Coercion-Resistant Electronic Elections

Ari Juels and Markus Jakobsson

RSA Laboratories
Bedford, MA 01730, USA
E-mail: {ajuels,mjakobsson}@rsasecurity.com

Abstract. We introduce a model for electronic election schemes that involves a more powerful adversary than in previous work. In particular, we allow the adversary to demand of coerced voters that they vote in a particular manner, abstain from voting, or even disclose their secret keys. We define a scheme to be *coercion resistant* if it is impossible for the adversary to determine whether a coerced voter complies with the demands. Furthermore, we relax the requirements made in some previous proposals from an untappable channel to only requiring the existence of an anonymous channel.
A first contribution of this paper is to carefully describe this new and strengtened adversary and the associated model; a second is to demonstrate a protocol that is secure in our model. While it is clear that a strengthening of attack models is of theoretical relevance, it is important to note that our results are also highly practical. This is true both in that we model real-life threats (such as vote-buying and vote-cancelling), and in that our proposed protocol combines high efficiency with an unusual lack of structural complexity. A surprising and counter-intuitive achievement of our protocol is that it combines universal verifiability – the requirement that everybody can verify that all ballots were counted – with coercion resistance.

Key words: coercion-resistance, electronic voting, mix networks, receipt-freeness

1 Introduction

Most voters participating in shareholder elections in the United States have the option of casting their ballots via a Web browser [1]. The same was true of voters participating in the Democratic Presidental primary in Arizona in 2000 [8]. These are just two instances of a broadening trend toward Internet-based voting. While voting of this kind appears to encourage higher voter turnout [29], it also brings with it the risks of any computer-based process, namely errors, failures, and vulnerability to attack. A number of papers in the cryptographic literature have described ways of achieving robust and verifiable *electronic* elections, i.e., elections in which ballots and processing data are posted to a publicly accessible bulletin board. For some recent examples, see [6, 13, 15, 19, 21, 24, 27, 33]. There are two other threats, however, that it is equally crucial to address in a fair and democratic election process: We speak of *voter coercion* and *vote buying*. Internet-based voting does not introduce these problems, but it does have the potential to exacerbate them by extending the reach and data collection abilities of an attacker. This is highlighted in one way by the presence of a notorious Web site that provides a forum for the auctioning of votes [2]. Seller compliance was in that case merely voluntary. Conventional Internet voting schemes, however, including those described in the literature, actually provide an attacker with ready-made tools for verifying voter behavior and thereby exerting influence or control over voters. Without careful system design, the threats of coercion and vote buying are potentially far more problematic in Internet voting schemes than in ordinary, physical voting schemes.

One commonly proposed way of achieving secure electronic voting systems is to use a cryptographic system known as a *mix network* [10]. This is a tool that enables a collection of servers

to take as input a collection of ciphertexts and to output the corresponding plaintexts according to a secret permutation. A straightforward way to achieve an election system that preserves the privacy of voters, then, is to assign a private digital signing key to each voter. To cast a ballot, the voter encrypts her choice and signs it, and then posts it to a bulletin board (i.e., a publicly accessible memory space). When all ballots have been collected and the corresponding signatures have been checked, the ciphertexts are passed through a mix network. The resulting plaintext voter choices may then be tallied. Thanks to the privacy preserving property of the mix network, an adversary cannot tell which vote was cast by which voter. This type of scheme is frequently advocated in the mix-network literature, as in, e.g., [6, 10, 15, 19].

In an ordinary mix-based scheme of this kind, an adversary can coerce a voter straightforwardly. The adversary can simply furnish the voter with a ciphertext on a target candidate chosen by the adversary, and then verify that the voter posted a ballot containing that ciphertext. Alternatively, the adversary can demand the private signing key of the voter and verify its correctness against the corresponding public key. An adversary attempting to buy votes can use the same means. Other types of cryptographic voting schemes, namely homomorphic schemes [4, 13] and schemes based on blind signatures [14, 27], suffer from similar vulnerabilities.

1.1 Previous work

Previous investigations of coercion-resistant voting have been confined to a property known as *receipt-freeness*. Roughly stated, receipt-freeness is the inability of a voter to prove to an attacker that she voted in a particular manner, even if the voter wishes to do so. For a more formal definition, see [27]. The property of receipt-freeness ensures that an attacker cannot determine exact voter behavior and therefore cannot coerce a voter by dictating her choice of candidate. It also protects against vote-buying by preventing a potential vote buyer from obtaining proof of the behavior of voters; voters can thereby pretend to sell their votes, but defraud the vote buyer. The notion of receipt-freeness first appeared in work by Benaloh and Tuinstra [4]; their scheme, based on homomorphic encryption, was shown in [18] *not* to possess receipt-freeness as postulated. An independent introduction of the idea appeared in Niemi and Renvall [25]. Okamoto [26] proposed a voting scheme which he himself later showed to lack the postulated receipt-freeness; a repaired version by the same author, making use of blind signatures, appears in [27]. Sako and Kilian [30] propose a multi-authority scheme employing a mix network to conceal candidate choices, and a homomorphic encryption scheme for production of the final tally. The modelling of their scheme was clarified and refined by Michels and Horster [23]. The Sako and Kilian scheme serves as a conceptual basis for the later work of Hirt and Sako [18], the most efficient (and correct) receipt-free scheme voting to date. A scheme by Magkos *et al.* [22] distinguishes itself by an approach relying on tamper-resistant hardware, but is flawed.[1]

All of these receipt-free voting schemes include somewhat impractical assumptions. For example, these schemes assume the availability of an *untappable channel* between the voter and the authorities, that is, a channel that provides perfect secrecy in an information-theoretic sense. The scheme in [27] makes the even stronger assumption of an *anonymous* untappable channel. (It is also not very practical in that it requires voter interaction with the system three times in the course of an election.) Moreover, all of these schemes (excepting [27]) lose the property of coercion-resistance if the attacker is able to corrupt even one of the tallying autorities in a

[1] We are unaware of any mention of a break of this scheme in the literature, and therefore briefly describe one here. The Magkos *et al.* system employs an interactive honest-verifier ZK proof made by a smartcard to the voter. Presumably because of the simulability of this proof, the authors describe the proof as being "non-transferable". This is not true. In particular, an adversary can stipulate that the voter engage in the proof using a challenge pre-selected by the attack. The proof then becomes transferable, yielding a means of receipt construction by the adversary.

distributed setting. The scheme of Hirt and Sako still retains coercion-resistance when such corruption takes place, but only under the strong assumption that the voter knows *which* tallying authorities have been corrupted.

A still more serious problem with of all of the receipt-free voting schemes described in the literature, however, is the fact that the property of receipt-freeness alone fails to protect an election system against several forms of serious, real-world attack, which we enumerate here:

Randomization attack: This attack was noted by Schoenmakers [34], who described its real-world applicability to the scheme of Hirt and Sako. The idea is for an attacker to coerce a voter by requiring that she submit randomly composed balloting material. In this attack, the attacker (and perhaps even the voter) is unable to learn what candidate the voter cast a ballot for. The effect of the attack, however, is to nullify the choice of the voter. For example, an attacker favoring the Republican party in a United States election would benefit from mounting a randomization attack against voters in a heavily Democratic district.

Forced-abstention attack: This is an attack related to the previous one based on randomization. In this case, the attacker coerces a voter by demanding that she refrain from voting. All of the schemes cited above are vulnerable to this simple attack. This is because the schemes authenticate voters directly in order to demonstrate that they are authorized to participate in the election. Thus, an attacker can see who has voted, and use this information to threaten and effectively bar voters from participation.[2]

Simulation attack: The receipt-free schemes described above assume that the attacker cannot coerce a voter by causing her to divulge her private keying material after the registration process but prior to the election process. Such an attack, however, is a real and viable one in previously proposed schemes, because an attacker can verify the correctness of private keying material. For example, in [27], the voter provides a digital signature which, if correct, results in the authority furnishing a blind digital signature. In [18], the voter proves knowledge of a private key relative to a publicly committed or published value on casting a ballot. Thus, receipt-freeness does not prevent an attacker from coercing voters into divulging private keys or buying private keys from voters and then *simulating* these voters at will, i.e., voting on their behalf.

