Faster coercion-resistant e-voting by encrypted sorting

Diego F. Aranha[1], Michele Battagliola[2], and Lawrence Roy[1]

[1] Aarhus University, Aarhus, Denmark
{dfaranha, lance.roy}@cs.au.dk
[2] Università degli Studi di Trento
michele.battagliola@unitn.it

Abstract. Coercion-resistance is one of the most challenging security properties to achieve when designing an e-voting protocol. The JCJ voting scheme, proposed in 2005 by Juels, Catalano and Jakobsson, is one of the first voting systems where coercion-resistance was rigorously defined and achieved, making JCJ the benchmark for coercion-resistant protocols. Recently, the coercion-resistance definition proposed in JCJ has been disputed and improved by Cortier, Gaudry, and Yang. They identified a major problem, related to leakage of the number of discarded votes by revoting; and proposed CHide, a new protocol that solves the issue and satisfies a stronger security notion. In this work we present an improved version of CHide, with complexity $O(n \log n)$ instead of $O(n^2)$ in the number n of received ballots, that relies on sorting encrypted ballots to make the tallying phase faster. The asymptotic complexity of our protocol is competitive with other state-of-the-art coercion-resistant voting protocols satisfying the stronger notion for coercion resistance.

1 Introduction

Internet voting is a type of electronic voting that allows voters to cast their ballot remotely through the Internet, without the need of physically going to a polling station. Since the first attempts of introducing the Internet as a legally binding way of casting votes in Estonia and the United States in the early 2000's, Internet voting solutions increased in popularity and are currently used to varying degrees in several countries around the world [18, 29]. Prominent examples include Switzerland [14], Canada [6] and Australia [16].

As with other electronic voting initiatives, the promises of Internet voting are higher voter turnout, lower cost and accessibility [21]; potentially at the expense of simplicity, transparency and privacy. Cryptographic protocols are particularly suited to the task, and in recent years many protocols were designed to achieve secure Internet-based elections that ensure voter privacy, vote verifiability and the correctness of the outcome [2, 9, 26].

There is one additional threat, however, that is equally crucial to address in a fair and democratic election process: *coercion resistance.* Informally, a coercion-resistant protocol must defend voters from attackers that pressure them to vote in a specific way, either through threats or rewards. Because of its remote nature, Internet voting substantially increases the attack surface with respect to coercion resistance, since it introduces new and convenient attacks compared to voting in person at the polling station. These include coercing voters to reveal all the voting material, or monitoring their behavior during the election day.

JCJ Protocol. Juels, Catalano and Jakobsson [20] achieved important progress in this field by proposing one of the first formal definitions of coercion resistance and designing a protocol to achieve it. To this date, JCJ still remains the reference point for research on the topic. Roughly speaking, a voting protocol is coercion resistant if and only if voters are able to generate some kind of *fake credential* that could be handed over to the coercer in case of attack, preserving the original legitimate ones and thus their ability to vote [20]. Votes with fake credentials are discarded later, in the *cleansing* phase of the election process.

One of the main drawbacks of JCJ and related protocols, such as Civitas [9], is their quadratic complexity[3], since they usually require to check each credential against the ones in the following votes. Consequently, this approach to voting has generally limited the scalability of the protocol.

JCJ Leakage. Recently, the security definition presented in JCJ was disputed, for example in [11] and [15], due to its limitation in handling revotes and ballots cast under invalid credentials. Ideally, the only types of leakage that should be allowed are those that inevitably arise from the election result. In particular, an unavoidable leakage is the difference between the total processed ballots and the number of valid votes. In [11] Cortier, Gaudry and Yang showed that the JCJ protocol leaks significantly more than this simple difference. Since the votes with duplicate credentials (i.e. the revotes) and votes with invalid credentials are handled and discarded separately, the JCJ protocol leaks the size of both sets individually, instead of leaking only the size of their union. Moreover, they presented CHide, an improved version of the original JCJ protocol that solves this security issue.

Contributions. However, CHide brings us back to the original scaling issue present in JCJ, as it also takes quadratic time in the number n of received ballots. [4] The techniques used to run JCJ in linear time [25] require leaking the duplicated and invalid credentials separately, and fail to generalize easily to CHide. In this work we overcome this issue and present an improved version of CHide, with complexity $O(n \log n)$ instead of $O(n^2)$, using sorting over encrypted data to make the tallying protocol faster. The resulting protocol achieves asymptotic complexity competitive with other state-of-the-art coercion-resistant

[3] Civitas mitigates this quadratic complexity: split the voters into blocks and tally each block separately. However, this significantly increases leakage, revealing how many votes were eliminated for each block, rather than just once for the whole election.

[4] The CHide preprint was independently updated by the authors to address this issue. We discuss further in Section 5.

voting protocols satisfying a stronger security notion. We also show an improved version of the protocol that removes the registrars altogether.

The rest of the paper is organized as follows. We start with the building blocks of our protocol in Section 2. In Section 3, we describe our protocol and prove its security for stronger coercion resistance in Section 4. We finish with estimates of efficiency gains in Section 5 and conclude in Section 6.

2 Cryptographic Primitives

ElGamal Encryption Scheme. Due to its homomorphic properties, the ElGamal encryption scheme [12] is a popular choice for designing voting schemes.

Let $\mathbb{G}$ be a multiplicatively written group of order q, with generator g, for which solving the Decisional Diffie Hellman (DDH) problem is hard. The private key sk is sampled at random from $\mathbb{Z}_q$, while the public key pk is g^{sk}. The encryption of a message m is defined as $\mathsf{Enc}(m, \mathsf{pk}; r) = (g^r, g^m \cdot \mathsf{pk}^r) \in \mathbb{G}^2$ where $r \in \mathbb{Z}_q$ is a random value. We omit the randomness when not explicitly necessary.

