Selections: Internet Voting with Over-the-Shoulder Coercion-Resistance*

Jeremy Clark and Urs Hengartner

University of Waterloo
`{j5clark,uhengart}@cs.uwaterloo.ca`

Abstract. We present Selections, a new cryptographic voting protocol that is end-to-end verifiable and suitable for Internet voting. After a one-time in-person registration, voters can cast ballots in an arbitrary number of elections. We say a system provides over-the-shoulder coercion-resistance if a voter can undetectably avoid complying with an adversary that is present during the vote casting process. Our system is the first in the literature to offer this property without the voter having to anticipate coercion and precompute values. Instead, a voter can employ a panic password. We prove that Selections is coercion-resistant against a non-adaptive adversary.

1 Introductory Remarks

From a security perspective, the use of electronic voting machines in elections around the world continues to be concerning. In principle, many security issues can be allayed with cryptography. While cryptographic voting has not seen wide deployment, refined systems like Prêt à Voter [13,33] and Scantegrity II [11] are representative of what is theoretically possible, and have even seen some use in governmental elections [9]. Today, a share of the skepticism over electronic elections is being apportioned to Internet voting.[1] Many nation-states are considering, piloting or using Internet voting in elections. In addition to the challenges of verifiability and ballot secrecy present in any voting system, Internet voting adds two additional constraints:

- Untrusted platforms: voters should be able to reliably cast secret ballots, even when their devices may leak information or do not function correctly.
- Unsupervised voting: coercers or vote buyers should not be able to exert undue influence over voters despite the open environment of Internet voting.

As with electronic voting, cryptography can assist in addressing these issues. The study of cryptographic Internet voting is not as mature. Most of the literature concentrates on only one of the two problems (see related work in Section 1.2). In this paper, we are concerned with the unsupervised voting problem. Informally, a system that solves it is said to be coercion-resistant.

1.1 Contributions

Coercion-resistant, end-to-end verifiable Internet voting systems have been proposed [1,4,17,27,36,41]. However, these systems all require the voter to remember cryptographic information after registration. Since the information is too long to memorize, authentication can be considered to be based on "something you have." Voters must prepare for the possibility of coercion by creating fake values, proofs, or transcripts. Our system works with passwords, "something you know," and it allows a voter to supply a panic password during ballot casting that can be created mentally in real-time by the voter. In summary, our system provides:

- Password-based authentication and cognitive coercion-resistance,

* An extended abstract of this paper appeared at Financial Cryptography 2011.

[1] One noted cryptographer, Ronald Rivest, infamously opined that "best practices for Internet voting are like best practices for drunk driving" [28].

- In-person registration that can be performed bare-handed,
- Tallying that is linear in the number of voters, and
- Efficient revocation of voters from the roster during and between elections.

We compare Selections to three systems: JCJ [27], Civitas [17], and AFT [4] (see Section 1.2). Of these properties, only Selections meets each while AFT achieves the third and both JCJ and Civitas achieve the fourth.

1.2 Related Work

The field of cryptographic voting is mature. One dissertation in 2011 reviews over 90 systems [14]. At this point, any new proposals for systems should be soundly motivated. Our system addresses the problem of coercion and vote selling when voters are not required to vote in a private booth. Only a small number of the most recent papers in cryptographic voting address this threat.

Coercion-resistance was first formalized by Juels *et al.* [27], who also provide a coercion-resistant system, often referred to as JCJ. JCJ was independently implemented as Civitas [17]. The main drawback of both is that tallying is quadratic in the number of voters. Aquisti [1] refined JCJ to use Paillier encryption and support write-in candidates, while both Smith [36] and Weber *et al.* [41] made the first attempts at reducing the complexity of tallying to linear. Unfortunately, all three are considered broken [4,17,6]. More recently (concurrent with Selections), Spycher *et al.* have proposed a different approach to making JCJ linear [37].

Araujo *et al.* provide a linear-time system we refer to as AFT [4]. Although no proofs are provided, the system appears sound. Both JCJ/Civitas and AFT provide registered voters with anonymous credentials. A voter submits a credential along with her vote and a procedure for computing a fake credential is provided (but cannot be done without a computer). In JCJ/Civitas, the credentials of registered voters are posted and these are anonymously and blindly compared to the credential accompanying each submitted vote. In AFT, the credentials of registered voters are essentially signed and the presence of a valid signature on a credential submitted during casting is anonymously and blindly checked. Due to the difficulty of revoking a signed value, voters cannot be revoked in AFT without a change of cryptographic keys.

Some Internet systems are designed for low-coercion elections. These include Helios [2], which was used in a binding university election [3]. Other Internet voting systems concentrate on the untrusted platform issue. A common approach is "code voting," where acknowledgement codes are returned to voters upon receipt of a vote. The codes are a function of the vote and not known in advance to the network carrier. This principle can be seen in SureVote [10], CodeVoting [26], Pretty Good Democracy [34], and Heiberg *et al.* [21].

2 Preliminaries

2.1 Selections: High-Level Overview

Selections is a protocol designed to allow voters to cast ballots over the Internet during a window of time prior to traditional in-person voting. Voters can opt out of Selections at any time prior to election day and cast a ballot in-person.

To be eligible for Selections, voters first complete a one-time, in-person registration protocol in a private booth without needing her own computational device. After this registration, the voter can vote in future elections over a tappable channel (see Section 2.3). The registration involves the voter choosing a password to be used for vote casting. However this password is non-traditional—it is a password from a panic password system (see Section 2.5). A semantically-secure homomorphic encryption of this password is posted on a public roster. The roster has an entry for each registered voter containing this ciphertext. The voter must be convinced that her entry is a correct encryption without being able to prove what it encrypts to anyone.

During vote submission, the voter asserts what her password is: it may be her actual password or a panic password. The voter creates a binding commitment to this asserted password. The voter then rerandomizes her entry off the roster. The voter proves in zero-knowledge the latter ciphertext is a re-encryption of some random subset of passwords off the public roster, without revealing which one. The commitment to her asserted password, re-encrypted roster entry, proof (and some additional proofs that things are well-formed), and an encryption of her vote are submitted over an anonymous channel to a public bulletin board.

When the voting period expires, a distributed group of trustees will eliminate submissions with invalid proofs, eliminate duplicate votes based on the password commitment, and then use a verifiable mix network to shuffle the order of the remaining submissions. After shuffling, voters can no longer determine where their submission is in the new permuted list. For each submission, the trustees will determine if the asserted password matches the roster entry without revealing either. If it does not, the entry is eliminated. The output of Selections is a list of encrypted votes from registered voters without duplicates. The entire protocol can be verified for soundness.

2.2 Coercion-resistance

Informally, Juels *et al.* define coercion-resistance as providing receipt-freeness, while preventing three attacks: randomization, abstention, and simulation [27]. A voting system is said to be receipt-free if the voter cannot produce a transcript that constitutes a sound argument for how they voted [8]. Adversaries should not be able to force a registered voter to cast a random vote or to abstain from voting. Finally, the system should protect against voters surrendering their credentials and allowing a coercer or vote buyer to cast their vote for them. The dominant approach to preventing such a simulation is providing voters with the ability to create fake credentials. If an adversary cannot distinguish a real credential from a fake one, he will only be willing to pay what a fake credential is worth, which is nothing.

2.3 Untappable Channels

The main challenge for coercion-resistant Internet voting is dealing with the elimination of the private voting booth, modelled as an untappable channel. One approach is to use multiple secure channels and assume that while any individual channel can be tapped, no adversary can tap all channels simultaneously. The second is to use an untappable channel just once, and bootstrap the output of this interaction into an arbitrary number of future interactions over secure (or anonymous) channels. We use the latter approach.

2.4 Registration Authority

In most coercion-resistant Internet voting systems, voters interact with a distributed registration authority [1,4,27]. To achieve coercion-resistance, it is assumed that at least one registrar is not corrupted by the adversary. Voters may be corrupted to retain a transcript, however the transcript has deniability by using a designated verifier proof [24].

While distributing trust is usually an effective approach for achieving correctness and secrecy in a protocol, it is more complex with coercion-resistance. The voter must be aware of which entity she trusts, so she can fake a proof that will not be compared to the original. If the voter discloses her private key to an adversary, it only requires a single malicious registrar to collude with the adversary and undetectably issue the voter an incorrect credential share (while retaining the correct value for potential adversarial use).

These concerns leave it unclear if the benefits of a distributed registration authority are worthwhile. While Selections is amenable to a distributed registration authority (voters would submit encryptions of shares of their password, which are homomorphically combined to create an encryption of the password), we describe the protocol using a single registrar that is assumed to not collude with a coercer (but may still misbehave in any other regard).

2.5 Panic Passwords

A panic password system [15] initializes three categories of passwords: a password, a set of panic passwords, and the residual set of inadmissible passwords. From the user's view, submission of a password or a panic password is indistinguishable, while an inadmissible password will prompt the user to try again. If the user registers a password and one panic password, an adversary can demand two distinct admissible passwords and submit the coerced vote with each—therefore, the number of panic passwords should be arbitrarily large to prevent these "iteration" attacks. If a user registers a password and all other values are panic passwords, an accidental mistyping will result in the vote being discarded—therefore, the distance between admissible and inadmissible passwords should be maximized. Finally, with an arbitrarily large number of panic passwords distributed sparsely among inadmissible passwords, set-membership tests for panic passwords should be cognitively easy to perform.

Clark and Hengartner propose the 5-Dictionary panic password system to meet these requirements [15]. Admissible passwords consist of five words from an agreed upon dictionary: the user chooses one combination as her password and any other combination is a panic password. A typo is likely to mutate the intended word into a string not found in the dictionary. With the Unix dictionary of English words, this system offers up to 70 bits of entropy, making exhaustive search infeasible at this time.[2] The authors also propose the 5-Click alternative based on graphical passwords, and new panic password schemes could be developed based on, for example, preferences [25]. Voters would be free to choose which to use.

3 The Selections Protocol

Selections involves four participants: a set of voters, a set of election trustees, an election authority, and a registrant. The system has six main protocols: registration set-up, voter preparation, registration, election set-up, casting, and pre-tallying. Let $\langle \mathsf{DKG}, \mathsf{Enc}, \mathsf{DDec} \rangle$ be a threshold encryption scheme. Distributed key generation $\mathsf{DKG}(n, m)$ generates public key, e, and a private key share, d_i, for each of n trustees. Encryption, $\mathsf{Enc}_e(m, r)$, is semantically secure and homomorphic with respect to one operation. Distributed decryption, $\mathsf{DDec}_{d_i}(c)$, on ciphertext c can be performed with $m+1$ trustees submitting shares d_i.[3] We use threshold Elgamal [31], which is CPA-secure [39].

3.1 Registration Setup

The registration set-up protocol involves a set of n trustees: $\mathcal{T}_1, \ldots, \mathcal{T}_n$ and the election authority. Primes p and q are chosen such that the DL-problem and DDH-problem are hard in the multiplicative subgroup $\mathbb{G}_q$ of $\mathbb{Z}_p^*$. Each T_j participates in $\mathsf{DKG}(n, m)$. Commitments are sent to the election authority, who posts them to an append-only broadcast channel called the Bulletin Board. At the end of the protocol, each $\mathcal{T}_j$ has private key share d_j and public key e is posted. The protocol is standard and will not be described here [31].

3.2 Voter Preparation

The voter preparation procedure is performed by each voter $\mathcal{V}_i$ on a trusted computational client. Let $\langle P, I \rangle$ be the domain of a panic password system. P represents the set of admissible passwords and $I = \neg P$ is the set of inadmissible passwords. $\mathcal{V}_i$ chooses a password $\hat{\rho}$. The client runs $\mathsf{PassSubmit}(\hat{\rho})$, which tests if $\hat{\rho} \in P$. If $\hat{\rho} \in I$, $\mathsf{PassSubmit}(\hat{\rho})$ returns an error. The set of panic passwords are the remaining passwords in P: $\{\forall \hat{\rho}^* \in P | \hat{\rho}^* \neq \hat{\rho}\}$. $\mathsf{PassSubmit}(\hat{\rho^*})$ will behave identically upon submission of a panic password (otherwise an adversary could distinguish the case where he is given a panic password).

