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 infeasible for the adversary to determine whether a coerced voter complies with the demands.
A first contribution of this paper is to describe and characterize a new and strengthened adversary for coercion in elections; a second is to demonstrate a protocol that is secure against this adversary. While it is clear that a strengthening of attack models is of theoretical relevance, it is important to note that our results lie close to practicality. 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 a fair degree of efficiency with an unusual lack of structural complexity. Furthermore, while previous schemes have required use of an untappable channel, ours only requires an anonymous one.
A surprising and counter-intuitive achievement of our protocol is that it combines universal verifiability – the ability for anyone to verify the correctness of the election – 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 [12], and in limited tests of electronic voting in France [3] and Japan [4] in 2002. Plans are to allow British voters to vote electronically in the 2006 general elections [25]. These are just a few instances of a broadening trend toward Internet-based voting. While voting of this kind appears to encourage higher voter turnout [40] and make accurate accounting for votes easier, it also carries the potential of making abuse easier to perform, and easier to perform at a large scale. 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 [10, 20, 22, 28, 30, 35, 38, 44].

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* [18]. 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 versions of the 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 approach is frequently advocated in the mix-network literature, as in, e.g., [10, 18, 22, 28].

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 particular candidate, 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 [6, 20] and schemes based on blind signatures [21, 38], 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 [38]. 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 [6]; their scheme, based on homomorphic encryption, was shown in [26] not to possess receipt-freeness as postulated. An independent introduction of the idea appeared in Niemi and Renvall [36]. Okamoto [37] 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 [38]. Sako and Kilian [42] proposed 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 [34]. The Sako and Kilian scheme served as a conceptual basis for the later work of Hirt and Sako [26], the most efficient (and correct) receipt-free scheme voting to date. A recenly proposed scheme by Magkos *et al.* [32] 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 [38] 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 [38]) lose the property of coercion-resistance if the attacker is able to corrupt even one of the tallying authorities in a 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:

[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 that the adversary has pre-selected. The proof then becomes transferable, yielding a means of receipt construction by the adversary. As noted in [26], this type of attack also explains why *deniable encryption* [17] does not solve the problem of coercion in a voting system.

Randomization attack: This attack was noted by Schoenmakers [45], who described its 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 with a large probability. 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 these permit an attacker to verify the correctness of private keying material. For example, in [38], the voter provides a digital signature which, if correct, results in the authority furnishing a blind digital signature. In [26], the voter, when casting a ballot, proves knowledge of a private key relative to a publicly committed or published value. In general, 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 a voting scheme that possesses the strong property of coercion-resistance proposed in this paper – and also naturally 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. A drawback of our scheme is that, even with use of asymptotically efficient mix networks as in [22, 35], the overhead for tallying authorities is quadratic in the number of voters. Thus the scheme is only practical for small elections. Our hope and belief, however, is that our proposed scheme might serve as the basis for refinements with a higher degree of practical application. We outline a security proof for our proposed scheme in the paper appendix.

[2] An exception is the scheme in [38], 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.

1.3 Intuition behind our scheme

In a conventional voting scheme, and also in receipt-free schemes like [26], 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 spirit to, e.g., [11, 13]. 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 the plaintext names of registered voters.

By means of mixing and blind comparison of ciphertext values, 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. 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, the individuals associated with bad ballots will remain concealed.) 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 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, a detailed security-proof outline, and details on our choice of primitives for realizing our proposed scheme.

2 Modelling

An election system consists of several sets of entities:

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 a public identifier for V_i.

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 write data to $\mathcal{BB}$, but cannot overwrite or erase existing data. Moreover, voters will be able to read the contents of $\mathcal{BB}$ once the vote casting phase has ended. For notational convenience, we assume that data are written to $\mathcal{BB}$ in μ-bit blocks for an appropriate choice of μ. 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_j may be identified according to its index j. 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_j indicates the number of votes cast for choice c_j. The protocols composing an election system are then as follows:

- **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 specification 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 incorrect and a '1' otherwise. (We characterize the behavior of verify more formally in the paper appendix.)

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. (Thus, this would combine the forced abstentation attack and the randomization attack.)

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 eligibility 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 credential permitting participation in the election. Previously registered voters may be able to re-use their credentials. $\mathcal{R}$ publishes a voter roll $\boldsymbol{L}$.
3. **Voting:** Referring to the candidate slate $\boldsymbol{C}$, registered voters use their credentials to 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 $\mathcal{BB}, P$ and $\boldsymbol{L}$ 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, on an interactive, computationally secure key-generation protocol such as [24] between the players in $\mathcal{R}$ and those in $\mathcal{T}$.

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 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 is coercion-resistant.

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 attempt to coerce voters outside its control by requesting that they divulge private keying material[3] or behave in a prescribed manner in voting. 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, or by use of anonymizing, asynchronous mix-networks, and so forth.

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

[3] We assume that the coercion takes place remotely. For example, the adversary may not continuously watch over the shoulder of a voter, monitor her hard-drive, etc. Our proposed protocol does potentially defend against some shoulder-surfing, however, by permitting voters to use fake keys and/or re-vote.

of '1'. Thus, for experiment $\mathbf{Exp}^{E}_{\mathsf{ES},\mathcal{A}}(\cdot)$ on property $E \in (ver, corr, c\text{-}resist)$, 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 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.

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 $\tilde{sk}$.

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 centers on a kind of game between the adversary $\mathcal{A}$ and a voter targeted by the adversary for coercive attack. A coin is flipped; the outcome is represented by a bit b. If $b = 0$, then the voter casts a ballot with a particular choice β, and provides the adversary with a false voting key $\tilde{sk}$; in other words, the voter attempts to evade adversarial coercion. If $b = 1$, on the other hand, then the voter submits to the coercion of the adversary; she simply furnishes the adversary with her valid voting key sk, and does not cast a ballot. The task of the adversary is to guess the value of the coin b, that is, to determine whether or not the targeted voter in fact cast a ballot. We permit the adversary in this definitional game to specify the ballot value β. While it is somewhat unnatural for the adversary thus to specify the intention of the voter, this permits us to achieve the strongest possible security definition.

If the adversary has perfect knowledge about the intentions of all voters, then coercion is unavoidable. For example, if the adversary is attempting to coerce one voter in a given election and knows that all hundred of the other eligible voters will cast ballots, then the adversary can mount an abstention attack straightforwardly. The adversary in this case simply threatens the voter in the case that the total tally for the election is one hundred and one. Similarly, suppose that the adversary does not know whether or not any given voter will cast a ballot, but knows that all participating voters will cast a ballot for the Republican party. In this case, the adversary can win the game we describe above by specifying a ballot value $\beta =$"Democrat".