Let $E_0 = (1, 1)$, $E_1 = (1, g)$, and $E_{-1} = (1, g^{-1})$ be the respective encryptions of $0, 1, -1$ with randomness 0. Re-encryption can be done by multiplying a ciphertext by an encryption of 0. In particular, let $X \in \mathbb{G}^2$ be an ElGamal ciphertext, then we define $\mathsf{ReEnc}(X, \mathsf{pk}; r) = X \cdot \mathsf{Enc}(0, \mathsf{pk}; r)$, where the multiplication operation is component-wise.

For a number n_T of election trusteers, we use a (t, n_T) threshold version of ElGamal, so pk is produced via a distributed key generation, and a minimum of $t + 1$ parties are required to jointly decrypt.

Designated-Verifier Zero-Knowledge Proof. Similarly to JCJ and CHide, our protocol uses Designated-Verifier Zero Knowledge Proofs (DVZKPs) [19]. Roughly speaking, a DVZKP is a zero-knowledge proof (ZKP) in which only the verifier designated by the prover is able to be convinced about the correctness of the proof. In particular the verifier V holds a key pair. Using the public key, the prover produces a proof for a statement, such that only V is convinced that the statement is true. This is achieved by allowing V to produce fake but valid DVZKPs for any statement, using their private key.

In particular, the usage of a DVZKP instead of a traditional ZKP is crucial for the evasion strategy, since it allows voters to be sure about the credentials received and, at the same time, they are able to produce fake credentials alongside fake proofs to hand over in case of attacks.

Circuits over encrypted bits. The basic building block for our tallying algorithm is the CGate protocol, originally presented in [27], in the re-randomized version [10]. Informally, on input of two encryptions X, Y of x and y, respectively, with $y \in \{0, 1\}$ it outputs a ciphertext Z which is the encryption of xy. If both x and y are bits, this allows to compute the conjunction And. Since the Not operator can be computed as $\mathsf{Not}(X) = E_1 \cdot X^{-1}$, every other Boolean operator can be easily implemented by combining these two. Algorithm 1 formalizes the idea.

In particular, for our sorting algorithm, we need an operator for equality $\mathsf{Eq}(X,Y) = \mathsf{Not}(XY/\mathsf{CGate}(X,Y)^2)$ and a less-than operator $\mathsf{Less}(X,Y) = Y/\mathsf{CGate}(X,Y)$. Indeed, let a, b be two values and $A_1, ..., A_k$ and $B_1, ..., B_k$ their bitwise encryptions. To check $a < b$ keeping the result encrypted we use this recursive formula: $L_0 = 0$, $L_i = \mathsf{Less}(A_i, B_i) \cdot \mathsf{CGate}(L_{i-1}, \mathsf{Eq}(A_i, B_i))$ for $i = 1, ..., n$. At the end L_k is the encryption of $a < b$.

Algorithm 1 CGate protocol

```
Require: $X, Y$ encryptions of $x, y$, with $y \in \{0,1\}$, number of participants $a$.
Ensure: $Z$ an encryption of $xy$.
 1: Compute $Y_0 = E_{-1} \cdot Y^2$ and set $X_0 = X, a = t+1$
 2: for $i = 1$ to $a$ do
 3:     Participant $P_i$ picks $r_1, r_2 \in \mathbb{Z}_q$ and $s \in \{-1, 1\}$ randomly
 4:     $P_i$ computes $X_i = \mathsf{ReEnc}(X_{i-1}^s, \mathsf{pk}; r_1)$ and $Y_i = \mathsf{ReEnc}(Y_{i-1}^s, \mathsf{pk}; r_2)$
 5:     $P_i$ produces a ZKP $\pi_i$ that $X_i$ and $Y_i$ are well formed
 6:     $P_i$ reveals $X_i, Y_i$ and $\pi_i$
 7: end for
 8: $P_1, ..., P_a$ verify all the proofs. Let $\Pi = (X_1, Y_1, \pi_1)||...||(X_a, Y_a, \pi_a)$.
 9: $P_1, ..., P_a$ jointly rerandomize $X_a, Y_a$ to get $X', Y'$, producing transcript $\Pi^{\mathsf{ReEnc}}$
10: $P_1, ..., P_a$ jointly compute $y_a = \mathsf{Dec}(Y')$ and transcript $\Pi^{\mathsf{Dec}}$
11: return $Z = (X{X'}^{y_a})^{\frac{1}{2}}$ and verification transcript $(y_a, \Pi^{\mathsf{Dec}})||(X', Y', \Pi^{\mathsf{ReEnc}})||\Pi$
```

In [27] the authors proved that the CGate algorithm is SUC-secure. We say that a protocol is SUC-secure if, for all adversary in the real process, there exists a simulator in the ideal process such that no PPT environment can tell whether they are interacting with the adversary in the real process or with the simulator in the ideal process.

Distributed Random Bit Generation. In the same way as CHide, credentials are generated by a particular set of authorities and are encrypted bit by bit. In order to do so, they need to use a distributed random bit generation protocol. In particular, they jointly produce an encrypted bit $\mathsf{Enc}(b, pk)$, for which each participant knows only a share b_i of b. Furthermore, the transcript of the protocol communication is used as a DVZKP for the correctness of the protocol. We use the $\mathsf{RandBit}$ protocol proposed in [11].

Mixnet. Mixnets are widely used in secure e-voting systems. Informally, a mixnet allows a set of participants to shuffle and re-encrypt a set of ciphertexts, without needing to know the secret key (or a secret sharing of it). On a high level, participants privately shuffle all inputs and eventually publish them re-encrypted in random order. Informally, we say that a mixnet is secure if, given at least one honest participant, the permutation from the input to the output remains secret for all the participants involved.

In the protocol we will need a verifiable mixnet, that ensures the correctness of the output (i.e. the output is indeed a permutation and re-encryption of the input). A suitable candidate for our protocol is the mixnet presented in [31].

3 Protocol Description

At its core, our protocol is very similar to CHide. The participants, the Setup phase, the Registration phase and the Voting phase are essentially the same, while we changed both the Cleansing and Tallying phase, to substantially reduce the computational complexity.