Once $\mathsf{PassSubmit}(\hat{\rho})$ accepts $\hat{\rho}$, the client encodes $\hat{\rho}$ as a bitstring and appends a non-secret salt to prevent accidental collisions with other users. This string is supplied as input to a password-based key derivation function (PBKDF) for strengthening and encoding into $\mathbb{Z}_q^*$. For brevity, we denote this entire password processing procedure as ϕ: $\rho \leftarrow \phi(\hat{\rho}) = \mathsf{PBKDF}(\mathsf{PassSubmit}(\hat{\rho}) \| \mathtt{salt})$.

Perhaps through a user-guided tutorial familiarizing the voter with the system, the voter will generate α admissible passwords: $\hat{\rho}_1, \ldots, \hat{\rho}_\alpha$. The value of α will determine the soundness of the registration protocol.

[2] In reality, users are unlikely to choose uniformly from the entire dictionary and reach this maximum. The number of words can be increased to compensate for this.

[3] Proactive security can maintain the secrecy of the shares over time, both the number of shares and the threshold can be adjusted without a dealer, and more a complex access structure than m-out-of-n can be created.

Algorithm 1: Registration Protocol

```
Participants : Voter $\mathcal{V}_i$ and registrant $\mathcal{R}$
Public Input: Encryption parameters $p, q, g$, public key $e$, and soundness parameter $\alpha > 1$
Private Input ($V_i$): Ciphertexts $\{c_1, \ldots, c_\alpha\}$ as described below
Prior to the protocol, each voter should:
    for $k$ from 1 to $\alpha$ do
1       Choose a password $\hat{\rho}_k$.
2       Process password: $\rho_k \leftarrow \phi(\hat{\rho}_k)$.
3       Encrypt $g^{\rho_k}$ with random $r_k$: $c_k \leftarrow \mathsf{Enc}_e(g^{\rho_k}, r_k)$.
4       Complete a NIZKP of knowledge of plaintext $g^{\rho_k}$:
            $\pi_k \leftarrow \mathrm{NIZKP}_{pok}\{(\rho_k, r_k) : c_k = \mathsf{Enc}_e(g^{\rho_k}, r_k)\}$.
5       Record $\langle c_k, \pi_k \rangle$.
Registrar should:
6   Receive $\{\langle c_1, \pi_i \rangle, \ldots, \langle c_\alpha, \pi_\alpha \rangle\}$.
7   for $k$ from 1 to $\alpha$ do
8       Check $\pi_k$.
9       Rerandomize $c_k$ with random $r'_k$: $c'_k \leftarrow \mathsf{ReRand}(c_k, r'_k)$.
10      Print $\langle c'_k, (c_k, r'_k) \rangle$.
Each voter should:
11  Receive for each $k$: $\langle c'_k, (c_k, r'_k) \rangle$.
12  Optionally, rewind to line 7.
13  Choose $s \leftarrow [1, \alpha]$.
14  Erase $(c_s, r'_s)$.
15  Send $s$ to $R$.
Registrar should:
16  Receive $s$.
17  Publish $\langle \mathtt{VoterID}, c'_s \rangle$ on the Roster.
Each voter should:
18  After leaving, check that $c'_k \leftarrow \mathsf{ReRand}(c_k, r'_k)$ for all $k \neq s$.
19  Check that received $c'_s$ matches $\langle \mathtt{VoterID}, c'_s \rangle$ on the Roster.
```

Remarks: This protocol is completed *bare-handed* [32] with pre-computations and erasures. The proof of knowledge of an Elgamal plaintext is standard. The option to rewind is included to prevent coercion contracts [16].

An example value for α is 10. The password the voter wishes to register is in a random location in the list. Each is encrypted by the voter under the trustees' public key e. The voter prints out the list of ciphertexts on to a piece of paper, *e.g.,* with the ciphertexts encoded into barcodes. The registration protocol in Algorithm 1 includes the voter preparation protocol.

3.3 Registration

The registration protocol is completed by each voter $\mathcal{V}_i$. It is a two-party cut-and-choose protocol between a voter $\mathcal{V}_i$ and the registrar $\mathcal{R}$. The protocol is described in Algorithm 1. It is an adaptation of the Benaloh's voter initiated auditing [7], with a predetermined number of challenges. The voter enters the protocol with a list of α encrypted passwords $\{c_1, \ldots, c_\alpha\}$ and the protocol completes with a re-encryption of one of the ρ's being posted to an append-only broadcast channel, called the Roster. The protocol itself is conducted over an untappable channel which is instantiated as an in-person protocol.

The voter presents identification and is authorized to register. The voter is given a blank transcript card and enters a private booth that has a computer in it capable of printing and scanning barcodes. A transcript card has α rows and two columns. The second column for each row has a scratch-off surface. The voter is provided the option of downloading and printing a document from the Internet—with the intention that the voter could print her voter preparation sheet in the event that an adversary ensured she entered the registration process without her sheet. The computer has a barcode scanner, which the voter uses to submit her α ciphertexts.

The computer will rerandomize each ciphertext and print the value in the first column of the transcript card. Beside this value on the scratch-off surface, it will print the original ciphertext and the randomization used. The voter chooses one password to register: for that password, the voter will erase the original ciphertext and randomization by scratching off the appropriate cell.[4] It is assumed the voter cannot memorize or copy the randomization (*e.g.,* it is encoded into a barcode). The voter shreds her preparation sheet and retains the transcript card. The remaining $\alpha - 1$ re-encryptions can be shown to anyone and checked for correctness at home.

3.4 Election Set-up

The Roster is a universal registration. To prepare for an election, entries from the Roster are copied to smaller lists, called ElectionRosters. An ElectionRoster is specific to a particular election, precinct or district. The trustees will also modify the encrypted message in each entry from g^{ρ} to g_0^{ρ}, where g_0 is a unique publicly-known generator for that election. This prevents information leakage across elections.

Recall that Roster entries are encrypted with ρ in the exponent: $\{c_1, c_2\} = \{g^r, g^{\rho} y^r\}$. For each ElectionRoster, each trustee chooses $b_i \leftarrow_r \mathbb{G}_q$. Then each trustee will in turn blind each ciphertext on the ElectionRoster as follows: output g^{b_i}, $c_1^{b_i}$ and $c_2^{b_i}$, and prove knowledge of b_i such that $g, c_1, c_2, g^{b_i}, c_1^{b_i}, c_2^{b_i}$ form a threewise DH-tuple with a NIZKP (*cf.* [12]). The next trustee will repeat the process using the previous trustee's output as input. All outputs are posted to an appendix on the ElectionRoster. Let $b_0 = \prod b_i$ and $g_0 = g^{b_0}$. The blinding sequence re-randomizes each ciphertext from r to $r' = r \cdot b_0$ and changes the encrypted message from g^{ρ} to g_0^{ρ}. The public and private key shares are the same. The public value g_0 will be used during the casting protocol.

3.5 Casting

The casting protocol involves a voter $\mathcal{V}_i$ and the election authority. The protocol is described in Algorithm 2. The communication occurs over an anonymous channel. The anonymity is to be built into the voter's client using an anonymous remailer or onion routing technology.

$\mathcal{V}_i$ submits a commitment to her asserted (*i.e.,* real or panic) password, $g_0^{\rho^*}$, and a rerandomization of her entry on the ElectionRoster, c'. If ρ^* matches the ρ encrypted in c', the pre-tallying protocol will ensure the ballot is included in the final result. Otherwise if it does not match, it will be discarded in a way that is unlinkable to the original submission.

$\mathcal{V}_i$ must prove that c' is from the ElectionRoster. Simply including her entry without rerandomizing it reveals that she submitted a vote. To prevent abstention attacks, she instead rerandomizes it, draws an additional $\beta - 1$ entries randomly from the ElectionRoster, and proves in zero-knowledge that c' is a rerandomization of one of these β entries (her entry plus the additional ones). β acts as an anonymity set. Most voters will use a small value of β, however privacy-conscious voters can also (at extra computational cost) cast a stealth vote where β includes all the entries on the ElectionRoster.

Selections is designed to be versatile with different options for capturing and tallying the votes themselves. Thus we leave the information the voter submits with regard to their vote abstractly as $\mathbf{B}$ while only requiring that $\mathbf{B}$ is submittable to a mix-network. For example, $\mathbf{B}$ could be an encryption of the preferred candidate(s) or a tuple of cryptographic counters for each option, accompanied by proofs of validity as appropriate. Note that our coercion-resistance guarantee extends only to the delivery of valid, eligible, and unique $\mathbf{B}$ values, and care should be taken to ensure that tallying these values does not break coercion-resistance.

[4] Under each scratch-off could be a pre-committed code in the form of a barcode, which the voter could scan to prove to the system that she scratched off the correct cell. We leave the details for such an augmented transcript card for future work.

Algorithm 2: Casting Protocol

```
Participants  : Voter $\mathcal{V}_i$ and election authority
Public Input: Encryption parameters $g, p, q$, election parameter $g_0$, public key $e$, ElectionRoster, and
              anonymity parameter $\beta$
Private Input ($V_i$): Password (either real or panic) $\hat{\rho}^*$
Each voter should:
1     Find $c$ for her VoterID from ElectionRoster.
2     Rerandomize $c$ with random $r$: $c' \leftarrow \mathsf{ReRand}(c, r)$.
3     Randomly select $\beta$-1 other $c_k$ from the ElectionRoster.
4     Form set $\mathcal{C} = \{c, c_1, \ldots, c_{\beta-1}\}$ in order of appearance on ElectionRoster.
5     Generate a NIZKP that $r$ rerandomizes 1-out-of-$\beta$ of $\mathcal{C}$.
        $\pi_1 \leftarrow \text{NIZKP}_{pok}\{(r) : c' = (\mathsf{ReRand}(c, r) \vee \mathsf{ReRand}(c_1, r) \vee \ldots)\}$.
6     Encode asserted password into $\mathbb{Z}_q^*$: $\rho^* \leftarrow \phi(\hat{\rho}^*)$.
7     Commit to $\rho^*$: $g_0^{\rho^*}$.
8     Complete an NIZKP of knowledge of $\rho^*$:
        $\pi_2 \leftarrow \text{NIZKP}_{pok}\{(\rho^*) : g_0, g_0^{\rho^*}\}$.
9     Complete a ballot and retain ballot information $\mathbf{B}$.
10    Send $\left\langle g_0^{\rho^*}, c', \mathbf{B}, \pi_1, \pi_2 \right\rangle$ to $A$.
Authority should:
11    Publish $\left\langle g_0^{\rho^*}, c', \mathbf{B}, \pi_1, \pi_2 \right\rangle$ on AllVotes.
```

Remarks: Rerandomization proofs are formed with a knowledge of a DDH-tuple proof due to Chaum and Pedersen [12]. 1-out-of-m proofs are due to a heuristic by Cramer, Damgard and Schoenmakers [18]. Proof of knowledge of a discrete log is due to Schnorr [35]. Parameter β represents the voter's anonymity set.

Each ZKP uses the Fiat-Shamir heuristic to make it non-interactive, and each uses the values $\left\langle g_0^{\rho^*}, c', \mathbf{B} \right\rangle$ in creating the challenge. This prevents an adversary from replaying any of the proofs individually. The submission is posted to an append-only broadcast channel called AllVotes.

If the voter is under coercion, she makes up a panic password and follows the rest of the protocol as specified. She can later cast a stealth vote with her real password. If a voter wants to overwrite a previous vote submitted under password ρ^*, the inclusion of the same $g_0^{\rho^*}$ will indicate in cleartext that it is an overwrite. Therefore, she should use the same β entries from the ElectionRoster as her anonymity set. Also note that the inclusion of the same $g_0^{\rho^*}$ across multiple elections would also be linkable if the value g_0 was not changed in each election.

3.6 Pre-tallying

The pre-tallying protocol involves an authorized subset of the N election trustees. The protocol is described in Algorithm 3. The protocol takes AllVotes and produces a shorter list of only the most recently cast votes for voters that supply the correct, registered password. Checking the validity of each vote is linear in β. For these voters, the list includes just the ballot information, $\mathbf{B}$, in an order that is unlinkable to the order of submission. How this list is further processed to produce a tally is dependent on the voting system our system interfaces with (which is why this is called a pre-tally). In a simple case, $\mathbf{B}$ is an encryption of the voter's selections (with a proof of knowledge) and the final step is jointly-decrypting each $\mathbf{B}$ from the list.