It is evident therefore that for any definition of coercion-resistance to be meaningful, the adversary must have uncertain knowledge about how – and indeed whether – some voters will cast their ballots. In other words, coercion-resistance requires that there be some "noise" or statistical uncertainty in the adversary's view of voting patterns. To our benefit, it is natural to expect that in a real-world election an adversary can obtain only fragmentary knowledge about the likely behavior of voters. This means that coercion-resistance is a viable possibility.[4] For a collection of n voters outside the control of the adversary – i.e., voters not subject to coercion – we characterize the view of the adversary in terms of a probability distribution D_{n,n_C}. We let ϕ be a symbol denoting a null ballot, i.e., an abstention, and let λ denote a ballot cast with an invalid credential. Then D_{n,n_C} is a distribution over vectors $(\beta_1, \beta_2, \ldots, \beta_n) \in (n_C \bigcup \phi \bigcup \lambda)^n$, i.e., over the set of possible ballot choices for an election plus abstentions and invalid ballots. Thus, the distribution D_{n,n_C} serves the purpose in our experiment of defining the distribution of the "noise" that conceals the behavior of voters targeted by the adversary for coercion. 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.

We are now ready to present an experiment *c-resist* that defines the game described above between an adversary and a voter targeted for coercion. Recall that k_1, k_2, and k_3 are security parameters defined above, n_V is the total number of eligible voters for the election, and n_C is the number of candidates, i.e., the size of the candidate slate. We let n_A denote the number of voters that may be completely controlled, i.e., corrupted by the adversary. We define $n_U = n_V - n_A - 1$. In other words, the number of uncertain votes n_U equals the total number of possible votes, minus those coming from voters controlled by the attacker, minus the vote coming from the voter the attacker is trying to coerce (in the experiment). Note that n_U is therefore the number of voters that contribute "noise" to the experiment.

We let $\leftarrow$ denote assignment and $\Leftarrow$ denote the append operation, while % denotes the beginning of an annotative comment on the experiment. Our experiment treats the case in which the adversary seeks to coerce a single voter; extension of the definition to coercion of multiple voters is straightforward. The experiments defined here halt when an output value is produced.

Experiment $\mathbf{Exp}^{c\text{-}resist}_{\mathsf{ES},\mathcal{A},H}(k_1, k_2, k_3, n_V, n_A, n_C)$

$\{(sk_i, pk_i) \leftarrow \mathsf{register}(SK_{\mathcal{R}}, i, k_2)\}_{i=1}^{n_V}$;	% voters are registered
$V \leftarrow \mathcal{A}(\{pk_i\}_{i=1}^{n_V}$, "control voters");	% $\mathcal{A}$ corrupts voters
$(j, \beta) \leftarrow \mathcal{A}(\{sk_i\}_{i \in V}$, "set target voter and vote");	% $\mathcal{A}$ sets coercive target
if $\lvert V \rvert \neq n_A$ or $j \notin \{1, 2, \ldots, n_V\} - V$ or $\beta \notin \{1, 2, \ldots, n_C\} \cup \phi$ then	% outputs of $\mathcal{A}$ checked for validity
output '0';	
$b \in_U \{0, 1\}$;	% coin is flipped
if $b = 0$ then	% voter evades coercion
$\tilde{sk} \leftarrow \mathsf{fakekey}(PK_{\mathcal{T}}, sk_j, pk_j)$;	
$\mathcal{BB} \Leftarrow \mathsf{vote}(sk_j, PK_{\mathcal{T}}, n_C, \beta, k_2)$;	
else	% voter submits to coercion
$\tilde{sk} \leftarrow sk_j$;	
$\mathcal{BB} \Leftarrow \mathsf{vote}(\{sk_i\}_{i \neq j, i \notin V}, PK_{\mathcal{T}}, n_C, D_{n_U,n_C}, k_2)$;	% ballots posted for honest voters
$\mathcal{BB} \Leftarrow \mathcal{A}(\tilde{sk}, \mathcal{BB}$, "cast ballots");	% $\mathcal{A}$ posts to $\mathcal{BB}$
$(\boldsymbol{X}, P) \leftarrow \mathsf{tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$;	% election results are tallied
$b' \leftarrow \mathcal{A}(\boldsymbol{X}, P$, "guess b");	% $\mathcal{A}$ guesses coin flip
if $b' = b$ then	% experimental output determined
output '1';	
else	
output '0';	

[4] Additionally, it is possible for voting authorities – or indeed any entity – intentionally to inject "chaff" in the form of blank and invalid ballots into an election system.

The adversary $\mathcal{A}$ in the above experiment is quite powerful, being capable (when $b = 1$) of complete coercion of the targeted voter. In order to characterize the success of $\mathcal{A}$, we must compare $\mathcal{A}$ with a second, weak adversary $\mathcal{A}'$. $\mathcal{A}'$ is capable of coercion only within the framework of an ideal voting system. In other words, $\mathcal{A}'$ characterizes the type of security against coercion that we would like to achieve in ES. The adversary $\mathcal{A}'$ may interact with the election system in only one way. In particular, $\mathcal{A}'$ may specify two ballot values: β_0, which represents the ballot that the targeted voter wishes to cast, and β_1, which represents the ballot that the adversary wishes the targeted voter to cast. A coin b is flipped determining which of these two values is counted in the election, i.e., added to the tally of votes cast by uncontrolled (i.e., honest) voters. The task of the adversary is to guess the value of b.

This weak adversary $\mathcal{A}'$ does not receive any keying material, and does not actively control any voters. (In fact, the election considered here is one in which only n_U uncontrolled voters participate, along with the single targeted voter.) Moreover, instead of the full contents of the bulletin board $\mathcal{BB}$, the adversary is furnished instead only with a value Γ, denoting the total number of ballots posted to $\mathcal{BB}$. Thus, $\mathcal{A}'$ is capable only of using knowledge of the tally $\boldsymbol{X}$, the input-ballot count Γ, and (implicitly) the distribution D_{n_U,n_C} in order to determine the behavior of the targeted voter.