The main difference is that for each ballot, CHide requires to compare the encrypted credential to every successive one and to every credentials in the register (thus having quadratic complexity), while in our protocol we first perform an a sorting algorithm on the encrypted votes. At the end of the protocol, votes with the same credentials and authorized credentials are consecutive, allowing the election authorities to recognize valid votes faster.

3.1 Participants

The participants in the protocol are:

- The *public board*, an append-only list of data, where all the other participants can write. The contents of the board can be read by anyone at any time, and the board is assumed to be honest.
- The election *trustees*, a set of n_T authorities that performs the cleansing and the tally. It is assumed that there are most t dishonest trustees, where $t < n_T$ is the threshold of the encryption protocol used.
- The *voters*. There are n_V voters and we assume that the adversary is able to control at most $n_V - 2$ of them.
- The *auditors*, a set of parties that check the consistency of the data published on the board. In particular auditors need to check the validity of all the ZKPs. We only need one auditor to be honest. Since every check involves only public data, any party could serve as auditor.
- The *registrars*, a second set of n_R authorities that provide credentials to voters. For coercion resistance it is assumed that all of them are honest.

Table 1 fixes the notation when referring to the various election participants.

Table 1. Parameters of an election conducted with CHide.

n_T	number of election trustees
n_R	number of registrars
n_C	number of candidates
n_V	number of voters
$n_{\mathcal{A}}$	number of voters controlled by $\mathcal{A}$
BB	the public board

3.2 Overview

Setup Phase. A security parameter k is chosen. The election trustees jointly run the distributed key generation (`DKeyGen`) protocol presented in [13], obtaining a public key pk at the appropriate security level. Each trustee publishes a commitment h_i to its private share of pk on the public board, as well as pk. The private shares are denoted sk_i for $i = 1, ..., n_T$.

Registration phase. As in CHide, credentials are created by a designated set of registrars, encrypted bitwise, sent to the voters and published on the public board. Let $s = (s_1, \dots, s_k)$ be the k-bit credential of voter v and let $S = (S^1, ..., S^k)$ the bitwise encrypted values published on the board. Each credential is sent privately to the voter, with designated zero-knowledge proofs to guarantee voters that their credential is valid.[5] Let R be the list of all the authorized credentials.

Voting Phase. To cast a vote for candidate ν, voter v computes an encryption of their vote $C^1 = \mathsf{Enc}(\nu, \mathsf{pk})$ and a bitwise encryption of their credential $C^2 = (\mathsf{Enc}(s_1, \mathsf{pk}), ..., \mathsf{Enc}(s_k, \mathsf{pk}))$, as well as two ZKPs: one to prove that C^2 contains encryptions of bits, and a second one proving knowledge of the randomness used in C^1 and that ν is a valid voting option. These ZKPs are also used to link together C^1 and C^2, making the tuple $C = (C^1, C^2)$ non-malleable. The tuple and the corresponding ZKPs are published on the public board using an anonymous channel.

During the Voting Phase, each voter can vote multiple times and only the last vote will be counted[6]. During this step the auditors verify the uniqueness of each ballot and that every ZKP is valid.

Cleansing and Tallying Phase. Once the Voting Phase is finished, the election trustees count the votes. Let $BB = \{C_i\}$ the list of all the votes, listed in chronological order, and $R = \{S_i\}$ the list of all authorized credentials.

First of all, all the invalid votes marked by the auditors are discarded. Then the election trustees parse each element e_i of $BB||R$ as $(\mathtt{Data}_i, \sigma_i, f_i, \mathsf{c}_i)$ where:

- $\mathtt{Data}_i \leftarrow C_i^1$ if $e_i \in BB$; otherwise $\mathtt{Data}_i$ is set to be a random encryption.
- $\sigma_i \leftarrow C_i^2$ if $e_i \in BB$; $\sigma_i \leftarrow S_i$ otherwise.
- $f_i \leftarrow \mathsf{Enc}(0, \mathsf{pk})$ if $e_i \in BB$; $f_i \leftarrow \mathsf{Enc}(1, \mathsf{pk})$ otherwise.
- c_i is the bitwise encryption of an increasing counter and represents the chronological order of the votes.

Then the trustees apply a mixnet protocol (for example [31] or [8]) on $BB||R$ and produce a verification transcript. For simplicity we will refer to each element

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

[6] Note that different policies about revoting are possible and could be achieved with a different ordering in the tallying phase.

after the mixnet using the same notation as before, i.e. each element is in the form $(\mathtt{Data}_i, \sigma_i, f_i, \mathsf{c}_i)$.

The election trustees perform a sorting algorithm on the set, with the following relation:

$$e_i <_{\mathtt{Tally}} e_j \Leftrightarrow \mathsf{Dec}(\sigma_i)||\,\mathsf{Dec}(f_i)||\,\mathsf{Dec}(\mathsf{c}_i) <_{\mathtt{Lex}} \mathsf{Dec}(\sigma_j)||\,\mathsf{Dec}(f_j)||\,\mathsf{Dec}(\mathsf{c}_j) \quad (3.1)$$

where, with an abuse of notation, $\mathsf{Dec}(\sigma_i)$ denotes the concatenation of the decryptions of every ciphertext in σ_i and $<_{\mathtt{Lex}}$ is the lexicographical order. It is important to note that:

– If two votes e_i, $e_j \in BB$ have the same credentials, then they are sorted chronologically thanks to the counters $\mathsf{c}_i, \mathsf{c}_j$. Moreover if e_h is such that $e_i <_{\mathtt{Tally}} e_h <_{\mathtt{Tally}} e_j$ then e_h has the same credential of both e_i and e_j.
– If $e_i \in BB$ and $e_j \in R$ have the same credentials (i.e. e_i is a ballot cast with an authorized credential) then $e_i <_{\mathtt{Tally}} e_j$. Moreover if e_h is such that $e_i <_{\mathtt{Tally}} e_h <_{\mathtt{Tally}} e_j$ then e_h has the same credential of both e_i and e_j.
– No two distinct elements e_i, e_j will compare equally in this ordering, thanks to the counter c_i in each ballot.

