

**Johns Hopkins University
Security Privacy Applied Research Lab**

**Protecting Against Privacy Compromise and Ballot
Stuffing by Eliminating Non-Determinism from
End-to-end Voting Schemes**

**Technical Report
SPAR-JHU:RG-SG-AR:245631**

Submitted as SPAR Technical Report on April 16th, 2008

Protecting Against Privacy Compromise and Ballot Stuffing by Eliminating Non-Determinism from End-to-end Voting Schemes

Ryan Gardner Sujata Garera Aviel D. Rubin
ryan@cs.jhu.edu sgarera@cs.jhu.edu rubin@jhu.edu
Johns Hopkins University

Abstract

End-to-end voting schemes have been researched significantly over the past two decades and aim to establish voter verifiable elections. Many of these schemes rely on non-deterministic inputs, such as the randomness used in encryption, for several tasks in the voting process. Karlof *et al.* show that the randomness in a scheme by Neff enables potential malicious subliminal channels. We revisit the issue of non-determinism and show that it leads to possible attacks on several end-to-end voting schemes including coercion and ballot stuffing. In particular, we argue that non-determinism in cryptographic voting schemes is problematic in general.

In this paper we present a technique for eliminating non-determinism in cryptographic voting protocols. Our approach provides the voter with a proof of deterministically generated pseudo-randomness. This requires trust in at least one of the political parties. Our method is general and we discuss its application to several existing voting protocols. As such, our constructions provide probabilistic assurance to the voter that her ballot was created deterministically.

1 Introduction

Electronic voting has received considerable attention since the 2000 presidential election in the United States. The election held that year in the state of Florida brought to light a significant number of usability and accuracy issues with paper ballots and consequently led to the widespread use of direct recording electronic (DRE) machines. While these machines address most of the usability issues of an election, they also generate a large number of new security concerns. A number of studies [32, 18, 25, 42, 21, 10, 26, 7] have independently analyzed DRE systems and shown that they have many vulnerabilities; they are even vulnerable to viruses that can undetectably change votes. As such, the election process is wholly unverifiable and voters 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 [40, 13, 43, 4, 3, 45], which are also commonly referred to as cryptographic voting schemes.¹ These schemes are designed to allow each voter to publicly verify both that her vote is accurately recorded (often referred to as individual verifiability) and that all recorded votes are correctly tallied

¹with one exception [45]

in the final sums (universal verifiability) [47, 15, 30]. At the same time, they are intended to prevent a voter from revealing her vote to a possible coercer. Specifically, end-to-end voting schemes aim to achieve strong accuracy guarantees *regardless* of the software that is on the machines.

In a typical end-to-end voting scheme, a voter first interacts with one or more voting machines (similar in appearance to a current DRE) to cast her intentions and create some form of an encoded ballot. The encoded ballot, along with other possible data for verification, is published to a publicly available bulletin board (most likely on the Internet), and it is also printed and taken by the voter as a receipt. After the election, each voter can use her receipt to check that her encoded ballot appears on the bulletin board. All the votes on the board are tallied using a cryptographic mechanism such that correctness of the tally can be publicly verified. Despite the verifiability, no voter can use her receipt or information on the bulletin board to prove how she voted.

During this process, many cryptographic voting schemes rely on randomness for a variety of purposes. For example, the scheme developed by Neff [40] encrypts bits using a form of ElGamal. Karlof *et al.* analyzed Neff's scheme and show that its use of the randomized encryption allows for malicious subliminal channels, which in turn can compromise voter privacy [30]. In this paper we revisit the issue of non-determinism in cryptographic voting schemes and argue that (with respect to the machine and poll-workers) it is problematic on the whole² because it adds an inherently unverifiable channel to the voting process. We show that the use of randomness enables a number of attacks in addition to the mentioned subliminal channels in Neff's scheme. These include possible coercion in the scheme of Riva and Ta-Shma [43], where the voter never even reveals her vote to the voting machine.

We propose a new technique to eliminate non-determinism from cryptographic voting schemes. Using the technique, pseudorandomness needed for randomized functions or operations is deterministically generated from a privately keyed function. Furthermore, voters are given proofs that the pseudorandomness was correctly generated. The scheme provides verifiability and assurance to the voter at a low cost. In order to achieve these guarantees, we place trust in at least one of the many candidate parties of the election. We propose two constructions of our scheme. The first construction is secure in the standard model and requires programmable smart cards (Section 4.2) while the second construction requires only standard ISO 7816 compliant smart cards and is secure in the random oracle model (Section 4.3). We focus on the cryptographic voting scheme of Benaloh [3, 4] as a central example for our paper primarily due to its simplicity and high usability. We also discuss ways that our techniques may be applied to the schemes of Neff [40] and of Riva and Ta-Shma [43] and finally conclude with some open problems.