1.2 Our contribution

Our contribution in this paper is twofold. First, we investigate a stronger and broader notion of coercive attacks than receipt-freeness. This notion, which we refer to as *coercion-resistance*, captures what we believe to be the fullest possible range of adversarial behavior in a real-world, Internet-based voting scheme. A coercion-resistant scheme offers not only receipt-freeness, but also defense against randomization, forced-abstention, and simulation attacks – all potentially in the face of corruption of a minority of tallying authorities. We propose a formal definition of coercion-freeness in the body of this paper. Two other properties are essential for any voting scheme, whether or not it is coercion-resistant. These are *correctness* and *verifiability*. As formal definitions for these properties are to the best of our knowledge lacking in the literature, we provide them as well in the paper appendix.

To demonstrate the practical realizability of our definitions, we describe voting scheme that possesses the strong property of coercion-resistance proposed in this paper – and also naturally

[2] An exception is the scheme in [27], which does not appear to be vulnerable to a forced-abstention attack. This is because the scheme seems to assume that the authority checks voter enrollment privately. In other words, the scheme does not permit public verification that participating voters are present on a published voter roll. This is potentially a problem in its own right.

possesses the properties of correctness and verifiability. Our scheme does not require untappable channels, but instead assumes voter access to an anonymous channel at some point during the voting process. (We note that anonymous channels are in fact a minimal requirement for *any* coercion-resistant schemes: An attacker that can identify which voters have participated can obviously mount a forced-abstention attack.) Despite its strong security, our proposal is eminently practical, requiring just three invocations of a verifiable mix network such as [15, 24], and some smaller additional overhead. Thus, our proposed system is more efficient than previously proposed receipt-free ones, in both a practical and asymptotic sense. As an example of the latter, our scheme involves constant work per voter for each tallying authority, while the Hirt-Sako scheme requires work linear in the number of candidates.[3]

1.3 Intuition behind our scheme

In a conventional voting scheme, and also in receipt-free schemes like [18], the voter V_i identifies herself at the time she casts her ballot. This may be accomplished by means of a digital signature on the ballot, or by an interactive authentication protocol. The key idea behind our scheme is for the identity of a voter to remain hidden during the election process, and for the validity of ballots instead to be checked blindly against a voter roll. When casting a ballot, a voter incorporates a concealed credential. This takes the form of a ciphertext on a secret value σ that is unique to the voter. The secret σ is a kind of *anonymous credential*, quite similar in flavor to, e.g., [7, 9]. To ensure that ballots are cast by legitimate voters, the tallying authority $\mathcal{T}$ performs a blind comparison between hidden credentials and a list $\boldsymbol{L}$ of encrypted credentials published by an election registrar $\mathcal{R}$ alongside of the plaintext names of registered voters.

By means of mixing and blinding, it is possible to check whether a concealed credential is in the list $\boldsymbol{L}$ or not, without revealing which voter the credential has been assigned to. Moreover, the verification of credentials can take place *en bloc*, i.e., multiple credentials can be checked against $\boldsymbol{L}$ at the same time. In consequence, an attacker who is given a fake credential $\tilde{\sigma}$ by a coerced voter cannot tell whether or not the credential is valid. (The attacker will learn how many ballots were posted with bad credentials. Provided, however, that some spurious ones are injected by honest players, authorities, or even outsiders, individual voters will be shielded from discovery when they provide fake credentials.) Moreover, the attacker cannot mount randomization or forced-abstention attacks, since there is no feasible way to determine whether an individual voter has posted a ballot or not. In particular, after divulging fake credential $\tilde{\sigma}$, a voter can go and vote again using her real credential σ.

1.4 Organization

In section 2, we describe our setup and attack models and sketch a few of the major forms of adversarial strategies. We provide formal definitions for the security property of coercion-resistance in section 3. We describe the particulars of our proposed scheme in section 4, prefaced by a summary of the underlying cryptographic building blocks. In the appendices to the paper, we offer formal definitions for the correctness and verifiability of election schemes, heuristic security proof sketches, and details on our choice of primitives for realizing our proposed scheme.

2 Modelling

An election system consists of several sets of entities:

[3] An extra log factor in the efficiency analysis originally presented in Hirt-Sako [18] may be eliminated by recent introduction of more efficient, universally verifiable mix networks [15, 24].

1. *Registrars:* Denoted by $\mathcal{R} = \{R_1, R_2, \ldots, R_{n_R}\}$, this is a set of n_R entities responsible for jointly issuing keying material, i.e., credentials to voters.
2. *Authorities (Talliers):* Denoted by $\mathcal{T} = \{T_1, T_2, \ldots, T_{n_T}\}$, authorities are responsible for processing ballots and jointly counting votes and publishing a final tally.
3. *Voters:* The set of n_V voters, denoted by $\mathcal{V} = \{V_1, V_2, \ldots, V_{n_V}\}$, are the entities participating in a given election administered by $\mathcal{R}$. We let i be an identifier for V_i that is available to all players.

We make use of a *bulletin board*, denoted by $\mathcal{BB}$. This is a piece of universally accessible memory to which all players have appendive-write access. In other words, any player can read or write new data to $\mathcal{BB}$, but cannot overwrite or erase existing data. For notational convenience, we assume that data are written to $\mathcal{BB}$ in μ-bit blocks. Shorter data segments may be padded appropriately. For simplicity of exposition, we assume no ordering on the contents of $\mathcal{BB}$.

2.1 Functions

We define a *candidate slate* $\boldsymbol{C}$ to be an ordered set of n_C distinct identifiers $\{c_1, c_2, \ldots, c_{n_C}\}$, each of which corresponds to a voter choice, typically a candidate or party name. In an election, choice c_i may be identified according to its index i. Thus, for cryptographic purposes the candidate slate consists of the integers $\{1, 2, \ldots, n_C\}$ and may be specified by n_C alone. We define a *tally* on an election under slate $\boldsymbol{C}$ to be a vector $\boldsymbol{X}$ of n_C positive integers $x_1, x_2, \ldots, x_{n_C}$ such that x_i indicates the number of votes cast for choice c_i.

- **Registering:** The function $\mathsf{register}(SK_{\mathcal{R}}, i, k_1) \rightarrow (sk_i, pk_i)$ takes as input the private registrar key $SK_{\mathcal{R}}$, a (voter) identifier i and a security parameter k_1, and outputs a key pair (sk_i, pk_i). This is computed jointly by players in $\mathcal{R}$, possibly in interaction with voter V_i.
- **Voting:** The function $\mathsf{vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k_2) \rightarrow \mathit{ballot}$ takes as input a private voting key, the public key of the authorities $\mathcal{T}$, the candidate-slate size n_C, a candidate selection β, and a security parameter k_2, and yields a ballot of bit length at most μ. The form of the ballot will vary depending on the design of the election system, but is in essence a digitally signed vote choice encrypted under $PK_{\mathcal{T}}$.
- **Tallying:** The function $\mathsf{tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3) \rightarrow (\boldsymbol{X}, P)$ takes as input the private key of the authority $\mathcal{T}$, the full contents of the bulletin board, the candidate-slate size, all public voting keys, and a security parameter k_3 and outputs a vote tally $\boldsymbol{X}$, along with a non-interactive proof P that the tally was correctly computed.
- **Verifying:** The function $\mathsf{verify}(PK_{\mathcal{T}}, \mathcal{BB}, n_C, \boldsymbol{X}, P) \rightarrow \{0, 1\}$ takes as input the public key of the authorities, the contents of the bulletin board, the candidate-slate size, the voting tally, and a non-interactive proof of correct tallying. It outputs a '0' if the tally is correct and a '1' otherwise. (We characterize the behavior of verify more formally in our security definitions.)

We define an election scheme ES as the collection of these functions. Thus $\mathsf{ES} = \{\mathsf{register}, \mathsf{vote}, \mathsf{tally}, \mathsf{verify}\}$.

Remark: There are many election models in use throughout the world. The model we propose here excludes important variants. In some systems, for example, voters are asked to rank candidate choices, rather than just listing those they favor. Many systems permit the use of *write-in* votes, i.e., the casting of a ballot in favor of a candidate not listed on the slate for the election. We exclude write-in voting from our model because it undermines the possibility of coercion resistance in any scheme where an observer can see a complete election tally including write-in votes. An attacker may, for example, require coerced voters to cast write-in ballots for candidate names consisting of random strings pre-specified by the attacker. This way, the attacker can: (1)

Verify that coerced voters complied with instructions, by looking for the random strings the attacker furnished, and (2) Ensure that the votes of coerced voters are not counted, since random strings will most likely not correspond to real election choices. This problem might be avoided using techniques to conceal write-in votes that do not impact the final tally – an interesting problem for further research.