Informally, the ordered list is formed by blocks of consecutive ballots cast with the same credential, ending with the corresponding element in R if they were made with an authorized one.

During sorting, it is safe to leak the comparison result $e_i <_{\mathtt{Tally}} e_j$, as the mixnet randomly permuted the votes and there are no two equal elements in the tally order. That is, the comparisons only reveal the order of the mixed values, which leaks nothing because they were initially in a random order.

After the sorting, for every pair of consecutive elements (e_i, e_{i+1}) in the ordered list, the election trustees check whether $\mathsf{Dec}(\sigma_i) = \mathsf{Dec}(\sigma_{i+1})$. This produces an encrypted bit I_i^1. Let I_i be the conjunction between the bit encrypted in I_i^1 and f_{i+1}. In particular I_i is an encryption of 1 if and only if e_i is a vote with a valid credential and the last vote with that credential. At this point the trustees multiply I_i and $\mathtt{Data}_i$ in the exponent for every i, computing $\mathsf{CGate}(\mathtt{Data}_i, I_i)$, apply a second mixnet on the resulting list, and decrypt every vote.

The $\mathtt{Sort}$ algorithm can be any suitable comparison sort, such as Quicksort or Mergesort, thanks to the mixnet (the stability property is guaranteed by the flag f_i and the counter c_i, that also ensures the absence of equalities). The crucial part is the evaluation of the comparison as per Equation (3.1). Indeed, let a, b be two values and $A_1, ..., A_k$ and $B_1, ..., B_k$ their bitwise encryption. To obtain an encryption of $a < b$ we use this recursive formula: $L_0 = 0$, $L_i = \mathsf{Less}(A_i, B_i) \cdot \mathsf{CGate}(L_{i-1}, \mathsf{Eq}(A_i, B_i))$ for $i = 1, ..., n$, with $\mathsf{Less}(X, Y) = Y/\,\mathsf{CGate}(X, Y)$.

The result of every comparison can then be decrypted and used according to the chosen sorting algorithm, without leaking anything because of the mixing. Sorting $BB||R$ without mixing would leak the number of votes between two authorized credentials and could lead to potential attack (for example, if an attacker votes with a fake credential that is greater than any authorized one it would easily detect the lie). In fact, due to the mixnet, any adversary would

Algorithm 2 `Tally`
The participants must share of the secret key sk matching the public key pk

```
Require: The list of votes in $BB$ and the list of keys $R$.
Ensure: The result of the election $X$ and a proof $\Pi$ for its correctness.
1: Parse each element in $BB||R$ as described
2: Compute $L, \Pi_1^{\mathtt{Mixnet}} = \mathtt{Mixnet}(BB||R)$
3: Compute $L_s, \Pi^{\mathtt{Sort}} = \mathtt{Sort}(L)$
4: for $e_i \in L_s$ do
5:     $I_i = \mathsf{CGate}(\mathsf{Eq}(\sigma_i, \sigma_{i+1}), f_i)$
6:     $\mathtt{Data}^i = \mathsf{CGate}(\mathtt{Data}_i, I_i)$
7: end for
8: Compute $L_f, \Pi_2^{\mathtt{Mixnet}} = \mathtt{Mixnet}(L_s)$
9: return $X = \mathsf{Dec}(\mathtt{Data})$ for all $\mathtt{Data} \in L_f$ and $\Pi = \Pi_1^{\mathtt{Mixnet}}||\Pi_2^{\mathtt{Mixnet}}||\Pi^{\mathtt{Sort}}||\Pi^{\mathtt{CGate}}$
   where $\Pi^{\mathtt{CGate}}$ is the verification transcript of all CGate computations in the cycle.
```

have no information about the terms of each comparison, thus the result of the comparison is meaningless and can be simulated, as shown in the next section.

In order to prove the correctness of the sorting algorithm, the trustees add the proofs of the correctness of every CGate computation as well the correctness of the decryption. For further details see Section 2.

Evasion Strategy. To evade coercion a voter can simply lie about their credential s, giving a fake credential $\bar{s}$ to the coercer, and manipulating the DVZKP accordingly. In this way, voters are also able to vote with their correct credentials.

4 Security Proof

The proof is very similar to the one presented in [11]. We consider a distribution $\mathcal{B}$ of sequence of pairs (j, ν) where j is a voter and ν is a voting option. Additionally, fake votes are modeled as pairs where $j \notin [1, n_V]$. In the following Algorithms 3 and 4, we employ the real-ideal paradigm.

In the real game (Algorithm 3), the adversary takes part of the setup process (line 2) and decides the set of voters $V_{\mathcal{A}}$ it controls and the coercion target (lines 3-5). Afterwards, votes are drawn according to a distribution $\mathcal{B}$ and added to the list B, containing all the votes in order. Lines 13-22 model the coercion: if $b = 1$, the coerced voter obeys, hence any vote from j is removed from B and the real credential s^j is handled to the adversary. If $b = 0$ the voter follows the evasion strategy, i.e. they cast a vote for their intended preference β and give to the adversary a fake credential.

Votes are then added to BB, according to the sequence B (lines 23-29). After each vote the adversary is allowed to see the board and add votes. Lastly the tally is performed and the adversary guesses whether the evasion strategy was followed or not.

In the ideal world Algorithm 4, the adversary only selects the set of voters $V_{\mathcal{A}}$ it controls and the coercion target (lines 3 and 5). Then votes from $V_{\mathcal{A}}$ (line

27) and, possibly, the coerced votes (line 24-26) are directly added to B. Then B is handled to the tally functionality that publishes the result of the election X, without revealing anything else.