2 Related Work

There has been a significant amount of research in cryptographic voting schemes over the past two decades. The first voting schemes were pioneered independently by Yao, Benaloh, and Chaum [49, 6, 12]. These schemes, however, were not coercion resistant and allowed the voter to use her receipt to prove how she voted to a coercer.

Benaloh *et al.* [5] were the first to introduce the notion of receipt freeness and describe a secret ballot election protocol that does not allow the voter to take the receipt with her. This protocol relies on a threshold scheme and is secure only against static adversaries. Sako *et al.* [47]

²In general, it is still okay for the *voter* to make non-deterministic choices.

subsequently proposed a receipt free system that relies on using mix networks. In their scheme, however, a coercer can force a voter to vote randomly.

Magkos *et al.* [35] proposed a receipt free scheme that relies on a tamper resistant smart card which collaborates with the voter to produce a valid encryption of her vote. The smart card provides a zero knowledge proof that encryptions are correct, but the authentication steps require significant voter participation.

Chaum and Neff [13, 40] independently pioneered electronic voting schemes that did not require the voter to possess any computational device to verify her vote. These types of voting schemes have come to be known as *bare handed* voting schemes. Chaum's scheme 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, introduced a scheme that encodes the voter's choice in an encrypted array of bits [40]. It further commits to the voter's choice by displaying a short string on voting screen. The verification of the voter's intent, depends on the voter's ability to compare short strings. In particular, the voter needs to verify that the commitment that was displayed near her chosen candidate is the same as printed on her receipt. If the voter can compare strings of length m then the voter can verify that her ballot was created correctly with probability $1 - (\frac{1}{35})^m$. Typically m is set to 4, which yields a significantly higher probability than the $1/2$ probability achieved in Chaum's scheme [13].

Karlof *et al.* were the first to analyze Chaum's and Neff's schemes from a systems perspective [30] and discover a number of flaws in them, including subliminal channels made possible by the use of randomness in Neff's scheme. They suggest possibly using a random tape to address the subliminal channels. However, they do not give a detailed implementation of the tape and the necessary zero knowledge proofs for such an implementation are unclear. More importantly, as noted by the authors, such zero knowledge proofs generally require randomness themselves and may lead to additional subliminal channels. Also, an adversary who obtained this tape would be able to compromise voter privacy. Finally, in cases where the ballot creation and casting devices are separated, as in our protocol, it is desirable to allow ballot creating machines to have as little storage as possible to prevent recording of voters' choices. This will be more difficult if a random tape is used.

Riva *et al.* [43] recently proposed a bare handed voting scheme which requires the voter to pre-process her ballot using a computational device and relying on any party he trusts. Unlike Neff's and Chaum's schemes, this scheme maintains privacy with respect to the voting booth by not revealing the voter's choice to the booth.

Benaloh was the first to introduce the notion of challenging a completely created ballot [3, 4]. His protocol is simple and gives the voter a choice to either cast a ballot or open it. The cut and choose nature of his scheme provides coercion resistance. However, his scheme relies on randomized encryption and is prone to several vulnerabilities. We discuss Benaloh's scheme in detail in Section 3.

The Prêt à Voter scheme [14, 46], introduced by Chaum *et al.*, is similar in concept to Chaum's scheme, but does not rely on visual cryptography. The paper ballot presents a randomized list of candidates to the voter, the encoding of which is printed on the ballot. Once the voter makes her choice, the ballot is split such that it is infeasible to determine the voter's selection without the ability to decrypt the ballot order. More recently, Rivest and Warren present 3 paper based voting protocols [45]. These protocols do not rely on any cryptographic techniques, yet they give the voter guarantees that her vote was cast correctly and included in the tally. Kelsey *et al.* present some

approaches to coercing voters using these paper based systems [31].

Several studies have found that pseudorandomness has been poorly generated in existing voting machines [21, 10, 26, 7].

In what follows, we detail and analyze the scheme proposed by Benaloh [3, 4] and then discuss our proposed solutions.

3 Benaloh's Scheme

We examine the recent cryptographic voting scheme of Benaloh [3, 4] as the primary illustrative example for this paper. The scheme presents a novel idea that may be able to provide strong guarantees to voters while imposing minimal burdens on them. It has been implemented as the foundation of the recent VoteBox voting system of Sandler *et al.* [48]. At the same time, we find that it exhibits several potential dangers of probabilistic voting machines. In Section 5, we will also discuss non-determinism and our technique with respect to several other existing schemes.

3.1 Primary Goals

As an end-to-end voting scheme, Benaloh's approach aims to achieve the following:

Cast-as-intended- The voter should be able to verify that her intentions were accurately recorded in her cast ballot. This property is also referred to as *individual verifiability* [47].