2.2 Summary of the attack model

We consider the process for a single election as proceeding in these phases, corresponding largely with the functions enumerated in section 2.1:

1. **Setup:** If not already available, key pairs are generated for or by $\mathcal{R}$ and $\mathcal{T}$. The candidate slate $\boldsymbol{C}$ for the election is published by $\mathcal{R}$ with appropriate integrity protection.
2. **Registration:** The identities and elegibility of would-be participants in the election are verified by $\mathcal{R}$. Given successful verification, an individual becomes a registered voter, receiving from $\mathcal{R}$ a set of credentials permitting participation in the election. Previously registered voters may be able to re-use their credentials.
3. **Voting:** Using their credentials with reference to the candidate slate $\boldsymbol{C}$, registered voters cast ballots.
4. **Tallying:** The authority $\mathcal{T}$ processes the contents of the bulletin board $\mathcal{BB}$ so as to produce a tally vector $\boldsymbol{X}$ specifying the outcome of the election, along with a proof of correctness P of the tally.
5. **Verification:** Any player, whether or not a participant in the election, can refer to $\mathcal{BB}$ and P to verify the correctness of the tally produced by $\mathcal{T}$ in the previous phase.

Assumptions in setup phase: Our security definitions permit the possibility of static, active corruption by the adversary of a minority of players in $\mathcal{R}$ and $\mathcal{T}$ in the setup phase. The security of our construction then relies on generation of the key pairs $(SK_{\mathcal{T}}, PK_{\mathcal{T}})$ and $(SK_{\mathcal{R}}, PK_{\mathcal{R}})$ by a trusted third party, or, alternatively, an interactive key-generation protocol such as [17] between the players in $\mathcal{R}$ resp. $\mathcal{T}$. In this latter case, the privacy and correctness of the key generation procedure depends on computational security assumptions.

Assumptions prior to registration: The adversary may coerce a voter prior to the registration phase in the sense of requesting in advance that the voter retain transcripts of the registration process, or by providing data in an attempt to dictate voter interaction with the registrar.

Assumptions in registration phase: We make the assumption that the registration phase proceeds without any corruption of voters. This assumption is at some level a requirement for a coercion-free election, as an attacker capable of corrupting and seizing the credentials of a voter in this initial phase can mount a simulation attack. More precisely, we must make *at least one* of three assumptions about the registration phase:

1. Erasure of data from voter interaction with $\mathcal{R}$ is forced (e.g., by difficult-to-modify voting software or smartcards provided to voters). This prevents an attacker from requesting registration transcript data after the fact; or
2. The adversary cannot corrupt any players in $\mathcal{R}$; or
3. Voters become aware of the identity of any corrupted player in $\mathcal{R}$.

The reason we require at least one of these assumptions is as follows. If none of these assumptions holds, then the adversary can, on demanding information from a voter, verify the

correctness of some portion thereof, where the voter would not know what portion is being checked. In other words, the adversary can perform spot checks, with a high probability of successfully detecting false transcripts. In consequence, the adversary can coerce voters into divulging full transcripts of their interactions with $\mathcal{R}$, thereby enabling a simulation attack. In contrast, if at least one of the assumptions holds, we show that it is possible to formulate a protocol that permits the voters to produce information that the adversary cannot distinguish from the requested information, but which will not allow the adversary to cast a vote that will influence the final tally.

Assumptions on voting, tallying and verification phases: Subsequent to the registration phase, we assume that the adversary may seize control of a minority of players in $\mathcal{T}$ and any number of voters in a static, active manner. (Since $\mathcal{R}$ does not participate in the process subsequent to registration, we need not consider adversarial corruption of $\mathcal{R}$ at this point.) The adversary may also coerce voters as desired by requesting that they divulge private keying material[4] or behave in a prescribed manner. Voters are assumed to be able to cast their ballots via fully anonymous channels, i.e., channels such that an attacker cannot determine whether or not a given voter cast a ballot. This assumption is a requirement for any election scheme to be fully coercion-resistant: If an attacker can tell whether or not a given voter cast a ballot, then the attacker can easily mount a forced-abstention attack. In practice, an anonymous channel may be achieved by enabling voters to cast ballots in public places, thereby mixing their votes with others, by use of anonymizing, asynchronous mix-networks, etc.

3 Formal definitions

We now turn our attention to formal security definitions of the essential properties of *correctness*, *verifiability*, and *coercion-resistance*, respectively abbreviated *corr*, *ver*, and $c - resist$. Our definitions hinge on a set of experiments involving an adversary $\mathcal{A}$ in interaction with components of the election system ES. This adversary is assumed to retain state throughout the duration of an experiment. We formulate our experiments such that in all cases, the aim of the adversary is to cause an output value of '1'. Thus, for experiment $\mathbf{Exp}^{E}_{\mathsf{ES},\mathcal{A}}(\cdot)$ on property $E \in (ver, corr, c - resist)$, we define:

$$\mathbf{Succ}^{E}_{\mathsf{ES},\mathcal{A}}(\cdot) = \mathsf{Pr}[\mathbf{Exp}^{E}_{\mathsf{ES},\mathcal{A}}(\cdot) = \text{'1'}].$$

According to the standard definition, we say that a quantity $f(k)$ is *negligible* in k if for every positive integer c there is some l_c such that $f(k) < k^c$ for $k > l_c$. In most cases, we use the term negligible alone to mean negligible with respect to the full set of relevant security parameters. Similarly, in saying that an algorithm has *polynomial running time*, we mean that its running time is asymptotically bounded by some polynomial in the relevant security parameters. As the properties of correctness and verifiability are of less relevance to our work than coercion-resistance, we relegate the first two definitions to appendices A and B.

Coercion resistance: Coercion resistance may be regarded as an extension of the basic property of privacy. Privacy in an election system is defined in terms of an adversary that cannot interact with voters during the election process. In particular, we say that an election is private if such an adversary cannot guess the vote of any voter better than an adversarial algorithm whose only input is the election tally. (Note, for example, in an election where all voters vote

[4] We assume that the coercion takes place remotely, i.e., the adversary cannot extract all user transcripts from, e.g., a hard drive.

Republican, the system may have the property of privacy, even though the adversary knows how all voters cast their ballots in that election.)

Coercion resistance is a strong form of privacy in which it is assumed that the adversary may interact with voters. In particular, the adversary may instruct targeted voters to divulge their private keys subsequent to registration, or may specify that these voters cast ballots of a particular form. If the adversary can determine whether or not voters behaved as instructed, then the adversary is capable of blackmail or otherwise exercising undue influence over the election process. Hence a coercion-resistant voting system is one in which the user can deceive the adversary into thinking that she has behaved as instructed, when the voter has in fact cast a ballot according to her own intentions. Observe that when deployed straightforwardly, the mix-network-based election system mentioned above in section 1 does not possess the property of coercion resistance. This is because an adversary that instructs a voter to divulge her private signing key can check the correctness of the key against the certificate of the voter. Also, note that the adversary may check how many votes were cast for each voter, which enables a successuful forced-abstention attack.

Our definition of coercion resistance requires addition of a new function to voting system ES:

- The function $\mathsf{fakekey}(PK_{\mathcal{T}}, sk, pk) \rightarrow \tilde{sk}$ takes as input the public key of the authorities, and the private/public key pair of the voter. It outputs a spurious key $\tilde{sk}$. (The term "key" may apply more broadly to other types of secret data.)

Of course, for the function $\mathsf{fakekey}$ to enable coercion resistance, the key $\tilde{sk}$ must be indistinguishable by the adversary $\mathcal{A}$ from a valid key, and only distinguishable by a majority of talliers $\mathcal{T}$. This property is captured in our experiment characterizing coercion resistance. To simplify the formulation of the experiment, we assume implicitly that tally is computed by an oracle (with knowledge of $SK_{\mathcal{T}}$). It suffices, however, for $\mathcal{T}$ to be computed via a protocol that achieves correct output and is computationally simulable by the adversary $\mathcal{A}$ (who, it will be recalled, may corrupt a minority of $\mathcal{T}$).