3.7 Voter Revocation

Between elections, Selections offers a way of choosing which registered voters are eligible or not to vote in a particular election. In Selections, it is also possible to revoke a voter at any point before the pre-tallying

Algorithm 3: Pre-Tallying Protocol

```
Participants : Authorized set of trustees $\mathcal{T}_1, \ldots, \mathcal{T}_m$ and election authority
Public Input: AllVotes
Private Input ($T_i$): Share of private key, $d_i$
Authority should:
1    For each entry, check $\pi_1$ and $\pi_2$.
2    Remove all tuples with invalid proofs to form list ProvedVotes
3    Find all entries in ProvedVotes with duplicate values for $g_0^\rho$.
4    Remove all but the most recent to form list UniqueVotes.
Each participating trustee should:
5    Participate in verifiable mix network for shuffling UniqueVotes.
       Note: the initial $g_0^{\rho^*}$ is treated as $c_\rho = \mathsf{Enc}_e(g_0^{\rho^*}, 0)$.
6    Output is AnonUniqueVotes.
Each participating trustee should:
7    for each entry in AnonUniqueVotes do
8        Read entry $\langle c_\rho, c', \mathbf{B} \rangle$.
9        Participate in a plaintext-equality test of $c_\rho$ and $c'$:
           $\{\mathtt{T}, \mathtt{F}\} \leftarrow \mathrm{PET}_{d_i}(c_\rho, c')$.
Authority should:
10   Remove all tuples with PET outcome of False to form list ValidVotes.
Each participating trustee should:
11   for each entry in ValidVotes do
12       Participate in threshold decryption of $\mathbf{B}$.
```

Remarks: Various protocols exist for verifiable mix networks. An efficient technique with statistical soundness is randomized partial checking [22]. The plaintext equality test (PET) is due to Juels and Jakobsson [23]. The output of this protocol is the ballot information for unique and registered voters in an order that is unlinkable to the order of submission.

protocol. This could arise because the voter forgot their password (and is issued a new one) or registered to vote online but decides to vote in person. For every submitted vote that includes the revoked voter among its β registered voters in its anonymity set (which will include any potentially valid vote by the revoked voter herself), the submitted password is checked against the revoked voter's entry on the ElectionRoster using a plaintext-equality test. Revocation of this type is the same in Civitas and is not possible in AFT. Coercion-resistance does not necessarily extend to all types of revocation.

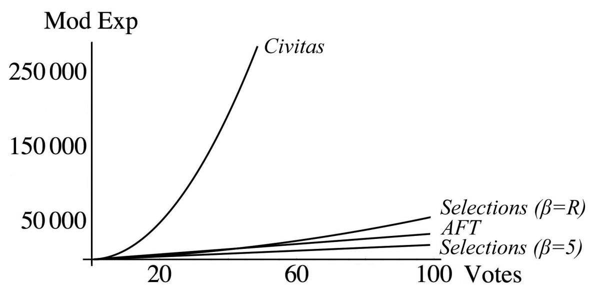
4 Performance

We compare the performance of Selections to JCJ as implemented in Civitas [17] and to AFT [4]. We make a number of standardizing assumptions to facilitate a better comparison. We assume a single registrar, T trustees, R registered voters, and V_0 submitted votes. We do not use the "blocking" technique of Civitas, which could improve the performance of all three systems. Of the V_0 submitted votes, $V_1 \leq V_0$ have correct proofs, $V_2 \leq V_1$ are not duplicates, and $V_3 \leq V_2$ correspond to registered voters. Recall that for Selections, α are the number of submitted ciphertexts in registration and β is the size of the voter's anonymity set during casting.

We use Elgamal encryption in each system, with proofs of knowledge of plaintexts where appropriate. We assume each trustee participates in decryption (*i.e.,* distributed instead of threshold). We assume that ballot material is encrypted with only a proof of knowledge (no additional proofs of well-formedness). The pre-tallying protocol ends with a list of V_3 encrypted ballots. Finally, we assume mixing is done with a re-

		Civitas	AFT	Selections
Registration	Registrar	7	9	2α
	Voter	11	10	4α-1
Casting	Voter	10	24	$(2\beta+9)$
Pre-Tally	Check Proofs	$4V_0$	$20V_0$	$(4\beta+6)V_0$
	Remove Duplicates	$(1/2)(V_1^2-V_1)(8T+1)$	—	—
	Check Removal	$(1/2)(V_1^2-V_1)(8T+1)$	—	—
	Mix	$8V_2T+4RT$	$20V_2T$	$12V_2T$
	Check Mix	$4V_2T+2RT$	$10V_2T$	$6V_2T$
	Remove Unregistered	$(8A+1)V_2R$	$(16T+8)V_2$	$(8T+1)V_2$
	Check Removal	$(8A+1)V_2R$	$(16T+10)V_2$	$(8T+1)V_2$

Table 1. Comparison of the efficiency of the main protocols in Civitas, AFT, and Selections, measured with modular exponentiations.



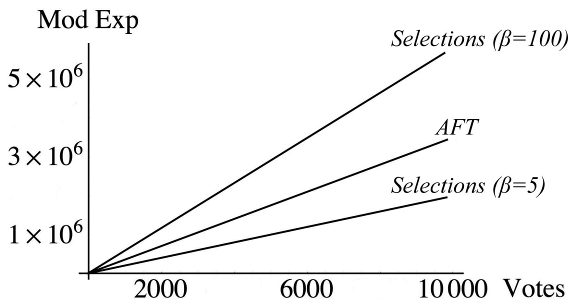


Fig. 1. Pre-tallying efficiency in modular exponentiations with $T = 5$ and variable $R = V_0 = V_1 = V_2$.

encryption mixnet and randomized partial checking [22], where each authority produces two mixes and half of these re-encryptions are checked. The complete details of our comparison are in Appendix A.

Table 1 shows the efficiency in terms of modular exponentiations and Figure 4 shows a comparison of the pre-tallying protocols. With full forced-abstention, Selections is quadratic like Civitas but with a smaller constant. When β is a constant, Selections is linear in the number of submitted votes like AFT. The exact value of β dictates which is exactly faster. Recall our goal was not to improve the efficiency of AFT but rather to create a password-based system with similar performance to AFT. To this end, we are successful.

5 Security Analysis (Abstract)

5.1 Soundness of Registration

In Appendix B, we show that the Registration protocol is a cut-and-choose argument for $\{(c, r) : c' = \mathsf{ReRand}_e(c, r)\}$. It takes soundness parameter α (*e.g.,* $\alpha = 10$). It is complete and has statistical soundness of $1 - \alpha^{-1}$ for a single run. After k runs, soundness increases to $1 - \alpha^{-k}$. Designing a bare-handed argument with stronger soundness (*e.g.,* $1 - 2^{-\alpha}$ for a single run) is open. With erasures, the protocol has deniability for c and computational secrecy for r.

The protocol does not protect against covert channels. This has been addressed in the literature with verifiable random functions [20] or pre-committed randomness [19]. The protocol protects against coercion contracts [16] with rewinds. Rewinds can be eliminated if the voter commits to their choice of password at the beginning of the protocol.

5.2 Coercion-Resistance

In Appendix C, we show several results concerning the coercion-resistance (cr) of Selections. Juels *et al.* define an experiment $\mathbf{Exp}^{\mathrm{cr}}_{ES,\mathcal{A}'}$ for non-adaptive adversary $\mathcal{A}$ in election system ES, as well as an ideal $\mathbf{Exp}^{\mathrm{cr-ideal}}_{ES,\mathcal{A}'}$. The critical component in $\mathbf{Exp}^{\mathrm{cr}}_{ES,\mathcal{A}'}$ is a coin flip $b \leftarrow_r \{0,1\}$ defining a corrupted voter's behaviour. If $b = 0$, the voter provides (in Selections) a panic password to the adversary and casts a vote with her real password. If $b = 1$, the voter complies with the adversary and provides her real password. In both cases, the adversary can use the supplied password to submit a vote. We define the advantage of $\mathcal{A}$, where an output of 1 is the adversary correctly stating b, as,

$$\mathbf{adv}^{\mathrm{cr}}_{ES,\mathcal{A}} = |\mathbf{Pr}[\mathbf{Exp}^{\mathrm{cr}}_{ES,\mathcal{A}'}(\cdot) = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathrm{cr-ideal}}_{ES,\mathcal{A}'}(\cdot) = 1]|.$$

Case 1: $\beta = R$. We show that when β is the full roster R, $\mathbf{adv}^{\mathrm{cr}}_{ES,\mathcal{A}}$ for Selections is negligible. Setting $\beta = R$ does impact performance. Vote casting is linear in the size of the ElectionRoster and Pre-Tallying is quadratic. However the only quadratic component is checking the 1-out-of-β rerandomization proof, where the proof length is linear in the size of the roster. These proofs can be pre-checked, while voters submit votes.

Case 2: $\beta =$ const. We show that when β is constant (*e.g.,* 5 or 100), $\mathbf{adv}^{\mathrm{cr}}_{ES,\mathcal{A}} < \delta$, where δ is small but non-negligible. Recall there are V_2 votes with valid proofs and R entries on the ElectionRoster. Let $\mathbf{F}(k;p,n)$ be the cumulative distribution function of a Binomial distribution with n trials, success probability p, and k successes. We show that δ for this case is,

$$\delta = \frac{1}{2}(F(\frac{\beta V_2}{R}; V_2, \frac{\beta}{R}) + 1 - F(\frac{\beta V_2}{R} - 1; V_2 - 1, \frac{\beta}{R})).$$

Case 3: $\beta \geq$ const. Finally we consider the case where β is required to be at least a constant value (*e.g.,* 5 or 100) but voters can submit stealth votes where $\beta = R$. We show that if a corrupted voter's coercion-resistant strategy is to submit their real vote as a stealth vote, $\mathbf{adv}^{\mathrm{cr}}_{ES,\mathcal{A}}$ is negligible. We do make one small change to $\mathbf{Exp}^{\mathrm{cr}}_{ES,\mathcal{A}'}$: instead of the corrupted voter's real vote being appended to the cast ballots, it is inserted at a random place (*i.e.,* she votes her real ballot at some arbitrary time after being coerced).

6 Concluding Remarks

Selections has many benefits: users can evade coercion without computations, registration does not require a computer, tallying the votes is linear in the number of voters, and voters can have their registration efficiently revoked. Future work includes providing protection against untrusted platforms, perhaps by merging Selections with existing work on code voting.

7 Acknowledgements

We acknowledge Richard Carback for suggesting that panic passwords could be employed in an Internet voting system to prevent undue influence. We thank Aleks Essex, Michael Clarkson, various participants of SecVote 2010, and the reviewers for useful feedback on the protocol and paper. This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC)—the first author through a Canada Graduate Scholarship and the second through a Discovery Grant.

References

1. A. Acquisti. Receipt-free homomorphic elections and write-in ballots. Technical report, IACR Eprint Report 2004/105, 2004.