One additional feature of the experiment occurs when $b = 0$, i.e., when the voter is regarded as attempting to evade coercion. In this case, when making an attempt to post a ballot using the fake credential supplied by the voter, the adversary may post an invalid ballot. This ballot potentially provides the adversary with some information, as it will be discarded during the weeding process, increasing by one the difference between Γ and the total number of tallied votes in $\mathcal{X}$. To reflect this situation, we assume, when $b = 0$, that an invalid ballot is posted to the bulletin board. (Provided that the associated credential is invalid, the form of this ballot does not affect our definition, and may be, e.g., random.) We loosely denote this invalid ballot in our experiment by λ.

We are now ready to present the experiment *c-resist-weak* that characterizes the success of $\mathcal{A}'$:

Experiment $\mathbf{Exp}^{\textit{c-resist-weak}}_{\mathsf{ES},\mathcal{A},H}(k_1, k_2, k_3, n_C, n_V, n_A)$

$\{(sk_i, pk_i) \leftarrow \mathsf{register}(SK_{\mathcal{R}}, i, k_2)\}_{i=1}^{n_U+1}$; % voters are registered

$\beta_0, \beta_1 \leftarrow \mathcal{A}'(\text{“set target votes”})$; % $\mathcal{A}$ selects vote choices

if $\beta_0, \beta_1 \notin \{1, 2, \ldots, n_C\} \cup \phi$ then % outputs of $\mathcal{A}$ checked for validity

    output ‘0’;

$b \in_U \{0, 1\}$; % coin is flipped

if $b = 0$ then % voter evades coercion

    $\mathcal{BB} \Leftarrow \mathsf{vote}(sk_1, PK_{\mathcal{T}}, n_C, \beta_0, k_2)$;

    $\mathcal{BB} \Leftarrow \lambda$;

else % voter submits to coercion

    $\mathcal{BB} \Leftarrow \mathsf{vote}(sk_1, PK_{\mathcal{T}}, n_C, \beta_1, k_2)$;

$\mathcal{BB} \Leftarrow \mathsf{vote}(\{sk_i\}_{i=2}^{n_U+1}, PK_{\mathcal{T}}, n_C, D_{n_U,n_C}, k_2)$; % ballots posted for honest voters

$(\boldsymbol{X}, \Gamma) \leftarrow \mathsf{tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_U+1}, k_3)$; % election results are tallied

$b' \leftarrow \mathcal{A}'(\boldsymbol{X}, \Gamma, \text{“guess } b\text{”})$; % $\mathcal{A}$ guesses coin flip

if $b' = b$ then % experimental output determined

    output ‘1’;

else

    output ‘0’;

For adversarial algorithm $\mathcal{A}$, we define $\mathbf{Adv}^{\textit{c-resist}}_{\mathsf{ES},\mathcal{A}}(\cdot) = \mathbf{Succ}^{\textit{c-resist}}_{\mathsf{ES},\mathcal{A}}(\cdot) - max_{\mathcal{A}'}[\mathbf{Succ}^{\textit{c-resist-weak}}_{\mathsf{ES},\mathcal{A}'}(\cdot)]$. We say that ES is coercion-resistant if for all n_V, n_A, n_C, all probability distributions D_{n_U,n_C}, and all algorithms $\mathcal{A}$ that are polynomial in the security parameters for the experiment, the quantity $\mathbf{Adv}^{\textit{c-resist}}_{\mathsf{ES},\mathcal{A}}(\cdot)$ is negligible. Briefly stated, a coercion-resistant election system is one in which a powerful adversary $\mathcal{A}$ can do no better than the weak adversary $\mathcal{A}'$ characterized by the above experiment.

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. Where appropriate, we model these as ideal primitives, as discussed in appendix D.

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 parameters and keys. 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 E to be perfectly hiding. We would like decryption to be possible only by having a majority of players in $\mathcal{T}$ agree on a ciphertext to be decrypted. We model this latter ideal property as in terms of a special decryption oracle denoted by $D\tilde{E}C$. We assume further that any decryption performed by $D\tilde{E}C$ is publicly verifiable.

Selected cryptosystem: El Gamal [23] 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 $\mathcal{G}$ [8, 47]. 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 $(\alpha, \beta) = (my^r, g^r)$ 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 E. An important feature of El Gamal is that it may be easily implemented in a threshold setting. In other words, the private key x may be distributed such that decryption can be performed by any quorum of share holders, without leakage of additional information. We exploit this distributed form of El Gamal in our proposed election scheme. As explained above, rather than focusing on a particular embodiment, we model the process by a decryption oracle denoted by $D\tilde{E}C$. We refer the reader to appendix E and to [16] for further discussion of threshold decryption in El Gamal.

Plaintext Equivalence Test (PET): A *plaintext equivalence test* (PET) [27, 31] is cryptographic primitive that operates on ciphertexts in a threshold cryptosystem. The input to PET is a pair of ciphertexts; the output is a single bit indicating whether the corresponding plaintexts are equal or not. PET may be realized as an efficient distributed protocol that reveals no additional, non-negligible information about plaintexts. For a detailed description of efficient methods to perform this verification, along with proofs of the properties of the construction, see [31]. Rather than focusing on a specific embodiment of PET, we model the ideal properties of the primitive by means of an oracle denoted by $P\tilde{E}T$, and with the property of public verifiability.

Mix network: A (re-encryption) mix network (MN) is a distributed protocol that takes as input an ordered set $\boldsymbol{E} = \{E_1, E_2, \ldots, E_d\}$ of ciphertexts generated in a cryptosystem like El Gamal that permits re-encryption. The output of MN is an ordered set $\boldsymbol{E'} = \{E'_{\pi(1)}, E'_{\pi(2)}, \ldots, E'_{\pi(d)}\}$. Here, $E'_{\pi(i)}$ is a re-encryption of E_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 $\boldsymbol{E}$ and the public key of the ciphertexts that $\boldsymbol{E}$ is correctly constructed. It is convenient to conceptualize MN as an ideal primitive in terms of an oracle $\tilde{MN}$ 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 [22] and Neff [35]. For further details, see appendix E.

Proofs of knowledge: As sketched in the above descriptions, we make use of NIZK (non-interactive zero-knowledge) proofs of knowledge [7] in a number of places. We do not describe these tools in detail, as they are standard tools in the cryptographic literature. Instead, we refer the reader to, e.g. [19], for discussion of construction and logical composition of such protocols, and [15] for a notational overview and discussion of efficient realization. As is the usual case, our use of NIZK proofs enforces a reliance on the random oracle model in the security proofs for our scheme [5].