Our definition of coercion resistance also requires a probability distribution D_{n,n_C} over the possible vote selections of n honest voters – including possible null ballots, i.e., abstention on the part of some voters. In other words, D_{n,n_C} is a distribution over vectors $(\beta_1, \beta_2, \ldots, \beta_n) \in (n_C \bigcup \phi)^n$. The need for such a distribution D_{n,n_C} in our definition is as follows. If an adversary knows exactly how all voters apart from the targeted one will cast their ballots, then coercion-resistance is impossible, since the adversary can simply subtract known votes from the total tally to determine whether and how the targeted voter cast a vote. Indeed, even if the adversary merely knows exactly how many distinct, valid ballots will be cast, then he can determine whether or not the targeted voter cast a valid vote. Thus, coercion resistance relies upon uncertainty in the way that voters that are not controlled cast their ballots. Abstention by some enrolled voters and use of invalid credentials by other voters or by the authorities themselves[5] serves the purpose of statistically hiding whether or not the targeted voter cast a ballot. The distribution D_{n,n_C} serves the purpose in our experiment of defining the distribution of the "noise" concealing targeted voter behavior. For a set of n voting credentials $\{sk_i\}$, we let $\mathsf{vote}(\{sk_i\}, PK_{\mathcal{T}}, n_C, D_{n,n_C}, k_2)$ denote the casting of ballots according to distribution D_{n,n_C}. In other words, a vector $(\beta_1, \beta_2, \ldots, \beta_n)$ is drawn from D_{n,n_C} and vote β_i is cast using credential sk_i.

Experiment $\mathbf{Exp}^{c-resist}_{\mathsf{ES},\mathcal{A},H}(k_1, k_2, k_3, n_V, n_A)$
$\{(sk_i, pk_i) \leftarrow \mathsf{register}(SK_{\mathcal{R}}, i, k_2)\}_{i=1}^{n_V}$;
$(V : |V| = n_A) \leftarrow \mathcal{A}(\{pk_i\}_{i=1}^{n_V}$, "control voters");

[5] This may be done to contribute globally to the coercion-resistance of the system, and amounts essentially to the injection of "chaff" into the system.

$(j, \beta) \leftarrow \mathcal{A}(\{sk_i\}_{i \in V}$, "set target voter and vote");
$n_C \leftarrow \mathcal{A}$("choose slate size");
$b \in_R \{0, 1\}$;
$n \leftarrow n_V - n_A - 1$;
if $b = 0$ then
    $\tilde{sk} \leftarrow \mathsf{fakekey}(PK_{\mathcal{T}}, sk_j, pk_j)$;
    $\mathcal{BB} \Leftarrow \mathsf{vote}(sk_j, PK_{\mathcal{T}}, n_C, \beta, k_2)$;
else
    $\tilde{sk} \leftarrow sk_j$;
$\mathcal{BB} \Leftarrow vote(\{sk_i\}_{i \neq j, i \notin V}, PK_{\mathcal{T}}, n_C, D_{n,n_C}, k_2)$;
$\mathcal{BB} \Leftarrow \mathcal{A}(\tilde{sk}, \mathcal{BB}$, "cast ballots");
$(\boldsymbol{X}, P) \leftarrow \mathsf{tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$;
$b' \leftarrow \mathcal{A}(\boldsymbol{X}, P$, "guess b");
if $b' = b$ then
    output '1';
else
    output '0';

A successful adversary $\mathcal{A}$ is one that can distinguish voter behavior better than an adversary $\mathcal{A}'$ that only has knowledge of the tally $\boldsymbol{X}$ and (perhaps implicitly of the distribution D_{n,n_C}). Thus, to characterize the advantage of $\mathcal{A}$, we require a second experiment:

Experiment $\mathbf{Exp}^{c-resist-simple}_{\mathsf{ES},\mathcal{A},H}(k_1, k_2, k_3, n_V, n_A)$
$\{(sk_i, pk_i) \leftarrow \mathsf{register}(SK_{\mathcal{R}}, i, k_2)\}_{i=1}^{n_V - n_A}$;
$\beta \leftarrow \mathcal{A}'$("set target vote");
$b \in_R \{0, 1\}$;
$n \leftarrow n_V - n_A - 1$;
if $b = 0$ then
    $\mathcal{BB} \Leftarrow \mathsf{vote}(sk_1, PK_{\mathcal{T}}, n_C, \beta, k_2)$;
$\mathcal{BB} \Leftarrow vote(\{sk_i\}_{i=2}^{n_V - n_C + 1}, PK_{\mathcal{T}}, n_C, D_{n,n_C}, k_2)$;
$(\boldsymbol{X}, P) \leftarrow \mathsf{tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$;
$b' \leftarrow \mathcal{A}'(\boldsymbol{X}, P$, "guess b");
if $b' = b$ then
    output '1';
else
    output '0';

For adversarial algorithm $\mathcal{A}$, we define:

$$\mathbf{Adv}^{c-resist}_{\mathsf{ES},\mathcal{A}}(\cdot) = \mathbf{Succ}^{c-resist}_{\mathsf{ES},\mathcal{A}}(\cdot) - max_{\mathcal{A}'}[\mathbf{Succ}^{c-resist-simple}_{\mathsf{ES},\mathcal{A}'}(\cdot)].$$

We say that ES is coercion-resistant if for all distributions D_{n,n_C} and polynomial-time algorithms $\mathcal{A}$, the quantity $\mathbf{Adv}^{c-resist}_{\mathsf{ES},\mathcal{A}}(\cdot)$ is negligible.

Remarks: Our definition of coercion resistance here considers the ability of $\mathcal{A}$ to target a single voter for coercion. It is a straightforward matter to adapt the definition for cases where $\mathcal{A}$ targets multiple voters for coercion simultaneously. Other extensions in which, e.g., voters may cast multiple valid ballots, are possible. It should also be noted that our definition is in one sense weaker than those for receipt-freeness. In particular, our definition only assumes statistical

concealment of voter behavior according to the behavior of other election participants, while receipt-free schemes retain their protective measures even if all voters are coerced. Of course, this aspect of receipt-free schemes is due to their exclusion of many types of attacks. Put another way, the weakened guarantee in our definition is an artifact of our covering a comprehensive range of attacks. In practice, we maintain that it is immaterial.

4 A Coercion-Resistant Election Protocol

We are now ready to introduce our protocol proposal. We begin by describing the cryptographic building blocks we employ.

Threshold cryptosystem with re-encryption: Our first building block is a threshold public-key cryptosystem CS that permits re-encryption of ciphertexts with knowledge only of public information. The private key for CS is held by $\mathcal{T}$ in our construction.

To describe our aim in the ideal, we would like any ciphertext C to be perfectly hiding. We would like decryption of a ciphertext to be possible only by having a majority of players in $\mathcal{T}$ agree on a query to a special decryption oracle $\tilde{\mathcal{D}}_{CS}()$. Any decryption performed by $\tilde{\mathcal{D}}_{CS}()$ should, in addition, be publicly verifiable.

Selected primitive: El Gamal: El Gamal [16] represents a natural choice of cryptosystem for our purposes, and is our focus in this paper. We let $\mathcal{G}$ denote the algebraic group over which we employ El Gamal, and q denote the group order. For semantic security, we require that the Decision Diffie-Hellman assumption hold over $\mathcal{G}$ [5, 36]. A public/private key pair in El Gamal takes the form $(y(= g^x), x)$, where $x \in_U Z_q$. We let $\in_U$ here and elsewhere denote uniform, random selection from a set. A ciphertext in El Gamal on message $m \in \mathcal{G}$ takes the form $(\alpha, \beta) = (my^r, y^r)$ for $r \in_U Z_q$. For succinctness of notation, we sometimes let $E_y[m]$ denote a ciphertext on message m under public key y. Further details on our use of El Gamal may be found in appendix D.

Blinding function: We also make use of what we call a *threshold blinding function*. This is a deterministic, one-way function $f : \mathbf{SK} \times \mathcal{G} \rightarrow \mathcal{G}$ that is keyed using a private key $b \in_U \mathbf{SK}$. The private key b is distributed in a $(t, n_{\mathcal{T}})$-threshold manner among the players in $\mathcal{T}$, where $t > n_{\mathcal{T}}/2$. A corresponding public key $\mathbf{PK}$ permits proof of the correct computation of $f_b(\cdot)$.

In the ideal, we would like $f_b(\cdot)$ to be computable only by having a majority of players in $\mathcal{T}$ query a random oracle $\tilde{\mathcal{F}}_b()$. Any query/response pair from $\tilde{\mathcal{F}}_b()$ should, in addition, be publicly verifiable. Another property we require of $f_b(\cdot)$ is that of commutativity with CS. In particular, for any ciphertext C on m, it should be the case that $\tilde{\mathcal{D}}_{CS}(\tilde{\mathcal{F}}_b(C)) = \tilde{\mathcal{F}}_b(m)$. In other words, it should be possible to apply the blinding function to a ciphertext and then decrypt so as to yield a plaintext blinded under $f_b(\cdot)$.