2. B. Adida. Helios: web-based open-audit voting. In *USENIX Security Symposium*, pages 335–348, 2008.
3. B. Adida, O. d. Marneffe, O. Pereira, and J.-J. Quisquater. Electing a university president using open-audit voting: Analysis of real-world use of Helios. In *EVT/WOTE*, 2009.
4. R. Araujo, S. Foulle, and J. Traore. A practical and secure coercion-resistant scheme for remote elections. In *Frontiers of Electronic Voting*, 2007.
5. R. Araujo, S. Foulle, and J. Traore. A practical and secure coercion-resistant scheme for internet voting. *Toward Trustworthy Elections*, LNCS 6000, 2010.
6. R. Araujo, N. B. Rajeb, R. Robbana, J. Traore, and S. Yousfi. Towards practical and secure coercion-resistant electronic elections. In *CANS*, 2010.
7. J. Benaloh. Simple verifiable elections. In *EVT*, 2006.
8. J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *ACM STOC*, 1994.
9. R. T. Carback, D. Chaum, J. Clark, J. Conway, A. Essex, P. S. Hernson, T. Mayberry, S. Popoveniuc, R. L. Rivest, E. Shen, A. T. Sherman, and P. L. Vora. Scantegrity II municipal election at Takoma Park: the first E2E binding governmental election with ballot privacy. In *USENIX Security Symposium*, 2010.
10. D. Chaum. Surevote: Technical overview. In *WOTE*, 2001.
11. D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A. Ryan, E. Shen, and A. T. Sherman. Scantegrity II: end-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In *EVT*, 2008.
12. D. Chaum and T. P. Pedersen. Wallet databases with observers. In *CRYPTO*, 1992.
13. D. Chaum, P. Y. A. Ryan, and S. Schneider. A practical voter-verifiable election scheme. In *ESORICS*, 2005.
14. J. Clark. *Democracy Enhancing Technologies: Toward deployable and incoercible E2E elections.* PhD thesis, University of Waterloo, 2011.
15. J. Clark and U. Hengartner. Panic passwords: Authenticating under duress. In *USENIX HotSec*, 2008.
16. J. Clark, U. Hengartner, and K. Larson. Not-so-hidden information: optimal contracts for undue influence in E2E voting systems. In *VOTE-ID*, 2009.
17. M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a secure voting system. In *IEEE Symposium on Security and Privacy*, pages 354–368, 2008.
18. R. Cramer, I. Damgard, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, 1994.
19. A. J. Feldman and J. Benaloh. On subliminal channels in encrypt-on-cast voting systems. In *EVT/WOTE*, 2009.
20. R. W. Gardner, S. Garera, and A. D. Rubin. Coercion resistant end-to-end voting. In *FC*, 2009.
21. S. Heiberg, H. Lipmaa, and F. v. Laenen. On e-vote integrity in the case of malicious voter computers. In *ESORICS*, 2010.
22. M. Jacobsson, A. Juels, and R. L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *USENIX Security Symposium*, pages 339–353, 2002.
23. M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In *ASIACRYPT*, 2000.
24. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT*, 1996.
25. M. Jakobsson, E. Stolterman, S. Wetzel, and L. Yang. Love and authentication. In *CHI*, 2008.
26. R. Joaquim and C. Ribeiro. Codevoting: protection against automatic vote manipulation in an uncontrolled environment. In *VOTE-ID*, 2007.
27. A. Juels, D. Catalano, and M. Jacobsson. Coercion-resistant electronic elections. In *ACM WPES*, 2005.
28. C. Kane. Voting and verifiability: interview with Ron Rivest. *RSA Vantage Magazine*, 7(1), 2010.
29. R. Kusters, T. Truderung, and A. Vogt. A game-based definition of coercion- resistance and its application. In *CSF*, 2010.
30. T. Moran and M. Naor. Split-ballot voting: Everlasting privacy with distributed trust. In *ACM CCS*, 2007.
31. T. P. Pedersen. A threshold cryptosystem without a trusted party. In *EUROCRYPT*, 1991.
32. R. L. Rivest and W. D. Smith. Three voting protocols: threeballot, VAV, and twin. In *EVT*, 2007.
33. P. Y. A. Ryan, D. Bismark, J. Heather, S. Schneider, and Z. Xia. Pret a voter: a voter-verifiable voting system. *IEEE TIFS*, 4(4), 2009.
34. P. Y. A. Ryan and V. Teague. Pretty good democracy. In *Workshop on Security Protocols*, 2009.
35. C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptography*, 4, 1991.
36. W. D. Smith. New cryptographic election protocol with best-known theoretical properties. In *Frontiers in Electronic Elections*, 2005.
37. O. Spycher, R. Koenig, R. Haenni, and M. Schlapfer. A new approach towards coercion-resistant remote e-voting in linear time. In *FC*, 2011.
38. V. Teague, K. Ramchen, and L. Naish. Coercion resistant tallying for STV voting. In *EVT*, 2008.

39. Y. Tsiounis and M. Yung. On the security of ElGamal based encryption. In *PKC*, pages 117–134, 1998.
40. D. Unruh and J. Muller-Quade. Universally composable incoercibility. In *CRYPTO*, 2010.
41. S. G. Weber, R. S. d. Araujo, and J. Buchmann. On coercion-resistant electronic elections with linear work. In *ARES*, 2007.

A Performance Comparison

Selections and Civitas both use Elgamal encryption, while JCJ and AFT use a modified variant of Elgamal that adds one modular exponentiation to encryption, re-randomization, and decryption. For performance comparison, we use Elgamal. Elgamal requires 2 modular exponentiations for encryption, 2 for reencryption, and 1 for decryption. Proof of knowledge of a plaintext requires 3 exponentiations to compute and 4 to verify.

Selections, JCJ, and AFT use threshold decryption while Civitas uses distributed decryption. For streamlining tasks involving all trustees with ones involving an authorized set, we assume all trustees are needed. Thus we use distributed decryption for performance comparison. In addition to the 1 modular exponentiation for plain decryption, we need an additional $3T$ where T are the number of trustees. To verify that the decryption was done correctly requires $4T+1$ exponentiations.

Part of the distributed decryption process involves each trustee proving that a tuple of values is a Diffie-Hellmen tuple. With Chaum-Pedersen, this is 2 exponentiations to compute and 4 to verify. This proof can also prove values are rerandomized correctly. The knowledge of discrete log, with Schnorr, is 1 to compute and 2 to verify. Knowledge of a representation requires 1 to compute and 3 to verify. For one of the proofs during casting, AFT requires knowledge of a representation with 3 generators. This requires 1 to compute and 4 to verify.

ZKPs can be made 1-out-of-m by scaling both the computing and verifying requirements by m. ZKPs can be made designated verifier by adding 1 exponentiation to both computing and verifying (if the verifier's key is known in advanced). We assume all commitments are based on hash functions, with the exception of the Chameleon commitment used in a designated verifier proof. We assume this commitment is a Pedersen commitment. As is standard, we also assume Pedersen commitments (and accumulators of a similar structure) require a single exponentiation while they actually contain multiple exponents.

A plaintext equality test requires each of the T trustees to blind both ciphertext components: $2T$ exponentiations; compute a Chaum-Pedersen proof: $2T$; and perform a distributed decryption: $4T+1$. This totals $8T+1$ for computing the test and $8T+1$ to verify it was performed correctly.

Finally, a mix-network requires each of the T trustees to reencrypt each of N ciphertexts in the input: $2N$ exponentiations. With randomized partial checking, each trustee actual performs two mixes: increasing the workload to $4N$. Finally, the input is not a list of single ciphertexts but rather tuples that include multiple ciphertexts: each of which has to be individually reencrypted. If there are N tuples with τ ciphertexts in each tuple, mixing costs $4NT\tau$. With randomized partial checking, half of the recencryptions have their randomization revealed. Checking these requires half the cost of mixing: $2NT\tau$.

A.1 Civitas

During registration, the registrar computes an encryption (2) and proves knowledge (3). It then computes a second encryption of the same value (2) and proves it is a rerandomization. This second proof is designated verifier (2+1). The voter is supplied with the plaintext and randomization for the second value. She checks the ciphertext is correct through encryption (2), verifies the knowledge of plaintext (4), and verifies the proof of randomization (4+1).

Vote casting consists of two encryptions (4) and a proof of simultaneous knowledge (6) of the plaintexts.

Tallying begins by checking each proof ($4V_0$). Then PETs are performed between each pair of submitted votes with correct proofs ($\binom{V_1}{2}(8T+1)$) and checking these PETs (same). The V_1 tuples, with two ciphertexts in each tuple, are mixed ($8V_2T$) and checked ($4V_2T$). The R values on the Roster are also mixed, with one ciphertext in each tuple: mix ($4RT$) and check ($2RT$). Finally, PETs are performed between each pair of entries in R and V_2 ($(8T+1)V_2R$) and checked ($(8T+1)V_2R$).

A.2 AFT

For the comparison, we use the 2010 version of AFT [5] (*cf.* [4])

During registration, the registrar forms a credential (3) and proves it's validity with respect to two keys using two designated verifier Chaum-Pedersen proofs (6). The voter checks these proofs (10).

Vote casting consists of four encrypted values (8), where two of the plaintexts were computed prior to encryption (2) and one submitted value is computed but not encrypted (1). The voter proves knowledge of the four encryptions (12). The voter also proves a representation (1) that relates an encrypted value to the non-encrypted value.

Tallying begins by checking the plaintext proofs (16) and representation (4). Duplicates can be removed without computing any exponentiations. AFT includes a step where a submitted value is encrypted: we omit this step as the first mix can perform this at no additional cost. The mixing is performed on tuples with five ciphertexts each ($20V_2T$ plus $10V_2T$ to check). Finally, removing unregistered voters requires, per vote, an exponentiation (1), Chaum-Pedersen to prove matching exponents (2), a PET ($8T+1$), and then the same three steps a second time. Finally, the two PETs and proofs are checked ($16T+10$) for each vote left.

A.3 Selections

Prior to registration, the voter prepares α encryptions (2α). The registrar rerandomizes each (2α). The voter checks the rerandomization of all but one ($2(\alpha-1)$).

Vote casting consists of a computed value (1), a proof of knowledge of this value (1), a rerandomization (2), a 1-out-of-β proof of rerandomization (2β), an encrypted value (2), and a proof of knowledge of the plaintext for this value (3).

During tallying, the three proofs are checked ($2+4\beta+4$) for each cast vote. Duplicates are removed without any exponentiations. The mix is of tuples with three ciphertexts ($12V_2T$ plus $8V_2T$ to check). Finally, a single PET is performed for each remaining tuple ($8T+1$).

B Registration

The Registration protocol is a cut-and-choose argument given by registrar $\mathcal{R}$ to $\mathcal{V}_i$ for $\{(c,r) : c' = \mathsf{ReRand}_e(c,r)\}$. It is a variant of Benaloh's voter initiated auditing [7]. The protocol actually includes $\alpha > 1$ ciphertexts, $c_1, \ldots, c_\alpha$, and the s^{th} one will be chosen for submission. In discussing general properties of Registration, we refer to c_s as simply c to keep the discussion applicable to protocols that do not have a set of ciphertexts (*i.e.,* are not based on cut-and-choose).

We first sketch informally several properties one may want from a registration (and voting) protocol.

1. *Integrity*: $\mathcal{V}_i$ should be convinced that c' rerandomizes c.
2. *Secrecy*: An adversary $\mathcal{A}$ should not be able to determine that c was rerandomized from the output of the protocol.
3. *Receipt-Free*: $\mathcal{A}$ in collusion with $\mathcal{V}_i$ should not be able to determine that c was rerandomized from any of the following: supplying inputs, examining the output, or examining a transcript of the protocol kept by $\mathcal{V}_i$.
4. *Covert-Free*: $\mathcal{R}$ cannot inject information into the transcript that would be useful for coercion or breaking secrecy.
5. *Dispute-Free*: If $\mathcal{R}$ does not follow the protocol, $\mathcal{V}_i$ can prove the protocol was not followed.
6. *Bare-Handed*: $\mathcal{V}_i$ can complete the protocol without performing computations during the protocol.

We will show that the protocol we provide has integrity (correctness and soundness), secrecy, receipt-freeness and is bare-handed. We do not attempt to prevent covert channels in the basic protocol but provide some discussion toward this point, as well as only discussing disputes.

Given a transcript of the Registration protocol, the c' can be either accepted or rejected as a rerandomization of c. If Registration is run correctly, the decision will always be to accept (completeness). If the protocol is not correct, the decision will be to reject with a high probability (soundness). Finally, the outputs do not provide any information that can be used by a computationally bounded adversary to determine any non-negligible information about the secrets in the protocol: c and r (computational secrecy). Finally, we show $\mathcal{V}_i$ has a coercion-resistant strategy (receipt-freeness).

Assumptions. We make the following assumptions regarding the security of Registration:

1. An adversary $\mathcal{A}$ can corrupt $\mathcal{V}_i$ prior to the protocol and provide $\mathcal{V}_i$ with inputs to the protocol,
2. $\mathcal{A}$ can corrupt V_i prior to the protocol and require $\mathcal{V}_i$ to provide a plausible transcript of the protocol afterward,
3. The protocol between $\mathcal{V}_i$ and $\mathcal{R}$ is run over an untappable channel unaccessible to $\mathcal{A}$,
4. V_i complies with erasures,
5. $\mathcal{A}$ cannot corrupt $\mathcal{R}$, and
6. $\mathcal{A}$ is bounded to probabilistic polynomial time (PPT).