Definition 1 *[11] A voting system is* coercion resistant *iff for all PPT adversary $\mathcal{A}$, for all parameters $n_T, t, n_V, n_{\mathcal{A}}, n_C$ and for all voting distribution $\mathcal{B}$, there exists a polynomial adversary $\mathcal{F}$ and a negligible function μ such that:*

$$|\mathbb{P}(\mathtt{Ideal}(\mathcal{F}, k, n_{\mathcal{A}}, n_C, \mathcal{B}) = 1) - \mathbb{P}(\mathtt{Real}(\mathcal{A}, k, n_T, t, n_V, n_{\mathcal{A}}, n_C, \mathcal{B}) = 1)| \leq \mu(k).$$

Theorem 1 *Under the DDH assumption and in the Random Oracle Model, the voting system presented in Section 3 is coercion resistant.*

Proof. Let $\mathcal{A}$ be an adversary for the real game. We give to $\mathcal{A}$ the power to impersonate t among n_T election trustees and up to $n_{\mathcal{A}}$ voters. Our goal is to build an adversary $\mathcal{F}$ that wins the ideal game by interacting with $\mathcal{A}$ and simulating the real game.

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

Then $\mathcal{F}$ follows the real game normally, until line 14, getting $V_{\mathcal{A}}, j$ and β. In the ideal game $\mathcal{F}$ sends the same choices for $V_{\mathcal{A}}, j, \beta$.

In line 22, $\mathcal{F}$ provides to $\mathcal{A}$ the real credential s^j of the coerced voter. From the ideal game $\mathcal{F}$ learns the size $|B|$ of the ideal board (line 23) and uses it to simulate the voting process (lines 23-29). For $|B|$ times:

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

Then $\mathcal{F}$ uses all the saved adversary ballots in lines 23-27, taking $\beta' = \nu_j$.

$\mathcal{F}$ learns X and its size in line 30 of the ideal game and use it to simulate the tallying process in the real game:

- $\mathcal{F}$ runs the first mixnet for the honest authorities, while $\mathcal{A}$ uses the dishonest ones.
- To perform the sorting, $\mathcal{F}$ simulates all the CGate operations. This can be done since CGate is a SUC-secure protocol, as shown in [10]. $\mathcal{F}$ also simulates the decryption step and thus randomly sorts the list.
- $\mathcal{F}$ runs the second mixnet for the honest authorities, while $\mathcal{A}$ uses the dishonest ones.
- $\mathcal{F}$ chooses $|X|$ entries at random and simulates its partial decryption: every entry not chosen is decrypted to 0, while such $|X|$ entries are decrypted such that the result is exactly X.

Algorithm 3 Real

```
Require: $\mathcal{A}, k, n_T, t, n_V, n_\mathcal{A}, n_C, \mathcal{B}$
 1: $BB \leftarrow \emptyset$
 2: $\mathsf{pk}, \mathsf{sk}_i, h_i \leftarrow \mathtt{Setup}^\mathcal{A}(k, n_T, t)$
 3: $V_\mathcal{A} \leftarrow A()$
 4: $\{s^i\}_{i\in[1,n_V]}, R \leftarrow \mathtt{Register}(\mathtt{k}, \mathsf{pk}, \mathtt{n_V})$
 5: $(j, \beta) \leftarrow \mathcal{A}(\{s^i\}_{i\in V]}, R)$
 6: if $|V| \neq n_A$ or $j \notin [1, n_V] \setminus V_\mathcal{A}$ or $\beta \notin [1, n_C] \cup \{\emptyset\}$ then
 7:     return 0
 8: end if
 9: $B \leftarrow \mathcal{B}(n_V - n_\mathcal{A}, n_C)$
10: for $(i, *) \in B, i \notin [1, n_V]$ do
11:     $s^i \leftarrow \mathtt{FakeCred}(\mathtt{s}^1)$
12: end for
13: $b \xleftarrow{\$} \{0, 1\}$
14: $\tilde{s} \leftarrow s^j$
15: if $b == 1$ then
16:     Remove all $(j, *)$ from $B$
17: else
18:     Remove all $(j, *)$ from $B$ but the last
19:     Replace it with $(j, \beta)$
20:     $\tilde{s} \leftarrow \mathtt{FakeCred}(s^j)$
21: end if
22: $\mathcal{A}(\tilde{s})$
23: for $(i, \alpha) \in B$ do
24:     $M \leftarrow \mathcal{A}(BB)$
25:     $BB \leftarrow BB \cup \{m \in M | m \text{ valid}\}$
26:     $BB \leftarrow \{\mathtt{Vote}(c_i, \alpha, \mathsf{pk})\}$
27: end for
28: $M \leftarrow \mathcal{A}(BB)$
29: $BB \leftarrow BB \cup \{m \in M | m \text{ valid}\}$
30: $X, \Pi \leftarrow \mathtt{Tally}^\mathcal{A}(BB, R, \mathsf{pk}, \{h_i, s_i\}, t)$
31: $b' \leftarrow \mathcal{A}()$
32: return $b == b'$
```

Algorithm 4 Ideal

```
Require: $\mathcal{A}, k, n_V, n_\mathcal{A}, n_C, \mathcal{B}$
 1:
 2:
 3: $V_\mathcal{A} \leftarrow A()$
 4:
 5: $(j, \beta) \leftarrow \mathcal{A}()$
 6: if $|V| \neq n_A$ or $j \notin [1, n_V] \setminus V_\mathcal{A}$ or $\beta \notin [1, n_C] \cup \{\emptyset\}$ then
 7:     return 0
 8: end if
 9: $B \leftarrow \mathcal{B}(n_V - n_\mathcal{A}, n_C)$
10:
11:
12:
13: $b \xleftarrow{\$} \{0, 1\}$
14:
15: if $b == 1$ then
16:     Remove all $(j, *)$ from $B$
17: else
18:     Remove all $(j, *)$ from $B$ but the last
19:     Replace it with $(j, \beta)$
20:
21: end if
22:
23: $(\nu_i)_{i\in V_\mathcal{A}}, \beta' \leftarrow \mathcal{A}(|B|)$
24: if $b == 1$ and $\beta \neq \emptyset$ then
25:     $B \leftarrow B \cup \{(j, \beta')\}$
26: end if
27: $B \leftarrow B \cup \{(i, \nu_i) | i \in V_\mathcal{A}, \nu_i \in [1, n_C]\}$
28:
29:
30: $X \leftarrow \mathtt{result}(\mathtt{cleanse}(B))$
31: $b' \leftarrow \mathcal{A}(X)$
32: return $b == b'$
```

