An Efficient and Provably-Secure Coercion-Resistant E-Voting Protocol

Alireza Toroghi Haghighat, Mohammad Sadeq Dousti, and Rasool Jalili

Data and Network Security Lab
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran

{*toroghi, dousti*}*@ce.sharif.edu, jalili@sharif.edu*

***Abstract*—We present an efficient and provably-secure e-voting protocol, which is a variant of the JCJ e-voting protocol (Juels et al., 2010). It decreases the total number of JCJ's operations from $O(n^2)$ to $O(n)$, where n is the number of votes or voters (whichever is the maximum). Note that since the operations under consideration are time-consuming (e.g., public-key encryption), the improvement is quite substantial. As a rough comparison, consider a nation-wide election with around ten million voters/votes. Assuming each operation takes one microsecond, and no parallelization is used, one can see a huge difference: our protocol tallies the votes in 10 seconds, while the JCJ protocol requires over 3 years to tally the votes. In order to achieve this level of efficiency, we change the ballot format and the tallying phase of the JCJ protocol. Moreover, we provide a complexity analysis and a detailed proof for coercion-resistance of our protocol.**

***Keywords*—E-Voting Protocol, Provable Security, Coercion-Resistance.**

I. Introduction

An appropriate e-voting protocol should provide many security and functional properties, such as correctness, verifiability, receipt-freeness, scalability, and robustness. Each proposed e-voting protocol attempted to satisfy some of these properties, see the survey of Sampigethaya and Poovendran [1].

Juels et al. [2] introduced a new security property for e-voting protocols, called *coercion-resistance*. The coercion-resistance property is stronger than privacy-preserving and receipt-freeness properties. If an e-voting protocol is coercion-resistant, it will be privacy-preserving and receipt-free as well. Moreover, it will prevent other attacks to the privacy of voters such as randomization, forced-abstention, and simulation attacks [2].

Juels et al. provided a formal definition for the coercion-resistance, and presented an e-voting protocol, called *JCJ*. They proved that the *JCJ* protocol is coercion-resistant according to their formal definition. To define coercion-resistance, they described and characterized a new adversary, whose aim is to coerce a voter.

The JCJ protocol has many useful properties, but it has an important shortcoming: inefficiency. In the tallying phase of the JCJ protocol, tallier authorities must perform PET [1] operations on the cast ballots, whose number is quadratic in the number of votes/voters. Although many solutions have been proposed to solve this problem (e.g., [7], [8], and [10]) to the best of our knowledge, there is no comprehensive and provably secure solution (see Table I). Note that providing security proof is very important, because several protocols that claimed to be coercion-resistant and did not provide a proof, were found to be flawed, e.g., [7], [4], and [8].

Our contribution is to present a provably secure variant of the JCJ protocol which needs a *linear* number of PET operations in the tallying phase. In order to decrease the running time of the JCJ protocol, we change the ballot format and the tallying phase. We propose that a voter adds the encryption of his credential—given to him in the registration phase—to the ballot. We also add a decryption sub-phase to the tallying phase, to remove duplicate votes. In order to prove that these changes will not violate coercion-resistance property, we provide a detailed proof for coercion-resistance of the proposed protocol in the JCJ model.

The rest of this paper is organized as follows. Section II provides a discussion of the related works. Section III describes the model and definition used in the proof. For self-containment, we describe the JCJ e-voting protocol in Section IV. The proposed e-voting protocol is presented in Section V. Section VI presents the proof of coercion-resistance, and Section VII concludes the paper and describes the future work.

II. Related Works

Since the seminal work of Juels et al. [12], [2], many coercion-resistant e-voting protocols have been proposed (see Table I). Most of these protocols are either an improvement on or similar to the JCJ protocol. Acquisti [3], Schweisgut [4], and

[1]PET is a cryptographic primitive in a threshold cryptosystem, which receives two ciphertexts as input, and determines whether their corresponding plaintexts are equal (see Section IV-A)

TABLE I: Comparison of several coercion-resistant e-voting protocols.

Protocol \ Property	Improvement of JCJ	Linear # of Operations	Coercion-Resistance Broken	Provably Coercion-Resistant	Assumptions		
					Untappable Channel	Anonymous Channel	Other
Acquisti [3]						✓	
Schweisgut [4]		✓	Yes [5]			✓	✓[a]
Selections [6]		✓		✓	✓	✓	✓[b]
JCJ [2]	—			✓	✓	✓	
Smith [7]	✓	✓	Yes [8]		✓	✓	
Weber et al. [8]	✓	✓	Yes [9]		✓	✓	
Araújo et al. [9]	✓	✓			✓	✓	
Araújo et al. [10]	✓	✓			✓	✓	
Spycher et al. [11]	✓	✓			✓	✓	
This paper	✓	✓		✓	✓	✓	

[a]Tamper-resistance hardware.
[b]Human behavioral, e.g., the voters cannot memorize or copy the randomness generated in the registration phase, and they will shred their preparation sheets.

Clark and Hengartner [6] proposed coercion-resistant e-voting protocols, which are similar to the JCJ protocol. Smith [7], Weber et al. [8], Araújo et al. [9], [10], and Spycher et al. [11] proposed improved versions of the JCJ protocol.

Acquisti [3] proposed a coercion-resistant e-voting protocol that allows *write-in ballots* (i.e., candidates are not predefined, and voters can vote for anyone they prefer). His protocol is similar to the preliminary version of the JCJ protocol [12], in that both rely on a number of authorities to issue the credentials of voters, and both tally election results by making a blind comparison between the list of encrypted votes and the list of encrypted credentials. A main difference is that Acquisti's protocol allows voters to encrypt a combination of their votes together with their credential shares.

Schweisgut [4] proposed a coercion-resistant e-voting protocol that relies on a tamper-resistant hardware. In this protocol, each voter has exactly two credentials, both of which are stored on the tamper-resistant hardware. One of the credentials is the valid credential of the voter, while the other one is his fake credential. Araújo et al. [5] described an attack on Schweisgut protocol that allows an adversary to check whether a voter submitted to coercion.