For the proof, $\mathcal{R}$ is unbounded in computation power, while $\mathcal{V}_i$ is PPT-bounded. These bounds are used to demonstrate that soundness is statistical and not computational. Either entity may employ a malicious strategy and we denote this with a prime ($\mathcal{R}', \mathcal{V}_i'$).

Completeness. If $\mathcal{R}$ follows the protocol, he will produce correct rerandomizations for each of the α ciphertexts, including the one chosen. $\mathcal{V}_i$ will accept this transcript by checking the $\alpha - 1$ ciphertexts and finding they are correct. Note that we are demonstrating the protocol is complete, not the rerandomization of the chosen ciphertext. It is possible that the rerandomization for the chosen ciphertext is correct but it is not correct for another ciphertext. In this case, the $\mathcal{V}_i$ will reject the transcript despite it being correct for the only value of importance. Completeness does not capture false-negatives.

Soundness. We now consider whether $\mathcal{V}_l$ will ever accept a transcript when c' is not a rerandomization of c. If the probability of rejecting a false proof is much greater than 1/2, we say Registration is sound. Let c_s be the ciphertext chosen for submission and let c_{-s} be one ciphertext that is not c_s.

We assume that the transcript will be checked by $\mathcal{V}_i$. In reality, only a fraction of $\mathcal{V}_i$'s will check. Any probabilities should be appropriately scaled according to the fraction of $\mathcal{V}_i$'s checking. For simplicity, we omit this factor.

A malicious $\mathcal{R}'$ rerandomizes $c_1, \ldots, c_\alpha$ before learning the value of s. If $\mathcal{R}'$ incorrectly rerandomizes more than one ciphertext, the transcript will be rejected with certainty. Therefore $\mathcal{R}'$ must choose one. If $\mathcal{V}_i$ chooses s at random, $\mathcal{R}'$ can do no better than choosing randomly. Denote $\mathcal{R}'$'s choice as $c_{\hat{s}}$. If $s \neq \hat{s}$, $\mathcal{V}_i$ rejects $\mathcal{R}'$'s false transcript. If $s = \hat{s}$, $\mathcal{V}_i$ accepts $\mathcal{R}'$'s false transcript. The probability $\mathbf{Pr}[s = \hat{s}] = \alpha^{-1}$. Therefore, the soundness of Registrar is:

$$\mathbf{Pr}[\mathtt{REJECT}_{(\mathcal{R}',\mathcal{V}_i)}] = 1 - \alpha^{-1}. \tag{1}$$

If $\mathcal{R}'$ creates false transcripts for k voters, the probability of at least one $\mathcal{V}_1, \ldots, \mathcal{V}_k$ rejecting the false transcript is:

$$\mathbf{Pr}[\mathtt{REJECT}_{(\mathcal{R}',\mathcal{V}_1)} \vee \ldots \vee \mathtt{REJECT}_{(\mathcal{R}',\mathcal{V}_k)}] = 1 - \alpha^{-k}. \tag{2}$$

For example, to achieve a probability of detection of at least 99.9%, $\{\alpha = 10, k = 3\}$ or $\{\alpha = 2, k = 10\}$ are sufficient.

Computational Secrecy. Under assumption 4, $\mathcal{V}_i$ complies with erasures. We assume additionally under this compliance that $\mathcal{V}_i$ cannot commit the values to memory. We do not consider here the issue of a voter memorizing a few bits of the values, which could make for interesting future work. We also note that while the voter could record the values with a device, this device could also be used to record how they vote in an in-person voting protocol. We only claim our system to be as secure as the baseline of an in-person voting protocol. Erasures can be enforced by having an erasure confirmation code printed under the scratch-off. In Section D, we collect features for enhanced verification cards to be explored in future work.

With erasures, a transcript of the protocol does not contain the values (c_s, r_s) for c_s'. However $\mathcal{V}_i'$ (*i.e.*, malicious $\mathcal{V}_i$) has the value c_s. Therefore $\mathcal{V}_i'$ can assert that the value of c_s is $\hat{c}_s$, which may or may not be true. The adversary must decide if the asserted $\hat{c}_s$ and the rerandomized value c'_s encrypt the same plaintext ρ. Toward a contradiction, assume the adversary has an efficient plaintext equality (peq) algorithm $\{\texttt{T}, \texttt{F}\} \leftarrow A^{\text{PEQ}}(\hat{c}_s, c'_s)$ to determine this equality.

$\mathcal{A}$ can use A^{PEQ} to win the CPA game. In the CPA game, $\mathcal{A}$ provides p_0 and p_1 to oracle $\mathcal{O}$, $\mathcal{O}$ flips coin $b \leftarrow_r \{0, 1\}$, and returns challenge ciphertext $c_b = \mathsf{Enc}_e(p_b, r)$. $\mathcal{A}$ encrypts p_0 as c_0 and submits queries $A^{\text{PEQ}}(c_0, c_b)$. If $\texttt{T}$, $\mathcal{A}$ guesses $b = 0$ and otherwise $b = 1$. Since the encryption is CPA-secure, $A^{\text{PEQ}}(\hat{c}_s, c'_s)$ cannot be efficient against the encryption and the PPT-bounded adversary (assumption 6) cannot decide if $\mathcal{V}'_i$ asserted a correct value.

Receipt-Freeness. Under assumption 1, $\mathcal{A}$ can provide inputs for $\mathcal{V}'_i$. For example, $\mathcal{A}$ could provide a list of α ciphertexts $c_1, \ldots, c_\alpha$ for $\mathcal{V}'_i$ to use. $\mathcal{V}'_i$ however can replace one of $\mathcal{A}$'s ciphertexts with her desired c_s and choose it for submission. After the erasure, $\mathcal{A}$ cannot distinguish if she supplied his ciphertext or her own ciphertext under the argument given above for computational secrecy.

However, $\mathcal{V}'_i$ has one further input to the protocol and that is the value of s itself. $\mathcal{A}$ can exploit this with a coercion contract. Assume $\mathcal{A}$ provides $\mathcal{V}'_i$ with α ciphertexts and $\mathcal{V}'_i$ substitutes her own for one of them. Further assume that a communication channel exists between $\mathcal{A}$ and V'_i. If such a channel existed, $\mathcal{V}'_i$ could tell $\mathcal{A}$ the values she received and $\mathcal{A}$ could tell her a value of s. Unless if $\mathcal{V}_i$ placed her substituted value in the correct place, her non-compliance will be caught.

$\mathcal{A}$'s best strategy is to chose a random value for s. The goal of a coercion contract is to replace the interactive channel between $\mathcal{A}$ and $\mathcal{V}'_i$ (which does not actually exist) with a non-interactive call to a random oracle made by the voter to receive s and re-queried by $\mathcal{A}$ to verify compliance. For example, $\mathcal{A}$ could require $\mathcal{V}_i$ to submit the smallest rerandomized value c'_i generated by $\mathcal{R}$. If this generation is random, then the implied value of s will be random as well. Since the voter has to choose where to substitute her value before seeing what such an s will be, she will get caught with probability $1 - \alpha^{-1}$.

In order to provide receipt-freeness, $\mathsf{Registration}$ allows $\mathcal{V}_i$ to rewind the protocol upon seeing the list of rerandomized values. In practice, they could be displayed and rewound until satisfactory and only then printed. An alternative approach is to have the voter commit to s prior to the protocol. This could be accomplished bare-handed by having the voter mark beside the cell she will scratch-off or putting s in a sealed envelope, which will be checked when she leaves.

Covert-Freeness. The protocol does not protect against covert channels. The issue of covert channels has been addressed in the literature with verifiable random functions [20] or pre-committed randomness [19]. However if $\mathcal{R}$ is printing values, it is very easy for $\mathcal{R}$ to leak information with slight variations in how things are printed. Further, $\mathcal{R}$ cannot prove anything that is interesting to the adversary. It can assert what V'_i supplied it with but these values could be easily simulated: the erased values are c and r' for c'. A simulator can generate such values: just rerandomize c' with r'' to get c'' and claim c'' and r'' were submitted by the voter.

Since under assumption 5 $\mathcal{A}$ cannot corrupt $\mathcal{R}$, $\mathcal{A}$ cannot trust any assertions leaked covertly by $\mathcal{R}$ on the face of the assertion itself. The assertions must include a sound argument for (or proof of) what is being asserted.

C Coercion-Resistance

We now turn our attention to the entire protocol and consider whether it is coercion-resistant under the game-based definition from Juels *et al.* [27]. Moving forward, we will model $\mathsf{registration}$ as an abstract protocol possessing the properties demonstrated. Following Juels *et al.*, we will also replace portions of the protocol with idealized primitives. These include the password-based key derivation function, mixing, plaintext equality tests, zero-knowledge proofs of knowledge, and hash functions used to make the proofs simultaneous, non-malleable, and non-interactive (*i.e.,* secure against a dishonest verifier).

Other cryptographic definitions of coercion-resistance (or the related receipt-freeness) exist in the literature. Kusters *et al.* provides another game-based definition [29] that can be seen as a generalization of the

Algorithm 4: Experiment $\mathbf{Exp}^{\mathrm{cr}}_{ES,\mathcal{A}'}$

```
Input: Security parameters $k_1$ (encryption) and $k_2$ (mixing), number of voters $n_V$, number of corrupted
    voters $n_A$, trustees public key $e$, distribution of types of votes cast by honest voters $\mathcal{D}$.
Private Input ($T_i$): Private key shares constituting $d_{\mathcal{T}}$.
1  $(V, U) \leftarrow \mathcal{A}(\texttt{VoterIDs}, \text{“corrupt”})$                    //$\mathcal{A}$ corrupts $n_A$ voters
2  $\{(\rho_i, \mathsf{Enc}_e(g^{\rho_i})) \leftarrow \mathsf{Register}(i, e, k_1)\}_{i=1}^{n_V}$       //All voters register
3  $\{\rho_i\}_{i \in V} \leftarrow \mathcal{A}(\texttt{VoterIDs}, \text{“obtain”})$        //$\mathcal{A}$ obtains corrupted passwords
4  $j \leftarrow \mathcal{A}(\texttt{VoterID}, \text{“coerce”})$                       //$\mathcal{A}$ coerces one voter $j$
5  Check $|V| = n_A$ and $j \notin V$ (upon failure, output $\perp$)       //Validity check on $\mathcal{A}$

6  $b \leftarrow_r \{0, 1\}$                                                //Flip coin $b$
   if $b = 0$ then
7      $\mathsf{AllVotes} \Leftarrow \mathsf{Cast}(\rho_j, \mathbf{B}_j, \beta_j, k_1)$                  //Voter casts ballot $\mathbf{B}$
8      $\rho^* \leftarrow \mathsf{PanicPassword}(\rho_j)$                 //Voter sets $\rho^*$ to a panic password
   else
9      $\rho^* \leftarrow \rho_j$                               //Voter sets $\rho^*$ to real registered password

10 $\mathsf{AllVotes} \Leftarrow \{\mathsf{Cast}(\rho_i, \mathbf{B}_i, \beta_i, k_1, \mathcal{D})\}_{i \neq j, i \notin V}$          //Honest voters cast ballots
11 $\mathsf{AllVotes} \Leftarrow \mathcal{A}(\{\rho_i, \mathbf{B}_i, \beta_i, k_1\}_{i \neq j, i \in V}, \text{“cast”})$      //$\mathcal{A}$ casts corrupt ballots
12 $\mathsf{AllVotes} \Leftarrow \mathcal{A}(\rho^*, \mathbf{B}, \beta, k_1, \text{“cast”})$               //$\mathcal{A}$ casts coerced ballot
13 $(\mathsf{ValidVotes}, \mathit{\Pi}, \mathit{\Gamma}) \leftarrow \mathsf{PreTally}(\mathsf{Roster}, \mathsf{AllVotes}, d_{\mathcal{T}}, k_2)$     //PreTally is conducted

14 $b' = \mathcal{A}(\mathsf{UniqueVotes}, \mathit{\Pi}, \mathit{\Gamma}, \text{“guess”})$                     //$\mathcal{A}$ guesses
15 Output 1 if $b' = b$, else 0.
```

Juels *et al.* definition to voting systems that do not fit the general architecture of the Juels *et al.* scheme (Selections does fit this general architecture). Gardner *et al.* provide a definition focused more on the elimination of covert channels and it is specific to the architecture of Benaloh's voter initiated auditing [20]. Other researchers have used simulation-based proofs [30,38,40]. See Kusters *et al.* [29] for an excellent overview of the different approaches as of 2010.