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

– In the real game $\mathcal{A}$ can get either the real credential s^j or a fake one. In the simulation $\mathcal{A}$ always receives s^j. Since in both the real and ideal worlds fake credentials have uniformly random distribution and the DVZKP could be simulated, $\mathcal{A}$ can only distinguish a real execution from a simulated one if and only if it is able to distinguish whether $\tilde{s}$ is a plaintext of one of the encrypted credentials in R or not. Since the ElGamal encryption is IND-CPA secure under the DDH Assumption this is impossible.
– During the simulation of the voting loop (line 23-29 of the real game) $\mathcal{F}$ adds random ballot, while in the real game ballots are drawn according to $\mathcal{B}$. As

before, since the ballots are encrypted, the simulation is indistinguishable from the real game under the DDH Assumption.

- During the tally $\mathcal{F}$ simulates the execution of the CGate protocol. By SUC-security, the simulation is indistinguishable from the real game[10].
- In the real game, the ballots are sorted as per relation 3.1, while in the ideal game each comparison is simulated and thus the order is random. Being able to distinguish between the correct order and a fake one would mean either being able to distinguish the ballots, that is unfeasible due to the IND-CPA security of the encryption scheme, or being able to recognize the ballots after the mixnet, that is unfeasible thanks to the security of the mixnet.
- In the simulation the result always include all the last valid ballots cast by honest voters. In a real execution the adversary may change it by casting ballots on behalf of an honest voter. However, to do so, the adversary must be able to create a valid ZKP about the credential used, and this is unfeasible.
- $\mathcal{F}$ simulates the decryption protocol at the end. This simulation is indistinguishable from the real world under the DDH assumption in the ROM.

4.1 Removing the registrars

Registrars are authorities whose only role is to provide authorized credentials to every user and publish the list of encrypted authorized credentials R.

In the base protocol we assume that all the n_R registrars are honest to achieve coercion resistance. Indeed, if the adversary is able to control at least one registrar, it clearly has probability of at least $\frac{1}{n_R}$ to detect the evasion strategy, since trivially it knows one of the share that forms the credential.

Informally, their only purpose is to provide some credential to the user, with the property that the user could later deny to have received them. The same result could be achieved by letting every user generate their own credential, encrypt them, delete the used randomness and publish the credential.[7] In this way we are able to remove a critical point of failure for coercion resistance.

To prove the security of the protocol without registrars we need to change the real world game (Algorithm 3) in line 4, replacing it with the following loop:

The security proof remains the same, except for the initial part and the voting loop. Instead of receiving the credential from the registrars, $\mathcal{F}$ performs the loop normally. At every iteration it checks whether the adversary created credential is already in R or not. If the credential is duplicate then $\mathcal{F}$ increases an internal counter of duplicate credentials by one.

The voting loop is simulated as before, but in the ideal world, $\mathcal{F}$ casts one additional null vote for every duplicate credential, such that the number of voter and credentials is consistent and the election result remains the same.

[7] This setting requires additional checks to avoid voters with multiple credentials and to verify their identity. Moreover, attackers should not be able to link a credential in R to its owner. A possible solution could be linkable ring signature [22], with the ring formed by long term authorized public keys. Lastly, k should be big enough such that the probability of collision is negligible.

Algorithm 5 Proposed improvement to remove registrars

```
1: $R \leftarrow \emptyset$
2: while $|R_\mathcal{A}| < n_\mathcal{A}$ do
3:     $S \leftarrow \mathcal{A}(R)$
4:     $R \leftarrow R \cup \{\ s \in S | s \text{ valid}\}$
5:     $R_\mathcal{A} \leftarrow R_\mathcal{A} \cup \{\ s \in S | s \text{ valid}\}$
6:     if $|R| < n_V - n_\mathcal{A}$ then
7:         $s \leftarrow \texttt{GenerateCred}()$
8:         $R \leftarrow R \cup \{s\}$
9:     end if
10: end while
```

5 Performance

5.1 Comparison with CHide

The main goal of the paper is to improve the performance of the tallying protocol in CHide and JCJ. This is achieved by performing a preliminary sorting step, that reduces the complexity of the tallying from quadratic to quasi-linear.

A performance comparison between our protocol and CHide can be performed by couting the number of CGate operations. We use as example the recent Estonian election, where for the first time more than half of the voters used a remote voting system, for a total of a little more than 3×10^5 valid votes. [17]. Since the Estonian voting system does not track the number of revotes and removed ballots, we suppose that a total of 6×10^5 votes where submitted (i.e. only half of the total votes are valid votes) and that every registered voter voted (i.e. the list of authorized credentials $|R|$ contains 3×10^5 registered credentials). In the following k is the bit-length of voters' credentials.

Each comparison during the sorting algorithm requires $3k$ CGate computations, as explained in 2. Thus for the sorting phase our algorithm requires $3k(9 \times 10^5 \times \log_2(9 \times 10^5)) \approx 54k \times 10^6$ CGate computations and 18×10^6 decryptions. Then, to compute the check bit I^i for every pair of votes the protocol requires $2k \times 9 \times 10^5$ CGate computations. In total, our protocols require around $56k \times 10^6$ CGate computations, 18×10^6 intermediate decryptions and two mixnet applications.

The CHide protocol instead requires to check that the credentials of each casted votes unique, comparing it with each subsequent vote, and that it is an authorized one, comparing it with every registered credentials. Each equality operation requires only k CGate computations, thus for finding duplicates CHide requires $k(2 \times 6 \times 10^5 \times 3 \times 10^5) = 360k \times 10^9$ CGate computations and the same number of computations for checking authorized credentials. Then a mixnet is applied and the votes are decrypted. Thus, CHide requires a total of $720k \times 10^9$ CGate computations and one mixnet application.