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 eligibility 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 [24]), 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 $\boldsymbol{L}$.[5] The voter roll $\boldsymbol{L}$ is maintained on the bulletin board $\mathcal{BB}$ and digitally signed as appropriate by $\mathcal{R}$.

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* [29] 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 $(E_1^{(i)}, E_2^{(i)})$ respectively on choice c_j and credential σ_i. In particular, for $a_1, a_2 \in_U Z_q$:

$$E_1^{(i)} = (\alpha_1, \beta_1) = (c_j y^{a_1}, g^{a_1}), E_2^{(i)} = (\alpha_2, \beta_2) = (\sigma_i y^{a_2}, g^{a_2}).$$

The first is a ciphertext on the candidate choice of the voter, the second a ciphertext on the credential of the voter.

Additionally, V_i includes NIZK proofs of knowledge of σ_i and c_j, and also an NIZK proof that $c_j \in \boldsymbol{C}$, i.e., that c_j represents a valid candidate choice. The latter can be accomplished, for example, using a disjuctive proof that the ciphertext constitutes a valid encryption of a candidate choice in $\boldsymbol{C}$. These three NIZK proofs, which we denote collectively by Pf, may be accomplished efficiently using techniques described in, e.g., [19] (essentially by conjoining a set of Schnorr signatures [43]). As is standard practice, the challenge values for Pf are constructed using a call to a cryptographic hash function, modeled in our security analysis by a random oracle $O\tilde{W}$. Input to $O\tilde{W}$ for these challenge values includes ϵ, E_1, and E_2 and commitment values required for realization of the NIZK proofs. V_i posts $B_i = (E_1, E_2, Pf)$ to $\mathcal{BB}$ via an anonymous channel.

[5] In our definitions above, we use the common terminology of private and public keys – with corresponding notation sk_i and pk_i – to describe the credentials associated with voters. Shifting from a general exposition to our specific protocol, we now use σ_i instead of sk_i to denote a voter credential, and S_i instead of pk_i to denote a public representation thereof. This change of notation aims to reflect the fact that voters do not employ a conventional form of public-key authentication in our scheme.

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 $\mathcal{BB}$. Any ballots with invalid proofs are discarded. For the valid, remaining ballots, let $\boldsymbol{A_1}$ denote the list of ciphertexts on candidate choices (i.e., the E_1 ciphertexts), and let $\boldsymbol{B_1}$ denote the list of ciphertexts on credentials (i.e., the E_2 ciphertexts).
2. **Eliminating duplicates:** The tallying authority $\mathcal{T}$ performs pairwise PETs on all ciphertexts in $\boldsymbol{B_1}$, and removes duplicates according to some pre-determined policy, using e.g., order of postings to $\mathcal{BB}$. When an element is removed from $\boldsymbol{B_1}$, the corresponding element (i.e., that with the same index) is removed from $\boldsymbol{A_1}$. We let $\boldsymbol{B'_1}$ and $\boldsymbol{A'_1}$ be the resulting "weeded" vectors. This is equivalent to retaining at most one ballot per given credential.
3. **Mixing:** $\mathcal{T}$ applies MN to $\boldsymbol{A'_1}$ 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 compares each ciphertext of $\boldsymbol{B_2}$ to the ciphertexts of $\boldsymbol{L}$ using PET. $\mathcal{T}$ retains a vector $\boldsymbol{A_3}$ of all ciphertexts of $\boldsymbol{A_2}$ for which the corresponding elements of $\boldsymbol{B_2}$ match an element of $\boldsymbol{L}$ according to PET. This step achieves the weeding of ballots based on invalid voter credentials.
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 simply 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. We discuss the process of faking voting keys in more detail in appendix C.

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

References

1. Proxyvote.com: Shareholder election website, 2002. URL: www.proxyvote.com.
2. Vote-auction, 2002. URL: www.vote-auction.net.
3. France to test electronic voting. *Computer Weekly CW360*, 26 March 2002.
4. Electronic voting debuts in Japan. *Japan Information Network*, 7 August 2002.
5. 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.
6. J.C. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *26th ACM STOC*, pages 544–553, 1994.
7. Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.
8. D. Boneh. The Decision Diffie-Hellman problem. In *ANTS '98*, pages 48–63. Springer-Verlag, 1998. LNCS no. 1423.
9. D. Boneh and M.K. Franklin. An efficient public key traitor tracing scheme. In M.J. Wiener, editor, *EUROCRYPT '99*, pages 338–353. Springer-Verlag, 1999. LNCS no. 1666.
10. D. Boneh and P. Golle. Almost entirely correct mixing with applications to voting. In *ACM CCS '02*, 2002. To appear.
11. S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy.* MIT Press, 2000.
12. L. Burke. The tangled web of e-voting. *Wired News*, 26 June 2000.
13. 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.
14. J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In J. Stern, editor, *EUROCRYPT '99*, pages 107–122. Springer-Verlag, 2001. LNCS no. 1592.
15. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In B. Kaliski, editor, *CRYPTO '97*, pages 410–424. Springer-Verlag, 1997. LNCS no. 1294.

16. R. Canetti, R. Gennaro, S. Jareckiand H. Krawczyk, and T. Rabin. Adaptive security for threshold cryptosystems. In M. Wiener, editor, *CRYPTO '99*, pages 98–115. Springer-Verlag, 1999. LNCS no. 1666.
17. Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In B. Kaliski, editor, *CRYPTO '97*, pages 90–104, 1997. LNCS no. 1294.
18. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.
19. 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.
20. 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.
21. 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.
22. 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.
23. T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.
24. 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.
25. D. Hencke. E-votes will push out ballot box 'by 2006'. *The Guardian*, July 17 2002.
26. 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.
27. M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In T. Okamoto, editor, *Advances in Cryptology - Asiacrypt '00*, pages 162–177. Springer-Verlag, 2000. LNCS No. 1976.
28. 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.
29. 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.
30. 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.
31. P. MacKenzie, T. Shrimpton, and M. Jakobsson. Threshold password-authenticated key exchange. In M. Yung, editor, *CRYPTO '02*, pages 385–400, 2002. LNCS no. 2442.
32. 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.
33. W. Mao. Verifiable partial sharing of integer factors. In *Selected Areas in Cryptography (SAC '98)*. Springer-Verlag, 1998. LNCS no. 1556.
34. 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.
35. 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.
36. 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.
37. T. Okamoto. An electronic voting scheme. In N. Terashima *et al.*, editor, *IFIP World Congress*, pages 21–30, 1996.
38. 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.
39. 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.
40. S. Parker. Shaking voter apathy up with IT. *The Guardian*, 11 Dec. 2001.
41. D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. M. Maurer, editor, *EUROCRYPT '96*, pages 387–398. Springer-Verlag, 1996. LNCS no. 1070.
42. 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.
43. C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

44. B. Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In M. Wiener, editor, *CRYPTO '99*, pages 148–164. Springer-Verlag, 1999. LNCS no. 1666.
45. B. Schoenmakers, 2000. Personal communication.
46. A. Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.
47. 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 $\langle \boldsymbol{Y} \rangle$ denote the multiset corresponding to entries in the vector $\boldsymbol{Y}$, and $|Y|$ denote the cardinality of set Y.