C.1 Security Game

High-level Overview. We present the experiment $\mathbf{Exp}^{\mathrm{cr}}_{ES,\mathcal{A}'}$ in Algorithm 4 and experiment $\mathbf{Exp}^{\mathrm{cr-ideal}}_{ES,\mathcal{A}}$ in Algorithm 5. These experiments are quite faithful to the original experiments proposed by Juels *et al.*, except we have modified them to correspond to user-chosen passwords instead of registrar-chosen credentials. We have replaced some of the notation to better integrate with Selections as presented in Section 3. Here we will briefly describe the game.

The game involves a challenger running the game, an adversary $\mathcal{A}$, and a set of voters. There are n_V voters and they are partitioned into three sets throughout the game. At the beginning of the protocol, the adversary corrupts n_A voters. We call this set of corrupted voters V and the remaining uncorrupted voters are U. The adversary then specifies a single voter from U to coerce. We call this voter j. So the makeup of the voters is n_A corrupted voters, 1 coerced voter, and $n_V - n_A - 1$ uncoerced (or honest) voters.

The adversary is non-adaptive: he must specify at the beginning of the protocol which voters are to be corrupted. He does this with a “corrupt” command. All voters register. The adversary is given the passwords registered by the corrupt voters (using the “obtain” command). The adversary then chooses the voter to coerce (using the “coerce” command). This voter cannot already be corrupted (*i.e.,* the adversary does not know her registered password). The challenger then makes sure the adversary did not cheat in how voters it corrupted or who it selected to coerce.

At the heart of the game, the adversary will decide between one of two scenarios selected with uniform randomness by the challenger. In the first scenario, the coerced voter does everything she can to escape

Algorithm 5: Experiment $\mathbf{Exp}^{\mathrm{cr-ideal}}_{ES,\mathcal{A}}$

```
1  $(V, U) \leftarrow \mathcal{A}'(\texttt{VoterIDs}, \text{“corrupt”})$
2  $\{(\rho_i, \mathsf{Enc}_e(g^{\rho_i})) \leftarrow \mathsf{Register}(i, e, k_1)\}_{i=1}^{n_V}$
3  $\square$                                                          //$\mathcal{A}'$ does not obtain corrupted passwords
4  $j \leftarrow \mathcal{A}'(\texttt{VoterID}, \text{“coerce”})$
5  Check $|V| = n_A$ and $j \notin V$ (upon failure, output $\perp$)

6  $b \leftarrow_r \{0, 1\}$
   if $b = 0$ then
7      $\mathsf{AllVotes} \Leftarrow \mathsf{Cast}(\rho_j, \mathbf{B}_j, \beta_j, k_1)$
8      $\boxed{\rho^* \leftarrow \rho_j}$                                  //$\mathcal{A}'$ always gets real password
   else
9      $\rho^* \leftarrow \rho_j$                                          //$\mathcal{A}'$ always gets real password

10 $\mathsf{AllVotes} \Leftarrow \{\mathsf{Cast}(\rho_i, \mathbf{B}_i, \beta_i, k_1, \mathcal{D})\}_{i \neq j, i \notin V}$
11 $\mathsf{AllVotes} \Leftarrow \mathcal{A}'(\{\square, \mathbf{B}_i, \square, k_1\}_{i \neq j, i \in V}, \text{“cast”})$      //$\mathcal{A}'$ just submits intent
12 $\mathsf{AllVotes} \Leftarrow \mathcal{A}'(\rho^*, \mathbf{B}, \square, k_1, \text{“cast”})$                //$\mathcal{A}'$ does not see AllVotes
13 $(\mathsf{ValidVotes}, \mathit{\Pi}, \mathit{\Gamma}) \leftarrow \boxed{\mathsf{IdealPreTally}}(\mathsf{Roster}, \mathsf{AllVotes}, d_{\mathcal{T}}, k_2)$     //Idealized

14 $b' = \mathcal{A}'(\mathsf{ValidVotes}, \square, \mathit{\Gamma}, \text{“guess”})$                     //$\mathcal{A}'$ guesses without $\mathit{\Pi}$
15 Output 1 if $b' = b$, else 0.
   Remarks: Boxed items represent changes from $\mathbf{Exp}^{\mathrm{cr}}_{ES,\mathcal{A}'}$.
```

coercion: she posts the ballot she wants to using her real password and provides a panic password to the adversary. In the second scenario, the coerced voter does everything she can to cooperate with the adversary: she does not cast any votes and she provides the adversary with her real password.

The election then proceeds with the honest voters casting their ballots, the adversary casting ballots on behalf of the corrupted voters (while having a full view of what the honest voters cast), and finally the adversary casting a ballot on behalf of the coerced voter using the password supplied to him (both through the “cast” command). The votes go through the pre-tally, and list of encrypted valid votes is returned, along with $\mathit{\Pi}$, a full transcript of the entire PreTally process, including all the proofs, and $\mathit{\Gamma}$ which replicates a subset of the information in $\mathit{\Pi}$ (see below). Based on this information, the adversary guesses which scenario. If the adversary cannot distinguish the scenarios, then he effectively cannot distinguish between the success or failure of his coercion.

Ideal Game. We can ask ourselves, what is the highest level of security we can achieve? It is tempting to conclude it is the scenario where a bounded adversary cannot distinguish between the scenarios. In other words, his advantage at guessing is ϵ-close to 1/2. Juels *et al.* point out this not always the case. In systems like JCJ, Civitas, and AFT, the final output of the protocol is the tally. In Selections, the protocol ends before this final decryption stage however consider for a moment that it produced the final tally as well. If the adversary were to cast a ballot for Alice using the password provided to him by the coerced voter and the final tally produced no votes for Alice, then the adversary could distinguish with certainty which scenario occurred in $\mathbf{Exp}^{\mathrm{cr}}_{ES,\mathcal{A}'}$.

However from this, we should not conclude anything about the coercion-resistance of Selections: the distinguishability of the scenarios in this case is invariant to the election system used. Since we are interested in the advantage an adversary would have with a particular election system, we need to measure it against an experiment that preserves the distinguishers that are invariant to the election system. To this end, Juels *et al.* define $\mathbf{Exp}^{\mathrm{cr-ideal}}_{ES,\mathcal{A}}$ where the election is conducted in a very similar manner but the adversary does not get any access to the cryptographic data involved.

In Selections, because a final tally is not produced, this illustrative distinguisher does not work. However, the adversary does learn Γ which contains four quantities: the number of cast votes, number of submissions eliminated due to invalid proofs, the number of submissions eliminated because they were duplicates, and the number of submissions eliminated for having invalid passwords. The last quantity could be distinguishing: if no other honest voters cast ballots in the election and the adversary abstains from casting the coerced voter's vote, he will learn if the coerced voter cast a vote of her own by the number of valid votes (relative to the number he cast through corrupted voters). In general, the number of votes cast will always be greater by one when the coin flip is 0, as opposed to 1. In order to negate the significance of this fact on the adversary's advantage, we introduce a notion of adversarial uncertainty.

Adversarial Uncertainty. If the adversary knows how all the other voters will vote, then he can easily and consistently win the game even without seeing the final tally. He will simply count how many votes were submitted. This number differs according to the coin flip. Juels *et al.* address this by specifying a distribution $\mathcal{D}$ over the honest voter's votes which specifies the number of ballots they will cast (voters can abstain, submit one, or submit more than one), and whether submitted ballots contain a valid proof and/or use a valid credential. $\mathcal{D}$ acts a "noise" that masks the coerced voter's actions. Additionally, they note, noise can be injected into the system by any interested party.

In Selections, voters rerandomize their registered encrypted password from Roster and include it in their vote, but do not reveal it is from their entry in Roster. They do this by forming an anonymity set of $\beta - 1$ other entries. When the size of β is not specified by the system (Case 3 below), we assume that $\mathcal{D}$ also specifies a distribution for the β values used by voters. It is likely to be bimodal, with voters either using the minimum β (for efficiency) or the largest β (for maximal privacy). We assume adversarial uncertainty with respect to this distribution.

C.2 Ideal Components

- $\widetilde{DKG}(k_1)$ is a distributed key generation algorithm that shares a secret decryption key. In the idealized form, we assume only the challenger learns the key. k_1 is a security parameter.
- $\widetilde{PBKDF}(\hat{\rho})$ is a password-based key derivation function. It takes input from an unspecified distribution and returns a value that is computationally indistinguishable from a uniform random selection in $\mathbb{G}_q$. It may rely on a random oracle.
- $\widetilde{POK}1(\beta)$ is a proof of knowledge that a given ciphertext reencrypts 1-out-of-β other ciphertexts. The computational secrecy of the witness is ϵ_{wh-pok} ("wh" for witness hiding) and it is non-interactive through a random oracle assumption. Using standard techniques, the component can output a simulated proof (by control of the random oracle) or the witness can be extracted (through rewinds).
- $\widetilde{POK}2$ is a proof of knowledge of a discrete log. It has the same properties as $\widetilde{POK}1$.
- $\widetilde{MIX}(\mathsf{UniqueVotes}, k_2)$ is a mix network. In the idealized form, it shuffles and rerandomizes a list of ciphertexts. k_2 is a security parameter that describes how unlinkable an output ciphertext is from its corresponding input (it differs from k_1 because some implementations reveal partial information about the permutation used).
- $\widetilde{PET}$ is a plaintext equality test. In the idealized form, it simply returns a bit describing whether two ciphertexts encrypt the same message or not.

C.3 Case 1: $\boldsymbol{\beta = R}$

We define the advantage of $\mathcal{A}$, where an output of 1 is the adversary correctly stating b, as,

$$\mathbf{adv}^{\mathrm{cr}}_{ES,\mathcal{A}} = |\mathbf{Pr}[\mathbf{Exp}^{\mathrm{cr}}_{ES,\mathcal{A}'}(\cdot) = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathrm{cr-ideal}}_{ES,\mathcal{A}}(\cdot) = 1]|.$$

Recall that when voter submit a ballot, they prove they re-encrypted one of β registered passwords. β represents the size of the anonymity set for the voter. We show that when β is the full roster R for all voters,

$\mathbf{adv}^{\text{cr}}_{ES,\mathcal{A}}$ for Selections is negligible. Setting $\beta = R$ does impact performance. Vote casting is linear in the size of the $\mathsf{ElectionRoster}$ and $\mathsf{Pre\text{-}Tallying}$ is quadratic.

We approach the proof through a series of modified games, beginning with a detailed implementation of $\mathbf{Exp}^{\text{cr}}_{ES,\mathcal{A}'}$ in Game 0 and ending with a game that, from inspection, has the same advantage as $\mathbf{Exp}^{\text{cr-ideal}}_{ES,\mathcal{A}}$. For each modification, we will show that $\mathcal{A}$'s advantage in distinguishing which game he is playing is bounded by a quantity that is negligible in some security parameter.

Game 0. We now describe the initial game. Note that the item numbers in this game correspond to the same step in $\mathbf{Exp}^{\text{cr}}_{ES,\mathcal{A}'}$ and $\mathbf{Exp}^{\text{cr-ideal}}_{ES,\mathcal{A}}$. For modified games, we will only include the modified lines.

Set-up assumptions. The experiment takes as input a number of parameters that were generated during some set-up phase. This includes a uniformly random private key $d_{\mathcal{T}} \leftarrow_r \mathbb{Z}_q$ which we assume is output in shares to a set of trustees with the $\widetilde{DKG}(k_1)$ oracle. The corresponding public key $e \in \mathbb{G}_q$ is published. Security parameter k_1 is specified such that computing d_{τ} from e is $\mathsf{negl}(k_1)$. A generator of $g_0 \leftarrow_r \mathbb{G}_q$ is also chosen by the trustees at random.