Recent Updates. The CHide preprint was independently updated by the authors to address the quadratic complexity of the protocol. Their solution is quite simi-

Table 2. Performance comparison between CHide and our protocol with respect of the security parameter k.

	CGate	Mixnet	Preliminary Decryptions
CHide	$720k \times 10^9$	1	-
Our Protocol	$56k \times 10^6$	2	18×10^6

lar to our solution, leveraging the CGate protocol to sort all the votes and achieve a quasi-linear complexity.

While sharing the same philosophy and the same asymptotic complexity, the two protocols have a meaningful difference that could lead to different running times. Updated CHide avoids the preliminary mixnet by using a swap operation between ciphertexts, instead of simply decrypting the output of each comparison. This restricts their choice of sorting algorithms to be *data-oblivious*, with complexity $O(n \log^2 n)$. Moreover, instead of using a single bit, they use a fixed "special" counter for registered credentials, thus performing more comparisons in the last part of the tally (the computation of I_i, as per our notation).

5.2 Comparison with related works

During the last years many different coercion resistant protocols have been proposed, usually with the goal of reducing the quadratic complexity that is typical of protocols descending from JCJ. Notable examples of more efficient protocols are VoteAgain [24], AFT [3], Athena [28] and protocols based on hash tables like [25] and [30]. The linear-time version of the JCJ protocol proposed in [25] also uses fully homomorphic encryption. Table 5.2 summarizes the comparison between this and related work in terms of security and complexity.

Table 3. Comparison with other coercion resistant protocols.

Protocol	Complexity	Security
JCJ[20]	$O(n^2)$	JCJ
Civitas [9]	$O(n^2)$	JCJ
AKLM [1]	$O(n^2)$	AKLM
Revote [23]	$O(n^2)$	AKLM
CHide[11]	$O(n^2)$ or $O(n \log^2 n)$	CHide
VoteAgain [24]	$O(n \log n)$	VoteAgain
AFT [3]	$O(n)$	JCJ
Athena [28]	$O(n)$	JCJ + Dups
Hash-based [25, 30]	$O(n)$	JCJ + Dups
This work	$O(n \log n)$	CHide

In the table, the security levels are defined as:

- JCJ is the security level achieved by the original JCJ protocol.
- JCJ+Dups is at lower security level than JCJ, where the number of votes for each credential also leak.

- AKLM is at lower security than JCJ, in which it is assumed that voters revote at the end of the voting period to escape from adversarial control.
- CHide is the security level achieved by CHide, higher than JCJ.
- VoteAgain follows its own coercion resistance definition introduced in [24] and it is not comparable with the others.

From the state of the art, achieving a better or equivalent complexity than our protocol requires to either change the security definition (as per [24]) or to increase the leakage.

6 Conclusions

In this work we presented an enhanced version of CHide, that drastically reduces the computational complexity of the tallying from $O(n^2)$ to $O(n \log n)$, which is currently the best efficiency among voting protocols satisfying a stronger notion of coercion resistance.

A possible way to speed up the tally even further is amortizing the process through the whole voting phase, instead of waiting until the end of the election. A possible approach would consist of using a bucket sorting algorithm, like the one presented in [4]. As votes come in, they are assigned to buckets. When the first two buckets are full, the first step of bucket sorting is performed. When the next two buckets are full, the authorities perform the first step of the sorting process on them and the second step on the whole for the bucket, and so on. While maintaining the same asymptotic complexity, this approach could lead to a vastly reduced delay between the end of the voting phase and the publication of the result. However, bucket sorting is usually susceptible to "overflow" attacks. Indeed, typical bucket sorting algorithms like [4] allow for a fixed maximum number of elements in each bucket, thus an attacker could vote multiple time with the same credential, causing the corresponding bucket to overflow and making the sorting fail. On the other hand, increasing drastically the bucket size to make these kind of attacks impractical, would greatly decrease the performance, thus making the use of bucket sorting meaningless. In the end we not find any solution to this problem but it is a topic worthy of further examination.

Unfortunately, our protocol still has the same issue of CHide regarding the dimension of the credentials, that are encryptions of individual bit instead of a single encrypted string. The bitwise encryption is required to realize a secure tally, since we need to multiply ciphertexts in the exponents. A possible solution to this problem, while keeping the overall structure of the tally in place, would be to change the encryption protocol. This could be achieved using class group encryption, originally presented in [7] and later studied in a threshold version in [5]. However this approach would need to design an ad-hoc mixnet suitable for this kind of encryption. Moreover, maintaining this level of efficiency could be challenging, since the sorting protocol would need some adaptations to work, in particular to avoid equal credentials.

References

[1] D. Achenbach, C. Kempka, B. Löwe, and J. Müller-Quade. “Improved Coercion-Resistant Electronic Elections through Deniable Re-Voting”. In: *USENIX Journal of Election Technology and Systems (JETS)* 3.2 (Aug. 2015), pp. 26–45. ISSN: 2328-2797. URL: `https://www.usenix.org/jets/issues/0302/achenbach`.

[2] B. Adida. “Helios: Web-based Open-Audit Voting”. In: *USENIX Security Symposium*. USENIX Association, 2008, pp. 335–348.

[3] R. Araújo, S. Foulle, and J. Traoré. “A Practical and Secure Coercion-Resistant Scheme for Internet Voting”. In: *Towards Trustworthy Elections*. Vol. 6000. LNCS. Springer, 2010, pp. 330–342.

[4] G. Asharov, T. H. Chan, K. Nayak, R. Pass, L. Ren, and E. Shi. “Bucket Oblivious Sort: An Extremely Simple Oblivious Sort”. In: *SOSA*. SIAM, 2020, pp. 8–14.

[5] L. Braun, I. Damgård, and C. Orlandi. “Secure Multiparty Computation from Threshold Encryption based on Class Groups”. In: *IACR ePrint Arch.* (2022), p. 1437. URL: `https://eprint.iacr.org/2022/1437`.