Experiment $\mathbf{Exp}^{corr}_{\mathsf{ES},\mathcal{A}}(k_1, k_2, k_3, n_C, n_V)$

$\{(sk_i, pk_i) \leftarrow \mathsf{register}(SK_{\mathcal{R}}, i, k_2)\}_{i=1}^{n_V}$; % voters are registered
$V \leftarrow \mathcal{A}(\{pk_i\}_{i=1}^{n_V}$, "choose controlled voter set"); % $\mathcal{A}$ corrupts voters
$\{\beta_i\}_{i \notin V} \leftarrow \mathcal{A}$("choose votes for uncontrolled voters"); % A chooses votes for honest voters
$\mathcal{BB} \Leftarrow \{\mathsf{vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta_i, k_2)\}_{i \notin V}$; % honest voters cast ballots
$(\boldsymbol{X}, P) \leftarrow \mathsf{tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$; % honest ballots are tallied
$\mathcal{BB} \Leftarrow \mathcal{A}$("cast ballots", $\mathcal{BB}$); % $\mathcal{A}$ posts ballots to $\mathcal{BB}$
$(\boldsymbol{X'}, P') \leftarrow \mathsf{tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$; % all ballots are tallied
if $\mathsf{verify}(PK_{\mathcal{T}}, \mathcal{BB}, n_C, \boldsymbol{X'}, P') =$ '1' and % does function verify accept?
($\{\beta_i\} \not\subset \langle \boldsymbol{X'} \rangle$ or $|\langle \boldsymbol{X'} \rangle| - |\langle \boldsymbol{X} \rangle| > |V|$) then % did $\mathcal{A}$ successfully tamper?
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 $\mathbf{Exp}^{ver}_{\mathsf{ES},\mathcal{A}}(k_1, k_2, k_3, n_C, n_V)$

$\{(sk_i, pk_i) \leftarrow \mathsf{register}(SK_{\mathcal{R}}, i, k_2)\}_{i=1}^{n_V}$; % voters are registered

$(\mathcal{BB}, \boldsymbol{X}, P) \leftarrow \mathcal{A}(SK_{\mathcal{T}}, \{(sk_i, pk_i)\}_{i=1}^{n_V}$, "forge election"); % $\mathcal{A}$ concocts full election

$(\boldsymbol{X'}, P') \leftarrow \mathsf{tally}(SK_{\mathcal{T}}, \mathcal{BB}, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$; % tally is taken on $\mathcal{BB}$

if $\boldsymbol{X} \neq \boldsymbol{X'}$ % does $\mathcal{A}$'s tally differ from correct $\mathcal{BB}$ tally?

and $\mathsf{verify}(PK_{\mathcal{T}}, \mathcal{BB}, n_C, \boldsymbol{X}, P) =$ '1' then % does function verify accept?

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 in appendix A to meet the minimal requirements for a fair election and to achieve some measure of conceptual simplicity. These definitions 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 voter controlled by $\mathcal{A}$ 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. Most importantly, we note that this definitional weakness does not apply to our proposed protocol, which meets the stronger definition we now set forth.

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 $\beta_0 \neq \beta_1$ 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 The Faking of Voting Keys

We provide some more detail here on the process whereby a voter fakes a voting credential in our proposed protocol. 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 discussed in the body of the paper; 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 there is a mechanism forcing erasure of voter 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 may if desired 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 convincingly release full *simulated* transcripts from the registration phase, corresponding to a credential $\tilde{\sigma}_i$.
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.

D Proving Coercion-Freeness

In this section, we provide a detailed outline for proof of the property of coercion-freeness in our proposed election protocol. (We do not consider correctness or verifiability here, as these are more standard properties, and the associated proofs are more straightforward.) For the purposes of this proof, we assume the use of the El Gamal cryptosystem over a preselected group $\mathcal{G}$ of order q. The coercion-freeness of our scheme is dependent on the Decision-Diffie Hellman (DDH) assumption on $\mathcal{G}$. Briefly stated, this assumption states that no algorithm with running-time polynomial in the security parameters for $\mathcal{G}$ can distinguish between the two distributions D and D' with non-negligible probability:

Here, D is the distribution of tuples of the form (y_1, g_1, y_2, g_2), where $g_1, g_2 \in_U \mathcal{G}$, $y_1 = g_1^x$, and $y_2 = g_2^x$ for $x \in_U Z_q$; i.e., the pair (y_1, g_1) and (y_2, g_2) are related by a common exponent. D' is the distribution of random tuples, i.e., tuples of the form (y_1, g_1, y_2, g_2), where $y_1, g_1, y_2, g_2 \in_U \mathcal{G}$. For detailed treatment of this assumption (expressed in an alternative, equivalent form), see, e.g., [8].

