and the relation R_T. It then simulates a NIZKP of knowledge for the relation R_T. Note that $\mathcal{S}$ re-randomizes the adversary ballots using public key pk_T and honest voters using public key $\bar{pk}_T$.
    c. **Decryption** $\mathcal{S}$ homomorphically adds the last ballots from $\{L_{\hat{ct}_T}\}_{\hat{ct} \in \mathcal{I} \setminus A}$ and decrypts the result using secret key sk_T. Similarly, the last ballots of $\{L_{\hat{ct}_T}\}_{\hat{ct} \in A}$ are added and decrypted. $\mathcal{S}$ computes the final result R and simulates the decrypting proof Π.
12. **Adversarial output.** Adversary $\mathcal{A}^R$ outputs a bit b'. $\mathcal{S}$ returns $d' = b'$ for the test instance of DDH problem.

If $d = d' = 2$, namely, $(g_1, g_2, h_1, h_2) = (g, g^a, g^b, g^{ab})$, the view of $\mathcal{A}^R$ from the simulation of the election process is indistinguishable from the real coercion resistance experiment $\mathbf{Exp}^{cr-real}$. Additionally, if $d = d' = 0$, the view of $\mathcal{A}^R$ from the simulation of the election is equal to the $\mathcal{A}^I$ in the ideal coercion resistance experiment $\mathbf{Exp}^{cr-ideal}$. The adversary $\mathcal{A}^I$ in experiment $\mathbf{Exp}^{cr-ideal}$ is given a list of numbers corresponding to the number of ballots per voter i and the final result. This means that the advantage of $\mathcal{S}$ in distinguishing the test (g_1, g_2, h_1, h_2) in DDH problem can be reduced to the advantage of $\mathcal{A}^R$ in $\mathbf{Exp}^{cr-real}$ over $\mathcal{A}^I$ in $\mathbf{Exp}^{cr-ideal}$. Formally,

$$Adv_{\mathcal{S}}^{DDH} = Adv_{\mathcal{A}^R}(\mathbf{Exp}^{cr-real}) - Adv_{\mathcal{A}^I}(\mathbf{Exp}^{cr-ideal}).$$

We now show that if $d = 1$ i.e. $(g_1, g_2, h_1, h_2) = (g, g^a, g^b, g^{ab})$. The view of $\mathcal{A}^R$ in the simulation process of $\mathbf{Exp}^{cr-real}$ is indistinguishable from the view of $\mathcal{A}^R$ in $\mathbf{Exp}^{cr-real}$. Let $\bar{pk}_T = (h_1^{x_1} h_2^{x_2}, (h_1, h_2))$ be the encryption public key used by $\mathcal{S}$, $\bar{ct}_c = \mathsf{Enc}(\bar{pk}_T, c; r)$, $c \in \mathbb{C}$. Since $d = 1$ we have $h_1 = g_1^b$, $h_2 = g_2^b$,

$$\mathsf{Enc}(\bar{pk}_T, c; r) = (h_1^r, h_2^r, c(\bar{pk}_T)^r) = (g_1^{br}, g_2^{br}, c(g_1^{x_1} g_2^{x_2})^{br}) = \mathsf{Enc}(pk_T, c; br)$$

The equation above shows that $\mathsf{Enc}(\bar{pk}_T, c; r)$ and $\mathsf{Enc}(pk_T, c; br)$ are different in the randomness. $\mathsf{Enc}(pk_T, c; br)$ and $\mathsf{Enc}(pk_T, c; t)$ have the same distribution of randomness for $t \xleftarrow{\$} \mathbb{Z}_p$, hence $Pr[\mathcal{S} = 1 | d = 1] = Adv_{\mathcal{A}^R}(\mathbf{Exp}^{cr-real})$, where $Adv_{\mathcal{A}^R}$ is defined as $|Pr[\mathbf{Exp}_{ES,\mathcal{A}^R}^{cr-real}(1^\lambda, 1^\kappa, \mathbb{I}, \mathbb{C}, n_c) = 1] - \frac{1}{2}|$.