Selected primitive: Threshold undeniable signatures: A convenient choice of f for implementation is a distributed variant of the *undeniable signature* scheme of Chaum and van Antwerpen [11] over a group $\mathcal{G}$ of order q. Here, $f_b(x) = x^b$. The private key $b \in_U Z_q$, and thus the computation of f_b, may be distributed using standard $(t, n_{\mathcal{T}})$ Shamir secret sharing [35] over $GF[q]$ with $t > n_{\mathcal{T}}/2$. Each player then holds a private/public key pair $(b_i, u_i(= g^{b_i}))$ for a published generator g. The one-wayness of f_b depends on the DDH assumption on $\mathcal{G}$. We require distributed one-way functions for two independent private/public key pairs (b, u_b) and $(b', u_{b'})$ in our construction. Commutativity of this primitive with El Gamal is easily seen. For further details, see appendix D.

Mix network: A (re-encryption) mix network (MN) is a distributed protocol that takes as input an ordered set $\boldsymbol{C} = \{C_1, C_2, \ldots, C_d\}$ of ciphertexts generated in a cryptosystem like El Gamal that permits re-encryption. The output of MN is an ordered set $\boldsymbol{C'} = \{C'_{\pi(1)}, C'_{\pi(2)}, , \ldots, C'_{\pi(d)}\}$. Here, $C'_{\pi(i)}$ is a re-encryption of C_i, while π is a uniformly random, secret permutation. This is to say that MN randomly and secretly permutes and re-encrypts inputs. Thus, the special privacy property of a mix network is this: An adversary cannot determine which output ciphertext corresponds to which input ciphertext, i.e., which inputs and outputs have common plaintexts. Stated another way, an adversary cannot determine $\pi(j)$ for any j with probability non-negligibly better than a random guess. A number of mix network constructions have been proposed that offer privacy and robustness against a static, active adversary capable of corrupting any minority of the n players (servers) performing the mix network operation. Some of these constructions offer the additional property of *verifiability*. In other words, a proof is output that is checkable by any party and demonstrates, relative to $\boldsymbol{C}$ and the public key of the ciphertexts that $\boldsymbol{C}$ is correctly constructed. It is convenient to conceptualize MN in terms of an oracle $\tilde{\mathcal{MN}}()$ for MN with the property of public verifiability.

There are many good choices of mix networks for our scheme; some examples of such schemes are those of Furukawa and Sako [15] and Neff [24]. For further details, see appendix D.

Proofs of knowledge and digital signatures: As sketched in the above descriptions, we make use of NIZK proofs in a number of places. We also require a digital signature scheme with security against adaptive chosen message attacks. We do not describe these tools in detail, as they are standard in the literature.

4.1 Our proposed protocol

Setup: The key pairs $(SK_{\mathcal{R}}, PK_{\mathcal{R}})$ and $(SK_{\mathcal{T}}, PK_{\mathcal{T}})$ are generated (in an appropriately trustworthy manner, as described above), and $PK_{\mathcal{T}}$ and $PK_{\mathcal{R}}$ are published along with all system parameters.

Registration: Upon sufficient proof of elegibility from V_i, the registrar $\mathcal{R}$ generates and transmits to V_i a random string $\sigma_i \in_U \mathcal{G}$ that serves as the credential of the voter. Such credentials can be generated in a distributed threshold manner (as in [17]), with each active server of $\mathcal{R}$ sending the voter V_i its credential. $\mathcal{R}$ then adds $S_i = E_{PK_{\mathcal{T}}}[\sigma_i]$ to the voter roll $\boldsymbol{L}$. The voter roll $\boldsymbol{L}$ is maintained on the bulletin board $\mathcal{BB}$ and digitally signed as appropriate by $\mathcal{R}$ using private key $\mathcal{R}$.

We assume that the majority of players in $\mathcal{R}$ are honest, and can thus ensure that the $\mathcal{R}$ provides V_i with a correct credential. Nonetheless, it is possible for $\mathcal{R}$ to furnish V_i with a proof that S_i is a ciphertext on σ_i. To enforce coercion-resistance in the case where erasure of secrets by voters is not automatic, a *designated verifier proof* [20] must be employed for this proof. We note that credentials may be used for multiple elections.

Candidate-slate publication: $\mathcal{R}$ or some other appropriate authority publishes a candidate slate $\boldsymbol{C}$ containing the names and unique identifiers in $\mathcal{G}$ for n_C candidates, with appropriate integrity protection. This authority also publishes a unique, random election identifier ϵ.

Voting: Voter V_i casts a ballot for candidate c_j comprising El Gamal ciphertexts $(C_1^{(i)}, C_2^{(i)})$ respectively on choice c_j and credential σ_i. In particular, for $a_1, a_2 \in_U Z_q$:

$$C_1^{(i)} = (\alpha_1, \beta_1) = (c_j y^{a_1}, g^{a_1}), C_2^{(i)} = (\alpha_2, \beta_2) = (\sigma_i y^{a_2}, g^{a_2}).$$

Additionally, V_i includes NIZK proofs of knowledge of c_j and σ_i. These proofs may be accomplished by providing what amount to Schnorr identification proofs under public keys β_1, β_2, since knowledge of encryption factor a_1 or a_2 implies knowledge of the corresponding plaintext. The challenge values for these proofs, Pf_1 and Pf_2, are dependent on ϵ, C_1, and C_2. V_i posts $B_i = (C_1, C_2, Pf_1, Pf_2)$ to $\mathcal{BB}$ via an anonymous channel.

Tallying: To tally the ballots posted to $\mathcal{BB}$, the authority $\mathcal{T}$ performs the following steps:

1. **Checking proofs:** $\mathcal{T}$ verifies the correctness of all proofs on $\mathcal{BB}$. In particular, $\mathcal{T}$ checks Pf_d for a given ballot by verifying that $g_d^s = w_d g^{\gamma_d}$. Any ballots with invalid proofs are discarded. For the valid, remaining ballots, let $\boldsymbol{A_1}$ denote the list of ciphertexts on vote choices, and let $\boldsymbol{B_1}$ denote the list of ciphertexts on (purported) credentials.
2. **Eliminating duplicates:** $\mathcal{T}$ applies the blinding function f_b to and decrypts each of the ciphertexts in $\boldsymbol{B_1}$. Let $\boldsymbol{B_1'}$ denote the resulting list. According to some pre-determined policy, e.g., timestamps on postings to $\mathcal{BB}$, the tallying authority $\mathcal{T}$ eliminates duplicate ballots. In other words, $\mathcal{T}$ removes all but one ballot sharing the same representative in $\boldsymbol{B_1'}$. This is equivalent to retaining at most one ballot per given credential.
3. **Mixing:** $\mathcal{T}$ applies MN to each of $\boldsymbol{A_1}$ and $\boldsymbol{B_1'}$ (using the same, secret permutation for both). Let $\boldsymbol{A_2}$ and $\boldsymbol{B_2}$ be the resulting lists of ciphertexts.
4. **Checking credentials:** $\mathcal{T}$ applies mix network MN to the encrypted list $\boldsymbol{L}$ of credentials from the voter roll. $\mathcal{T}$ then applies the blinding function $f_{b'}$ to and decrypts each of the ciphertexts in $\boldsymbol{B_2}$ and $\boldsymbol{L}$. Let $\boldsymbol{B_2'}$ and $\boldsymbol{L'}$ denote the resulting lists. If a given value in $\boldsymbol{B_2'}$ does not correspond to any value in $\boldsymbol{L'}$, then the corresponding ballot is removed. Let $\boldsymbol{A_3}$ denote the resulting list of ciphertext voter choices.
5. **Tallying:** $\mathcal{T}$ decrypts all ciphertexts in $\boldsymbol{A_3}$ and tallies the final result.

How to cheat a coercer: One possible implementation of the function fakekey is for the coerced voter V_i to select and reveal a random group element $\tilde{\sigma}_i$, claiming that this is the credential σ_i. (If coerced multiple times – whether for one or more elections – the voter V_i would, of course, release the same value $\tilde{\sigma}_i$.) In addition, partial or full transcripts from the registration phase may be given to the adversary, as will be discussed below.

Upon receiving a claimed credential $\tilde{\sigma}_i$, the adversary would like to verify if it is correct. Let us consider the possibility of doing so under each of our three possible assumptions on the registration phase; in doing so, recall that we always assume that the adversary can corrupt only a minority of servers in $\mathcal{T}$, and so, will not be able to decrypt any of the semantically secure encryptions of credentials.