1. **Corrupt voters.** The adversary $\mathcal{A}$ selections a subset of the n_V eligible voters to corrupt. He outputs two sets: U the uncorrupted voters and V the corrupted voters.
2. **Registration**. For each voter i, the challenger chooses a random password $\hat{\rho}$, queries $\widetilde{PBKDF}(\hat{\rho})$ and obtains a random ρ. With $r \leftarrow_r \mathbb{Z}_q$, the challenger encrypts ρ under key e. The output is Roster, which includes for each voter an entry of the form: $\langle \mathtt{VoterID}_i, \mathsf{Enc}_e(g_0^{\rho_i}, r_i)\rangle$.
3. **Obtain corrupted passwords**. For each voter in V, the challenger supplies the adversary with the passwords: $\langle \rho_i \rangle_{i \in V}$ and a proof of correctness using $\widetilde{POK}1(1)$.
4. **Choose coercion target**. From the uncorrupted voters U, $\mathcal{A}$ selects voter j to coerce. Voter j is removed from set U.
5. **Validity Check**. The challenger checks that the subset of corrupted voters is according to input parameter n_A and that $\mathcal{A}$ did not obtain the password for the coerced voter j in step 4. If any test fails, the game halts and outputs $\perp$.
6. **Flip coin**. The challenger flips a coin $b \leftarrow_r \{0, 1\}$.
7. **Heads: Cast real ballot**. If $b = 0$, the challenger constructs a ballot for coerced voter j using password ρ_j. Let $c_j = \mathsf{Enc}_e(g_0^{\rho_j}, r_j)$ for voter j from Roster and c'_j be a rerandomization of c_j. The challenger appends to $\mathsf{AllVotes}$ a ballot of the form $\langle g_0^{\rho_j}, \mathsf{ReRand}(c_j), \mathbf{B}_j, \pi_1, \pi_2\rangle$ where π_1 is the output of $\widetilde{POK}1(\beta)$ and π_2 is the output of $\widetilde{POK}2$.[5] Recall our assumption that the knowledge error for both oracles is $\mathsf{negl}(k_1)$, the same security parameter for the encryption. Also note that due to the use of an append operation, the adversary knows exactly where this ballot, if it exists, will be posted—a point we will return to later.
8. **Heads: Generate panic password**. Continuing the case that $b = 0$, the challenger chooses a random password $\hat{\rho}^*$, queries $\widetilde{PBKDF}(\hat{\rho}^*)$, and obtains a random ρ^*. He sets $\rho_{(\mathcal{A},j)} = \rho^*$ and provides the adversary with $\rho_{(\mathcal{A},j)}$.
9. **Tails: Give real password**. The challenger sets $\rho_{(\mathcal{A},j)} = \rho_j$ and provides the adversary with $\rho_{(\mathcal{A},j)}$.
10. **Honest voters vote**. For each voter remaining in U, the challenger posts a ballot following the procedure used in step 7 for the coerced voter. The output to $\mathsf{AllVotes}$ is $\langle g_0^{\rho_i}, \mathsf{ReRand}(c_i), \mathbf{B}_i, \pi_1, \pi_2\rangle_{i \in U}$.
11. **Corrupt voters vote**. $\mathcal{A}$ constructs votes for the remaining candidates. The adversary is not bound to the protocol and can post tuples of any form (*e.g.,* duplicate votes, invalid votes, votes with invalid proofs, *etc.*). These are provided to the challenger, who appends them to $\mathsf{AllVotes}$.
12. **Coerced voter votes**. $\mathcal{A}$ constructs one remain vote for the coerced voter j, potentially using the $\rho_{(\mathcal{A},j)}$ he was provided with.

[5] Note that in JCJ, the adversary can specify the candidate the voter votes (so that the security does not depend on the adversary not knowing) for however in Selections, we never decrypt the ballots and we leave ballot information $\mathbf{B}$ generic. However, we if assume $\mathbf{B}$ to be a ciphertext, we can allow the adversary to choose the corresponding plaintext.

13. **PreTally**. The challenger operates the PreTally protocol. It is detailed earlier in the paper, so we only briefly outline it here. The challenger first verifies the proofs and removes tuples with invalid proofs, producing ProvedVotes. The challenger then inspects the tuples for duplicate values for g^{ρ_i}. For pairs with the same value, he only retains the most recent. The new list is UniqueVotes. He queries $\widetilde{MIX}(\mathsf{UniqueVotes}, k_2)$ which produces a shuffled output and proof of correctness. Each output entry is distinguishable from its corresponding input by a $\mathsf{negl}(k_2)$ factor. Finally, for each tuple, the first two entries are supplied to $\widetilde{PET}$. If the test is negative, the entry is removed. The remaining list ValidVotes is output to $\mathcal{A}$ along with each preceding list.
14. **Guess**. The adversary produces a guess $b' \in \{0, 1\}$. If $b' = b$, Game 0 outputs a 1. Otherwise, it outputs 0.

Intuition. The intuition behind our modifications is as follows. The coerced voter behaves differently in at least two regards depending on the coin flip: (1) she submits a second vote and (2) she supplies either a real or panic password to the adversary. The second vote has two potentially distinguishing features: (1a) the rerandomized value from the Roster belongs to the voter[6] and (1b) the asserted password matches the value on the Roster. Game 1 addresses (1a) and (1b) and Game 2 addresses (2). These games will show that a bounded adversary cannot use values associated with the coerced voter directly. However he could potentially learn the actions of all the honest voters in the system. If he were successful, he would know the actions of everyone except the coerced voter and could then indirectly determine the actions of the coerced voter based on the result. We address the honest voters in Game 3.

Game 1. For Game i, define $S_i = \mathbf{Pr}[\text{Game i} = 1]$. We describe Game 1 and show that $|S_1 - S_0| \leq \epsilon_{cpa} + \epsilon_{wh-pok} + \epsilon_{ddh} + \epsilon_{wh-pok}$.

2 **Registration**. Registration proceeds as in Game 0.
6 **Flip coin**. Coin is flipped as in Game 0.
7 **Heads: Cast ballot**. The challenger chooses a random value $\hat{z} \in \mathbb{Z}_q$, computes $\hat{c} = \mathsf{Enc}_e(g_0^{\hat{z}}, r)$, and simulates a proof $\hat{\pi}_1$ that $\hat{c}$ is one of β entries on the Roster using $\widetilde{POK1}(\beta)$. The challenger appends $\left\langle \boxed{g_0^{\hat{z}}}, \boxed{\hat{c}}, \mathbf{B}_j, \boxed{\hat{\pi}_1}, \boxed{\hat{\pi}_2} \right\rangle$ to AllVotes.
8 **Heads: Generate panic password**. $\mathcal{A}$ is provided $\rho_{(\mathcal{A},j)} = \rho^*$ as in Game 0.
9 **Tails: Give real password**. $\mathcal{A}$ is provided $\rho_{(\mathcal{A},j)} = \rho_j$ as in Game 0.
13 **PreTally**. PreTally proceeds as in Game 0.

We now consider whether $\mathcal{A}$ can distinguish is he is playing Game 0 or Game 1. If $b = 1$, then the games are identical and he will have no advantage. For this reason, the advantage we compute should be scaled by a factor of 1/2, however since we will be dealing with negligible quantities, we omit this. If $b = 0$, $\mathcal{A}$ is asked to distinguish $\langle g_0^{\rho_j}, \mathsf{ReRand}(c_j), \mathbf{B}_j, \pi_1, \pi_2 \rangle$ from $\left\langle \boxed{g_0^{\hat{z}}}, \boxed{\hat{c}}, \mathbf{B}_j, \boxed{\hat{\pi}_1}, \boxed{\hat{\pi}_2} \right\rangle$. Since votes are appended, $\mathcal{A}$ can find this submission when $b = 0$.

We first consider if $\mathcal{A}$ can distinguish $g_0^{\hat{z}}$ for a random $\hat{z}$ from $g_0^{\rho_j}$. The adversary can see an encryption of ρ_j, $(c_{(1,j)}, c_{(2,j)}) = \mathsf{Enc}(g_0^{\rho_j}, r)$, from the coerced voter's entry on the Roster. In Game 0, the following values form a Diffie-Hellman tuple: $\langle g, g^r = c_{(1,j)}, y, y^r = c_{(2,j)}/g_0^{\rho_j} \rangle$. In Game 1, the last element of this tuple is masked by z and is distributed randomly in $\mathbb{G}_q$. Thus distinguishing them is an example of the decisional Diffie-Hellman problem.

Toward a contradiction, let A^{G1} be an efficient algorithm for distinguishing Game 1 from Game 0 based on this potentially distinguishing factor (we deal with three others next). We can use A^{G1} to gain a non-negligible advantage at the DDH game in the underlying group. The adversary receives challenge tuple $\langle a_1, a_2, a_3, a_4 \rangle$ where $a_1^x = a_2$ for some x. $\mathcal{A}$ must decide if $a_3^x = a_4$. $\mathcal{A}$ computes a random exponent $\omega \leftarrow_r \mathbb{Z}_q$, sets $g = a_1$,

[6] Note that this is a necessary but not sufficient condition for the submission being a second vote from the coerced voter. There is no restriction on other voters submitting votes using the coerced voter's Roster entry. These will be eliminated.

sets $e = a_3$, replaces RosterEntry with $c'_j = \langle a_2, a_4 * g_0^{\omega} \rangle$, and places g_0^{ω} in the submitted vote. He submits this transcript to A^{G1}. Upon output Game 0, $\mathcal{A}$ guesses it is a Diffie-Hellman tuple and upon output Game 1, he guesses it is not. Let ϵ_{ddh} be the adversary's advantage at the DDH game.

Next, $\mathcal{A}$ again obtains the coerced voter's entry, c_j, from the Roster. Now he guesses if the submitted vote contains a rerandomization of c_j or not. Let the submitted value, either $\mathsf{ReRand}(c_j)$ or $\hat{c}$, be c^*.

Toward a contradiction, let A^{G2} be an efficient algorithm for distinguishing Game 1 from Game 0 based on this potentially distinguishing factor. We can use A^{G2} to gain a non-negligible advantage at the CPA game of the underlying encryption. $\mathcal{A}$ submits two random messages, $m_1, m_2 \leftarrow_r \mathbb{G}_q$ and receives challenge ciphertext c_{m_b}. The adversary takes a transcript of a game, replaces voter j's entry of Roster with $\mathsf{Enc}_e(m_1, r)$ and replaces c^* with c_{m_b}. $\mathcal{A}$ submits this to A^{G1} and upon a guess of Game 0, $\mathcal{A}$ guess m_0. Let ϵ_{cpa} be the adversary's advantage at the CPA game.

There are two other values in the transcript of the Game that are functionally dependent on $\hat{z}$. First, π_1 will be constructed the same in both games, but will rely on different witnesses. We assumed an idealized proof of knowledge, and recall that we assumed it is witness hiding. Let we bound the information $\mathcal{A}$ can gain from π_1 about the witness at ϵ_{wh-pok}. Second, the submission contains a transcript π_2 that proves knowledge of $\hat{z}$ in $g_0^{\hat{z}}$. We assume $\mathcal{A}$'s advantage at extracting $\hat{z}$ from this transcript is ϵ_{wh-pok}.

By the triangle inequality, $|S_1 - S_0| \le \epsilon_{cpa} + \epsilon_{wh-pok} + \epsilon_{ddh} + \epsilon_{wh-pok}$

Game 2. We now describe Game 2 and show that $|S_2 - S_1| \le \epsilon_{cpa}$

6 **Flip coin**.
7 **Heads: Cast ballot**. As in Game 1.
8 **Heads:** $\boxed{\textbf{Give real password}}$. $\mathcal{A}$ is provided $\rho_{(\mathcal{A},j)} = \boxed{\rho_j}$ instead of ρ^*.
9 **Tails: Give real password**. As in Game 1.

We now consider whether $\mathcal{A}$ can distinguish is he is playing Game 1 or Game 2. As before, if $b = 1$, then the games are identical and he will have no advantage. If $b = 0$, $\mathcal{A}$ is asked to distinguish ρ_j from ρ^* given an encryption of ρ_j from Roster. Distinguishing the games provides an advantage in the CPA game following almost the identical reduction already used in Game 1.

Toward a contradiction, let A^{G3} be an efficient algorithm for distinguishing Game 2 from Game 1 based on this potentially distinguishing factor. We can use A^{G3} to gain a non-negligible advantage at the CPA game of the underlying encryption. $\mathcal{A}$ submits the messages, m_0, m_1 and receives challenge ciphertext c_{m_b}. The adversary replaces the coerced voter's entry on Roster with c_{m_b} and uses m_1 as the provided value. He submits this to A^{G3}. Upon output Game 1, $\mathcal{A}$ guesses m_0 (since in Game 1, the roster and provided value are different) and upon Game 2, it guesses m_1 (since in Game 2, the values are the same).