D.1 Assumptions

As explained above, we simplify our analysis by assuming ideal constructions for a number of components in our election protocol. Our aim in doing so is twofold: (1) Our protocol is flexible enough to accommodate a range of cryptographic building blocks from the literature and (2) We wish to retain a focus on the conceptual and definition elements of our paper, and not on protocol details. Hence, we assume the availability of oracles for the four following cryptographic operations in our protocol: mixing, plaintext equivalence testing (PET), threshold ciphertext decryption, and calls to the one-way or hash function required for NIZK proofs. As in the main body of the paper, denote these oracles respectively by $\tilde{MN}, \tilde{PET}, \tilde{DEC}$ and $\tilde{OW}$. Although the functioning of these oracles should be clear from our protocol description, we present it again here:

- The oracle $\tilde{MN}$ performs exactly the same function as a mix network. It accepts as input an ordered list $\boldsymbol{E} = \{E_1, E_2, \ldots, E_d\}$ of ciphertexts under the public key $PK_{\mathcal{T}}$ of the tallying authorities. Its output on $\boldsymbol{E}$ is an ordered set $\boldsymbol{E'} = \{E'_{\pi(1)}, E'_{\pi(2)}, \ldots, E'_{\pi(d)}\}$ for a secret, random permutation π, where $E'_{\pi(i)}$ represents a re-encryption of ciphertext E_i.
- The oracle $\tilde{PET}$ takes as input a pair of ciphertexts (E, E') under $PK_{\mathcal{T}}$. It outputs a '1' if E and E' have identical corresponding plaintexts, and outputs '0' otherwise.
- The oracle $\tilde{DEC}$ takes as input a ciphertext E under $PK_{\mathcal{T}}$. It outputs the corresponding plaintext.
- The oracle $\tilde{OW}$ takes as input a query value in $\{0,1\}^*$, and outputs a random value $\{0,1\}^{k_4}$, where k_4 is a security parameter (that may depend on k_1, k_2 and k_3). The output of $\tilde{OW}$ is consistent, in the sense that a given input value always yields the same output value. This oracle may be viewed as the ideal embodiment of a cryptographic hash function.

Each of these oracles accepts publicly viewable input from all participating authorities (talliers). Each tallier may be thought of as having a publicly readable tape to which it may write input values for a given oracle; each tape contains a write portion for each time-step of the protocol, which we assume to be synchronous. At the end of a given timestep, an oracle produces output according to the following procedure. If a majority of talliers have furnished identical non-null values Z on their tapes, then the oracle processes input Z and yields the corresponding output. If there is no non-null majority input, then the oracle simply outputs the special symbol $\perp$. The requirement for majority input ensures that the protocol execution is determined by honest players, i.e., effectively reduces $\mathcal{A}$ to an honest-but-curious adversary once the ballot-posting phase for the election is complete.

We additionally assume for simplicity that key setup and registration are performed by a trusted entity. Our proofs may be extended to accommodate more general assumptions in which these two processes are performed in a distributed manner.

D.2 Proof overview

Recall that our definition of coercion-freeness revolves around a game played between an adversary $\mathcal{A}$ and a voter targeted for coercion. The aim of $\mathcal{A}$ is to guess which of the following two behaviors the voter has adopted during the execution of an election system ES: (1) The voter has divulged valid voting credentials and abstained from voting or (2) The voter has divulged fake credentials and cast a ballot. In order to demonstrate that ES possesses coercion-freeness, we must show that $\mathcal{A}$ can guess successfully with probability only negligibly better than a weaker poly-time adversary $\mathcal{A}'$ interacting with an ideal election system. This adversary $\mathcal{A}'$ is passive, and its only input is the final tally $\mathcal{X}$ of votes cast by honest voters in the completed election plus Γ, the number of ballots eliminated for invalid associated credentials.

Our proof strategy is to construct a polynomial-time algorithm $\mathcal{S}$ that takes a set of ballots W of honest voters and simulates the election system ES in the experiment *c-resist*. If the simulation is indistinguishable to $\mathcal{A}$ from use of the true functional components of ES, and $\mathcal{A}$ cannot cause the simulation to deviate from correct execution, then we see that $\mathcal{A}$ learns nothing more than the correct election tally $\boldsymbol{X}$ and the number of bad ballots Γ. This means in turn that $\mathcal{A}$ is no more powerful than the weak adversary $\mathcal{A}'$ characterized in our experiment *c-resist-weak*. Thus ES is coercion-free.

The inability of the adversary to cause deviation in the experiment from correct execution hinges on our oracle definitions, which require majority agreement on input values. Given this, we show that the simulation produced by $\mathcal{S}$ is indistinguishable by $\mathcal{A}$ from a real experimental execution of *c-resist* under the DDH assumption on $\mathcal{G}$. Our proof relies on the semantic security of El Gamal [47]. In particular, we make use of the following, useful fact implied by the DDH assumption: A poly-time adversary that selects a plaintext m cannot distinguish between the distribution of El Gamal ciphertexts on m and the distribution of random pairs (α, β) for $\alpha, \beta \in_U \mathcal{G}$ with non-negligible probability (in the security parameters for G). In consequence of this observation, it is possible for $\mathcal{S}$ to simulate the election process by substituting *random ciphertexts*, i.e., random pairs of group elements, for the real ciphertexts that would be processed in a true execution of the experiment *c-resist*. In particular, $\mathcal{S}$ can simulate the ballots of voters not controled by $\mathcal{A}$ with a list of random ciphertexts. Additionally, $\mathcal{S}$ can simulate the oracle $\tilde{MN}$ by setting its simulated output to a list of random ciphertexts. Under the DDH assumption, $\mathcal{A}$ cannot distinguish between the random ciphertexts furnished by $\mathcal{S}$ and the ciphertexts that would be processed in a true execution of ES.

The standard proof for showing the indistinguishability between an El Gamal ciphertext on plaintext m and a random ciphertext runs roughly like this. Let $K = (y_1, g_1, y_2, g_1)$ be a DDH input tuple. With probability $1/2$, $K \in D$; otherwise, $K \in D'$. A simulator publishes g_1 as a generator for the group $\mathcal{G}$, and y_1 as an El Gamal public key. On input consisting of plaintext $m \in \mathcal{G}$, the simulator outputs a putative El Gamal plaintext $(\alpha, \beta) = (my_2, g_2)$. Observe that if $K \in D$, then (α, β) is a valid ciphertext on m; otherwise, (α, β) is a random ciphertext. Suppose an adversary that can distinguish between a ciphertext on m and a random ciphertext with non-negligible probability. By furnishing this adversary with input (α, β), it is possible to distinguish between $K \in D$ and $K \in D'$, and therefore to break the DDH assumption. See, e.g., [9] for a clear and concise exposition of this proof technique. We apply the same technique in our proof to show that $\mathcal{A}$ cannot distinguish between the simulation of $\mathcal{S}$ and a true execution of ES. In particular, we rely on a diagonalization argument over all simulated, i.e., random ciphertexts in the transcript produced by $\mathcal{S}$.