1. Assume that voters can erase information no longer needed at the end of the registration phase, and that only a minority of servers in $\mathcal{R}$ may be corrupted. At the end of the registration process, each voter will erase information specifying what part of the transcript leading to the credential σ_i he got from what registration server. Without proofs or transcripts from individual servers of $\mathcal{R}$, it is not possible for the adversary to verify the correctness of $\tilde{\sigma}_i$.
2. Assume that the adversary cannot corrupt *any* server in $\mathcal{R}$. As mentioned, the registration servers will use designated verifier proofs to prove to each voter that the share they send is authentic (i.e., will be part of the recorded transcript S_i). While the voter will be convinced of these proofs, the adversary will not; in fact, he cannot distinguish between real such proofs and proofs simulated by V_i. Therefore, V_i can release full *simulated* transcripts from the registration phase, corresponding to a credential $\tilde{\sigma}_i$. The adversary cannot compare this to actual transcripts produced by $\mathcal{R}$, since he does not corrupt any player in $\mathcal{R}$.
3. Assuming that the user knows what (minority of) servers in $\mathcal{R}$ are corrupted, but is not necessarily able to erase data, he can present the adversary with registration transcripts that

are consistent with the view of the servers he knows to be corrupted, but inconsistent (in terms of the real share of σ_i) with the view of the servers that are not. The latter transcripts will be accompanied by simulated designated verifier proofs. Since the adversary may only corrupt a minority of servers in $\mathcal{R}$, and a majority is required to compute the credential σ_i, there will be at least one share of σ_i that V_i can change to obtain a fake credential $\tilde{\sigma}_i \neq \sigma_i$, without the detection of the adversary.

We offer further discussion of security in appendix C in the form of security proof sketches based on oracles defined for our building blocks.

References

1. Proxyvote.com: Shareholder election website, 2002. URL: www.proxyvote.com.
2. Vote-auction, 2002. URL: www.vote-auction.net.
3. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.
4. J.C. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *26th ACM STOC*, pages 544–553, 1994.
5. D. Boneh. The Decision Diffie-Hellman problem. In *ANTS '98*, pages 48–63. Springer-Verlag, 1998. LNCS no. 1423.
6. D. Boneh and P. Golle. Almost entirely correct mixing with applications to voting. In *ACM CCS '02*, 2002. To appear.
7. S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy.* MIT Press, 2000.
8. L. Burke. The tangled web of e-voting. *Wired News*, 26 June 2000.
9. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In B. Pfitzmann, editor, *EUROCRYPT '01*, pages 93–118. Springer-Verlag, 2001. LNCS no. 2045.
10. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.
11. D. Chaum and H. van Antwerpen. Undeniable signatures. In G. Brassard, editor, *CRYPTO '89*, pages 212–216. Springer-Verlag, 1989. LNCS no. 435.
12. R. Cramer, I. Damgard, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. Desmedt, editor, *CRYPTO '94*, pages 174–187. Springer-Verlag, 1994. LNCS no. 839.
13. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In W. Fumy, editor, *EUROCRYPT '97*, pages 103–118. Springer-Verlag, 1997. LNCS no. 1233.
14. A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In J. Seberry and Y. Zheng, editors, *ASIACRYPT '92*, pages 244–251. Springer-Verlag, 1992. LNCS no. 718.
15. J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In J. Kilian, editor, *CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387. Springer-Verlag, 2001.
16. T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.
17. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. The (in)security of distributed key generation in dlog-based cryptosystems. In J. Stern, editor, *EUROCRYPT '99*, pages 295–310. Springer-Verlag, 1999. LNCS no. 1592.
18. M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In B. Preneel, editor, *EUROCRYPT '00*, pages 539–556, 2000. LNCS no. 1807.
19. M. Jakobsson, A. Juels, and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In D. Boneh, editor, *USENIX '02*, pages 339–353, 2002.

20. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In U. Maurer, editor, *EUROCRYPT '96*, pages 143–154. Springer-Verlag, 1996. LNCS no. 1070.
21. A. Kiayias and M. Yung. Self-tallying elections and perfect ballot secrecy. In D. Naccache and P. Paillier, editors, *PKC '02*, pages 141–158. Springer-Verlag, 2000. LNCS no. 2274.
22. E. Magkos, M. Burmester, and V. Chrissikopoulos. Receipt-freeness in large-scale elections without untappable channels. In B. Schmid *et al.*, editor, *First IFIP Conference on E-Commerce, E-Business, E-Government (I3E)*, pages 683–694, 2001.
23. M. Michels and P. Horster. Some remarks on a receipt-free and universally verifiable mix-type voting scheme. In K. Kim and T. Matsumoto, editors, *ASIACRYPT '96*. Springer-Verlag, 1996. LNCS no. 1163.
24. A. Neff. A verifiable secret shuffle and its application to e-voting. In P. Samarati, editor, *ACM CCS '01*, pages 116–125. ACM Press, 2001.
25. V. Niemi and A. Renvall. How to prevent buying of votes in computer elections. In J. Pieprzyk and R. Safavi-Naini, editors, *ASIACRYPT '94*, pages 164–170. Springer-Verlag, 1994. LNCS no. 917.
26. T. Okamoto. An electronic voting scheme. In N. Terashima *et al.*, editor, *IFIP World Congress*, pages 21–30, 1996.
27. T. Okamoto. Receipt-free electronic voting schemes for large scale elections. In B. Christianson *et al.*, editor, *Security Protocols Workshop*, pages 25–35. Springer-Verlag, 1997. LNCS no. 1361.
28. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *EUROCRYPT '99*, pages 223–238. Springer-Verlag, 1999. LNCS no. 1592.
29. S. Parker. Shaking voter apathy up with IT. *The Guardian*, 11 Dec. 2001.
30. K. Sako and J. Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In L. Guillou and J.-J. Quisquater, editors, *EUROCRYPT '95*, pages 393–403. Springer-Verlag, 1995. LNCS no. 921.
31. C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
32. C.-P. Schnorr and M. Jakobsson. Security of signed ElGamal encryption. In *ASIACRYPT*, pages 73–89. Springer, 2000.
33. B. Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In M. Weiner, editor, *CRYPTO '99*, pages 148–164. Springer-Verlag, 1999. LNCS no. 1666.
34. B. Schoenmakers, 2000. Personal communication.
35. A. Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.
36. Y. Tsiounis and M. Yung. On the security of ElGamal-based encryption. In *Workshop on Practice and Theory in Public Key Cryptography (PKC '98)*. Springer, 1998.

A Definitions of Correctness and Verifiability

Correctness: We first consider the property of correctness. This property is in fact twofold: First, it stipulates that an adversary $\mathcal{A}$ cannot pre-empt, alter, or cancel the votes of honest, i.e., voters that are not *controlled*; Second, it stipulates that $\mathcal{A}$ cannot cause voters to cast ballots in such a way as to achieve double voting, i.e., use of one credential to vote multiple times, where more than one vote per credential is counted in the tally.

In our experiment characterizing correctness, we give the adversary powers she does not normally have. Namely, apart from getting to select a set V of voters she will control, we also allow her to choose the candidate-slate size n_C, and to choose what votes will be cast by voters she does not control. The latter voters will indeed vote according to the adversary's wish – but only for the purposes of our thought experiment defining correctness, of course. If the adversary still cannot cause an incorrect tally to be computed (i.e., one not corresponding to the votes cast), then the scheme has the correctness property even in the real-world scenario in which the adversary has less power. The aim of the adversary is to cause more than $|V|$ ballots to be counted in the final tally on behalf of the controlled voters, or to alter or delete the vote of at least one honest voter. (This corresponds to the the condition that: (1) The verification of

the tally succeeds, and (2) That either a vote is "dropped" or "added".) Our definition assumes implicitly that tally is computed correctly by the authority $\mathcal{T}$. (The next property we consider, namely verifiability, addresses the possibility that this is not so.) In what follows, we let $\leftarrow$ denote assignment and $\Leftarrow$ denote the append operation. We let $\langle \boldsymbol{Y} \rangle$ denote the multiset corresponding to entries in the vector $\boldsymbol{Y}$, and $|Y|$ denote the cardinality of set Y.

