Coercion Resistant End-to-end Voting*

Ryan W. Gardner, Sujata Garera, and Aviel D. Rubin

Johns Hopkins University, Baltimore MD 21218, USA

Abstract. End-to-end voting schemes have shown considerable promise for allowing voters to verify that tallies are accurate. At the same time, the threat of coercion has generally been considered only when voting devices are honest, and in many schemes, voters can be forced or incentivized to cast votes of an adversary's choice. In this paper, we examine the issue of voter coercion and identify one example method for coercing voters in a scheme by Benaloh. To address such attacks, we present a formal definition of coercion resistance for end-to-end voting. We then present a new scheme, extended from Benaloh's, that is provably coercion resistant. In addition to providing accuracy and coercion resistance, our scheme emphasizes ease-of-use for the voter.

Key words: end-to-end voting, coercion, privacy, cryptography

1 Introduction

Many parts of the world have been witnessing a rapid adoption of electronic voting systems to address the usability issues of the paper ballot and tally votes more conveniently. While these systems have offered many benefits, they have also generated a large number of new security concerns. Several studies have independently analyzed electronic voting systems and shown that they are vulnerable to a multitude of threats [1–8]. As these machines are added to an already unverifiable voting process, voters today are left with very few assurances of the integrity of their recorded votes or their tabulation.

To address these concerns, several researchers have explored the notion of end-to-end voting schemes [9–14]. These schemes are designed to allow each voter to publicly verify both that her vote is accurately recorded and that all recorded votes are correctly tallied in the final sums [15–17]. Specifically, end-to-end voting schemes aim to provide these properties without trusting the software that runs on the voting machines.

Although end-to-end voting schemes have potential to greatly increase the transparency and integrity of elections, strong privacy guarantees are generally secondary foci and have often been missed. For example, Moran and Naor [18] identify a vote buying attack in the Punchscan system as it is demonstrated [19]. Karlof *et al.* [17] identify possible means for revealing voter and vote information in schemes by Neff [9] and Chaum [10]. Other examples exist [20–22]. However, privacy is critical and not independent of integrity. Websites dedicated to the

* This work was supported by the National Science Foundation grant CNS-0524252.

sale of votes have been found on the Internet [23], and people have been caught selling votes on eBay [24, 25]. Selling is simply a voluntary form of coercion.

In this paper, we explore the issue of coercion in end-to-end voting protocols. We examine a scheme by Benaloh [12, 13] and describe a possible method for coercing voters in that scheme. We then present the first formal definition of coercion resistance we are aware of that encompasses a voter's actions at the polls and the final output from casting a ballot in the end-to-end voting setting. We construct a new end-to-end voting scheme that is provably coercion resistant. The scheme provides verifiability to the voter at a low cost. In order to achieve these guarantees, we assume at least one of each election's candidate parties is honest and rely on a private channel [21] between the voting device and the parties, which could be instantiated by an inexpensive smart card, for example.

In addition to security, our scheme's primary goal is to maintain the simplicity of the voting process for the voter. By building on the technique by Benaloh [12, 13], our scheme requires a typical voter only to answer a single, simple question in addition to making her ballot selections. The assurances provided by the scheme are then probabilistic. We begin with related work, including a description of Benaloh's scheme, continue with our new definition and construction, and end with some brief practical considerations.

2 Related Work

Significant research has been conducted in cryptographic voting over the past 2 decades. The first voting schemes are pioneered independently by Yao [26], Benaloh [27], and Chaum [28] although they allow a voter to prove how she voted to a coercer. Benaloh and Tuinstra [21] introduce the notion of receipt freeness and describe a secret ballot election protocol that relies on private channels between some parties and utilizes a threshold scheme. Their scheme is later shown not to be receipt free by Hirt and Sako [22]. Sako *et al.* propose a receipt free system that uses mix networks [15]. In their scheme, however, a coercer can force a voter to vote randomly. A scheme by Magkos *et al.* relies on a tamper resistant smart card which collaborates with the voter to produce a valid encryption of her vote [29] but requires significant voter participation.

Chaum [10] and Neff [9] independently create the first electronic voting schemes that do not require the voter to possess any computational device to verify her vote. Chaum's scheme [10] relies on visual cryptography and provides the voter with a receipt that is a visual share of her cast ballot. Neff, on the other hand, introduces a scheme that encodes the voter's choice in an encrypted array of bits [9]. It further commits to the voter's choice by displaying a short string on the voting screen. Verification depends on the voter's ability to compare short strings. Karlof *et al.* analyze Chaum's and Neff's schemes [17] and discover a number of potential problems in them from a system's perspective, including possible methods for adversaries to leak information about voters' choices.

Riva *et al.* recently propose a voting scheme which requires the voter to prepare her ballot prior to arriving at the polls [11]. Unlike Neff's and Chaum's

schemes, this scheme attempts to maintain vote secrecy even with respect to the voting booth.[1] Benaloh introduces the notion of auditing a completely marked ballot [12, 13, 30]. We discuss his scheme further in Section 4.

Prêt à Voter, introduced by Ryan *et al.*, is a paper based scheme [31, 32]. It is similar in concept to Chaum's scheme but does not rely on visual cryptography. Ryan *et al.* extend their scheme to use Pallier encryption [33] although Xia *et al.* examine the scheme and show that it can leak information about voters' intentions under several circumstances [34]. Rivest and Smith present 3 paper based voting protocols [14] that do not use any cryptography although the 2 schemes that allow the voter to verify her *own* vote require her to fill 3 separate ballots. Scantegrity II [35], introduced by Chaum *et al.*, extends the original Scantegrity proposal [36] to avoid the need to locate physical ballots to resolve voter disputes. This paper based system relies on the use of invisible ink to allow each voter to verify the inclusion of her vote by looking up a confirmation code.

3 Notation

We consider a voting protocol to consist of interactions between several entities: a voter $\mathcal{V}$, a ballot marking machine $\mathcal{M}$, a public bulletin board $\mathcal{B}$,[2] and a receipt $\mathcal{R}$ (which is also often the physical ballot). We write $\mathcal{A} \xmapsto{x} \mathcal{B}$ to describe entity $\mathcal{A}$ sending datum x to entity $\mathcal{B}$. The notation $x \in_R A$ is used to denote a variable x drawn uniformly at random from set A. Further, we write $\mathsf{poly}(k)$ to denote any polynomial function of k and $\mathsf{negl}(k)$ to denote any function that is negligible in k.[3] Let v represent a voter's candidate vote or ballot.

In addition to the above notation, we assume the existence of 3 functions:

- $\mathsf{KeyGenEnc}(1^k, N_T)$: a key generation function that takes as input security parameter k and generates a public encryption key e and N_T shares of a corresponding distributed private key $\{d_1, \ldots, d_{N_T}\}$.
- $\mathsf{Enc}_{e,r}(p)$: an IND-CCA2 [37] secure encryption function that accepts a public key e, a random value r, and plaintext p. It gives ciphertext c as output.
- $\mathsf{EncVerify}(p, c, e, r)$: a ballot verification function that checks that encrypted ballot c is a valid encryption of plaintext vote p using public key e and randomness r. It returns $\mathsf{SUCCESS}$ if $c = \mathsf{Enc}_{e,r}(p)$ and $\mathsf{FAILURE}$ otherwise.

4 Benaloh's Scheme

We examine the recent cryptographic voting scheme of Benaloh [12, 13] as an example for this paper. We focus on the scheme because it has a minimal impact on the traditional voting process while providing voters with guarantees of the tally's accuracy. At the same time, we find that, like several other schemes, its implicit use of probabilistic cryptographic operations enables possible coercion.