Counted-as-cast- Voters should be able to verify that all cast ballots were properly included in the tallies. This notion is also referred to as *universal verifiability* [47].

Mandatory Privacy- No one should be able to learn how another voter voted. This must be true even if the voter wants someone else to know how they voted or the voter may be susceptible to coercion.

No Additional Votes- All counted votes should come exclusively from legitimate voters who voted.

Also, Benaloh specifically attempts to achieve the following additional goal with his scheme:

Minimal User Imposition- The scheme should impose as few requirements on the voter as possible.

This last goal is also particularly important and emphasized slightly more in Benaloh's scheme than many previous schemes. Voters come from widely varying levels of education and familiarity with computers, mathematics, and cryptography. Additional voting requirements have potential to confuse and even cast doubt on the legitimacy of the voting process.

3.2 Overview of the Scheme

We now present a technical overview of the Benaloh scheme [3, 4]. The steps a voter takes to cast and potentially verify her vote are illustrated in Figure 1.

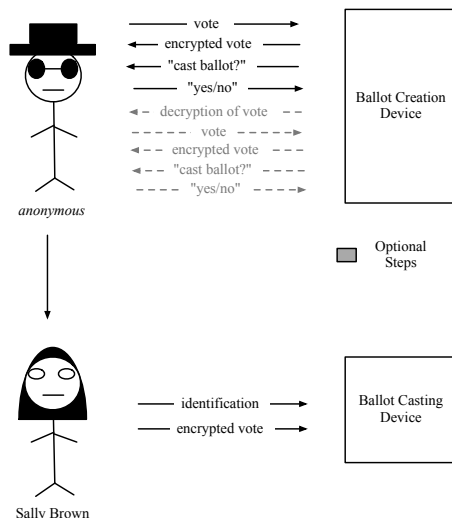


Figure 1: The general interactions between a voter and machines to cast a vote using the Benaloh scheme. Messages in gray are optional and may be repeated as many times as desired by the voter.

BALLOT CREATION. As illustrated, an implementation of the scheme uses two distinct, non-communicating devices: a *ballot creation* device and a *ballot casting* device. On the day of an election, the voter enters the polling place empty handed³ and first approaches the ballot creation device. At this point, the voter does not need or want to identify herself in any way so that the machine knows nothing unique about the voter (overlooking some minor details such as time) that might increase its probability of successfully cheating without detection. The voter simply interacts with the machine just as she would with any direct recording electronic (DRE), and when she has finished indicating her selections, she chooses to create her ballot.⁴ In response, the voting machine computes and prints an encrypted ballot allegedly representing the voter's choices.⁵

OPTIONAL AUDITING. Before the voter is finished interacting with the machine, she is presented with a final option. Rather than taking her castable, encrypted ballot, she can choose to use the ballot to *audit* the machine. (Benaloh suggests that the question presented to the voter may be as simple as "Do you wish to cast this vote?" [3].) In the case where the voter chooses to audit, the machine further opens the encryption of the ballot, by providing the randomness that was used to encrypt.⁶ The randomness is also printed on the paper receipt along with the encrypted ballot. The voter can then take the receipt home and verify that the ballot was properly formed. We also suggest that the plaintext ballot be printed on the receipt so the voter can immediately verify her selections in the polling place. The ballot must also now be marked as invalid since it would reveal the voter's selections if cast. Notice that the voter can interact with the ballot creation device as much as she likes as in the figure and audit an unbounded number of ballots.

Suppose there were an election with 100,000 voters and the machine attempted to dishonestly

³aside from whatever identification information is required by the precinct

⁴The ballot creation machine should have very limited storage to prevent it from simply recording all votes.

⁵Benaloh mentions that the ballot may actually be put on a variety of media. However, we will assume it simply prints the encryption on paper.

⁶In his paper, Benaloh also suggests an interactive proof option that the voter may choose from [3].

encrypt 500 ballots. Using Neff's analysis [39], if 1% of the created ballots were randomly audited, the cheating would be detected with greater than 99% probability.

BALLOT CASTING. At some point, the voter chooses to cast the ballot and leaves the ballot creation device with an unopened, encrypted ballot. She then identifies herself as a valid voter and casts her vote at the vote casting device. (It might scan her encrypted ballot, for example.) She keeps her printed, encrypted ballot as a receipt.

TALLYING AND VERIFICATION. All of the cast ballots are posted, along with the name of each voter, to a public bulletin board, which is typically thought of as a public web site. With her receipt, each voter can verify that her vote is posted. We also strongly suggest that the vote casting device scan and *anonymously* post opened ballots. This will allow auditing voters to simply verify that their open ballots are posted while anyone can verify that they were properly formed. Furthermore, it will allow those who wish to verify that the ballots were properly formed to simply download the data, rather than tediously enter it. It could then be provided as input to verification software available from a variety of sources.