Clark and Hengartner [6] proposed a coercion-resistant e-voting protocol, called *Selections*. It shares several similarities with the JCJ protocol:

- In the registration phase of *Selections*, voters use an untappable channel to choose a (specific) password, and an encryption of this password is sent to a public board.
- In the voting phase of *Selections*, voters cast their votes consisting of commitments of their asserted passwords, re-encryptions of their passwords, encryptions of their votes, and some zero-knowledge proofs.
- The tallying phase of *Selections* consists of eliminating votes with invalid zero-knowledge proofs, eliminating duplicate votes, applying votes to a verifiable mix-net, and checking the validity of the asserted passwords.

While similar to the JCJ protocol, the *Selections* protocol needs untappable channel and anonymous channel, it uses some other physical assumptions as well. For instance, the *Selections* protocol assumes that a voter cannot memorize or copy the randomness generated in the registration phase, and he will shred his preparation sheet. That said, the *Selections* protocol has several desirable properties, such as linear overhead, revocation of voters, and password-based authentication.

Smith [7] proposed a variant of the JCJ protocol to mitigate the inefficiency problem of the JCJ. However, Smith's protocol is ambiguous in some aspects, and his new comparison method (that is the alternative of the PET method of the JCJ) may fail in some special cases, and cause inaccurate election results [8].

Weber et al. [8] proposed another variant of the JCJ protocol with $O(n)$ operations, which is based on the Smith's ideas [7]. They proposed a new method of comparison, which is based on Pedersen's distributed key generation [13]. However, Araújo et al. [9] found a flaw in the coercion-resistance property of this protocol.

Araújo et al. [9], [10] proposed two other e-voting protocols based on the JCJ protocol, with $O(n)$ operations. They suggested the use of approaches based on group signatures. While similar to the JCJ protocol, the voters obtain their credentials at the registration phase, there is no public voter roll. Spycher et al. [11] asserted that protocols based on this approach have an inherent weakness: Due to the lack of a public voter roll, a colluding majority of registrars can issue valid credentials to a great number of illegitimate parties.

Spycher et al. [11] proposed another e-voting protocol based on the JCJ protocol. They removed duplicate votes using the linear-time method of Smith [7] and Weber et al. [8]. Specifically, to check credentials of voters in linear-time, each voter should indicate the voter roll entry with which the vote's credential is to be compared. Spycher et al. [11] did not provide a proof for coercion-resistance.

Table I compares the aforementioned protocols with our proposed protocol in several aspects. As is shown in the table, our protocol achieves the $O(n)$ bound, does not rely on assumptions such as tamper-proof hardware or voter behavior, and most importantly includes a detailed proof for the coercion-resistance property.

III. Model and Definition

We use the election model and definition of coercion-resistance provided by Juels et al. [14]. Let us describe them in brief.

A. System Entities & Parameters

The election model consists of several entities:

1) **Registrars:** R is the set of n_R registrar authorities, who jointly generate the key pairs of voters.
2) **Talliers:** T is the set of n_T tallier authorities, who jointly process ballots cast by voters and compute the final election results.
3) **Voters:** V is the set of n_V voters, who participate in an election administered by T and R.
4) **Candidates:** C is the set of n_C candidates, for which the ballots are cast.

The security parameter of the system is denoted by k, which is implicitly given to all entities/functions of the system. Moreover, we assume that all entities in the system have memory. Therefore, once given an input, they can memorize that input to use it in a later stage.

B. Attack Model

The model of the election protocol consists of five phases: setup, registration, voting, tallying, and verification. The adversary is assumed to have an *a priori* knowledge on the distribution of the votes that will be cast by the voters which is denoted by $\mathcal{D}$. The adversary can perform the following actions on the election protocol:

- **Corrupt registrars and talliers:** Before the *setup* phase, the adversary may corrupt a minority of registrars and talliers. This is denoted by $CR \leftarrow A(R, \text{"corrupt } R\text{"})$ and $CT \leftarrow A(T, \text{"corrupt } T\text{"})$, where CR and CT denote the set of corrupted registrars and talliers, respectively.
- **Corrupt voters:** After the *setup* phase, the adversary selects a subset of voters to corrupt. This is denoted by $CV \leftarrow A(V, \text{"corrupt } V\text{"})$, where CV denotes the set of corrupted voters.
- **Coerce a single voter:** After the *registration* phase, the adversary coerces a single voter which is denoted by $(j, \beta) \leftarrow A\big(\{sk_i\}_{i \in CV}, \text{"coerce"}\big)$. The adversary has to output the ID of the voter he wants to corrupt (j) and a ballot value β. If the coerced voter does *not* submit to coercion, he *must* vote for β.
- **Cast ballots:** After all honest voters cast their ballots, the adversary reads the contents of the bulletin board BB, and cast her own ballots.

The validity of the adversary's choices are governed by the *check_validity* function, which makes the following checks:

- Only a minority of registrars and talliers can be corrupted: $|CR| < n_R/2$ and $|CT| < n_T/2$.
- The number of corrupted voters, $|CV|$, is n_A.
- The coerced voter j is a valid voter and is not corrupted: $j \in V \setminus CV$.
- The vote selected for the coerced voter is valid: $\beta \in C \cup \{\emptyset\}$.

If any of the above validity checks fails, the adversary will be deemed unsuccessful in his attack.

Exp. 1 $Exp^{c\text{-}resist}_{ES,A}(k, n_A, V, R, T, C)$

```
1: $CR \leftarrow A(R, \text{"corrupt } R\text{"})$ and $CT \leftarrow A(T, \text{"corrupt } T\text{"})$
2: $setup\_r(R)$ and $setup\_t(T)$
3: $CV \leftarrow A(V, \text{"corrupt } V\text{"})$;
4: for $i \leftarrow 1$ to $n_V$ do $(sk_i, pk_i) \leftarrow register(SK_R, i)$;
5: $(j, \beta) \leftarrow A(\{sk_i\}_{i \in CV}, \text{"coerce"})$;
6: if $check\_validity(\text{outputs of } A) = \text{false}$ then return 0;
7: $b \leftarrow_R \{0,1\}$;
8: if $b = 0$ then
9:     $\tilde{sk} \leftarrow fakekey(PK_T, sk_j, pk_j)$;
10:    $BB \Leftarrow vote(sk_j, PK_T, n_C, \beta)$;
11: else $\tilde{sk} \leftarrow sk_j$;
12: $BB \Leftarrow vote(\{sk_i\}_{i \notin CV \cup \{j\}}, PK_T, n_C, \mathcal{D})$;
13: $BB \Leftarrow A(\tilde{sk}, BB, \text{"cast ballots"})$;
14: $(\mathbf{X}, P) \leftarrow tally(SK_T, BB, n_C, \{pk_i\}_{i=1}^{n_V})$;
15: $b' \leftarrow A(\mathbf{X}, P, \text{"guess } b\text{"})$;
16: if $b' = b$ then return 1, else return 0;
```