[1] An attack similar to one we describe on Benaloh's scheme is actually possible here since the scheme implicitly trusts the machine to produce truly random values.

[2] Generally this is thought of as a web-page on the Internet.

[3] A function $\mathsf{negl}(k)$ is negligible in k if $\forall d \geq 1 \ \exists \ell > 0 \ \forall k > \ell \ \mathsf{negl}(k) < \frac{1}{k^d}$.

4.1 Security Model

Benaloh states that we can never *guarantee* voter privacy, citing an example that one can never prove that hidden cameras are not installed at the voting booth [13]. He is correct in the absolute sense and considers strict privacy only in the setting where parties and machines are honest. Despite the impossibility of *unconditional* certainties in practice, however, it remains important to consider privacy in the face of dishonest machines. We can still reduce the types of attacks adversaries can perform and minimize threats of large scale coercion by applying a stronger theoretical model. Ideally, end-to-end voting schemes achieve the following properties:

Individual Verifiability- The voter should be able to verify that her intentions were accurately recorded in her cast ballot.

Universal Verifiability- Voters should be able to verify that all cast ballots were properly included in the final tallies and came from legitimate voters.

Mandatory Privacy- No one should be able to learn how another voter voted with certainty even if the voter would like that person to know.

4.2 Overview of the Scheme

We now present an overview of the steps involved in an election under the Benaloh scheme [12, 13]. Since the original description of the scheme is informal, we necessarily make some assumptions about the details, particularly with respect to the encryption process. However, we believe our description is accurate with respect to the intentions of the scheme. We focus on the casting process and summarize the tallying and verification somewhat more informally since they are straightforward and less critically relevant to this study.

Initialization

Before the start of election day, a group of N_T trusties runs $\mathsf{KeyGenEnc}(1^k, N_T)$ with a k of their choice, distributes private key shares $d_1, \ldots, d_{N_T}$, and writes the public encryption key e to the ballot marking machine $\mathcal{M}$.

Ballot Marking

1. $\mathcal{V} \overset{v}{\longmapsto} \mathcal{M}$: The voter enters her candidate selections v into the ballot marking machine.
2. $\mathcal{M} \overset{c=\mathsf{Enc}_{e,r}(v)}{\longmapsto} \mathcal{R}$: The ballot marking machine generates a random r and prints a corresponding encryption of the voter's alleged ballot to a receipt.
3. $\mathcal{M} \overset{\text{“Cast vote?”}}{\longmapsto} \mathcal{V}$: The machine asks the voter if she wants to cast her vote.

Option 1: Auditing

1. $\mathcal{V} \overset{\text{“No”}}{\longmapsto} \mathcal{M}$: The voter optionally indicates that she would not like to cast this ballot. (Rather, she is choosing to audit the machine with it.)

2. $\mathcal{M} \overset{r,v}{\longmapsto} \mathcal{R}$: The ballot marking machine reveals the randomness r used for encryption and adds it to the receipt. It also prints the plaintext ballot. This marks the ballot invalid for casting. The voter can verify the plaintext and take the receipt home to test that $\mathsf{EncVerify}(v, c, e, r)$ returns $\mathsf{SUCCESS}$.

The voter may choose to repeat the ballot marking and optional auditing steps on new ballots an unbounded number of times (even after she has cast her vote) to increase her certainty that the machine is behaving honestly. (See Section 7 for a brief discussion on the effectiveness of such auditing.)

Option 2: Casting

1. $\mathcal{V} \overset{\text{“Yes”}}{\longmapsto} \mathcal{M}$: The voter indicates that she would like to cast this ballot.
2. $\mathcal{R} \overset{c}{\longmapsto} \mathcal{B}$: The voter takes her valid receipt to a ballot casting station where it is used to cast her encrypted ballot, and it is posted to the public bulletin board. (She also presents any necessary identification.)

Tallying and Verification

When the voting period has ended, a group of trustees anonymizes the posted ballots through a mix-net [38–40]. Each trustee re-encrypts all the ballots and posts them back to the public bulletin board in random order along with a zero-knowledge proof [41] of correctness. Finally, a sufficiently large subset of the trustees uses a threshold scheme [42] to jointly decrypt the ballots, Again, a proof is provided with each decryption to allow public verification. Each voter can use her receipt $\mathcal{R}$ to verify that her vote has been cast and counted correctly.

4.3 Compromising Voter Privacy

The voting and verification processes of Benaloh's scheme are simple and, when described informally, seem to accomplish the goals of end-to-end voting clearly. However, the scheme's need to make random choices can be exploited to compromise voter privacy. One possible attack consists of an adversary replacing the code for obtaining the randomness used in the scheme's encryptions with a pseudorandom number generator, for which she exclusively knows the key. Such an attack effectively gives the adversary knowledge of the randomness r used in each encryption $\mathsf{Enc}_{e,r}(v)$ and allows her to determine the plaintext of each posted encrypted ballot since the message space is likely to be very small. As a result, such an adversary can also coerce voters into casting particular votes.

Another attack involves the adversary compromising the machine to actually encode information into the ciphertexts c themselves by trying new encryptions until a desired ciphertext is obtained. For example, in one naive approach, the parity of c could indicate a vote for republican or democrat. Similar observations were made by Karlof *et al.* [17] with respect to possible subliminal channels enabled by the randomness used in the cryptographic voting scheme of Andrew Neff [9]. Obviously, more sophisticated approaches could encode many more bits and information about the ballot in a more covert manner. Similar attacks are also possible against other proposed schemes [11, 20].

5 Coercion Resistance

With nuanced attacks that compromise voter privacy such as those against Benaloh's scheme, the issue of coercion resistance needs to be treated rigorously. Several definitions for coercion resistance have been proposed in the literature. Juels *et al.* offer a definition centered around voters' potential use of fake keys to avoid coercion and is more specifically tailored for coercion resistance in a remote voting setting where machines are assumed uncompromised [43]. It also does not allow the adversary to adaptively interact with the voting system. Teague *et al.* offer a nice definition that considers the information content of the plaintext votes (but no other output from the protocol) [44]. For example, they consider attacks where an adversary requests that a voter fill out a specific permutation of votes on a portion of the ballot to identify it as belonging to that voter. Benaloh and Tuinstra introduce the notion of "receipt freeness" for end-to-end voting protocols although they do not give a formal definition [21]. Moran and Naor subsequently define receipt-freeness based on an ideal functionality of a voting protocol [20], extending from the work of Canetti and Gennaro [45]. However, their definition focuses on the adversary's view of a voter's interactions with a machine and allows privacy leaks in the final output of the protocol, such as the ones we describe.

We introduce a new definition of a coercion resistant vote casting protocol. Intuitively, it requires that an adversary who can adaptively interact with the protocol cannot distinguish between a vote cast using inputs of her choice and a vote cast using inputs of the voter's choice, including any possible vote. Alternatively, if the protocol is not functioning honestly, it can be detected by the voter (with probability varying by scheme).

Our definition is more direct than previous definitions, and by separating the vote casting from the entire voting protocol, we are able to address coercion enabled by examination of the protocol's final output. Note that our definition does not account for privacy leaks in the plaintext ballots themselves, such as information that would allow a coercer to identify ballots like specific permutations of votes or write-in candidate strings. This problem is independent of the one we examine, and we believe it is addressable by combining other approaches using disjoint definitions such as the one by Teague *et al.* [44].[4]

We consider a vote casting protocol to consist of a series of interactions with a vote caster $\mathcal{C}^*$[5] that takes a set of ordered inputs X, minimally including some ballot choice or vote v. The caster's output is an ordered set Ψ including some encoding of the voter's ballot c. We also introduce what we refer to as a unique seed $s \in \mathbb{S}$. The seed s is part of the vote casting input X. We refer to the set of all output that *could* be made available to an adversary from an interaction with $\mathcal{C}^*$ (if, for example, a voter were forced to reveal it) as $\Psi_{\mathcal{A}}$.[6] Let $\mathcal{P}$ represent a set of all public information.