Experiment $\mathbf{Exp}^{corr}_{\mathsf{ES},\mathcal{A}}(k_1, k_2, k_3, n_V)$
$\{(sk_i, pk_i) \leftarrow \mathsf{register}(SK_{\mathcal{R}}, i, k_2)\}_{i=1}^{n_V}$;
$n_C \leftarrow \mathcal{A}(\{sk_i\}_{i \in V}$, "choose slate size");
$V \leftarrow \mathcal{A}(\{pk_i\}_{i=1}^{n_V}$, "choose controlled voter set");
$\{\beta_i\}_{i \notin V} \leftarrow \mathcal{A}$("choose votes for uncontrolled voters");
$\mathcal{BB} \Leftarrow \{\mathsf{vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta_i, k_2)\}_{i \notin V}$;
$(\boldsymbol{X}, P) \leftarrow \mathsf{tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$;
$\mathcal{BB} \Leftarrow \mathcal{A}$("cast ballots", $\mathcal{BB}$);
$(\boldsymbol{X}', P') \leftarrow \mathsf{tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$;
if $\mathsf{verify}(PK_{\mathcal{T}}, \mathcal{BB}, n_C, \boldsymbol{X}', P') =$ '1' and
$(\{\beta_i\} \not\subset \langle \boldsymbol{X}' \rangle$ or $|\langle \boldsymbol{X}' \rangle| - |\langle \boldsymbol{X} \rangle| > |V|)$ then
output '1';
else
output '0';

We say that ES possesses the property of correctness if for all polynomial-time adversaries $\mathcal{A}$, it is the case that $\mathbf{Succ}^{corr}_{\mathsf{ES},\mathcal{A}}(k_1, k_2, k_3, n_V)$ is negligible.

Verifiability: As explained above, an election system has the property of correctness if computation of tally always yields a valid tabulation of ballots. Given the ability of an adversary $\mathcal{A}$, however, to corrupt some number of authorities among $\mathcal{T}$, we cannot be assured that tally is always computed correctly. The property of verifiability is the ability for any player to check whether the tally $\boldsymbol{X}$ has been correctly computed, that is, to detect any misbehavior by $\mathcal{T}$ in applying the function tally.

A strong security definition for verifiability is appropriate given the high level of auditability required for trustworthy elections. Such a definition considers an attacker $\mathcal{A}$ capable of controlling *all* of the voters and tallying authorities in $\mathcal{T}$. This attacker seeks to construct a set of ballots on $\mathcal{BB}$ and a corresponding tally $\boldsymbol{X}$ and proof P of correct tabulation such that the proof is accepted by verify, but the tally is in fact incorrect. By an incorrect tally, we mean one in which all of the valid ballots of a particular voter (i.e., corresponding to a particular credential) are discounted, or else where multiple votes are tallied that could have been generated by the same voting credential. Our experiment characterizing verifiability is as follows.

Experiment $\mathbf{Exp}^{ver}_{\mathsf{ES},\mathcal{A}}(k_1, k_2, k_3, n_V)$
$\{(sk_i, pk_i) \leftarrow \mathsf{register}(SK_{\mathcal{R}}, i, k_2)\}_{i=1}^{n_V}$;
$n_C \leftarrow \mathcal{A}(\{sk_i\}_{i=1}^{n_V}$, "choose slate size");
$(\mathcal{BB}, \boldsymbol{X}, P) \leftarrow \mathcal{A}(SK_{\mathcal{T}}$, "forge election");
$(\boldsymbol{X}', P') \leftarrow \mathsf{tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$;
if $\boldsymbol{X} \neq \boldsymbol{X}'$ and $\mathsf{verify}(PK_{\mathcal{T}}, \mathcal{BB}, n_C, \boldsymbol{X}, P) =$ '1' then
output '1';
else
output '0';

We say that ES possesses the property of verifiability if for all positive integers n_V and all adversaries $\mathcal{A}$ with polynomial running time, the quantity $\mathbf{Succ}^{ver}_{\mathsf{ES},\mathcal{A}}(k_1, k_2, k_3, n_V)$ is negligible. A technical strengthening of this definition and that for correctness is possible, and discussed in the next section, appendix B, of this paper.

Another aspect of verifiability that we do not formally define, but do mention here and incorporate into our proposed protocol is that of verification against voter rolls. In particular, it may be desirable for any election observer to check that credentials were assigned only to voters whose names are on a published roll. This is not technically a requirement if we rule out corruption of players $\mathcal{R}$, but may still be desirable for high assurance of election integrity. Our definitions can be modified accordingly.

B Remark on strong verifiability

We set forth our definitions of correctness and verifiability to meet the minimal requirements for a fair election and to achieve some measure of conceptual simplicity. The definitions given above are adequate for most election scenarios, but have a technical deficiency that may be of concern in some cases. In particular, our definitions allow for the possibility that a controlled voter casts a ballot corresponding to vote β, but that the ballot gets counted as a vote for β'. Since $\mathcal{A}$ can choose the vote cast by a controlled voter in any case, this technical deficiency only means that $\mathcal{A}$ can potentially cause the votes of *controlled voters only* to change in the midst of the election process. It does not provide $\mathcal{A}$ with control of a larger number of votes.

Nonetheless, one can envisage some (somewhat artificial) scenarios in which stronger guarantees may be desirable. For example, $\mathcal{A}$ might have the aim of causing the victor in an election to win by the slimmest possible margin. In this case, if $\mathcal{A}$ controls a majority of $\mathcal{T}$, then $\mathcal{A}$ might seek to decrypt all of the ballots cast in an election and alter the votes of controlled voters so as to favor the losing candidate.

We discuss now how our definition of verifiability may be modified to discount the possibility of this type of attack. (Analogous modifications may be made to the definition of correctness.) In particular, we can require that P be a proof that every tallied vote corresponds uniquely to a credential for which a valid ballot has been cast. For this, we require a natural technical restriction on vote. Let $\langle \mathsf{vote}(\cdot) \rangle$ denote the set of possible outputs for the randomized function vote on a particular input. We require that an output ballot be wholly unambiguous with respect to both the vote β and the credential sk. In other words, we require $\langle \mathsf{vote}(sk_0, PK_{\mathcal{T}}, n_C, \beta_0, k_2) \rangle \bigcap \langle \mathsf{vote}(sk_1, PK_{\mathcal{T}}, n_C, \beta_1, k_2) \rangle = \phi$ if $\beta_0 \neq \beta_2$ or $sk_0 \neq sk_1$.

To achieve our strengthened definition of verifiability, we alter experiment $\mathbf{Exp}^{ver}_{\mathsf{ES},\mathcal{A}}(k_1, k_2, k_3, n_V)$ such that if the following conditions 1 and 2 are met, then the output of the experiment is '1'. Otherwise it is '0'.

1. $\mathsf{verify}(PK_{\mathcal{T}}, \mathcal{BB}, n_C, \boldsymbol{X}, P) =$ '1'
2. For every injective mapping $f : \langle \boldsymbol{X} \rangle \rightarrow Z_{n_V}$ one of two conditions holds:
   (a) $\exists B : B \in \mathcal{BB}, B \in \langle \mathsf{vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta, k_2) \rangle, \forall j f(j) \neq i$
   (b) $\exists \beta \in \boldsymbol{X} : f(\beta) = i, \forall B \in \mathcal{BB}, B \notin \langle \mathsf{vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta, k_2) \rangle$

Conditions 2(a) and 2(b) here respectively specify that the adversary has successfully defeated the verifiability of the system either by causing all of the valid ballots associated with a particular credential not to be counted or else enabling multiple votes to be tallied for a single credential.

Given use of a verifiable mix network, our proposed protocol meets this stronger security definition for verifiability.

C Security Discussion

Both for conciseness and generality in the following, we consider our building blocks to be oracles, as described in appendix D, and offer heuristic proof sketches relative to the known properties of these, and relative to the assumptions on which our suggested building blocks rest. We show that our proposed protocol satisfies correctness and verifiability, and that it is coercion-resistant given the corruption model outlined.

Coercion-resistance: Under our assumptions, and given the above suggested functionality of fakekey, we know that it is not possible for the adversary to distinguish a valid credential σ_i from an invalid credential $\tilde{\sigma}_i$. It is important to note here that what makes a credential σ_i valid is only the inclusion of a ciphertext S_i on σ_i in the voter roll $\boldsymbol{L}$.

We can see that the first step of the tallying process does not distinguish between valid and invalid ballots, but only removes malformed ballots; similarly, the second step only removes duplicates. Neither of these processes allows the adversary to infer whether some credential he was given was valid or not – only whether the corresponding ballot was correctly cast, resp. how many times a particular credential was used.

The third step is performed by a mix oracle: the elements of its output vector are re-encryptions of the elements of the input list – randomly permuted. It is clear that this step it itself does not allow the adversary to determine whether a given credential was valid or not. The same argument holds for the mixing of the voter roll of the fourth step, and the blinding of elements of the two resulting mixed vectors.