Game 3. We describe Game 3 and show that $|S_3 - S_2| \le \epsilon_{cpa} + \epsilon_{wh-pok} + \epsilon_{ddh} + \epsilon_{wh-pok}$ and that $S_3 = \mathbf{Pr}[\mathbf{Exp}^{\text{cr-ideal}}_{ES,\mathcal{A}}(\cdot) = 1]$.

10 **Honest voters vote**. For each voter remaining in U, the challenger posts a ballot following the procedure used in Game 2 for the coerced voter, except the commitments and encryptions do not match. The output to AllVotes for each voter is $\left\langle \boxed{g_0^{\hat{z}_1}}, \boxed{\mathsf{Enc}_e(g_0^{\hat{z}_2}, r)}, \mathbf{B}_j, \boxed{\hat{\pi}_1}, \boxed{\hat{\pi}_2} \right\rangle$ for a random values $\hat{z_1}$ and $\hat{z_2}$. If the vote is chosen to be not valid, the simulated submission is modified in the same way (*e.g.,* invalid proof). If the vote is a duplicate, the same $\hat{z_1}$ value is used. If the vote is not supposed to match the Roster, this is noted by the challenger.
11 **Corrupt voters vote**. Same as in Game 2.
12 **Coerced voter votes**. Same as in Game 2.
13 **PreTally**. The challenger operates the PreTally protocol as in Game 2 with some modifications. Since none of the honest votes will having matching $g_0^{\hat{z}_1}$ and $\mathsf{Enc}_e(g_0^{\hat{z}_2}, r)$ values, the challenger will track the votes through the protocol. The challenger eliminates submissions with invalid proofs or duplicate committed passwords as in Game 2, and outputs UniqueVotes. Instead of querying $\widetilde{MIX}(\mathsf{UniqueVotes}, k_2)$,

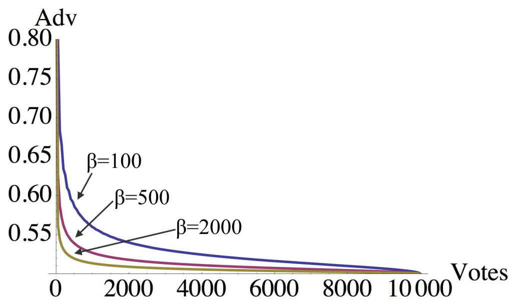


Fig. 2. Adversary's advantage over one half (δ) versus β and votes cast (V_2) in Case 2.

the challenger simulates a mix (as a reencryption mix, it does not depend on $d_{\mathcal{T}}$) and remembers where each honest vote ends up. For each corrupted or coerced tuple, it submits the first two entries to $\widetilde{PET}$. If the test is negative, the entry is removed. For honest votes, it simulates a positive or negative output of $\widetilde{PET}$ according to whether the submission was intended to match the Roster. The remaining list ValidVotes is output to $\mathcal{A}$ along with each preceding list.

In Game 1, we have seen an argument for why replacing commitments to passwords with a random $g_0^{\hat{z}_1}$ and rerandomized roster entries with a random $\mathsf{Enc}_e(g_0^{\hat{z}_2}, r)$ cannot be distinguished with advantage greater than $\epsilon_{ddh} + \epsilon_{cpa}$. The rest of the modifications consist of replacing queries to the ideal components with simulations. This could be detected if the witnesses could be extracted from the real proofs in Game 2 as opposed to the simulated proofs in Game 3. We denote this advantage as $\epsilon_{mix} + \epsilon_{pet}$ for the mixing and plaintext equality tests.

Since AllVotes, ProvedVotes, UniqueVotes, and ValidVotes contains only random values in each vote submission other than the adversary's own submissions (through corrupt voters and/or the coerced voter), the only useful information it provides is the final output,ValidVotes, and the relative size of each list, Γ. This is exactly the output of $\mathbf{Exp}_{ES,\mathcal{A}}^{\mathrm{cr-ideal}}$.

Putting everything together,

$$\begin{aligned}|S_3 - S_0| &= |\mathbf{Pr}[\mathbf{Exp}_{ES,\mathcal{A}}^{\mathrm{cr-ideal}}(\cdot) = 1] - \mathbf{Pr}[\mathbf{Exp}_{ES,\mathcal{A}'}^{\mathrm{cr}}(\cdot) = 1]| \\ &= \epsilon_{cpa} + \epsilon_{wh-pok} + \epsilon_{ddh} \\ &= \mathbf{adv}_{ES,\mathcal{A}}^{\mathrm{cr}}\end{aligned}$$

C.4 Case 2: fixed β

We show that when β is constant (*e.g.,* 5 or 100), $\mathbf{adv}_{ES,\mathcal{A}}^{\mathrm{cr}} < \delta$, where δ is small but non-negligible. Recall there are V_2 votes with valid proofs and R entries on the ElectionRoster. Each voter submits a vote within an anonymity set. Assume the members of this set are chosen with uniform randomness. The expected number of votes cast in which the coerced voter j is in one or more anonymity set can be described with a binomial distribution.

When $b = 0$, at least one vote will include j in an anonymity set because the coerced voter must include herself in the vote to be valid. The additional inclusions are distributed randomly. When $b = 1$, all the inclusions are distributed randomly. Thus, there is a measurable difference between these distributions, which we call δ. Let $\mathbf{F}(k; p, n)$ be the cumulative distribution function of a Binomial distribution with n trials, p success probability, and k successes. The adversary's advantage is δ,

$$\delta = \frac{1}{2}(F(\frac{\beta V_2}{R}; V_2, \frac{\beta}{R}) + 1 - F(\frac{\beta V_2}{R} - 1; V_2 - 1, \frac{\beta}{R})).$$

While we could perhaps provide a bound on this value, we are ultimately uninterested in Case 2 and are considering it only to motivate Case 3. Therefore we simply note it is a non-negligible amount, and provide in Figure 2 some values of δ plotted against the number of cast votes and β.

In Case 1, we made assumptions about adversarial uncertainty with respect to certain voting behaviours. For example, the total number of votes cast will always differ by one between $b = 0$ and $b = 1$ but it also differs according to $\mathcal{D}$. (Similarly the number of votes in the final output will as well, but in $\mathbf{Exp}^{\mathrm{cr-ideal}}_{ES,\mathcal{A}}$, the number of votes in the final output is also provided to $\mathcal{A}$.) Unfortunately, the anonymity set membership distribution cannot be described within adversarial uncertainty: in $\mathbf{Exp}^{\mathrm{cr-ideal}}_{ES,\mathcal{A}}$, $\mathcal{A}$ does not have access to any information posted to $\mathsf{AllVotes}$ and the protocol clearly provides an expected value for j's inclusion in an anonymity set. Therefore, we conclude Case 2 is not secure.

Game 0-weak. We consider a modified experiment, $\mathbf{Exp}^{\mathrm{cr-weak}}_{ES,\mathcal{A}'}$, for which $\mathbf{adv}^{\mathrm{cr-weak}}_{ES,\mathcal{A}}$ is negligible for Case 2. We do this to clearly illustrate what blocks the proof from going through. We hope this makes the motivation for and solution in Case 3 easier to understand. We illustrate this through a game: Game 0-weak.

6 **Flip coin**.
7 **Heads: Cast ballot**. As in Game 0.
8 **Heads: Give panic password**. As in Game 0.
9 **Tails: Give real password**. As in Game 0.
10 **Honest voters vote**. Iff $b = 1$, the challenger will select for the first honest voter an anonymity set that includes voter j. The other members of the set are chosen randomly, and the process for the rest of the honest voters proceeds as in the original Game 0 (with randomly selected, β-sized anonymity sets).

Game 0-weak is too strong of a modification to consider Case 2 coercion resistant; however were it made, the security would follow as in Case 1 (with this modification preserved through the sequence of games).

C.5 Case 3: stealth votes

We consider the case where β is required to be at least a constant value (*e.g.,* 5 or 100) but voters can submit $\mathsf{stealth\ votes}$ where $\beta = R$. We show that if a coerced voter's coercion-resistant strategy is to submit their real vote as a stealth vote, $\mathbf{adv}^{\mathrm{cr}}_{ES,\mathcal{A}}$ is negligible. We do make one small change to $\mathbf{Exp}^{\mathrm{cr}}_{ES,\mathcal{A}'}$: instead of the coerced voter's real vote being appended to the cast ballots, it is inserted at a random place (*i.e.,* she votes her real ballot at some arbitrary time after being coerced). If we used Game 0 directly, the use of appends means that the coerced voter's vote will always be first (when $b = 0$). $\mathcal{A}$ can simply inspect this vote and see if it is a stealth vote. It will always be when $b = 0$ and will only be so with non-certain probability when $b = 1$. We believe this gives the adversary unnecessary power. We describe our change as Game 0-iii and then argue for its validity.

Game 0-iii.

6 **Flip coin**.
7 **Heads: Cast ballot**. If $b = 0$, the challenger constructs a ballot for coerced voter j using password ρ_j as in Game 0. Likewise, the challenger constructs a ballot of the form $\langle g_0^{\rho_j}, \mathsf{ReRand}(c_j), \mathbf{B}_j, \pi_1, \pi_2 \rangle$ where π_1 is the output of $\widetilde{POK}1(\beta)$ and π_2 is the output of $\widetilde{POK}2$. In Game 0, due to the use of an append operation, the adversary knows exactly where this ballot, if it exists, will be posted. In this game, the challenger holds the ballot until he posts the votes from the honest voters and at that time, he posts this ballot as well in a $\boxed{\text{random location}}$.
8 **Heads: Give real password**. As in Game 0.
9 **Tails: Give real password**. As in Game 0.

Given the adversary is passive and must corrupt voters at the beginning of the protocol, we believe that Game 0 gives the adversary too much power. This is not a problem until we have a protocol where the security should hold under reasonable assumptions but the proof will not go through, as is the case here. We propose Game 0-iii as reasonable modification to $\mathbf{Exp}^{\mathrm{cr}}_{ES,\mathcal{A}'}$ that preserves the security of the system against a *passive* adversary, who must corrupt voters at the beginning of the protocol. With a non-adaptive adversary, the coerced voter knows they are being coerced and in the $b = 0$ case, they are attempting to deceive the adversary. Game 0-iii simply adds another dimension to the deception.

In the real-world, the original $\mathbf{Exp}^{\mathrm{cr}}_{ES,\mathcal{A}'}$ does not model the intended strategy for a coerced voter. If a coercer demands a voter's password, the voter cannot cast a real vote immediately because they are in the presence of the adversary. It is likely they may have already voted with their real password, or will wait some period of time before submitting. In other words, the timing of when they submit their real vote is invariant to the time that they are coerced, unless if they happen to coincide.

If we accept Game 0-iii as a definition of coercion resistance against a passive adversary, then security of Case 3 follows from the negligible distinguishability between Game 0-iii and a similarly adjusted Game 1-iii and on through the same steps we used in Case 1. If we consider this potentially distinguishing feature that not all votes have anonymity sets of the full size in isolation (Case 1 proof covers the others), we must introduce a new assumption: we assume that other voters are submitting stealth votes (because they are privacy sensitive) and the adversary has no reliable expected value for the quantity of stealth votes in the election. To accomplish this, we expand $\mathcal{D}$ to include the distribution of stealth and non-stealth votes, and as with the other aspects of this distribution, we assume adversarial uncertainty.

D Augmented Transcript Cards

Future work could explore the creation of an augmented transcript card that provides verifiable, bare-handed mechanisms for erasures, pre-committed randomness and a voter commitment to which of the α submissions they will choose. It could include the following features,

- A serial number.
- A commitment to the contents for the Roster prior to the protocol.
- A place for the voter to commit to which ciphertext she will retain prior to the reencryptions being printed. This could be accomplished with a simple hole-punch. The voter performs the hole-punch in front of the registration agent and the card is checked before the voter leaves to ensure the scratched off cell matches the committed spot.
- Confirmation codes printed under each scratch-off surface for the voter to prove to the registrar she complied with the erasure.
- To eliminate covert channels, randomization is precommitted to.
- The randomization values could be supplied to a one-way function to generate the erasure codes.
- A commitment to the contents of the Roster after the protocol.