[4] One trivial solution is to simply use separate logical ballots for each race.

[5] In practice, this might be one or several devices, poll workers, etc.

[6] This would minimally include data such as that available on voter receipts and posted to public bulletin boards.

For our end-to-end voting scenario, we let $\mathcal{C}^*$ also produce a proof of correctness π. To verify the correctness of a cast vote and to evade coercion, we refer to two functions respectively:

- $\mathsf{BallotVerify}(X, \Psi, \pi, \mathcal{P})$: takes the vote casting input X, the caster's output Ψ, the proof of correctness π, and public information $\mathcal{P}$. It returns SUCCESS if Ψ is a valid output for input X with proof π and FAILURE otherwise.
- $\mathsf{GenerateInput}(X_{\mathcal{A}}, s, v)$: outputs a coercion resistant vote casting input $X_{\mathcal{V}}$ with vote v and seed s when an adversary demands that the voter use input $X_{\mathcal{A}}$ instead.

We write our definition in terms of a game between several algorithms (PTMs): an adversary $\mathcal{A}$, a challenger $\mathcal{G}$, a vote caster $\mathcal{C}^*$, and a verifier $\mathcal{Z}$. The adversary's goal is to distinguish between the visible output of the vote casting protocol for 2 distinct votes. Formally:

Security Game: Indistinguishability of Encoded Votes (IEV)

1. An initialization phase establishes public data $\mathcal{P}$.
2. $\mathcal{A}$ adaptively sends inputs X to $\mathcal{C}^*$ and obtains corresponding outputs Ψ and π. $\mathcal{Z}$ is also given each (X, Ψ, π) tuple.
3. $\mathcal{A}$ selects an input X_0 including vote v_0, with the constraint that X_0's seed s' has not been the seed in any of $\mathcal{A}$'s previous queries to $\mathcal{C}^*$. $\mathcal{A}$ also selects a second vote v_1[7] to be part of an input $X_1 = \mathsf{GenerateInput}(X_0, s', v_1)$. $\mathcal{A}$ sends X_0 and X_1 to $\mathcal{G}$.
4. $\mathcal{G}$ chooses a random bit b and sends X_b to $\mathcal{C}^*$. $\mathcal{C}^*$ gives $\mathcal{A}$ the visible output $\Psi_{b,\mathcal{A}}$ corresponding to the input X_b. It also gives the corresponding tuple (X_b, Ψ_b, π_b) to $\mathcal{Z}$.
5. $\mathcal{A}$ again adaptively sends inputs X to $\mathcal{C}^*$ under the constraint that s' is not the seed of any X and obtains corresponding outputs Ψ and π. $\mathcal{Z}$ receives each (X, Ψ, π).
6. $\mathcal{Z}$ runs $\mathsf{BallotVerify}(X, \Psi, \pi, \mathcal{P})$ with public $\mathcal{P}$ for each tuple (X, Ψ, π) it was given. If BallotVerify returns SUCCESS for each tuple, $\mathcal{Z}$ outputs $z =$ SUCCESS. Otherwise, it outputs $z =$ FAILURE.
7. $\mathcal{A}$ outputs b', its best guess of the value b.

The adversary's advantage in the game $\mathsf{adv}_{IEV}(\mathcal{A})$ is defined as:

$$\mathsf{adv}_{IEV}(\mathcal{A}) = Pr\left((b' = b) \cap (z = \mathsf{SUCCESS})\right) - \frac{1}{2}.$$

Definition 1. *A vote casting protocol with security parameter k is coercion resistant if for all probabilistic $\mathsf{poly}(k)$ time algorithms $\mathcal{A}$ and all probabilistic vote casters $\mathcal{C}^*$, $\mathsf{adv}_{IEV}(\mathcal{A}) < \mathsf{negl}(k)$.*

Notice that in this definition, the coercion resistance of a protocol depends entirely on the BallotVerify function and a correct GenerateInput. Although initially it may seem counterintuitive to discard the entire vote encoding process,

[7] This is analogous to the voter's "desired vote".

the threat model our definition aims to address is one where the caster $\mathcal{C}^*$ (such as a voting machine) may be completely corrupted by an adversary. In other words, the BallotVerify function of the scheme must lock $\mathcal{C}^*$ into a scheme where an adversary cannot distinguish the output from different votes. On the other hand, also note that our definition models the caster as something without any post-election communication with the adversary. Unfortunately, this is largely unavoidable since, for the most part, each voter must[8] divulge her vote to the caster. We may be able to approximate this model in practice by building voting machines with an exact, minimum amount of writable memory although this possibility requires more rigorous exploration. Nevertheless, our primary objective is to remove information that could be used to compromise a voter's privacy from the *public* domain. A significantly more powerful adversary is required to launch large scale coercion attacks by communicating with voting machines after they are used than an adversary who can determine votes by examining public information.

6 A Coercion Resistant End-to-end Voting Scheme

We now present a construction for a coercion resistant end-to-end voting scheme. It is an extension of Benaloh's [12, 13] (Section 4) and roots the source of all entropy required of the scheme in a small number of keys distributed among parties with conflicting interests, which we assume to be all the candidate parties.[9] Voters can then verify uniquely correct outputs, and the scheme is secure as long as at least one candidate party behaves honestly. To utilize the key provided by each party, we rely on the existence of a private channel [21] between the voting machine and each party. In practice, this could be instantiated by inexpensive trusted hardware such as smart cards. Again, in addition to providing coercion resistance, our primary aim is to keep the voting process as simple as possible for the voter.

We begin with our assumptions and continue to a description of the vote casting protocol and a security proof.

6.1 Preliminaries

In addition to the variables and entities listed in Section 3, we also refer to N_C, the number of candidate parties, and $\mathcal{T}_i$ for $i = 1, \ldots, N_C$, an entity with a private channel to the voting machine and whose key, K_i, is written by candidate party i. In practice, each entity $\mathcal{T}_i$ could be instantiated by an inexpensive smart

[8] A scheme proposed by Riva and Ta-Shma [11] is one exception to this although it is arguably quite impractical in terms of complexity for the voter. The scheme is also susceptible to an attack very similar to the one we describe against Benaloh's.

[9] The entropy could be distributed among any variety of parties, but for the sake of concreteness, we assume it is distributed among each candidate's party in our descriptions.

card inserted into the ballot marking machine.[10] We write $\mathbb{S}$ to refer to the set of all valid ballot serial numbers s.

Our construction utilizes the verifiable random function of Dodis and Yampolskiy [46]. Its security relies on several assumptions. First, it requires an IND-CCA2 secure [37] public key encryption function $\mathsf{Enc}_{e,r} : \{0,1\}^* \rightarrow \{0,1\}^*$.[11] The construction also depends on the existence of groups $\mathbb{G}$ (of prime order $p > 2^k$) and $\mathbb{G}_1$ such that an (admissible) bilinear map exists between the groups and for which the the q-decisional bilinear Diffie-Hellman inversion assumption (q-DBDHI) [47] holds. We briefly review each of these below:

Definition 2. *An* (admissible) bilinear map *is a function* $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ *with the following 3 properties:*

1. Bilinear: $\forall g_1, g_2 \in \mathbb{G}$ *and* $x, y \in \mathbb{Z}$, $e(g_1^x, g_2^y) = e(g_1, g_2)^{xy}$.
2. Non-degenerate: $e(g, g) \neq 1$.
3. Computable: *There is an efficient (polynomial time) algorithm to compute* $e(g_1, g_2)$ $\forall g_1, g_2 \in \mathbb{G}$.