C. Formal Definition of Coercion-Resistance

In order to provide a formal definition for coercion-resistance, Juels et al. defined a game between the adversary and the participants of the election protocol, called *c-resist* (Exp. 1). They defined the coercion-resistance as a property of a voting system which holds if and only if the advantage of the adversary in the *c-resist* experiment is at most negligibly more than the advantage of a weaker adversary. In order to define this weaker adversary, Juels et al. defined an idealized version of *c-resist*, called *c-resist-ideal* (Exp. 2), where the advantage of the adversary in coercing a voter is negligible (*by definition*).

They proved their proposed protocol to be coercion-resistant, by showing that the advantage of any efficient adversary A in winning the *c-resist* experiment is at most negligibly more than the advantage of any efficient adversary A' in the *c-resist-ideal* experiment. To this end, they presented a simulator S, which could simulate the environment for the adversary A. The simulator gets as input a quadruple, and then simulates the e-voting protocol for A. If the simulator's input

Exp. 2 $Exp^{c\text{-}resist\text{-}ideal}_{ES,A}(k, n_A, V, R, T, C)$

```
1: $CR \leftarrow A'(R, \text{"corrupt } R\text{"})$ and $CT \leftarrow A'(T, \text{"corrupt } T\text{"})$
2: $setup\_r(R)$ and $setup\_t(T)$
3: $CV \leftarrow A'(V, \text{"corrupt } V\text{"})$;
4: for $i \leftarrow 1$ to $n_V$ do $(sk_i, pk_i) \leftarrow register(SK_R, i)$;
5: $(j, \beta) \leftarrow A'(\text{"coerce"})$;
6: if $check\_validity(\text{outputs of } A') = \text{false}$ then return 0;
7: $b \leftarrow_R \{0,1\}$;
8: if $b = 0$ then $BB \Leftarrow vote(sk_j, PK_T, n_C, \beta)$;
9: $\tilde{sk} \leftarrow sk_j$;
10: $BB \Leftarrow vote(\{sk_i\}_{i \notin CV \cup \{j\}}, PK_T, n_C, \mathcal{D})$;
11: $BB \Leftarrow A'(\tilde{sk}, |BB|, \{sk_i\}_{i \in CV}, \text{"cast ballots"})$;
12: $(\mathbf{X}, P) \leftarrow ideal\_tally(SK_T, BB, n_C, \{pk_i\}_{i=1}^{n_V})$;
13: $b' \leftarrow A'(\mathbf{X}, |BB|, \text{"guess } b\text{"})$;
14: if $b' = b$ then return 1, else return 0;
```

constitutes a valid DDH quadruple, the simulated environment will mimic the *c-resist* experiment; otherwise, it will mimic the *c-resist-ideal* experiment. It is then proven that A cannot distinguish between the simulated and real environments with non-negligible advantage. Finally, they proved that the success probability of the simulator in solving the DDH challenge is equal to the "difference between the advantage of A in *c-resist* experiment, and the advantage of A' in *c-resist-ideal* experiment." As a result, if A's advantage is non-negligibly more than A''s advantage, the simulator can solve the DDH problem with non-negligible probability, contrary to the assumption that DDH is hard.

Let us briefly sketch some important functions used in these experiments:

- *setup_r* (*setup_t*): This function denotes a distributed key generation among the registrars (talliers), at the end of which all registrars (talliers) will get a single public key PK_R (PK_T), and a share of the corresponding secret key SK_R (SK_T). Moreover, in the *setup_r*, registrars will jointly generate and sign the candidate slate.
- *check_validity*: This functions verifies the validity of the output produced by the adversary. The details are described in Section III-B.
- *fakekey*: On input the public key of the talliers PK_T, and the key pair of the coerced voter, generates a fake secret key $\tilde{sk}$.
- *tally*: This function denotes a distributed protocol among the talliers, in which the election results ($\mathbf{X}$) is jointly computed. The talliers output a proof of correctness (P) as well (*ideal_tally* is described at the followings).

In the Exp. 1, when the adversary coerces a voter, a coin is flipped, whose outcome is represented by a bit b. If $b = 0$, the coerced voter (j) evades coercion by giving the adversary a fake secret key and voting for his favorite candidate, β. Otherwise, if $b = 1$, the coerced voter submits to coercion by giving the adversary his real secret key, and by abstention. Looking forward, the adversary will win the *c-resist* experiment if she can correctly guess the bit b at the end of the election.

Exp. 2 is different from Exp. 1 in several ways. For brevity, let us denote by x.y the yth line of Experiment x.

- The adversary A' in Exp. 2 cannot use the secret keys of voters in her control (CV) to infer any additional information, or to perform an adaptive attack (cf. 2.5 with 1.5).
- Exp. 2 tallies ballots by *ideal_tally* function, while Exp. 1 uses the *tally* function (cf. 2.12 with 1.14). The *ideal_tally* function tallies ballots cast by honest parties normally, but it tallies ballots cast by A' in a special manner. More specifically, this function tallies all ballots cast by A' with secret keys of corrupted voters normally, but discards all ballots cast by A' with secret keys of honest voters. Moreover, this function tallies ballots cast by A' with $\tilde{sk}$ in the following manner: if $b = 0$, then *ideal_tally* discards these ballots; otherwise it counts them normally.
- The adversary A' in Exp. 2 never gets to see the contents of the bulletin board: she merely sees garbled strings. To model this, we only gave the size of the bulletin board ($|BB|$) to A' (cf. 2.11 with 1.13 and cf. 2.13 with 1.15).