When the voting period has ended and all ballots have been posted, a group of trustees anonymizes the ballots through a mix-net [27, 28, 24]. Each trustee re-encrypts all the ballots and posts them back to the public bulletin board in random order. Each shuffled set of ballots is accompanied by a zero-knowledge proof [23] that a one-to-one correspondence has been maintained from the previous set. Finally, a sufficiently large subset of the trustees use a threshold scheme [41] to jointly decrypt the ballots. Again, a proof is also provided with each decryption so that the correctness of the process can be publicly verified.

3.3 Non-Determinism

Abstractly, the voting, tallying, and verification processes of the Benaloh scheme are simple, and as such, seem to accomplish their goals clearly. However, there are several problems inherent in the scheme's implicit non-determinism that have potential to arise if they are not carefully addressed. Karlof *et al.* [30] make some similar observations with respect to possible subliminal channels caused by the randomness used in the cryptographic voting scheme of Andrew Neff [40].

Semantic security is a strong notion of security intended to guarantee that an adversary cannot learn any information (aside from size) about the plaintext corresponding to a given ciphertext generated using an asymmetric encryption scheme [22]. In particular, schemes with this property prevent an adversary from encrypting guessed plaintexts until she obtains the target ciphertext. The property is essential to encryption schemes in electronic voting applications such as Benaloh's where the plaintext message space is small. Because such plaintext space searching is possible with any deterministic encryption process for which all encryption keys are known, semantic security is achieved by incorporating randomness into the ciphertexts.

CHOOSING "RANDOMNESS". One of the first problems that arises from using randomness in the encryption process is simply that we cannot verify whether or not a given number is truly random. A first possible attack enabled by this problem consists of an adversary simply replacing the code for obtaining the randomness used in the encryption with the output of a pseudo-random number generator, for which she exclusively knows the key. Such an attack will effectively give the adversary knowledge of the randomness used in each encryption and allow her to determine the plaintext of each posted encrypted ballot.

A second possible attack involves breaking the one-to-one correspondence between votes cast and votes posted. Benaloh mentions that the name of each voter may be posted with her corresponding encrypted vote or the correspondence may be omitted to further ensure privacy [3, 4]. Suppose the names of the voters are not posted with their encrypted votes. If the machine sees that Alice and Bob voted the exact same way, it could choose to use the same randomness for each of their ballots. Both would get identical receipts that would verify correctly, and when checking the public bulletin board, each would find that her/his ballot was posted although it would really only be one ballot. An adversary could then stuff a ballot of her choice. (Benaloh also mentions encrypting counters in the ballots in his subsequent paper [4], which would probably make this attack difficult. However, the uniqueness of the counters is not guaranteed, and a coercer can instruct voters on how to vote for different counters.)

GENERAL NON-DETERMINISM. Even if the randomness used by the voting machine could be verified as having come from a truly random source, the non-deterministic nature of the machine output remains highly problematic. As was similarly noted of Neff’s scheme [40], non-determinism allows for subliminal channels in the encryptions [30]. An adversary can encode information in the ciphertext by simply trying new encryptions until a desired ciphertext is obtained. For example, an adversary might simply allow the parity of a random subset of the ciphertext bits to indicate a vote for democrat or republican. Such encoding would require only an expected 2 encryption attempts per ballot. As Karlof *et al.* suggest, an adversary could easily encode much more information by choosing ciphertexts according to characteristics of their hashes [30].

4 Our Approach to Removing Non-determinism

Benaloh’s voting scheme requires semantic security, which necessitates pseudorandomness. Many other cryptographic protocols need general randomness as well. Two natural ways to generate or obtain that randomness are through the voting machine or through the voter. However, as we have found, trusting the voting machine is problematic for several reasons. We cannot be guaranteed that the apparent randomness is actually random, and even if it were, non-determinism in the output creates potential subliminal channels.

On the other hand, the randomness could be obtained from the voter by asking him to enter a random number during the voting process, for example. This technique is also undesirable, however, since it places an unnecessary burden on the voter and is potentially confusing. Humans are also notoriously bad at choosing random numbers. As such, we observe a natural tension in our problem between a need for completely unpredictable values and a need for determinism. To address this, we adopt a distributed trust model, and aim at minimizing the effort of the voter.

We distribute trust among the candidate parties of the election and assume that each candidate party will provide a unique, private key, which can be used in the generation of pseudorandomness. As long as one party behaves correctly, the pseudorandomness generated will be computationally indistinguishable from random and verifiable by the voter. As a result of the trust placed in one of the many parties, we reduce the voter’s burden and keep the voting protocol simple.