Suppose $\mathbb{G}$ and $\mathbb{G}_1$ are groups with an admissible bilinear map e and $|\mathbb{G}| = p$ with prime $p > 2^k$ and g a generator of $\mathbb{G}$. The q-DBDHI problem asks an algorithm $\mathcal{A}_D$ to distinguish $e(g, g)^{1/x}$ from random $\Gamma \in \mathbb{G}_1$ given $g, g^x, \ldots, g^{(x^q)}$. Formally, let $\beta_{0,D} = e(g, g)^{1/x}$ and $\beta_{1,D} = \Gamma$. If $b \in_R \{0, 1\}$, $\mathcal{A}_D$'s advantage $\mathsf{adv}_{DBDHI}(\mathcal{A}_D)$ is defined as

$$\mathsf{adv}_{DBDHI}(\mathcal{A}_\mathcal{D}) = Pr\left(\mathcal{A}_\mathcal{D}(g, g^x, \ldots, g^{(x^q)}, \beta_{b,D}) = b\right) - \frac{1}{2}$$

where the probability is over $\mathcal{A}_D$'s random tape and the choice of $x \in \mathbb{Z}_p^*$ and $\Gamma \in \mathbb{G}_1$.

Definition 3. *The* q-decisional bilinear Diffie-Hellman inversion assumption (q-DBDHI) [47] *for groups* $\mathbb{G}$ *and* $\mathbb{G}_1$ *with an admissible bilinear map* e *states that no algorithm* $\mathcal{A}_D$ *can win game q-DBDHI with advantage* $\mathsf{adv}_{DBDHI}(\mathcal{A}_D) > \mathsf{negl}(k)$ *in time* $t_D \leq \mathsf{poly}(k)$.

Certain groups over elliptic curves or abelian varieties are believed to satisfy these properties with bilinear maps that can be constructed from the Weil or Tate pairings [46, 48–50].

We also constrain the valid ballot serial numbers $\mathbb{S}$ to a set $\mathbb{S} \subset \mathbb{Z}_p^*$, $|\mathbb{S}| \leq \mathsf{poly}(k)$ for a protocol security parameter k.[12] Lastly, our scheme requires an efficiently computable mapping $\varphi : \mathbb{G}_1 \rightarrow \mathbb{H}$ where $\mathbb{H}$ is the partial domain

[10] Smart cards allow writing of keys such that they can never be read off the cards.

[11] For the complete voting process, the function must also allow publicly provable re-encryption or specific homomorphic operations to enable anonymous tallying although, in this paper, we go into depth on only the ballot casting phase.

[12] Although this size constraint has no practical effect on our scheme, it is necessary for our proof of security. Dodis and Yampolskiy show that an approximately 1000 bit $p = |\mathbb{G}|$ is sufficient for an $\mathbb{S}$ with $|\mathbb{S}| = 2^{160}$ as we use it [46].

of the encryption function $\mathsf{Enc}_{e,r}(v)$ that pertains to r, and φ has a uniform probability distribution over $\mathbb{H}$.[13]

6.2 Vote Casting Protocol

We now outline our scheme, presenting the steps required of an election. Again, we focus primarily on the ballot casting portion of the scheme.

Initialization

Prior to the election, a group of N_T trusties computes $\mathsf{KeyGenEnc}(1^k, N_T)$ with a k of their choice, distributes private key shares $d_1, \ldots, d_{N_T}$, and writes the public encryption key e to the ballot marking machine $\mathcal{M}$. Each of N_C candidate parties i also selects a $K_i \in \mathbb{Z}_p^*$ independently and uniformly at random, and writes secret key K_i to $\mathcal{T}_i$ (e.g. a smartcard, which is inserted into the machine for election day). The party sets the public key to $\lambda_i = g^{K_i}$. All public keys are made available on the public bulletin board $\mathcal{B}$. Physical ballots/receipts $\mathcal{R}$ are created and a unique serial number is printed to each.

Ballot Marking

1. $\mathcal{R} \overset{s}{\longmapsto} \mathcal{M}$: The ballot marking machine reads a serial number s off of the voter's receipt (i.e. a new, blank ballot with a printed serial number).
2. $\mathcal{V} \overset{v}{\longmapsto} \mathcal{M}$: The voter enters her candidate selections into the ballot marking machine.
3. $\mathcal{M} \overset{s}{\longmapsto} \mathcal{T}_i$: The ballot marking machine sends the serial number to each $\mathcal{T}_i$.
4. $\mathcal{T}_i \overset{\pi_i = g^{1/(s+K_i)}}{\longmapsto} \mathcal{M}$: Each $\mathcal{T}_i$ computes a pseudorandomness proof value $\pi_i = \mathsf{GenProof}_i(s)$ and sends it to the ballot marking machine where:
   - $\mathsf{GenProof}_i(s) = g^{1/(s+K_i)}$.
5. $\mathcal{M}$ computes $\mu = \prod_{i=1}^{N_C} \pi_i$ and pseudorandom value $r' = \varphi\left(e(g, \mu)\right)$.
6. $\mathcal{M} \overset{c=\mathsf{Enc}_{e,r'}(v)}{\longmapsto} \mathcal{R}$: The ballot marking machine encrypts the voter's alleged ballot v using pseudorandom value r' and prints it to the receipt.
7. $\mathcal{M} \overset{\text{"Cast vote?"}}{\longmapsto} \mathcal{V}$: The ballot marking machine asks the voter if she would like to cast this ballot.

Option 1: Auditing

1. $\mathcal{V} \overset{\text{"No"}}{\longmapsto} \mathcal{M}$: The voter optionally indicates that she would not like to cast this ballot. (Rather, she is choosing to audit the machine with it.)
2. $\mathcal{M} \overset{s,v,c,\pi_1,\ldots,\pi_{N_C}}{\longmapsto} \mathcal{B}$: The ballot marking machine reveals proofs π_i for the pseudorandom values from each $\mathcal{T}_i$ to the public bulletin board. It also posts the encrypted and plaintext ballots c, v along with the serial number s.

[13] I.e. $\forall r_1, r_2 \in \mathbb{H}\ \ Pr\left(\varphi(x) = r_1\right) = Pr\left(\varphi(x) = r_2\right)$ over the random choice of $x \in \mathbb{G}_1$. This property would likely be approximated in practice with a negligible error (varying inverse exponentially with $\log|\mathbb{G}_1|$) since achieving it exactly requires that $|\mathbb{G}_1| = N|\mathbb{H}|$ for some $N \in \mathbb{Z}$. For readability, however, we assume complete uniformity.