Note that similar to the normal experiment, in the ideal experiment the coerced voter may submit to or evade the coercion according to the flipped coin. However, in the ideal experiment, the coerced voter can safely give away his real secret key to the adversary. This is because the *ideal_tally* is aware of the result of the flipped coin.

IV. The JCJ Protocol

In this section, we provide a brief description of the JCJ protocol [14] to keep the discussion self-contained.

A. Building Blocks

Juels et al. employed the following building blocks in their proposed protocol:

Modified ElGamal: The JCJ protocol relies on a modified ElGamal cryptosystem proposed by Juels et al. [2]. Assume that $\mathbb{G}$ is a cyclic group of prime order q, and let g_1 and g_2 be two elements in $\mathbb{G}$. Moreover, let x_1 and x_2 be random elements in $\mathbb{Z}_q$, and define $h = g_1^{x_1} g_2^{x_2}$. The triple (g_1, g_2, h) constitute the public key pk, and the pair (x_1, x_2) denotes the secret key sk.
The encryption of a message m under pk is computed as $E(m) = (g_1^r, g_2^r, mh^r)$, where r is a random element of $\mathbb{Z}_q$. To decrypt this message, the owner of the secret key should compute $m = mh^r * ((g_1^r)^{x_1} * (g_2^r)^{x_2})^{-1}$. It is straightforward to prove the semantic security of this cryptosystem under the Decisional Diffie-Hellman (DDH) assumption in $\mathbb{G}$ [14].
The JCJ protocol requires a threshold variant of the cryptosystem. Cramer et al. [15] described an ElGamal threshold cryptosystem.

Plaintext Equivalence Test: The JCJ protocol requires a cryptographic primitive called *plaintext equivalence test* (PET), which operates on two ciphertexts in a threshold cryptosystem. The output is a single bit indicating whether the input ciphertexts correspond to the same plaintext. Efficient realizations of PET exist [16]. The JCJ protocol requires a PET which satisfies the property of *public verifiability* as well.

Mix-net: The JCJ protocol requires a re-encryption mix-net that operates on Modified ElGamal encrypted messages, and also require it to be *public verifiable*. Mix-net proposed by Neff [17] is a good choice for the JCJ protocol.

NIZK proofs: A non-interactive zero-knowledge (NIZK) proof (of knowledge) consists of a single string transmitted by a prover to a verifier, which convinces the verifier that a statement is true, and does not leak any additional knowledge. NIZK proofs can be obtained in the common reference string (CRS) and random oracle (RO) models.

B. E-Voting Protocol

This section describe the JCJ e-voting protocol. In the description of the protocol, everywhere we say that registrars (or talliers) do some operation, we mean that the majority of them agree on the operation and its input(s), and do it jointly.

Setup: R and T generate their secret/public key pairs (SK_R, PK_R) and (SK_T, PK_T). They generate their keys using a distributed key generation protocol such as [18], and publish the public keys.

Registration: After verifying the eligibility of a voter V_i, the registrars generate a credential[2] for V_i, denoted as σ_i, which is a random member of $\mathbb{G}$. The credential is jointly generated by the registrars using a distributed key generation protocol (e.g., [18]).

Next, R assures the voter V_i that $\hat{E}_{PK_T}(\sigma_i)$ is an encryption of his credential σ_i using a *designated-verifier proof* π_i [19]. Finally, R appends $\hat{E}_{PK_T}(\sigma_i)$ to the voter roll $\mathbf{L}$, which is a public bulletin board digitally signed by R.

Candidate-slate release: R publishes the candidate slate C, which contains the names and unique identifiers of all n_C candidates, encoded as elements of $\mathbb{G}$, and digitally signed by R.

Voting: The voter V_i casts his vote for candidate C_j by casting $(E_{PK_T}(C_j), E_{PK_T}(\sigma_i), Pfs)$ via an *anonymous* channel to bulletin board. Let us dissect this vote:

$E_{PK_T}(C_j)$ is the encryption for the candidate choice of voter C_j under the public key PK_T (a Modified ElGamal cryptosystem is used for all encryptions). *Pfs* represents three NIZK proofs. The first two NIZKs prove that the voter knows C_j and σ_i respectively, and the third one proves that $C_j \in C$.

Tallying: After the end of the voting time, T starts to compute the election results as follows:

1) **Verifying NIZK proofs:** T verifies the NIZK proofs associated with each ballot and discards ballots with invalid proofs. Ciphertexts of the remaining ballots are divided into two lists; $\mathbf{A_1}$ denotes the list of encrypted candidate choices, and $\mathbf{B_1}$ denotes the list of encrypted credentials.
2) **Eliminating duplicate votes:** By applying pairwise PET operations on all encrypted credentials in the $\mathbf{B_1}$ list, T removes duplicate votes according to the pre-defined policy, e.g., "last-vote-count" policy. Let $\mathbf{A_2}$ and $\mathbf{B_2}$ be the resulting lists.
3) **Mixing:** T Sends the $\mathbf{A_2}$ and $\mathbf{B_2}$ lists to a verifiable mix-net, using a unique secret permutation for both lists. Let $\mathbf{A_3}$ and $\mathbf{B_3}$ be the resulting mixed lists. T also sends the voter roll $\mathbf{L}$ to another verifiable mix-net; let $\mathbf{L}'$ denote the resulting list.
4) **Checking credentials:** T compares each element of the $\mathbf{B_3}$ list to all elements of $\mathbf{L}'$ list using PET, in order to find a match. If there is no match for a ballot, T will discard it. Let $\mathbf{A_4}$ and $\mathbf{B_4}$ be the resulting lists.
5) **Computing result:** T decrypts all ciphertexts of $\mathbf{A_4}$ list and computes the election results.

All lists generated above are publicly readable. In other words, all participants in the system (as well as the adversary) can read the lists $\mathbf{A_1}, \ldots, \mathbf{A_4}$, $\mathbf{B_1}, \ldots, \mathbf{B_4}$, $\mathbf{L}$, and $\mathbf{L}'$.

V. Proposed Modifications

Our proposed e-voting protocol is a variant of the JCJ protocol, and requires the same building blocks. The tallying phase of the JCJ protocol has two sub-phases with PET operations, each of which has quadratic complexity. First one is the "eliminating duplicate votes" sub-phase and second one is the "checking credentials" sub-phase.