We also prove that if $d = 0$ then $(g_1, g_2, h_1, h_2) = (g, g^a, g^b, g^z)$ where $z \xleftarrow{\$} \mathbb{Z}_p$. The view of $\mathcal{A}^R$ in the simulation process of $\mathbf{Exp}^{cr-real}$ can be presented by the view of $\mathcal{A}^I$ in $\mathbf{Exp}^{cr-ideal}$. Let $az' = z$ and $b + b' = z'$,

$$\begin{aligned}
\mathsf{Enc}(\bar{pk}_T, c; r) &= (h_1^r, h_2^r, c(\bar{pk}_T)^r) \\
&= (g_1^{br}, g_2^{zr}, c(g_1^{bx_1} g_2^{zx_2})^r) \\
&= (g_1^{br}, g_2^{zr+br-br}, c(g_1^{bx_1} g_2^{rzx_2 - rbx_2 + rbx_2}) \\
&= (g_1^t, g_2^t g_2^{zr-br}, c(pk_T)^t g_2^{rzx_2 - rbx_2}) \\
&= (g_1^t, g_2^t g_2^{t'}, c(pk_T)^t g_2^{t'}))
\end{aligned}$$

The random group element $g_3^{t'}$ completely hides vote c, and the adversary $\mathcal{A}^R$ does not learn anything from $\mathsf{Enc}(\bar{pk}_T, c; r)$. In this case, the view of $\mathcal{A}^R$ in the simulation process of $\mathbf{Exp}^{cr-real}$ can be compared to the view of $\mathcal{A}^I$ in $\mathbf{Exp}^{cr-ideal}$ where in the latter $\mathcal{A}^I$ is given a list containing the total number of ballots of each voter as $Pr[\mathcal{S} = 1 | d = 0] = Adv_{\mathcal{A}^I}(\mathbf{Exp}^{cr-ideal})$.

5.2 Ballot privacy

A voting scheme ensures ballot privacy if the information published during the election does not reveal how a voter voted. We informally show that our scheme achieves ballot privacy based on the following assumptions:

- the ballot encryption scheme with the NIZKP of knowledge is NM-CPA secure
- the registrar and the public bulletin board are honest

– up to t talliers and a subset of voters can be corrupted

A voting system can be vulnerable to replay attacks if an attacker can copy a voter's ballot from the bulletin board and then submit it as their own legitimate ballot, violating ballot privacy. Our ballot encryption scheme includes NIZKPoK, which prevents malleability during the voting phase. Since the registrar and the majority of talliers are honest, no legitimate ballot can be generated with honest voter credentials. In the tally phase, the manipulation of the voting ballot beyond re-randomization and nullifying defined in the relation R_T is prevented, as the talliers generate NIZKPoK for each ballot during the tally phase.

6 Verifiability

For verifiability, we consider the voting device being trusted and an adversary that can corrupt a subset of voters. First, we show that the final result is accurate and computed on the last ballots on the bulletin board. Assume that the adversary outputs a set of final ballots, the result R, and the corresponding proof Π at the tally phase. The *last* ballots from the tally processed lists namely $\{L_{\hat{ct}_T}\}_{\hat{ct}\in\mathbb{L}}$ form a set i.e., $T = \{ct_{T_1}, ct_{T_2}, \ldots, ct_{T_n}\}$. The set T, the result R, and the proof of valid decryption are published on the $\mathcal{BB}$. The homomorphic property of ElGamal and the soundness of proof of valid decryption verifies that the result R is obtained from the decryption of $\Pi_{i=1}^{n} ct_{T_i}$. We can conclude that $\mathsf{VerifyTally}(R, \Pi)$ only returns $\top$, when R is the correct result of $T = \{ct_{T_1}, ct_{T_2}, \ldots, ct_{T_n}\}$ on $\mathcal{BB}$.