3. $\mathcal{M} \overset{v}{\longmapsto} \mathcal{R}$: The machine appends the plaintext ballot to the receipt, so the voter can verify its correctness. This marks the ballot invalid for casting.
4. $\mathcal{V}$ keeps her receipt $\mathcal{R}$ as evidence of her (uncastable) audit vote. To verify its correctness, she, herself needs only check that the printed plaintext accurately represents her vote and that the exact serial number s, plaintext v, and encryption c all appear together on the public bulletin board and that there is no additional information on the receipt.[14]
5. Any verifier, including $\mathcal{V}$, can read the posted data $s, v, c, \pi_1, \ldots, \pi_{N_C}$ off the bulletin board. That person then checks that $\mathsf{BallotVerify}(s, v, c, \pi_1, \ldots, \pi_{N_C}, e, \lambda_1, \ldots, \lambda_{N_C})$ outputs $\mathsf{SUCCESS}$ where it is defined as follows:
   - $\mathsf{BallotVerify}(s, v, c, \pi_1, \ldots, \pi_{N_C}, e, \lambda_1, \ldots, \lambda_{N_C})$: verifies that $e(g^s \lambda_i, \pi_i) = e(g, g)$ for each $i \in \{1, \ldots, N_c\}$. It computes $r' = \varphi\left(e\left(g, \prod_{i=1}^{N_C} \pi_i\right)\right)$ and $c'' = \mathsf{Enc}_{e,r'}(v)$, and checks that $c'' = c$. Lastly it verifies that $s \in \mathbb{S}$. It outputs $\mathsf{SUCCESS}$ if all of these checks hold and $\mathsf{FAILURE}$ otherwise.

   (Indeed for a correct π_i, $e(g^s \lambda_i, \pi_i) = e(g^s g^{K_i}, g^{1/(s+K_i)}) = e(g, g)$.)

Again, the voter may choose to repeat the ballot marking and optional auditing steps on new ballots an unbounded number of times, even after she has cast her vote. Section 7 discusses the effectiveness of such auditing.

Option 2: Casting

1. $\mathcal{V} \overset{\text{"Yes"}}{\longmapsto} \mathcal{M}$: The voter indicates that she would like to cast this ballot.
2. $\mathcal{R} \overset{c}{\longmapsto} \mathcal{B}$: The voter takes her receipt to a ballot casting station where it is used to cast her encrypted ballot, and it is posted to the bulletin board.

Tallying and Verification.

Voters check that no serial number s appears on the bulletin board more than once. Then tallying and verification occur exactly as in Benaloh's scheme (Section 4.2) where encrypted votes are anonymously shuffled and decrypted with corresponding zero-knowledge proofs of correctness, which voters can verify.

6.3 Security

For readability, we first clarify the generic symbols of our definition with the elements of our specific scheme. The correlations are as follows:

- $\mathcal{P} = \{e, \lambda_1, \ldots, \lambda_{N_C}\}$: public information consists of all the public keys
- $X = \{s, v\}$: casting input consists of the ballot serial number and the vote
- $\Psi = \{c\}$: vote casting output consists of the encrypted ballot
- $\pi = \{\pi_1, \ldots, \pi_{N_C}\}$: proofs of correctness consist of the proofs of pseudorandomness from each $\mathcal{T}_i$
- $\Psi_{\mathcal{A}} = \{s, c\}$: adversary's visible output consists of the ballot serial number and the encrypted vote[15]

[14] "Helper organizations" may also be created to assist voters in this step, and the voter may go to a "helper organization" sponsored by a party she trusts.

[15] Implicitly, as we make this correlation, we are assuming the existence of a private channel [21] between $\mathcal{V}$ and $\mathcal{M}$, and between $\mathcal{M}$ and each $\mathcal{T}_i$.

- $s = s$: seed is the ballot serial number
- $\mathsf{BallotVerify}(X, \Psi, \pi, \mathcal{P}) = \mathsf{BallotVerify}(s, v, c, \pi_1, \ldots, \pi_{N_C}, e, \lambda_1, \ldots, \lambda_{N_C})$: verification function is directly represented by the one used in our scheme (where we slightly abuse the notation)
- $\mathsf{GenerateInput}(X_{\mathcal{A}}, s, v) = \{s, v\}$: our coercion resistant inputs are generated by the trivial function giving the specified input vote and seed

Theorem 1. *For all inputs $X = \{s, v\}$ and public data $\mathcal{P}$, there do not exist two output value pairs $(\Psi, \pi) \neq (\Psi', \pi')$ such that $\mathsf{BallotVerify}(X, \Psi, \pi, \mathcal{P}) = \mathsf{SUCCESS}$ and $\mathsf{BallotVerify}(X, \Psi', \pi', \mathcal{P}) = \mathsf{SUCCESS}$.*

Proof. Assume there exist $X = \{s, v\}, (\Psi, \pi), (\Psi', \pi')$ for which this does not hold. $\mathsf{BallotVerify}(s, v, c, \pi_1, \ldots, \pi_{N_C}, e, \lambda_1, \ldots, \lambda_{N_C})$ recomputes c'' as a deterministic function of $s, v, \pi_1, \ldots, \pi_{N_C}$ and compares to the input c for equivalence. Hence, $\Psi \neq \Psi'$ implies $\pi \neq \pi'$.

Let $\pi = \{\pi_0, \ldots, \pi_{N_C}\}$ and $\pi' = \{\pi'_0, \ldots, \pi'_{N_C}\}$, and without loss of generality, assume $\pi_j \neq \pi'_j$. Because $\forall a, b \in \mathbb{G}\ e(a, b)^{|\mathbb{G}|} = e(a^{|\mathbb{G}|}, b) = e(a^0, b) = e(a, b)^0$, $|\mathbb{G}_1|$ divides $|\mathbb{G}|$. Since $e(g, g) \neq 1$, $|\mathbb{G}_1| = p$ prime. Let $\pi_j = g^x$ and $\pi'_j = g^y$. $\mathsf{BallotVerify}$ outputs $\mathsf{SUCCESS}$ implies $e(g^s \lambda_j, \pi_j) = e(g^s \lambda_j, \pi'_j) = e(g, g)$, so $e(g^s \lambda_j, g)^x = e(g^s \lambda_j, g)^y$. Thus, $x \equiv y \mod p$, and $\pi_j = \pi'_j$.

Theorem 2. *Suppose the $|\mathbb{S}|$-DBDHI assumption holds and $\mathbb{S}$, the input set of queries to $\mathsf{GenProof}_i$, satisfies $|\mathbb{S}| \leq \mathsf{poly}(k)$. Then, if key K_j, $j \in \{1, \ldots, N_C\}$ is chosen independently at random from $\mathbb{Z}_p^*$,[16] and $\mathcal{K} = \{K_1, \ldots, K_{j-1}, K_{j+1}, \ldots, K_{N_C}\}$, for any $\mathsf{poly}(k)$ time algorithm $\mathcal{A}_R$:*

$$Pr\left[\left. b = b'_R \right| \begin{array}{c} (s, \mathsf{state}) \leftarrow \mathcal{A}_R^{\mathsf{GenProof}_j(\cdot)}(\lambda_j, \mathcal{K}); \\ \beta_{0,R} = \varphi\left(e(g, \prod_{i=1}^{N_C} \mathsf{GenProof}_i(s))\right); \beta_{1,R} \xleftarrow{R} \mathbb{H}; \\ b \xleftarrow{R} \{0,1\}; b'_R \leftarrow \mathcal{A}_R^{\mathsf{GenProof}_j(\cdot)}(\beta_{b,R}, \lambda_j, \mathcal{K}, \mathsf{state}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(k)$$

with the constraint that $\mathcal{A}_R$ never queries $\mathsf{GenProof}$ on s.[17]

Proof. Dodis and Yampolskiy prove [46] that for any $\mathsf{poly}(k)$ time algorithm $\mathcal{A}_Y$:

$$Pr\left[\left. b = b'_Y \right| \begin{array}{c} (s, \mathsf{state}) \leftarrow \mathcal{A}_Y^{\mathsf{GenProof}_j(\cdot)}(\lambda_j); \\ \beta_{0,Y} = e(g, g)^{1/(s+K_j)}; \beta_{1,T} \xleftarrow{R} \mathbb{G}_1; \\ b \xleftarrow{R} \{0,1\}; b'_Y \leftarrow \mathcal{A}_Y^{\mathsf{GenProof}_j(\cdot)}(\beta_{b,Y}, \lambda_j, \mathsf{state}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(k)$$

under the $|\mathbb{S}|$-DBDHI assumption and with the constraints that $\mathcal{A}_Y$ never queries $\mathsf{GenProof}$ on s and $|\mathbb{S}| \leq \mathsf{poly}(k)$. (We refer to this property as DY.) Our slightly modified theorem follows.