To overcome the first complexity barrier, we propose decrypting of all encrypted credentials before checking credentials sub-phase and encrypting them again. To overcome the second complexity barrier, we suggest the following: each voter specifies the entry of the voter roll $\mathbf{L}$, with which his credential should be compared. The two modifications are carefully applied, so that the desirable properties of the JCJ protocol is preserved for all voters. We briefly describe why this is the case (Section V-A), and also prove the coercion-resistance of our proposed protocol in Section VI.

As pointed above, the modifications are limited to the ballot format and the tallying phase. Therefore, we only describe the voting and tallying phases of the proposed protocol.

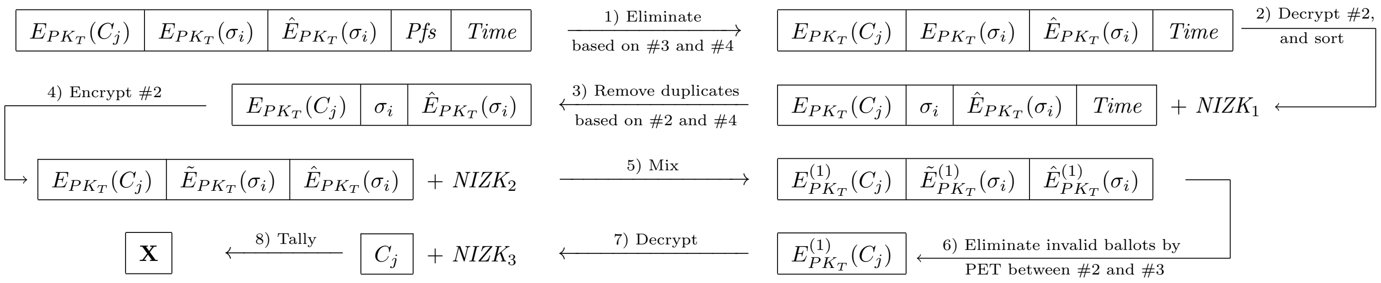
Voting: The voter V_i casts his ballot by appending $\big(E_{PK_T}(C_j), E_{PK_T}(\sigma_i), \hat{E}_{PK_T}(\sigma_i), Pfs\big)$ via an *anonymous* channel to the bulletin board. There is only one modification here: the addition of the third component $\hat{E}_{PK_T}(\sigma_i)$. It is the entry of the voter roll $\mathbf{L}$ which corresponds to the credential of the voter[3].

Tallying: After the voting deadline expires, T starts to compute the election results as depicted in Fig. 1. Below, we describe each sub-phase in more details:

1) **Eliminate invalid ballots based on columns #3 and #4:** Similar to the JCJ protocol, T verifies the NIZK proofs *Pfs* associated with each ballot (column #4), and discards all ballots with invalid proofs. Moreover, T removes ballots whose third column $\hat{E}_{PK_T}(\sigma_i)$ is not in $\mathbf{L}$.
2) **Decrypt column #2, and sort:** T decrypts encrypted credentials $E_{PK_T}(\sigma_i)$ in column #2, and sorts the list of ballots based on the decrypted values. T also computes a NIZK proof to prove that it decrypted column #2 correctly.
3) **Remove duplicates based on columns #2 and #4:** If multiple ballots with the same credential exist in column #2, only the last one is retained, based on the timestamp in column #4. Then T discards column #4.
4) **Encrypt column #2:** T encrypts the credentials in column #2 again, which are denoted by $\tilde{E}_{PK_T}(\sigma_i)$. Moreover, T computes a NIZK proof to prove that $\tilde{E}_{PK_T}(\sigma_i)$ is a valid ciphertext on σ_i.
5) **Mix:** T delivers the list of ballots to a (verifiable) mix-net, which applies a secret unique permutation to all three columns of the ballot.
6) **Eliminate invalid ballots by PET between columns #2 and #3:** T applies a (verifiable) PET to columns #2 and #3 of each ballot, and only retains the column #1 of those ballots whose columns #2 and #3 correspond to identical credentials.
7) **Decrypt:** T decrypts the column #1, and computes a NIZK proof to prove that it performed decryption correctly.
8) **Tally:** Finally, T tallies the results $\mathbf{X}$, and outputs it.

[2]In the JCJ protocol, the credential of a voter is a secret key, and he has no corresponding public key.

[3]In order to find the most recent version of duplicate votes of a voter, the bulletin board keeps a timestamp along with each ballot. However, the timestamp should be removed after removing duplicate votes.



1) Eliminate based on #3 and #4
2) Decrypt #2, and sort
3) Remove duplicates based on #2 and #4
4) Encrypt #2
5) Mix
6) Eliminate invalid ballots by PET between #2 and #3
7) Decrypt
8) Tally
$E_{PK_T}(C_j)$ $E_{PK_T}(\sigma_i)$ $\hat{E}_{PK_T}(\sigma_i)$ Pfs Time
$+ NIZK_1$
$+ NIZK_2$
$+ NIZK_3$
$E^{(1)}_{PK_T}(C_j)$ $\tilde{E}^{(1)}_{PK_T}(\sigma_i)$ $\hat{E}^{(1)}_{PK_T}(\sigma_i)$
C_j
X

Fig. 1: Diagram of the tallying phase. Each data structure represents a single ballot, whose i^{th} column is denoted by #i.

The intermediate computations of T, as depicted in Fig. 1, is posted to the bulletin board as the proof of correctness P.

A. *Discussion*

In this section, we compute the complexity of the modified protocol, and (intuitively) show that it preserves the correctness and universal verifiability properties.

Let n be the number of ballots. The modified voting phase has no overhead compared to the corresponding phase in the JCJ protocol. Let us now consider the sub-phases of the modified tallying phase. Each sub-phase requires $O(n)$ operations, including sub-phase 2 (sorting) which can be performed by $O(n)$ operations using algorithms such as the *radix sort*. (One can also remove duplicate votes in linear time by using a hash table and bypass the sorting sub-phase.) Therefore, the total complexity of our protocol is linear, compared to the quadratic complexity of the JCJ protocol.