The tricky part of our proof comes in the simulation of the oracles $\tilde{PET}$ and $\tilde{DEC}$. In order to obtain a reduction to the DDH assumption, we require that $\mathcal{S}$ publish $g = g_1$ as a generator for $\mathcal{G}$ and $y = PK_{\mathcal{T}} = y_1$ as the public key for the tallying authorities. Since g_1 and y_1 are randomly distributed, this means that $\mathcal{S}$ does not know the corresponding private El Gamal decryption key. This is problematic because it means that $\mathcal{S}$ cannot decrypt any of the ciphertexts produced by $\mathcal{A}$. In order to perform a successful simulation, however, it is essential that $\mathcal{S}$ be able to learn the plaintexts in the ballots posted by $\mathcal{A}$. Otherwise, $\mathcal{S}$ cannot produce a correct election tally, and the simulation may be detectable by $\mathcal{A}$.

In order to perform a successful simulation, therefore, we require that $\mathcal{S}$ be able to extract the plaintext ballots posted by $\mathcal{A}$. We might hope to accomplish this by exploiting the random oracle assumption on the one-way function used for the NIZK proofs in ballot construction, i.e., by making use of the oracle $\tilde{OW}$. In particular, by using the "Forking Lemma" [41] on this oracle and rewinding, it is possible in principle to extract the encryption factors used by $\mathcal{A}$, and therefore to extract the plaintexts for the ballots posted by $\mathcal{A}$. The problem is that $\mathcal{A}$ can interleave calls to $\tilde{OW}$ and ballot postings arbitrarily. Hence the number of rewindings required by the simulator to perform a successful extraction of all plaintexts could potentially be exponential in the number of voters n_V in the system.

Instead, we adopt a different approach to permit $\mathcal{S}$ to extract the plaintexts of ballots posted by $\mathcal{A}$. Since we assume that the registrar $\mathcal{R}$ is a trusted entity, we may have $\mathcal{S}$ simulate $\mathcal{R}$ and therefore learn all credentials $\{\sigma_i\}_{i=1}^{n_V}$. In particular, we let $\mathcal{S}$ generate a credential as $\sigma_i = g^{s_i}$ for $s_i \in_U Z_q$; thus s_i specifies the credential σ_i. We alter the ballot construction in our protocol in order to permit

these credentials to serve effectively as trapdoors for plaintext extraction. Recall that in our protocol as described above, a ballot comprises the ciphertext pair:

$$E_1^{(i)} = (\alpha_1, \beta_1) = (c_j y^{a_1}, g^{a_1}), E_2^{(i)} = (\alpha_2, \beta_2) = (\sigma_i y^{a_2}, g^{a_2}).$$

To add the trapdoor, we append a third ciphertext:

$$E_3^{(i)} = (\alpha_3, \beta_3) = (c_j \sigma_i^{a_3}, g^{a_3}).$$

The value s_i effectively serves as a private key for this ciphertext. It is easy to see that knowledge of s_i permits decryption of the ballot plaintext c_j. In order to prove correct ballot construction, we require the following NIZK proof in ES (using the notation introduced in [15]). As a technical requirement, we let any candidate identifier $c_i = g_1^{r_i}$ for some published value r_i.

$$PK\{a_1, a_2, a_3, r, s : (\alpha_1, \beta_1) = (g^r y^{a_1}, g^{a_1}) \wedge (\alpha_2, \beta_2) = (g^s y^{a_2}, g^{a_2}) \wedge (\alpha_3, \beta_3) = (g^r (g^s)^{a_3}, g^{a_3})\}.$$

In contrast to the NIZK proofs specified in our main protocol, the NIZK proof specified here requires "secret multiplication". It depends, in particular, on the product sa_3 being correctly formed according to the expression above. This portion of the proof may be accomplished efficiently using techniques described in [14, 33].

Remark: We omit the ciphertext (α_3, β_3) from our protocol in the body of the paper as it does not contribute conceptually to our construction. We believe, in fact, that it is really not required to achieve coercion-freeness, but merely imparts the property of provability to our election system ES. One might object to the fact that use of credentials $\{\sigma_i\}$ as trapdoors weakens the security of our election system, as it potentially permits $\mathcal{R}$ to learn the values of ballots cast by voters. We argue on the contrary that this aspect of our construction does not imply any weakening of our security model. First, we recall that $\mathcal{R}$ must be the most highly trustworthy entity in the election process to begin with, or else coercion-freeness is not possible. And of course, $\mathcal{R}$ may be distributed if desired. Second, it is possible for $\mathcal{R}$ to erase plaintext credentials once it has issued them. This is not possible for $\mathcal{T}$, which itself has the ability, if compromised, to learn the ballot values cast by voters. Thus, $\mathcal{T}$ constitutes the real weak link in our construction, rather than $\mathcal{R}$.

D.3 The simulation

We now outline the steps of the simulation of *c-resist* executed by $\mathcal{S}$. Throughout the simulation, according to the usual technique in the literature, $\mathcal{S}$ maintains state for the simulated oracle $O\tilde{W}$ so as to ensure consistency of output values. Let $W \in D_{n_U, n_C}$ represent a set of ballots input into the simulation as representing the posting of honest voters.

1. **Setup:** $\mathcal{S}$ publishes the generator g and public key y, and also a randomized candidate slate $\boldsymbol{C} = \{c_i\}_{i=1}^{n_C}$ such that $c_i = g^{r_i}$ for $r_i \in_U Z_q$. (For technical reasons in our proof, we require that candidate identifiers here be random, rather than comprising the set $\{1, 2, \ldots, n_C\}$.)
2. **Registration:** $\mathcal{S}$ simulates the registrar $\mathcal{R}$, generating a set of credentials $\{\sigma_i = g^{s_i}\}$ for $s_i \in_U Z_q$. For the encrypted credential list $\boldsymbol{L}_0$, the simulator $\mathcal{S}$ publishes a list of n_V random ciphertexts.
3. **Adversarial corruption:** The adversary $\mathcal{A}$ selects a set V of n_A voters to corrupt, as well as a voter j for coercion and a target vote β. If any of these selections are invalid, i.e., if $V \neq n_A$ or $j \notin \mathcal{V} - V$ or $\beta \notin \boldsymbol{C} \bigcup \phi$, then the simulation is terminated.
4. **Coin flip:** A coin $b \in_U \{0, 1\}$ is flipped.
5. **Credential release:** $\mathcal{S}$ gives $\mathcal{A}$ the set of credentials $\{\sigma_i\}_{i \in V}$ as well as a credential σ for the targeted voter j. If $b = 1$, then $\mathcal{S}$ gives $\sigma = \sigma_j$; otherwise σ is a random string.