[16] I.e. at least one candidate party behaves honestly.

[17] Intuitively, no adversary with all but one of the party secret keys and the ability to adaptively query $\mathsf{GenProof}_j$ with different seeds can efficiently distinguish the generated pseudorandom number r' from a random element in $\mathbb{H}$.

Assume there exists a $t_R \leq \mathsf{poly}(k)$ time algorithm $\mathcal{A}_R$ that contradicts our theorem. Then we can create a $\mathsf{poly}(k)$ time algorithm $\mathcal{A}_Y$ that interacts with $\mathcal{A}_R$ to contradict DY: Let K_i be any keys $K_i \in \mathbb{Z}_p^*$ $\forall i = 1, \ldots, N_C, i \neq j$ and known to $\mathcal{A}_Y$. Assume key K_j, unknown to $\mathcal{A}_Y$, is chosen independently and at random from $\mathbb{Z}_p^*$. $\mathcal{A}_Y$'s goal is to use $\mathcal{A}_R$ to distinguish $e(g,g)^{1/(s+K_j)}$ from a random element of $\mathbb{G}_1$ with non-negligible advantage.

$\mathcal{A}_Y$ responds to each of $\mathcal{A}_R$'s queries to $\mathsf{GenProof}_j$ by querying its own $\mathsf{GenProof}_j$ oracle and passing the response to $\mathcal{A}_R$. When $\mathcal{A}_R$ is ready for its challenge $\beta_{b,R}$, $\mathcal{A}_Y$ requests its challenge value $\beta_{b,Y}$ with $\mathcal{A}_R$'s query s. $\mathcal{A}_Y$ replies to $\mathcal{A}_R$ with $\beta_{b,R} = \varphi\Big(e\big(g,g\big)^{\sum_{i\neq j} 1/(s+K_i)}\beta_{b,Y}\Big)$. Because $\varphi(x)$ has a uniform probability distribution over x and $\beta_{1,Y}$ is chosen independently and randomly from $\mathbb{G}_1$, $\beta_{1,R}$ is a uniformly randomly chosen element of $\mathbb{H}$.

$$\begin{aligned}\beta_{0,R} &= \varphi\left(e\big(g,g\big)^{\sum_{i\neq j} 1/(s+K_i)} e\big(g,g\big)^{1/(s+K_j)}\right) \\ &= \varphi\left(e\big(g, g^{\sum_{i=1}^{N_C} 1/(s+K_i)}\big)\right) = \varphi\left(e\big(g, \textstyle\prod_{i=1}^{N_C} \mathsf{GenProof}_i(s)\big)\right)\end{aligned}$$

Thus, $\mathcal{A}_R$'s bit b from the Theorem 2 simulation corresponds exactly to the challenger's bit b, and $\mathcal{A}_Y$, which replies to the challenger with $\mathcal{A}_R$'s output $b'_Y = b'_R$, is correct ($b'_Y = b$) exactly when $\mathcal{A}_R$ is. $\mathcal{A}_Y$'s work consists mainly of of responding to $\mathcal{A}_R$'s queries on $\mathsf{GenProof}_j$, so its run-time is bounded by $t_Y < n_1 t_R + n_2$ for some constants n_1, n_2, and therefore $t_Y \leq \mathsf{poly}(k)$.

Theorem 3. *Suppose at least one key K_j is selected independently at random from $\mathbb{Z}_p^*$ and is kept secret, $\mathsf{Enc}_{e,r}(\cdot)$ is an IND-CPA secure public key encryption function [51, 52],*[18] *groups $\mathbb{G}$, $\mathbb{G}_1$ have an admissible bilinear map such that the $|\mathbb{S}|$-DBDHI assumption holds and $|\mathbb{G}| = p$, a function $\varphi : \mathbb{G}_1 \to \mathbb{H}$ exists as described above, and the set of possible seeds $\mathbb{S}$ satisfies $|\mathbb{S}| \leq \mathsf{poly}(k)$ (for security parameter k). Then the scheme presented in Section 6.2 is coercion resistant.*

Proof. We briefly review IND-CPA security for a public key encryption function (that uses a random number r as part of its input) below. It simply states that for any $\mathsf{poly}(k)$ time algorithm $\mathcal{A}_C$:

$$Pr\left[b = b'_C \,\middle|\, \begin{array}{c}(e,d) \leftarrow \mathsf{KeyGenEnc}(1^k, N_T); \\ (m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}_C(e); b \xleftarrow{R} \{0,1\}; r \xleftarrow{R} \mathbb{H} \\ \beta_C = \mathsf{Enc}_{e,r}(m_b); b'_C \leftarrow \mathcal{A}_C(e, \beta_C, \mathsf{state})\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(k).$$

$\mathcal{A}$ cannot win IEV if BallotVerify outputs FAILURE. Hence, by Theorem 1, $\mathcal{A}$ wins IEV implies that $\mathcal{C}^*$ provides outputs $(\Psi, \pi) = (\{c = \mathsf{Enc}_{e,r'}(v)\}, \{\pi_i = \mathsf{GenProof}_i(s)\}_{i=1,\ldots,N_C})$ where $r' = \varphi\Big(e\big(g, \prod_{i=1}^{N_C} \pi_i\big)\Big)$ for each input $X = \{s, v\}$.

[18] The use of an IND-CCA2 encryption scheme [37] for our construction is important for the decryption and tallying process of the scheme, which we separate from the ballot casting and do not analyze in depth in the paper. However, to be precise, coercion resistance of the ballot casting requires only IND-CPA security. Of course, IND-CCA2 security implies IND-CPA security.

From here we see that IEV closely models the CPA game where the random r is replaced by a pseudorandom r'. If we have a $t_{\mathcal{A}} < \mathsf{poly}(k)$ time algorithm $\mathcal{A}$ where $\mathsf{adv}_{IEV}(\mathcal{A}) > \mathsf{negl}(k)$, then we can either create an algorithm $\mathcal{A}_C$ that contradicts the IND-CPA security of Enc or an algorithm $\mathcal{A}_R$ that contradicts Theorem 2. Assume such an $\mathcal{A}$ exists. Let $K_i, i \neq j$ be any elements of $\mathbb{Z}_p^*$ known to $\mathcal{A}$, $\mathcal{A}_C$, and $\mathcal{A}_R$ and let $K_j \in_R \mathbb{Z}_p^*$ be known only to $\mathcal{A}_C$. $\lambda_j = g^{K_j}$ is public.

First we consider the possibility of an algorithm $\mathcal{A}_C$ running the IND-CPA security simulation whose goal is to distinguish 2 encrypted ciphertexts. For every query (s, v) made by $\mathcal{A}$ to $\mathcal{C}^*$, $\mathcal{A}_C$ replies to $\mathcal{A}$ with $c = \mathsf{Enc}_{e,r'}(v)$ and π_i values such that $\mathsf{BallotVerify}$ succeeds. When $\mathcal{A}$ submits its challenge votes $((s, v_0), (s, v_1))$, $\mathcal{A}_C$ passes (v_0, v_1) to its challenger, and receives β_C (computed as above), which it forwards to $\mathcal{A}$. $\mathcal{A}_C$ continues to reply to $\mathcal{A}$'s queries as before and finally submits $\mathcal{A}$'s value $b'_C = b'$. $\mathcal{A}_C$'s work consists mainly of computing values c and π_i in response to $\mathcal{A}$'s queries, so it's runtime is bounded by $t_C < n_1 t_{\mathcal{A}} + n_2$ for constants n_1, n_2. Let $Pr(b = b'_C) = \varepsilon_C$. If $\varepsilon_C > \frac{1}{2} + \mathsf{negl}(k)$, then we have contradicted that Enc is IND-CPA secure.