[6] A. Cardillo, N. Akinyokun, and A. Essex. “Online Voting in Ontario Municipal Elections: A Conflict of Legal Principles and Technology?” In: *E-VOTE-ID*. Vol. 11759. LNCS. Springer, 2019, pp. 67–82.

[7] G. Castagnos and F. Laguillaumie. “Linearly Homomorphic Encryption from DDH”. In: *CT-RSA*. Vol. 9048. LNCS. Springer, 2015, pp. 487–505.

[8] D. Chaum. “Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms”. In: *Commun. ACM* 24.2 (1981), pp. 84–88. URL: `https://doi.org/10.1145/358549.358563`.

[9] M. R. Clarkson, S. Chong, and A. C. Myers. “Civitas: Toward a Secure Voting System”. In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2008, pp. 354–368.

[10] V. Cortier, P. Gaudry, and Q. Yang. “A Toolbox for Verifiable Tally-Hiding E-Voting Systems”. In: *ESORICS (2)*. Vol. 13555. LNCS. Springer, 2022, pp. 631–652.

[11] V. Cortier, P. Gaudry, and Q. Yang. “Is the JCJ voting system really coercion-resistant?” working paper or preprint. 2022. URL: `https://hal.inria.fr/hal-03629587`.

[12] T. E. Gamal. “A public key cryptosystem and a signature scheme based on discrete logarithms”. In: *IEEE Trans. Inf. Theory* 31.4 (1985), pp. 469–472. URL: `https://doi.org/10.1109/TIT.1985.1057074`.

[13] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. “Secure Distributed Key Generation for Discrete-Log Based Cryptosystems”. In: *J. Cryptol.* 20.1 (2007), pp. 51–83.

[14] T. Haines, O. Pereira, and V. Teague. “Running the Race: A Swiss Voting Story”. In: *E-Vote-ID*. Vol. 13553. LNCS. Springer, 2022, pp. 53–69.

[15] T. Haines and B. Smyth. “Surveying definitions of coercion resistance”. In: *IACR ePrint Arch.* (2019), p. 822. URL: `https://eprint.iacr.org/2019/822`.

[16] J. A. Halderman and V. Teague. "The New South Wales iVote System: Security Failures and Verification Flaws in a Live Online Election". In: *VoteID*. Vol. 9269. LNCS. Springer, 2015, pp. 35–53.

[17] *How did Estonia carry out the world's first mostly online national elections*. `https://e-estonia.com/how-did-estonia-carry-out-the-worlds-first-mostly-online-national-elections/`. Accessed: 2023-05-06.

[18] *Internet Voting: Past, Present and Future*. `https://www.ifes.org/news/internet-voting-past-present-and-future`. Accessed: 2023-02-18.

[19] M. Jakobsson, K. Sako, and R. Impagliazzo. "Designated Verifier Proofs and Their Applications". In: *EUROCRYPT*. Vol. 1070. LNCS. Springer, 1996, pp. 143–154.

[20] A. Juels, D. Catalano, and M. Jakobsson. "Coercion-Resistant Electronic Elections". In: *Towards Trustworthy Elections*. Vol. 6000. LNCS. Springer, 2010, pp. 37–63.

[21] N. Licht, D. Duenas-Cid, I. Krivonosova, and R. Krimmer. "To i-vote or Not to i-vote: Drivers and Barriers to the Implementation of Internet Voting". In: *E-VOTE-ID*. Vol. 12900. LNCS. Springer, 2021, pp. 91–105.

[22] J. K. Liu, V. K. Wei, and D. S. Wong. "Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups (Extended Abstract)". In: *ACISP*. Vol. 3108. LNCS. Springer, 2004, pp. 325–335.

[23] P. Locher, R. Haenni, and R. E. Koenig. "Coercion-Resistant Internet Voting with Everlasting Privacy". In: *Financial Cryptography Workshops*. Vol. 9604. LNCS. Springer, 2016, pp. 161–175.

[24] W. Lueks, I. Querejeta-Azurmendi, and C. Troncoso. "VoteAgain: A scalable coercion-resistant voting system". In: *USENIX Security Symposium*. USENIX Association, 2020, pp. 1553–1570.

[25] P. B. Rønne, A. Atashpendar, K. Gjøsteen, and P. Y. A. Ryan. "Short Paper: Coercion-Resistant Voting in Linear Time via Fully Homomorphic Encryption". In: *Financial Cryptography and Data Security*. Ed. by A. Bracciali, J. Clark, F. Pintore, P. B. Rønne, and M. Sala. Cham: Springer International Publishing, 2020, pp. 289–298. ISBN: 978-3-030-43725-1.

[26] P. Y. A. Ryan, P. B. Rønne, and V. Iovino. "Selene: Voting with Transparent Verifiability and Coercion-Mitigation". In: *Financial Cryptography Workshops*. Vol. 9604. LNCS. Springer, 2016, pp. 176–192.

[27] B. Schoenmakers and P. Tuyls. "Practical Two-Party Computation Based on the Conditional Gate". In: *ASIACRYPT*. Vol. 3329. LNCS. Springer, 2004, pp. 119–136.

[28] B. Smyth. "Athena: A verifiable, coercion-resistant voting system with linear complexity". In: *IACR ePrint Arch.* (2019), p. 761. URL: `https://eprint.iacr.org/2019/761`.

[29] *Use of E-Voting Around the World*. `https://www.idea.int/news-media/media/use-e-voting-around-world@misc`. Accessed: 2023-05-06.

[30] S. G. Weber, R. Araujo, and J. Buchmann. "On Coercion-Resistant Electronic Elections with Linear Work". In: *The Second International Confer-*

ence on Availability, Reliability and Security (ARES'07). 2007, pp. 908–916. DOI: `10.1109/ARES.2007.108`.

[31] D. Wikström. "A Commitment-Consistent Proof of a Shuffle". In: *ACISP*. Vol. 5594. LNCS. Springer, 2009, pp. 407–421.