We show that each ballot $\beta = (ct_c, ct_\sigma, \hat{ct}, \pi)$ on $\mathcal{BB}$ corresponds to one of the following sets: i) the ballots of the honest voters who have checked their ballots; ii) the ballots with fake credential σ^f; iii) the ballots of the corrupted voters.

The knowledge soundness of the proof π on the ballot β ensures that β is well-formed and valid. Thus, one can verify that the ballot β on $\mathcal{BB}$ is either a well-formed ballot with real or fake credential. Given that i) the registration authority is honest, ii) up to the threshold of tallier are dishonest, iii) DDH problem assumption holds, and iv) the knowledge-soundness of NIZKP proves that if $\beta \in \mathcal{BB}$ has a valid credential, it is cast by either honest voters or corrupted voter. In addition, the adversary cannot generate a new ballot with a legitimate credential except by using the corrupted voter's credential. The cleansing process on the tuples $(ct_c, ct_\sigma) \in L_{\hat{ct}}$ result in either a vote $c \in \mathbb{C}$ or $c = 0$. The tally servers generate a new pair (ct_T, π_T) corresponding to $(ct_c, ct_\sigma) \in L_{\hat{ct}}$ based on the relation $R_T = R^{\text{eq}} \vee R^{\text{Uneq}}$. The knowledge soundness of the proof π_T ensures that ct_T is either a re-randomized version of ct_c or the null vote with deterministic randomness ct_0. According to relation R_T, ct_T is a re-randomization of ct_c if the decryption of ct_σ and $\hat{ct}$ are equal, or ct_0 otherwise.

7 Performance and conclusion

Our prototype is written in Python [14]. We use the zksk library [23] for the implementation of the disjunctive zero-knowledge proofs. We run our experiments

Table 1: Tallying times (including cleansing) in our scheme.

nr. of ballots	1000			10000			100000			1000000		
nr. of candidates	2	4	10	2	4	10	2	4	10	2	4	10
tallying time	2s	3s	5s	18s	30s	50s	3m	5m	8m	30m	50m	1.3h

in a M2 MacBook Pro laptop with 16GB of RAM. Table 1 shows the average times to tally the results according to different numbers of ballots and candidates. The prototype implementation confirms that the tallying time is linear to the number of the ballots, which also includes the noise ballots. Since each voter list can be cleansed independently from the others, cleansing is fully parallelizable in our scheme. This means that our scheme can accommodate a very large number of noise ballots and still provide fast tallying. Better performance can be achieved by implementing the scheme in a more efficient language.

In conclusion, we presented a scheme that provides an efficient cleansing procedure for coercion-resistant voting. Since any participant can cast a ballot for any candidate, the scheme is subject to ballot flooding attacks. This is mitigated by fast cleansing and can be further mitigated by using slot times for casting ballots. With this work, we introduce a new trade-off between coercion-resistance and efficiency, and aim at stimulating the voting community to further investigate the implications of publicly cleansing-hiding in coercion-resistant voting.

References

1. Achenbach, D., Kempka, C., Löwe, B., Müller-Quade, J.: Improved coercion-resistant electronic elections through deniable re-voting. {USENIX} Journal of Election Technology and Systems ({JETS}) **3**, 26–45 (2015)
2. Aranha, D.F., Battagliola, M., Roy, L.: Faster coercion-resistant e-voting by encrypted sorting. Cryptology ePrint Archive (2023)
3. Araújo, R., Foulle, S., Traoré, J.: A practical and secure coercion-resistant scheme for remote elections. In: Dagstuhl Seminar Proceedings (2008)
4. Bernhard, D., Kulyk, O., Volkamer, M.: Security proofs for participation privacy and stronger verifiability for helios. Tech. rep., TU Darmstadt (2016)
5. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In: ASIACRYPT 2012. pp. 626–643. Springer (2012)
6. Clark, J., Hengartner, U.: Selections: Internet voting with over-the-shoulder coercion-resistance. In: Danezis, G. (ed.) Financial Cryptography and Data Security. pp. 47–61. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
7. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. In: 2008 IEEE Symposium on Security and Privacy (sp 2008). pp. 354–368 (2008)
8. Cortier, V., Gaudry, P., Yang, Q.: Is the jcj voting system really coercion-resistant? Cryptology ePrint Archive (2022)
9. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: CRYPTO '94 (1994)
10. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. European transactions on Telecommunications (1997)

11. Elgamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. Information Theory, IEEE Transactions on (1985)
12. EU Commission: Compendium of e-voting and other ICT practices. `https://commission.europa.eu/publications/compendium-e-voting-and-other-ict-practices_en`, 2023-12-06
13. Giustolisi, R., Garjan, M.S., Schuermann, C.: Thwarting last-minute voter coercion. In: 2024 IEEE Symposium on Security and Privacy (SP) (2024)
14. Giustolisi, R., Sheikhi, M.: Scheme prototype. `https://github.com/fgiustol/Evoteid24` (2024)
15. Grewal, G.S., Ryan, M.D., Bursuc, S., Ryan, P.Y.: Caveat coercitor: Coercion-evidence in electronic voting. In: IEEE Symposium on Security and Privacy (2013)
16. Haghighat, A.T., Dousti, M.S., Jalili, R.: An efficient and provably-secure coercion-resistant e-voting protocol. In: Privacy, Security and Trust (2013)
17. Haines, T., Mueller, J., Querejeta-Azurmendi, I.: Scalable coercion-resistant e-voting under weaker trust assumptions. In: ACM SAC'23 (2023)
18. Haines, T., Smyth, B.: Surveying definitions of coercion resistance. IACR Cryptol. ePrint Arch. p. 822 (2019)
19. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: ACM Workshop on Privacy in the Electronic Society (2005)
20. Kulyk, O., Teague, V., Volkamer, M.: Extending helios towards private eligibility verifiability. In: E-Voting and Identity - VoteID 2015 (2015)
21. Kusters, R., Truderung, T., Vogt, A.: A game-based definition of coercion-resistance and its applications. In: IEEE Computer Security Foundations (2010)
22. Locher, P., Haenni, R., Koenig, R.E.: Coercion-resistant internet voting with everlasting privacy. In: Financial Cryptography and Data Security: FC 2016 International Workshops. pp. 161–175. Springer (2016)
23. Lueks, W., Kulynych, B., Fasquelle, J., Le Bail-Collet, S., Troncoso, C.: Zksk: A library for composable zero-knowledge proofs. In: WPES (2019)
24. Lueks, W., Querejeta-Azurmendi, I., Troncoso, C.: VoteAgain: A scalable coercion-resistant voting system. In: USENIX Security (2020)
25. Müller, J., Pejó, B., Pryvalov, I.: Devos: Deniable yet verifiable vote updating. Proceedings on Privacy Enhancing Technologies (2024)
26. Rønne, P.B., Atashpendar, A., Gjøsteen, K., Ryan, P.Y.A.: Coercion-resistant voting in linear time via fully homomorphic encryption: Towards a quantum-safe scheme. CoRR **abs/1901.02560** (2019)
27. Schoenmakers, B., Tuyls, P.: Practical two-party computation based on the conditional gate. In: International conference on the theory and application of cryptology and information security. pp. 119–136. Springer (2004)
28. Smyth, B.: Athena: A verifiable, coercion-resistant voting system with linear complexity. Cryptology ePrint Archive (2019)
29. Spycher, O., Koenig, R., Haenni, R., Schläpfer, M.: Achieving meaningful efficiency in coercion-resistant, verifiable internet voting. Gesellschaft für Informatik eV (2012)
30. Spycher, O., Koenig, R., Haenni, R., Schläpfer, M.: A new approach towards coercion-resistant remote e-voting in linear time. In: Financial Cryptography and Data Security. pp. 182–189. Springer (2012)
31. Weber, S.G., Araujo, R., Buchmann, J.: On coercion-resistant electronic elections with linear work. In: ARES. pp. 908–916. IEEE (2007)