Suppose otherwise. Let $\mathcal{A}$ guess $b' = b$ in IEV with probability $\varepsilon > \frac{1}{2} + \mathsf{negl}(k)$ (when all verifications succeed) and $\mathcal{A}_C$ guess $b'_C = b$ in the IND-CPA simulation with probability $\varepsilon_C < \frac{1}{2} \pm \mathsf{negl}(k)$. We create a second algorithm $\mathcal{A}_R$ that contradicts Theorem 2. $\mathcal{A}_R$'s goal is to distinguish a truly random $r \in \mathbb{H}$ from an $r' = \varphi\Big(e\big(g, \prod_{i=1}^{N_C} \pi_i\big)\Big)$ following the simulation defined in Theorem 2. For each query (s, v) that $\mathcal{A}$ makes to $\mathcal{C}^*$, $\mathcal{A}_R$ computes $\mathsf{GenProof}_i$ for $i \neq j$ using K_i and queries its $\mathsf{GenProof}_j$ oracle to obtain each $\pi_i = \mathsf{GenProof}_i(s)$. It computes the corresponding r' and $c = \mathsf{Enc}_{e,r'}(v)$, with which it replies to $\mathcal{A}$. On the challenge $((s, v_0), (s, v_1))$, $\mathcal{A}_R$ submits s to its challenger and receives a value $r'' = \beta_{b,R}$, which it uses to compute $c = \mathsf{Enc}_{e,r''}(v_{b''})$ after choosing $b'' \in_R \{0, 1\}$. $\mathcal{A}_R$ answers $\mathcal{A}$'s queries as before until $\mathcal{A}$ outputs b'. If $b' = b''$, $\mathcal{A}_R$ outputs $b'_R = 0$. Otherwise, it outputs $b'_R = 1$. The probability of $\mathcal{A}_R$'s success is $Pr(b'_R = b) = Pr(b'_R = b | b = 0 \cap b = 0) + Pr(b'_R = b | b = 1 \cap b = 1) = \frac{1}{2}\varepsilon + \frac{1}{2}\varepsilon_C = \frac{1}{4} + \frac{1}{2}\varepsilon \pm \frac{1}{2}\mathsf{negl}'(k) > \frac{1}{2} + \mathsf{negl}(k)$. Since $\mathcal{A}_R$'s primary work is done computing responses to $\mathcal{A}$'s queries, its runtime is bounded by $t_R < n_1 t_{\mathcal{A}} + n_2$ for some constants n_1, n_2, so $t_R \leq \mathsf{poly}(k)$.

Although we focus on coercion resistance, lastly, we informally note that there is a guarantee that each output c is an encryption of the voter's vote v since $\mathsf{BallotVerify}$ explicitly recomputes the encryption of v and checks for equivalence (*individual verifiability*). Furthermore, each voter can compute and verify the tally using the public, decrypted ballots (*universal verifiability*).

7 Practical Considerations

To this point, we have focused largely on more rigorous coercion resistance, with minimal discussion of deployment. In this section, we briefly clarify a few of the more practical aspects of our protocol.

As mentioned, our protocol, adapted from Benaloh's technique [12, 13], attempts to minimize impact on the voter. The typical voter in our scheme can simply walk into the voting booth, mark her ballot as she would normally vote on a common touchscreen system, choose to cast it, and take her receipt to a separate machine where it is scanned and cast. The choice not to cast a marked ballot and audit is completely optional. At the same time, recall that all a voter needs to do to successfully audit the machine with an uncastable receipt (marked ballot) is verify that the plaintext printed on it is correct and then check that the printed serial number, encrypted ballot, and plaintext all appear correctly on the bulletin board. Philanthropic or politically motivated organizations may also assist voters in this task. Lastly, as long as at least one honest voter verifies the cryptographic operations on the board, fallacious computation of the tally is detected, although anyone has the option of doing so.

This simplicity introduces several intricacies regarding verification. One result we notice, is that the scheme can directly prove nothing about the correctness of the content of ballots that are actually cast. Instead, it relies on the option for voters, officials, etc. to audit to provide probabilistic assurances. Because the ballot marking machine is separate from the receipt scanning, casting machine, the ballot marking machine does not need to know anything about the voter. With no voter information, the machine can do approximately no better than to cheat at random. As a result, only a small number of audits are necessary to achieve a relatively high guarantee of accuracy. An analysis by Neff [53], shows that in general, in an election with N_V voters, M_V compromised votes, and A_V audit votes, a very crude approximation of the probability of detection Pr_D when $N_V \gg M_V + A_V$ is $Pr_D \approx 1 - (1 - \frac{M_V}{N_V})^{A_V}$. More concretely (and using more precise calculations [53]), suppose there were an election with 100,000 voters and a machine attempted to dishonestly encrypt 500 ballots. If 1% of the created ballots were randomly audited, the cheating would be detected with greater than 99% probability.

As a final practical note, we notice from our definition that queried seeds s must be unique. To address this issue in practice, we suggest that each party supply a $\mathcal{T}_i$ that responds only to queries of strictly increasing seeds[19] and that ballots are provided to voters in sequence with respect to their serial numbers. Again, as long as at least one party behaves honestly, seed uniqueness and thus voter privacy are assured. To maintain usability, the ballot marking machine can check that each serial number is within an expected range prior to querying the $\mathcal{T}_i$s and provide a warning requesting poll worker assistance if not. Note that such functionality purely prevents accidental usage problems and does not place any trust on the machine with respect to vote integrity or privacy.

8 Conclusion

Coercion resistance is vital to election integrity. Because attacks are often subtle, it must be addressed rigorously. We formally define coercion resistance for end-

[19] This only requires that $\mathcal{T}_i$ can store a single serial number and compute a comparison.

to-end voting. We then construct an end-to-end voting scheme that is provably coercion resistant and minimally impacts the voting process for the typical voter.

Acknowledgments

National Science Foundation grant CNS-0524252 supported this work. We thank Josh Benaloh, Ariel Feldman, and Susan Hohenberger for their insights.