To prove the *correctness*, we observe that if the majority of the authorities are honest, every sub-phase in the modified protocol is carried out correctly.

We now attend to the *universal verifiability* property:

1) Sub-phases 1,3, and 8 of the tallying phase can be verified by any observer. In fact, these sub-phases can be performed by a polynomial-time algorithm, without requiring the secret key of the talliers.
2) In sub-phases 2, 4, and 7 of the tallying phase, talliers will provide NIZK proofs for decryption or encryption operations, so these phases are also verifiable.
3) Sub-phase 5 (resp., 6) is universally verifiable due to using verifiable mix-nets (resp., verifiable PET).

VI. Proving Coercion-Resistance

In this section, we provide a proof for coercion-resistance of our proposed protocol. To prove coercion-resistance of our protocol, we assume that Decisional-Diffie Hellman (DDH) is a hard problem in the cyclic group $\mathbb{G}$ of prime order q.

A. *Proof Overview*

Based on definition provided in Section III, in order to prove coercion-resistance of an e-voting protocol, we should prove that the success probability of A in guessing b in Exp. 1 is at most negligibly more than the success probability of A' in guessing b in Exp. 2. The proof strategy is as follows: we construct a probabilistic polynomial time algorithm S that simulates the election protocol operations for the adversary A in Exp. 1. S takes two inputs:

- A test quadruple (g_1, g_2, h_1, h_2), which is either a DDH instance, or a random instance. More specifically, let g be a generator of $\mathbb{G}$, and assume that $g_1 = g$, $g_2 = g^a$, and $h_1 = g^b$, where $a, b \in_R \mathbb{Z}_q$. The quadruple is a DDH instance if $h_2 = g^{ab}$, and it is a random instance if $h_2 = g^c$ for some $c \leftarrow_R \mathbb{Z}_q$.
- A sequence $\mathbf{W}$, consisting of pairs (V_i, C_j) of voters and candidates. $\mathbf{W}$ is a list which denotes for which candidate(s) C_j each voter V_i votes. The votes in $\mathbf{W}$ must be distributed according to adversary's prior knowledge, $\mathcal{D}$.

The idea of the proof is that S constructs the adversary's view based on the test quadruple and the input ballots. If the test quadruple is a DDH instance, the adversary's view will be identical to her view in the *c-resist* experiment. Otherwise, the adversary's view will include garbled values, identical to A''s view in the *c-resist-ideal* experiment. We will prove that if A has non-negligible advantage over A', S can solve the DDH problem with non-negligible probability, contrary to the assumption that DDH is hard in $\mathbb{G}$.

B. *Simulation*

In this section we construct our simulator S. The challenger of the DDH problem constructs the test quadruple (g_1, g_2, h_1, h_2) by first flipping a random coin d, and then giving S a DDH instance if $d = 1$, and a random instance otherwise. On input the test quadruple and the set $\mathbf{W}$ of votes, S simulates Exp. 1 for A, and determines the bit d as follows:

1) **Adversarial corruption:** S lets the adversary corrupt a minority of registrars CR, and a minority of talliers CT.
2) **Setup:** The distributed key generation protocols $setup\_r(R)$ and $setup\_t(T)$ are simulated for the adversary. S is in control of the majority of registrars, and the majority of talliers, and can therefore recover the secret keys of both R and T at the end of key generation protocols. The details for each game follows.
In *setup_r*, the simulator allows the registrars in its control to follow the protocol. The registrars jointly run a distributed key generation protocol. At the end of key generation protocol, S will deviate from the protocol, and recovers SK_R from the shares of the registrars in its control. Subsequently, the candidate slate is jointly

generated and signed by the registrars, exactly as specified in the real execution.

In *setup_t*, the talliers must share a secret/public key pair to be used in the Modified ElGamal cryptosystem (Section IV-A). The simulator publishes (g_1, g_2) as the input for the distributed key generation protocol between talliers. The output is a single public key PK_T, and each registrar will hold a share of the corresponding secret key SK_T. Let us denote the secret key as $SK_T = (x_1, x_2)$, and the public key as $PK_T = (g_1, g_2, h)$, where $x_1, x_2 \in_R \mathbb{G}$, and $h = g_1^{x_1} g_2^{x_2}$. At this stage, S will deviate from the protocol, and recovers SK_R from the shares of the registrars in its control.