As in Benaloh’s scheme, the voter in our scheme has the option to challenge the encryption of his vote. In addition to the decryption information provided in Benaloh’s scheme, our scheme provides the voter with a proof of correct, deterministic pseudorandomness. This, in turn ensures him a unique encryption of his ballot. In the following discussion, we describe our protocol at a

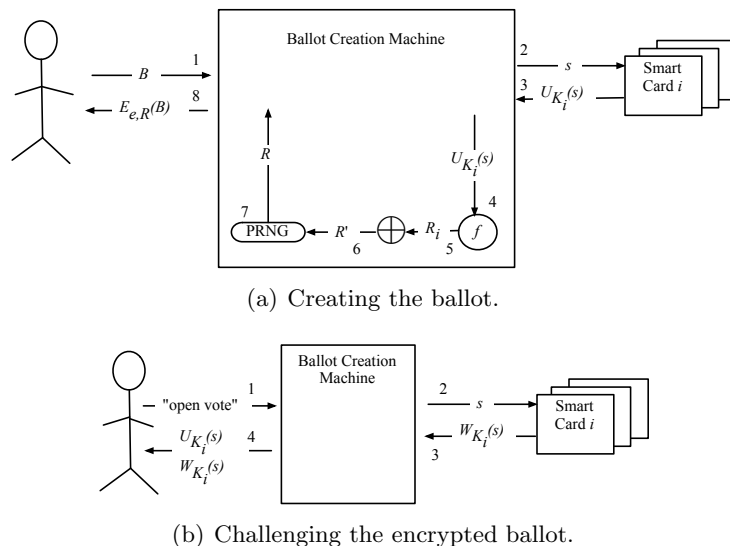


Figure 2: High level interactions between the voter, voting machine, and smart cards.

high level followed by two possible constructions.

4.1 High Level Protocol Overview

The high level steps of our protocol are illustrated in Figure 2. Ballot casting and tallying remain the same as in the Benaloh scheme, so we focus on the processes of ballot creation and challenging encrypted ballots. The basic idea behind our solution is to use a keyed, deterministic function to generate the pseudorandomness used for the encryptions, and when an encrypted ballot is challenged, the voter is given proof that allows her to verify that the randomness used in the encryption of the ballot was generated correctly.

Let n denote the number of candidate parties competing in the election. Each candidate party i generates a public-private key pair (v_i, K_i) prior to the election and programs K_i onto a smart card for each ballot creation machine.⁷ The public keys v_1, \dots, v_n are posted on the public bulletin board. Each smart card is capable of computing an unpredictable, deterministic function $U_{K_i}(\cdot)$, and a corresponding proof function $W_{K_i}(\cdot)$ that provides a form of witness to the output of U . The user has access to an associated verification function $\text{Ver}_{v_i}(\cdot, \cdot, \cdot)$ that verifies the correctness of outputs from U given the proof W . Let d be the private decryption key shared among trustees and e be the public encryption key on the voting machines.

At the beginning of the election day, poll workers install n smart cards corresponding to the n candidate parties on each ballot creation machine. These contain the private keys K_1, \dots, K_n . When a voter wishes to create a ballot B , the ballot creation machine queries each smart card $1, \dots, n$ with the same unique seed s (step 2 in Figure 2(a)). In return, each card computes and sends a deterministic unpredictable value $U_{K_i}(s)$ (step 3). The machine then uses the values $U_{K_i}(s)$, $i = 1, \dots, n$ as input to a public function f to generate numbers R_1, \dots, R_n , each of which is indistinguishable from random without knowledge of the corresponding K_i (step 4). Finally, R' is

⁷Smart cards allow key entry and storage such that the keys can never be read from the card.

created as $R' = \bigoplus_{i=1}^n R_i$ (where \oplus indicates exclusive-or), and a pseudorandom number generator (PRNG) is used to expand R' to $R = \text{PRNG}(R')$ of the desired size (steps 5 and 6). Since R' is computationally indistinguishable from random as long as at least one R_i is indistinguishable, R is as well. R is then used as the randomness for encrypting the ballot B (step 7). We will discuss possible methods for generating the seed s in Section 4.1.1 below.

As in Benaloh’s scheme, the voter can choose to either cast the created, encrypted ballot $E_{e,R}(B)$, or she may choose to challenge it. In the case where she decides to issue a challenge, the machine further queries each smart card, this time for proof values $W_{K_i}(s)$, $i = 1, \dots, n$ (steps 2 and 3 in Figure 2(b)). The machine prints both the values $U_{K_i}(s)$ and $W_{K_i}(s)$, $i = 1, \dots, n$ on the voter’s receipt (step 4).⁸ (In some cases U and W may actually be identical functions.) Each value $W_{K_i}(s)$, along with public verification key v_i , enables the voter to verify that the corresponding $U_{K_i}(s)$ was in fact correctly computed for seed s by computing $\text{Ver}_{v_i}(s, U_{K_i}(s), W_{K_i}(s))$. After verifying the correctness of each $U_{K_i}(s)$, the voter can lastly compute $R = \text{PRNG}(\bigoplus_{i=1}^n f(U_{K_i}(s)))$ and check that her ballot was properly encrypted using the random value R .