In the fourth step, ballots corresponding to invalid credentials are removed. Assume that the adversary can determine whether the ballot corresponding to a particular entry in the voter roll was removed or not. This would contradict the assumed properties of the mix network, as it would allow the construction of a distinguisher with non-negligible advantage in detremining relationships between input and output elements of the mix. Similarly, if the adversary can determine whether the ballot corresponding to a given entry on $\mathcal{BB}$ was removed or not, then this would lead to a contradiction. Finally, if the adversary could determine whether the ballot corresponding to a particular (and to him known) credential was removed, then this would contradict the assumptions on the blinding function, as it would allow a non-negligible advantage for a corresponding distinguisher. Either of these contradictions would imply that the DDH assumption does not hold, given our suggested choice of building blocks.

Remark: As is natural for a model addressing electronic (particularly Internet-based) voting, our main emphasis is on an attacker that is not physically present, but coerces victims remotely. We note, however, that there are ways in which our propsed scheme can help resist *shoulder-surfing* attacks, e.g., physical oversight by a family member. For example, suppose that the secret key σ for a voter arrives on a smartcard encrypted under her password. Then if the voter provides the wrong password while subjected to oversight, an incorrect secret key is undetectably released. The voter can vote again later using the correct password.

Correctness: In accordance with the experiment associated with our correctness definition, we let the adversary choose the slate size, choose the set of voters he controls, and choose votes for the uncontrolled voters (who will vote accordingly). The adversary further gets to corrupt a minority of servers in $\mathcal{T}$. Her goal is to remove at least one correct ballot vast by a voter she does not control, or to have at least one vote cast by a corrupted voter counted at least twice in the tally. The former corresponds to causing the encrypted credential of a coerced voted to be altered, whether in the processed ballots or in the voter roll, or to otherwise cause the two not to result in the same value after being mixed and blinded. Given that servers in $\mathcal{T}$ only accept transcripts accompanied with valid proofs of correct computation, neither of these

approaches will work, or we obtain an attack against the soundness of at least one building block used in the tallying process. Namely, if a server provides incorrect transcripts, this will be detected with overwhelming probability, and the cheater replaced, after which the computation will restart. Similarly, it is only possible to count a given vote more than once by making the second step (elimination of duplicates) not succeed; this can only be achieved with a non-negligible probability, or some building block must not be sound. The failure of the soundness of any building block translates directly into the failure of either the DDH assumption or the rndom oracle assumption.

Verifiability: Following the experiment associated with the definition of verifiability, we allow the adversary to select the slate size, corrupt any number of servers in $\mathcal{T}$, and corrupt all voters. It is the adversary's goal to cause an incorrect tally from being produced from the ballots on $\mathcal{BB}$, while making it appear that the tally is correctly computed. Turning to our protocol, we can see that each step of the tallying process results in a universally verifiable proof of correct computation – therefore, given the soundness of the building blocks, any modification of a ballot will be detected, as will any incorrect elimination or lack thereof. Therefore, any strategy allowing an adversary to defeat the correctness requirements implies an attack on the soundness of at least one of the building blocks, contradicting either the DDH on $\mathcal{G}$ or the random-oracle assumption on underlying hash functions.

D Details on primitives

Selected primitive: El Gamal: As explained in the body of the paper, El Gamal [16] represents a natural choice of cryptosystem for our purposes, and is our focus in this paper. Recall that we let $\mathcal{G}$ denote the algebraic group over which we employ El Gamal, and q denote the group order. For semantic security, we require that the Decision Diffie-Hellman assumption hold over $\mathcal{G}$ [5, 36]. A public/private key pair in El Gamal takes the form $(y(= g^x), x)$, where $x \in_U Z_q$. We let $\in_U$ here and elsewhere denote uniform, random selection from a set. The private key x may be distributed among the n_T players in $\mathcal{T}$ using (t, n_T)-Shamir secret sharing [35] over $GF[q]$, for $t > n_T/2$. This private key may be generated by a trusted third party or via a computationally secure simulation of this process [17]. Each player then holds a public/private key pair $(y_i(= g^{x_i}), x_i)$, where x_i is a point on the polynomial used for the secret sharing. A ciphertext in El Gamal on message $m \in \mathcal{G}$ takes the form $(\alpha, \beta) = (my^r, y^r)$ for $r \in_U Z_q$. For succinctness of notation, we sometimes let $E_y[m]$ denote a ciphertext on message m under public key y.

To decrypt a ciphertext (α, β), the plaintext $m = \alpha/\beta^x$ is computed. We let $D_x[C]$ denote the decryption of a ciphertext C under private key x. To achieve a threshold decryption of ciphertext (α, β), each active player i publishes a decryption share $\beta_i = \beta^{x_i}$. The value β^x, and thus m, may be computed using standard LaGrange interpolation. Player i may prove the correctness of its share using an NIZK proof of the form $PK\{s : \beta_i = \beta^s \bigwedge u_i = g^s\}$ – essentially two Schnorr identification proofs [31] with conjunction achieved using techniques described in [12]. (The security of such proofs depends on the dicrete-log assumption and the random-oracle assumption [3] on an underlying hash function.). To re-encrypt a ciphertext (α, β), it suffices to multiply it pairwise by a ciphertext on $m = 1$, i.e., to compute a new ciphertext $(\alpha', \beta') = (y^{r'}\alpha, g^{r'}\beta)$ for $r' \in_U Z_q$. We note that another good possible choice of cryptosystem for our scheme is that of Paillier [28].

Selected primitive: Threshold undeniable signatures: A convenient choice of f for implementation is a distributed variant of the *undeniable signature* scheme of Chaum and van Antwerpen [11] over a group $\mathcal{G}$ of order q. Here, $f_b(x) = x^b$. The private key $b \in_U Z_q$ may be distributed using standard $(t, n_\mathcal{T})$ Shamir secret sharing [35] over $GF[q]$ with $t > n_\mathcal{T}/2$. Each

player then holds a private/public key pair $(b_i, u_i(= g^{b_i}))$ for a published generator g. To compute $x' = f_b(x)$ on public input x, each player outputs $x' = x^{b_i}$, along with a non-interactive zero-knowledge (NIZK) proof of correctness. This proof assumes the form $PK\{s : x'_i = x^s \bigwedge u_i = g^s\}$. The one-wayness of f_b depends on the DDH assumption on $\mathcal{G}$. We require distributed one-way functions for two independent private/public key pairs (b, u_b) and $(b', u_{b'})$ in our construction. Where appropriate for generality and simplicity of exposition, we assume the availability of a oracles $\tilde{\mathcal{F}_b}()$ for $f_b(\cdot)$ and $\tilde{\mathcal{F}_{b'}}()$ for $f_{b'}(\cdot)$.

Note that our choice of f_b has the desired property of commutativity with El Gamal. This is because $((my^r)^b, (g^r)^b) = (m^b y^{r'}, y^{r'})$ for encryption factor $r' = rb$. Assuming correct computation, the value of m is concealed in a semantically secure sense under the DDH assumption on $\mathcal{G}$. Blinding yields a natural way to test whether two El Gamal ciphertexts C_0 and C_1 have equal plaintexts without revealing any additional information (provided that neither plaintext is unity, and under the DDH assumption). In particular, for ciphertext $C = (\alpha, \beta)$, let $f_b(C)$ denote the ciphertext resulting from blinding of each element in the pair (α, β). Then it suffices to test whether $D_x[f_b(C_0)] = D_x[f_b(C_1)]$.

Mix networks: As explained above, there are many good choices of mix networks for our scheme. The examples with the strongest security properties are the constructions of Furukawa and Sako [15] and Neff [24]. Both of these employ El Gamal as the underlying cryptosystem, i.e., an input ciphertext $C_i = (\alpha, \beta) = (my^k, g^k)$ for some public key y and published generator g. Security in these constructions is reducible to the Decision Diffie-Hellman assumption and a random-oracle assumption on a hash function. We also note that the security of these and most other mix network constructions relies on a second input $\mathcal{P} = \{P_1, P_2, \ldots, P_d\}$, where P_i is an NIZK proof of knowledge of the plaintext for C_i that serves the purpose of rendering the cryptosystem chosen-ciphertext-attack secure while permitting re-encryption. A common choice for achieving non-malleability on El Gamal ciphertexts is $P_i = PK\{k : \beta_i = g^k\}$ – essentially a Schnorr signature [32].