3) **Adversarial corruption:** The adversary A chooses a subset of voters CV, whom she wants to corrupt.
4) **Users registration:** S allows the registrars to jointly generate shares of the credential σ_i of each voter i, as well as the voter roll $\mathbf{L}$. They also give a designated-verifier proof to each voter. Same as above, S deviates from the protocol, and recovers the credentials of all voters from the shares of the registrars in its control.
5) **Adversarial coercion:** The adversary A selects the voter that she wants to coerce, and also chooses the vote β of the coerced voter.
6) **Checking the outputs of the adversary:** If any choices of A in corruption and coercion phases are illegitimate (as verified by the *check_validity* function), the simulator returns 0.
7) **Coin flip:** S flips a random coin $b \leftarrow_R \{0,1\}$.
   If $b = 0$, the simulator gives a random element of $\mathbb{G}$ as the credential of the coerced voter to A, and votes on behalf of the coerced voter to candidate β. The simulator can votes on behalf of him, since it has the credentials of all voters, including the coerced one.
   If $b = 1$, the simulator gives A the credential of the coerced voter.
8) **Honest voter simulation:** Let $\mathbf{W}'$ denote the sequence of W, with the elements corresponding to honest voters. S creates a ballot for each $(V_i, C_j) \in \mathbf{W}'$ as follows: S finds the element of the voter roll $\mathbf{L}$ corresponding to the credential of V_i. Let us call it $\hat{E}_{PK_T}(\sigma_i)$. The simulator can perform this step easily, since it has both SK_T and σ_i, and can therefore decrypt $\mathbf{L}$ and find $\hat{E}_{PK_T}(\sigma_i)$.
   Next, S generates valid-looking encryptions of the vote and credential of V_i, respectively. More precisely, S chooses two random values $r_i, s_i \leftarrow_R \mathbb{Z}_q$, and sets $E_{\widetilde{PK_T}}(C_j) = \left(h_1^{r_i}, h_2^{r_i}, (h_1^{x_1} h_2^{x_2})^{r_i} C_j\right)$ and $E_{\widetilde{PK_T}}(\sigma_i) = \left(h_1^{s_i}, h_2^{s_i}, (h_1^{x_1} h_2^{x_2})^{s_i} \sigma_i\right)$. Note that the honest voter would generate the encryptions of his vote and his credential using the "real" public key of T. That is, $E_{PK_T}(C_j) = (g_1^{r_i}, g_2^{r_i}, h^{r_i} C_j)$ and $E_{PK_T}(\sigma_i) = (g_1^{s_i}, g_2^{s_i}, h^{s_i} \sigma_i)$. We will discuss that this deviation goes unnoticeable to the adversary due to the semantic security of the Modified ElGamal.
   Finally, S has to generate valid-looking NIZK proofs *Pfs*. It can do this easily because it can choose the common reference string (in the CRS model), or program the random oracle (in the RO model).
   The ballot generated for the voter i has the format $\left(E_{\widetilde{PK_T}}(C_j), E_{\widetilde{PK_T}}(\sigma_i), \hat{E}_{PK_T}(\sigma_i), Pfs\right)$. Let $\mathbf{A}$ be the list of all ballots produced by S.
9) **Adversarial ballot casting:** Adversary A casts ballots with credentials of the corrupted voters, as well as that of the coerced voter. Let $\mathbf{B}$ be the list of all ballots cast by A.
10) **Tallying:** Let $\mathbf{E}$ be the union of $\mathbf{A}$ and $\mathbf{B}$, to be counted by the talliers. S allows the talliers in its control to interact honestly with the adversary-controlled talliers. At the end of the interaction, they will jointly post the result $\mathbf{X}$, as well as the proof of correctness P, to the bulletin board.
11) **Adversarial output:** On input $(\mathbf{X}, P)$, the adversary outputs her guess bit b'.

At the end of simulation, S returns 1 if $b = b'$, and 0 otherwise. Its output denotes the simulator's guess for the DDH challenge bit, d. In the of proof [14], the simulator outputs b', which seems to be incorrect.

We now show that the advantage of A in Exp. 1 over her advantage in Exp. 2 is equal to the advantage of S in solving DDH problem.

If test quadruple is a DDH instance ($d = 1$), then $(g_1, g_2, h_1, h_2) = (g, g^a, g^b, g^{ab})$. Therefore, the encryptions will be valid encryptions for any $m \in \mathbb{G}$:

$$h_1 = g_1^b, \quad h_2 = g_2^b \tag{1}$$

$$E_{\widetilde{PK_T}}(m) = \left(h_1^{r_i}, h_2^{r_i}, (h_1^{x_1} h_2^{x_2})^{r_i} m\right) \tag{2}$$

$$\begin{aligned}(1),(2) \Rightarrow E_{\widetilde{PK_T}}(m) &= \left(g_1^{br_i}, g_2^{br_i}, (g_1^{x_1} g_2^{x_2})^{br_i} m\right) \\ &= \left(g_1^{t_i}, g_2^{t_i}, h^{t_i} m\right) = E_{PK_T}(m) \ .\end{aligned} \tag{3}$$

In equation (3), we replaced br_i with t_i. We conclude that the simulation is perfectly indistinguishable from Exp. 1. In other words:

$$\begin{aligned}\Pr[S = 1 \mid d = 1] &= \Pr\left[\mathbf{Exp}^{c\text{-}resist}_{ES,\,A}(k, n_A, V, R, T, C) = 1\right] \\ &= \mathbf{Succ}^{c\text{-}resist}_{ES,\,A}(k, n_A, V, R, T, C) \ .\end{aligned} \tag{4}$$

On the other hand, if the test quadruple is a random instance ($d = 0$), then $(g_1, g_2, h_1, h_2) = (g, g^a, g^b, g^c)$. Therefore, the ciphertexts generated by S reveal no information about the message (in the information-theoretic sense). More specifically, let $c' = \frac{c}{a}$ and $c'' = \frac{c'}{b}$:

$$\begin{aligned}E_{\widetilde{PK_T}}(m) &= \left(h_1^{r_i}, h_2^{r_i}, (h_1^{x_1} h_2^{x_2})^{r_i} m\right) \\ &= \left(g_1^{br_i}, g_2^{c'r_i}, (g_1^{bx_1} g_2^{c'x_2})^{r_i} m\right) \\ &= \left(g_1^{t_i}, g_2^{t_i c''}, h^{t_i} g_2^{(c''-1)bx_2 r_i} m\right) \ ,\end{aligned} \tag{5}$$

which totally masks m (compare to (3)). We conclude that, in case $d = 0$, the adversary will not learn anything from ciphertexts, beside those information specifically decrypted during the execution of the protocol. Such information require a special treatment, which we attend to next.

Below, we argue that in case $d = 0$, the adversary sees essentially whatever she would in Exp. 2:

1) **Credentials *before* coercion:** In Exp. 1, the adversary is given the credentials of the corrupted voters before deciding whom to coerce, while in Exp. 2, the credentials of the corrupted voters are only available to her at later stages.

– *Argument:* We argue that since each credentials is a random element of $\mathbb{G}$, it does not help the adversary in collecting information about other voters. Specially, having access to the credentials of the corrupted voters does not help her to decide whom to coerce. Therefore, from adversary's point of view, Exp. 1 and Exp. 2 are indistinguishable in this respect.

2) **Access to the Bulletin Board:** In Exp. 1, the adversary can see the bulletin board, while in Exp. 2, she does not enjoy such luxury.

– *Argument:* We argue that due to the semantic security of the Modified ElGamal, the adversary does not gain any knowledge by accessing the credentials on the bulletin board. Specifically, the simulator can replace the credentials of honest voters by random elements of $\mathbb{G}$, in a manner unnoticeable to the adversary.

We conclude that from adversary's point of view, Exp. 1 and Exp. 2 are indistinguishable in this respect.

Finally, since the adversary can guess a valid credential only with negligible probability, the probability that the adversary can cast a valid ballot on behalf of an honest voter (specifically the coerced voter) is negligible. In this regard, the *tally* function acts exactly like the *ideal_tally* function (except with negligible probability, where A guesses a valid credential).