In summary, the voter can verify that the randomness was deterministically produced using the unique seed s . Our protocol thus eliminates the non-determinism in Benaloh’s scheme in exchange for trust in one of the n candidate parties.

4.1.1 Generating the Unique Seed s

A definitive, unique seed for each ballot is sufficient to ensure non-determinism and non-repetition of the “randomness” used to encrypt. This seed can be generated using a combination of the precinct number, the machine number within the precinct, and an additional input that is tied to the voter. We discuss several possibilities.

VOTER NAME. A fairly ideal, (mostly) unique, and invariable such input would be the voter’s name or other distinguishing information about the voter. However, unfortunately, any identifying information about the voter may allow the machine to increase its probability of successfully cheating and must be avoided. Even deterministic functions of the voter’s name, birth date, etc., such as a hash, can still allow the machine to obtain information about the voter’s identity since these deterministic constructs are not semantically secure.

VOTER CHALLENGE. Another option involves having the voter enter some sort of challenge, which could be used as a seed. We discussed previously, however, that a random challenge puts an unnecessary burden on the voter and it additionally introduces a channel that a coercer could exploit. For instance, the voter could vote exactly as the coercer does and also provide the same challenge, simply allowing the coercer to verify that the voter’s encrypted ballot is identical to his own.

TIME. While using time seems to provide sufficient uniqueness, it presents several challenges in verification and can result in a subliminal channel. Suppose time were used at a minute granularity, for instance. If a vote were cast at 2:02 pm, a voting machine could try to produce a desired encryption by submitting the seed as 2:01, 2:02, 2:03, and 2:04. It is unlikely that a voter will have enough certainty to complain about a 1 or 2 minute difference. This problem lessens as we increase the granularity by which time is used for the seed, but new problems are generated where voters who vote within the same short time frame may obtain encryptions that use the same “randomness”.

⁸In practice, these values could simply be put on the electronic bulletin board and omitted from the receipt.

BALLOT SERIAL NUMBER. We choose the ballot serial number published on the ballot for the unique seed. If produced honestly, this number is unique for each ballot created and consequently results in unique values of R . If two ballot serial numbers were not unique, however, and two voters vote in the exact same manner on the same machine, possibly upon being instructed to do so, then their resulting encryptions will be the same. This scenario, as mentioned previously, could be maliciously exploited to display only a single encryption on the bulletin board and stuff an additional ballot. To avoid this problem, we suggest that the names of voters are published along with each corresponding encrypted ballot and serial number. This practice has been used in previous schemes [3, 1] although it does allow a subset of people (the trustees sharing the decryption key) to potentially learn how every voter voted simply from the bulletin board, if dishonest.

4.2 A Direct Solution in the Standard Model

We now examine constructions by which verifiable pseudorandomness can be obtained using smart cards. We choose a solution that can leverage smart cards as a source of secure computation primarily because the cards are tamper resistant, readily available, and inexpensive (8-20 USD⁹).

The first solution we present utilizes programmable smart cards, such as JavaCards [37]. Of course, it could also be implemented with non-programmable, factory cards if the necessary cryptographic operations were made available on the cards. However, currently, such operations are not available. The solution is secure in the standard model.

4.2.1 Verifiable Random Functions

Several cryptographic primitives to provide verifiable pseudorandomness have been explored in the literature, namely verifiable random functions (VRFs) [36, 34, 16, 17]. Our first solution leverages the primitive of Dodis and Yampolskiy [17], mainly due to its overwhelming computational efficiency compared to other VRFs. We informally review the Dodis-Yampolskiy verifiable random function scheme below.

A verifiable random function scheme consists of 4 functions:

- $\text{KeyGen}(k)$: a probabilistic algorithm that takes a security parameter k as input and returns a secret key K and a public key v .
- $F_K(s)$: a deterministic function that takes a secret key K and a seed s and returns a pseudorandom value R , i.e. a value that is computationally indistinguishable from random to someone without knowledge of K or the proof $\pi = \text{Prove}_K(s)$ defined below.
- $\text{Prove}_K(s)$: a deterministic function that takes a secret key K and a seed s and returns a proof π that $F_K(s)$ is the correct, unique output of F for input seed s .
- $\text{Verify}_v(s, y, \pi)$: a probabilistic function that takes input seed s , supposed F_K function output y , and proof π , and returns YES if $y = F_K(s)$ and NO otherwise.