6. **Honest voter simulation:** For each ballot element in W, the simulator posts a ballot consisting of three random ciphertexts (α_1, β_1), (α_2, β_2), and (α_3, β_3). $\mathcal{S}$ also furnishes an associated NIZK proof of the form specified above. Since the associated challenge value comes from $O\tilde{W}$, and may therefore be predetermined by $\mathcal{S}$, the NIZK proof may be simulated using standard techniques. Let $\boldsymbol{A}_0$ be the list of these ballots. Let $\boldsymbol{A}^*$ be the associated set of plaintext ballot choices in W for which the associated credential is correct, i.e., excluding λ elements.
7. **Adversarial ballot posting:** The adversary $\mathcal{A}$ posts a set of ballots $\boldsymbol{B}_0$ and associated NIZK proofs.
8. **Plaintext extraction:** $\mathcal{S}$ checks the NIZK proofs in $\boldsymbol{B}_0$. Let $\boldsymbol{B}_1$ be the list of ballots with correct proofs. For each ballot in $\boldsymbol{B}_1$ and each credential in $\{\sigma_i\}_{i \in V} \bigcup \sigma_j$, the simulator attempts to decrypt the third ciphertext (i.e., using our notation above, the ciphertext (α_3, β_3). A decryption is regarded as successful if it yields a plaintext vote in $\boldsymbol{C}$. Let $\boldsymbol{B}_2$ be the list of ballots for which decryption is successful, i.e., the list of ballots that appear to be based on valid credentials. As a final step, $\mathcal{S}$ eliminates ballots with duplicate credentials according to the election policy. Let $\boldsymbol{B}_3$ be the list of ballots that results from this elimination process. Let $\boldsymbol{B}^*$ be the corresponding set of plaintext ballot choices.
9. **Tallying simulation:** $\mathcal{S}$ simulates the behavior of honest tallying authorities. Since these are a majority, any deviating behavior by tallying authorities in the control of $\mathcal{A}$ may be ignored. This part of the simulation proceeds as follows:
   (a) **Proof checking:** Let $\boldsymbol{E}_0$ denote the combined list of input ballots $\boldsymbol{A}_0$ and $\boldsymbol{B}_0$. $\mathcal{S}$ simulates the behavior of honest tallying authorities in rejecting all ballots with invalid associated NIZK proofs. Let $\boldsymbol{E}_1$ be the resulting ballot list.
   (b) **Eliminating duplicates:** Since no mixing has yet occurred, $\mathcal{S}$ may simulate the elimination of duplicate ballots via PET straightforwardly. A comparison via PET is simulated for every ballot pair. When the comparison is applied to a pair of ballots determined during the plaintext-extraction process to be based on the same credential, the simulated output of $P\tilde{E}T$ is '1'; otherwise, it is '0'. In the former case, one of the two ballots is eliminated in accordance with election policy. Let $\boldsymbol{E}_2$ be the resulting ballot list.
   (c) **Mixing:** $\mathcal{S}$ simulates the oracle $\tilde{MN}$ as applied to $\boldsymbol{E}_2$ by outputting an equal-length list $\boldsymbol{E}_3$ of random ciphertext triples. Likewise, $\mathcal{S}$ simulates the mixing of $\boldsymbol{L}_0$ by outputting an equal-lengthed list $\boldsymbol{L}_1$ of random ciphertexts.
   (d) **Checking credentials:** $\mathcal{S}$ simulates the process of credential checking. In a true protocol execution, this would involve sequential comparison using $P\tilde{E}T$ between each ballot in $\boldsymbol{E}_3$ (more precisely, the credential ciphertext therein) and the ciphertexts in $\boldsymbol{L}_1$. Either a match is found, in which case a ballot is deemed to be based on a valid credential, or else the list $\boldsymbol{L}_1$ is exhausted, and the ballot is rejected.
   $\mathcal{S}$ simulates the output of $P\tilde{E}T$ for this phase of the protocol as follows. $\mathcal{S}$ selects a random permutation π on n_V elements, and a random permutation ρ on $|\boldsymbol{E}_3|$ elements. For a comparison between the i^{th} element of $\boldsymbol{E}_3$ and the j^{th} element of $\boldsymbol{L}_1$, $\mathcal{S}$ simulates the output of $P\tilde{E}T$ as follows. If the $\rho(i)^{th}$ element of $\boldsymbol{E}_2$ has a valid credential corresponding to $\sigma_{\pi(j)}$, then $\mathcal{S}$ simulates an output from $P\tilde{E}T$ of '1'. Otherwise, $\mathcal{S}$ simulates an output value of '0'. Based on this simulation, presumed duplicate ballots are eliminated according to the election policy. Let $\boldsymbol{E}_4$ be the resulting ballot list.
   (e) **Decryption:** $\mathcal{S}$ simulates the decryption oracle $D\tilde{E}C$ as follows. Let $\boldsymbol{C}^* = \boldsymbol{A}^* \bigcup \boldsymbol{B}^*$. Note that $z = |\boldsymbol{C}^*| = |\boldsymbol{E}_4|$ (if the simulation has proceeded correctly). Let ζ be a random permutation on z elements. When decryption is to be performed for the i^{th} element of $\boldsymbol{E}_4$, $\mathcal{S}$ simulates the output of $D\tilde{E}C$ as the $\zeta(i)^{th}$ element of $\boldsymbol{C}^*$.

E Some details on primitives

El Gamal: As explained in the body of the paper, El Gamal [23] 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 $\mathcal{G}$ [8, 47]. 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 [46] 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 [24]. 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 $(\alpha, \beta) = (my^r, g^r)$ for $r \in_U Z_q$. For succinctness of notation in the body of the paper, we sometimes let $E_y[m]$ denote a ciphertext on message m under public key y. 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 $r' \in_U Z_q$.

To decrypt a ciphertext (α, β), the plaintext $m = \alpha/\beta^x$ is computed. 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 [43] with conjunction achieved using techniques described in, e.g., [19]. We omit many details in this description regarding the scheduling of these operations and the use of commitments to avoid adversarial bias. (The reader is referred to, e.g., [16, 24] for some discussion of these issues in relation to key generation.)

We note that another possible choice of cryptosystem for our voting scheme is that of Paillier [39].

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 [22] and Neff [35]. Both of these employ El Gamal as the underlying cryptosystem, i.e., an input ciphertext $E_i = (\alpha, \beta) = (my^k, g^k)$ 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 E_i. This serves the purpose of rendering the cryptosystem chosen-ciphertext-attack secure while still permitting re-encryption.