All in all, we proved that in case $d = 0$, the adversary is actually presented with the *ideal* experiment. Therefore we have:

$$\begin{aligned}\Pr[S = 1 \mid d = 0] &= \Pr\left[\mathbf{Exp}^{\textit{c-resist-ideal}}_{ES,\,A}(k, n_A, V, R, T, C) = 1\right] \\ &= \mathbf{Succ}^{\textit{c-resist-ideal}}_{ES,\,A}(k, n_A, V, R, T, C) \ .\end{aligned} \tag{6}$$

Subsequently:

$$\begin{aligned}\mathbf{Adv}^{ddh}_{S} &= \Pr[S = 1 \mid d = 1] - \Pr[S = 1 \mid d = 0] \\ &= \mathbf{Succ}^{\textit{c-resist}}_{ES,\,A} - \mathbf{Succ}^{\textit{c-resist-ideal}}_{ES,\,A} \\ &= \left(\mathbf{Succ}^{\textit{c-resist}}_{ES,\,A} - \tfrac{1}{2}\right) - \left(\mathbf{Succ}^{\textit{c-resist-ideal}}_{ES,\,A} - \tfrac{1}{2}\right) \\ &= \mathbf{Adv}^{\textit{c-resist}}_{ES,\,A} - \mathbf{Adv}^{\textit{c-resist-ideal}}_{ES,\,A} \ .\end{aligned} \tag{7}$$

Equation 7 means that the advantage of A in *c-resist* experiment over her advantage in the *c-resist-ideal* experiment is equal to the advantage of S in solving DDH problem, which is negligible under DDH assumption.

VII. Conclusions & Future Work

In this paper, we presented a variant of the JCJ e-voting protocol. Similar to the JCJ protocol, our protocol has a proof of coercion-resistance. However, our protocol reduces the total number of operations from quadratic in the JCJ protocol to linear. While there is a practical implementation of the JCJ protocol called Civitas [20], it relies on partitioning the voters to bypass the quadratic running time of the JCJ protocol. Partitioning makes the protocol inconvenient for the voters, contrary to one of the objectives to be achieved in e-voting: *convenience*.

Our protocol provides provable security, efficiency, and convenience simultaneously. This flexibility, however, comes at a little price: Since the talliers of our protocol decrypt the credentials, the credentials of the voters must be regenerated in the next election. The JCJ protocol does not suffer from this limitation, but its running time is much higher. It will be great to see a protocol enjoying the best of both worlds.

References

[1] K. Sampigethaya and R. Poovendran, "A framework and taxonomy for comparison of electronic voting schemes," *Computers & Security*, vol. 25, no. 2, pp. 137–153, 2006.

[2] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *Proceedings of the 2005 ACM workshop on Privacy in the electronic society (WPES '05)*. ACM, 2005, pp. 61–70.

[3] A. Acquisti, "Receipt-free homomorphic elections and write-in ballots," Cryptology ePrint Archive, 2004, http://eprint.iacr.org/2004/105.

[4] J. Schweisgut, "Coercion-resistant electronic elections with observer," in *2nd International Workshop on Electronic Voting, Bregenz*, 2006.

[5] R. Araújo, N. Rajeb, R. Robbana, J. Traoré, and S. Youssfi, "Towards practical and secure coercion-resistant electronic elections," in *Cryptology and Network Security*. Springer, 2010, pp. 278–297.

[6] J. Clark and U. Hengartner, "Selections: Internet voting with over-the-shoulder coercion-resistance," in *Financial Cryptography and Data Security*. Springer, 2012, pp. 47–61.

[7] W. Smith, "New cryptographic election protocol with best-known theoretical properties," in *Proc. of Workshop on Frontiers in Electronic Elections*, 2005.

[8] S. G. Weber, R. Araujo, and J. Buchmann, "On coercion-resistant electronic elections with linear work," in *The Second International Conference on Availability, Reliability and Security (ARES 2007)*, april 2007, pp. 908–916.

[9] R. Araújo, S. Foulle, and J. Traoré, "A practical and secure coercion-resistant scheme for remote elections," *Frontiers of Electronic Voting*, vol. 7311, 2007.

[10] R. Araújo, S. Foulle, and J. Traoré, "A practical and secure coercion-resistant scheme for internet voting," in *Towards Trustworthy Elections*. Springer, 2010, pp. 330–342.

[11] O. Spycher, R. Koenig, R. Haenni, and M. Schlapfer, "A new approach towards coercion-resistant remote e-voting in linear time," in *Financial Cryptography and Data Security*. Springer, 2012, pp. 182–189.

[12] A. Juels and M. Jakobsson, "Coercion-resistant electronic elections," Cryptology ePrint Archive, 2002, http://eprint.iacr.org/2002/165.

[13] T. Pedersen, "A threshold cryptosystem without a trusted party," in *Advances in Cryptology—EUROCRYPT '91*. Springer, 1991, pp. 522–526.

[14] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *Towards Trustworthy Elections*. Springer, 2010, pp. 37–63.

[15] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," *European Transactions on Telecommunications*, vol. 8, no. 5, pp. 481–490, 1997.

[16] M. Jakobsson and A. Juels, "Mix and match: Secure function evaluation via ciphertexts," in *Advances in Cryptology—ASIACRYPT 2000*. Springer, 2000, pp. 162–177.

[17] C. A. Neff, "A verifiable secret shuffle and its application to e-voting," in *Proceedings of the 8th ACM conference on Computer and Communications Security (CCS '01)*. ACM, 2001, pp. 116–125.

[18] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," in *Advances in Cryptology—EUROCRYPT '99*. Springer, 1999, pp. 295–310.

[19] M. Jakobsson, K. Sako, and R. Impagliazzo, "Designated verifier proofs and their applications," in *Advances in Cryptology—EUROCRYPT '96*. Springer, 1996, pp. 143–154.

[20] M. Clarkson, S. Chong, and A. Myers, "Civitas: Toward a secure voting system," in *IEEE Symposium on Security and Privacy (SP 2008)*, may 2008, pp. 354 –368.