The Dodis-Yampolskiy scheme relies on the existence of groups \mathbb{G} and \mathbb{G}_1 such that an (admissible) bilinear map exists between the groups and the q -Diffie-Hellman inversion assumption (q -DHI) [38] and the q -decisional bilinear Diffie-Hellman inversion assumption (q -DBDHI) [8] hold. We briefly review each of these below:

⁹available at: <http://www.usasmartcard.com>

Definition 1 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 algorithm to compute $e(g_1, g_2) \quad \forall g_1, g_2 \in \mathbb{G}$.

Definition 2 The q-Diffie-Hellman inversion assumption (q-DHI) [38] for groups \mathbb{G} states that no polynomial time algorithm can compute $g^{1/x}$ given $g, g^x, \dots, g^{(x^q)}$.

Definition 3 The q-decisional bilinear Diffie-Hellman inversion assumption (q-DBDHI) [8] for groups \mathbb{G} and \mathbb{G}_1 with an admissible bilinear map e states that no polynomial time algorithm can distinguish $e(g, g)^{1/x}$ from random with probability greater than $1/2$ given $g, g^x, \dots, g^{(x^q)}$.

The scheme also requires the existence of a generator $g \in \mathbb{G}$ and a bijection $h : \mathbb{G}_1 \rightarrow \mathbb{H}$ where \mathbb{H} is the desired range of the pseudorandom function.

Certain groups over elliptic curves are believed to satisfy these properties with bilinear maps that can be constructed from the Weil or Tate pairings [17, 20, 29, 9].

The Dodis-Yampolskiy scheme is then constructed as follows:

- **KeyGen**(k): selects a random $r \in \mathbb{G}$ and sets the secret key to $K = r$ and the public key to $v = g^r$.
- $F_K(s)$: returns $e(g, g)^{1/(s+K)}$ as the VRF output.
- **Prove** $_K(s)$: returns $\pi_K(s) = g^{1/(s+K)}$ as a proof of correctness.
- **Verify** $_v(s, y, \pi)$: returns YES if $e(g^s v, \pi) = e(g, g)$ and $y = e(g, \pi)$ and returns NO otherwise.

We refer the reader to the original paper for the proofs that $F_K(s)$ is computationally indistinguishable from random and that **Verify** $_v(s, y, \pi)$ outputs YES if and only if $y = F_K(s)$ [17].

4.2.2 Application to Our Scheme

We now apply the Dodis-Yampolskiy VRF to our scheme.

To reiterate, aside from the standard semantically secure encryption function E , our scheme requires 4 functions:

- $U_{K_i}(s)$: a deterministic function that takes a seed s and key K_i and returns a computationally unpredictable value u .
- $W_{K_i}(s)$: a deterministic function that takes a seed s and key K_i and returns a value w that can be used as proof that $U_{K_i}(s)$ was computed correctly.
- $f(u)$: a deterministic function that takes an unpredictable value $u = U_K(s)$ for some key K and seed s and returns a value R' that is indistinguishable from random.

- $\text{Ver}_{v_i}(s, u, w)$: a function that takes a seed s , an alleged unpredictable output u from function U , verification proof value w , and public verification key v_i and returns YES if $u = U_{K_i}(s)$ and NO otherwise.

If we have each of these functions, then we have a deterministic source of pseudorandomness $f(U_{K_i}(s))$ with correctness that can be verified from proof $W_{K_i}(s)$ using Ver .

The application of the Dodis-Yampolskiy scheme follows almost immediately from these requirements. Each candidate party i first runs $\text{KeyGen}(k)$ to generate public and private keys v_i and K_i . We then define the functions as follows:

- $U_{K_i}(s)$ returns $F_{K_i}(s)$. Since $F_{K_i}(s)$ is indistinguishable from random, its value is also unpredictable.
- $W_{K_i}(s)$: returns $w = \text{Prove}_{K_i}(s) = \pi_K(s)$.
- $f(u)$: returns the value of the bijection $h(u)$. Since $u = U_{K_i}(s) = F_{K_i}(s)$ is indistinguishable from a random element in \mathbb{G} , $h(u)$ is indistinguishable from a random element in the range of h . (Recall that the range of h , \mathbb{H} , is a desirable range for the pseudorandom values.)
- $\text{Ver}_{v_i}(s, u, w)$: returns $\text{Verify}_{v_i}(s, u, w)$, which returns YES if and only if $u = F_{K_i}(s) = U_{K_i}(s)$.

The correctness and security of the construction follow directly from those of the Dodis-Yampolskiy VRF [17].

4.3 A Solution Using Standard ISO 7816 Smart Cards

The previous construction presented to provide verifiable randomness is direct and secure in the standard model. However, it requires programmable smart cards. In this section, we present a second solution that can be implemented using standard ISO 7816 compliant smart cards. It is secure in the random oracle model under the well known RSA assumption [44].