References

1. Kohno, T., Stubblefield, A., Rubin, A.D., Wallach, D.S.: Analysis of an electronic voting system. In: IEEE Symposium on Security and Privacy. (2004)
2. Feldman, A.J., Halderman, J.A., Felten, E.W.: Security analysis of the Diebold AccuVote-TS voting machine. In: EVT '07:USENIX/ACCURATE Electronic Voting Technology Workshop. (2007)
3. Hursti, H.: Diebold TSx evaluation: Critical security issues with Diebold TSx (May 2006) Available at `http://www.blackboxvoting.org/BBVreportIIunredacted.pdf`.
4. Proebstel, E., Riddle, S., Hsu, F., Cummins, J., Oakley, F., Stanionis, T., Bishop, M.: An analysis of the Hart Intercivic dau eslate. In: EVT '07: USENIX/ACCURATE Electronic Voting Technology Workshop. (2007)
5. Gardner, R., Yasinsac, A., Bishop, M., Kohno, T., Hartley, Z., Kerski, J., Gainey, D., Walega, R., Hollander, E., Gerke, M.: Software review and security analysis of the Diebold voting machine software. Technical report, Florida Department of State (July 2007)
6. Calandrino, J.A., Feldman, A.J., Halderman, J.A., Wagner, D., Yu, H., Zeller, W.P.: Source code review of the Diebold voting system. Technical report, California Secretary of State (July 2007)
7. Inguva, S., Rescorla, E., Shacham, H., Wallach, D.S.: Source code review of the Hart InterCivic voting system. Technical report, California Secretary of State (July 2007)
8. Blaze, M., Cordero, A., Engle, S., Karlof, C., Sastry, N., Sherr, M., Stegers, T., Yee, K.P.: Source code review of the Sequoia voting system. Technical report, California Secretary of State (July 2007)
9. Neff, A.: Practical high certainity intent verification for encrypted votes (2004) Available at `http://www.votehere.com/vhti/documentation`.
10. Chaum, D.: Secret-ballot receipts: True voter-verifiable elections. IEEE Security and Privacy **2**(1) (2004) 38–47
11. Riva, B., Ta-Shma, A.: Bare-handed electronic voting with pre-processing. In: EVT '07:USENIX/ACCURATE Electronic Voting Technology Workshop. (2007)
12. Benaloh, J.: Ballot casting assurance via voter-initiated poll station auditing. In: EVT '07:USENIX/ACCURATE Electronic Voting Technology Workshop. (2007)
13. Benaloh, J.: Simple verifiable elections. In: EVT '06:USENIX/ACCURATE Electronic Voting Technology Workshop. (2006)
14. Rivest, R.L., Smith, W.D.: Three voting protocols: Threeballot, VAV, and twin. In: EVT '07: USENIX/ACCURATE Electronic Voting Technology Workshop. (2007)
15. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme: A practical solution to the implementation of a voting booth. In: EUROCRYPT '95: Advances in Cryptology. (1995)

16. Cranor, L.F., Cytron, R.K.: Sensus: A security-conscious electronic polling system for the internet. In: HICSS '97: Hawaii International Conference on System Sciences. (1997)
17. Karlof, C., Sastry, N., Wagner, D.: Cryptographic voting protocols: A systems perspective. In: USENIX Security Symposium. (2005)
18. Moran, T., Naor, M.: Split-ballot voting: everlasting privacy with distributed trust. In: CCS '07: ACM conference on Computer and Communications Security. (2007)
19. Chaum, D.: Punch scan Available at `http://www.punchscan.org/learnmore.php`.
20. Moran, T., Naor, M.: Receipt-free universally-verifiable voting with everlasting privacy. In: CRYPTO '06: Advances in Cryptology. (2006)
21. Benaloh, J., Tuinstra, D.: Receipt-free secret-ballot elections (extended abstract). In: STOC '94: ACM Symposium on Theory of Computing. (1994)
22. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: EUROCRYPT '00: Advances in Cryptology. (2000)
23. Stenger, R.: Vote-selling web site to be revived, possibly offshore. CNN (August 2005) Available at `http://archives.cnn.com/2000/TECH/computing/08/24/internet.vote/index.h%tml`.
24. Tribune, S.: U student who offered his vote on eBay gets community service. Star Tribune (2008) Available at `http://www.startribune.com/politics/state/26063069.html`.
25. Local 6, O.: Man accused of trying to sell vote (October 2004) Available at `http://www.local6.com/news/3834797/detail.html`.
26. Yao, A.C.: Protocols for secure computations. In: FOCS '82: IEEE Symposium on Foundations of Computer Science. (1982)
27. Benaloh, J.D.C.: Verifiable Secret-ballot Elections. PhD thesis, Yale University (1987)
28. Chaum, D.: Elections with unconditionally secret ballots and disruption equivalent to breaking RSA. In: EUROCRYPT '88: Advances in Cryptology. (1988)
29. Magkos, E., Burmester, M., Chrissikopoulos, V.: Receipt-freeness in large-scale elections without untappable channels. In: I3E '01: IFIP Conference on Towards The E-Society. (2001)
30. Benaloh, J.: Administrative and public verifiablity: Can we have both? In: EVT '08:USENIX/ACCURATE Electronic Voting Technology Workshop. (2008)
31. Chaum, D., Ryan, P.Y., Schneider, S.: A practical voter-verifiable election scheme. In: ESORICS '05: European Symposium on Research in Computer Security. (2005)
32. Ryan, P.Y., Peacock, T.: Prêt à voter: A systems perspective. Technical report, University of Newcastle (2005)
33. Ryan, P.: Prêt à Voter with Pallier encryption. Technical report (2006) Available at `http://www.cs.ncl.ac.uk/research/pubs/trs/papers/965.pdf`.
34. Xia, Z., Schneider, S.A., Heather, J., Traore, J.: Analysis, improvement and simplification of prêt à voter with pallier encryption. In: EVT '08:USENIX/ACCURATE Electronic Voting Technology Workshop. (2008)
35. Chaum, D., Carback, R., Clark, J., Essex, A., Popoveniuc, S., Rivest, R., Ryan, P., Shen, E., Sherman, A.T.: Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In: EVT '08:USENIX/ACCURATE Electronic Voting Technology Workshop. (2008)
36. Chaum, D., Essex, A., Carback, R., Clark, J., Popveniuc, S., Sherman, A.T., Vora, P.: Scantegrity: End-to-end voter verifiable optical scan voting. Security and Privacy, IEEE **6** (2008)
37. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: CRYPTO '91: Advances in Cryptology. (1992)

38. Jakobsson, M., Juels, A.: Millimix: Mixing in small batches. Technical Report 99-33, DIMACS (1999)
39. Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: USENIX Security Symposium. (2002)
40. Golle, P., Jakobsson, M., Juels, A., Syverson, P.F.: Universal re-encryption for mixnets. In: Topics in Cryptology - CT-RSA '04: Cryptographers' Track at RSA. (2004)
41. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing **18**(1) (1989)
42. Pedersen, T.P.: A threshold cryptosystem without a trusted party. In: EUROCRYPT '91: Advances in Cryptology. (1991)
43. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: WPES '05:ACM Workshop on Privacy in the Electronic Society. (2005)
44. Teague, V., Ramchen, K., Naish, L.: Coercion-resistant tallying for STV voting. In: EVT '08:USENIX/ACCURATE Electronic Voting Technology Workshop. (2008)
45. Canetti, R., Gennaro, R.: Incoercible multiparty computation (extended abstract). In: FOCS '96: IEEE Symposium on Foundations of Computer Science. (1996)
46. Dodis, Y., Yampolskiy, A.: A verifiable random function with short proofs and keys. In: PKC '05: Workshop on Theory and Practice of Public Key Cryptography. (2005)
47. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: EUROCRYPT '04: Advances in Cryptology. (2004)
48. Galbraith, S.D.: Supersingular curves in cryptography. (2001)
49. Joux, A., Nguyen, K.: Separating decision diffie-hellman from computational diffie-hellman in cryptographic groups. Journal of Cryptography **16**(4) (2001) 239–247
50. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: CRYPTO '01: Advances in Cryptology. (2001)
51. Goldwasser, S., Micali, S.: Probabalistic encryption. Journal of Computer and System Science **28**(2) (1984)
52. Micali, S., Rackoff, C., Sloan, B.: The notion of security for probabilistic cryptosystems. SIAM Journal on Computing **17**(2) (1988) 412–426
53. Neff, A.: Election confidence: A comparison of methodologies and their relative effectiveness at achieving it (2003) Available at `http://www.votehere.com`.