We base the construction on the classical, PKCS #1 RSA signature [33], a functionality listed in the ISO 7816-8 standard [19]. We choose the RSA signature over other signatures specified by the standard (such as DSA) because it is deterministic. Briefly, the signature scheme is defined by the 3 following functions:

- $\text{KeyGen}(k)$: chooses random, secret primes p and q and sets $N = pq$ to be the RSA modulus. Let $v = 2^{16} + 1$ be the verification key. The function sets $K \equiv v^{-1} \pmod{\phi(N)}$ and returns the public key (v, N) and private signing key K .
- $\text{Sign}_K(m)$: signs a message m by first computing an encoded message $M = H(m)$ where H is a public, collision resistant encoding function. (H may actually vary across versions of PKCS #1, but its details are not essential for our purposes.) It returns a signature using the classic formula: $\sigma \equiv M^K \pmod{N}$.
- $\text{Verify}_v(m, \sigma)$: verifies the authenticity of the signature σ on message m . It first computes $M = H(m)$. It then verifies that $\sigma^v \equiv M \pmod{N}$ and returns YES if so and NO otherwise.

Let \mathcal{O} be a random oracle [11, 2] $\mathcal{O} : \mathbb{Z}_N^* \rightarrow \mathbb{H}$ where \mathbb{H} is a desired output range of the pseudorandom function. Recall the 4 functions sufficient for a practical construction and their requirements as outlined in Section 4.2.2. We can now define them using the RSA signature functions:

- $U_{K_i}(s)$: returns RSA $\text{Sign}_{K_i}(s) = u$. Since the RSA signature is known to be unforgeable under the RSA assumption [44], this value is computationally unpredictable.
- $W_{K_i}(s)$: simply returns $U_{K_i}(s)$. In this case, the output of U is its own proof of correctness since anyone with the public key can verify the signature.
- $f(u)$: returns the value of the random oracle $\mathcal{O}(u)$. Because u is computationally unpredictable, $\mathcal{O}(u)$ is computationally indistinguishable from random.
- $\text{Ver}_{v_i}(s, u, w)$: returns NO if $u \geq N$ to ensure that equivalent values of $u \bmod N$ do not verify. Otherwise, the function returns $\text{Verify}_{v_i}(s, u)$, which returns YES if and only if $H(s) \equiv u^{v_i} \bmod N$, i.e. u is a valid signature on s . Because the function $g(s) = s^v \bmod N$ is a permutation, there is only one such $u \pmod n$ for which the function will return YES. Thus, verification ensures that u is the *unique*, correct output of $U_{K_i}(s)$.

Thus, this second approach also provides deterministic values $f(U_{K_i}(s))$ that are indistinguishable from random and that can be verified with provided proofs $W_{K_i}(s)$ using Ver .

4.4 Smart Card Security

To this point, we have focused primarily on protocol for obtaining verifiable pseudorandomness without discussion of the physical entities of our system or their implications on privacy. In this section, we briefly discuss possible approaches to mitigating a requirement of smart card security.

As many have mentioned, guaranteeing privacy in voting protocols with absolute certainty is extremely difficult in real world settings [4]. The challenges in securing the physical source of pseudorandomness in our protocol also exemplify this fact. Although secret keys cannot be read from smart cards, an adversary who can gain physical access to a card can repeatedly query it to obtain the pseudorandomness it generates. (Notice that such compromise does not affect the integrity of the tally.) We suggest several modifications to the full election procedure to mitigate this issue:

- Each party should use a different key for each machine. This ensures that even an adversary who obtains one card from each party can compromise the privacy of the votes cast on at most one machine. Also, the parties should not produce more than one card for any given machine to ensure limited availability.
- A large random number should be chosen the morning of election day to be used as part of each seed s to the verifiable random function. This prevents adversaries with access to the cards prior to election day from being able to make meaningful queries to them. Similarly, the cards could be destroyed immediately following the election in a process requiring several observers.

- The cards should require authentication from the ballot creation machine¹⁰ prior to responding to queries. Although this will not solve the situation where the adversary compromises the machine as well, it increases the level of compromise required by the adversary.

Although these suggestions make an adversary’s task significantly difficult, we leave the issue of smart card security as a partially open problem. We discuss further open problems that apply to cryptographic voting schemes more generally in Section 6.

5 Application to Other Schemes

While the primary focus of our discussion so far has been the application of our construction to Benaloh’s scheme [4], our techniques are general enough to be applied to other cryptographic voting protocols as well. Here we briefly discuss how these techniques for eliminating non-determinism, may be applied to the schemes by Neff [40] and Riva and Ta-Shma [43]. Chaum’s visual cryptographic scheme [13] is actually already deterministic, and we observe that our construction in the random oracle model is very similar to his technique of using a signature to generate randomness deterministically.

5.1 Neff’s Scheme

In Neff’s scheme [40], the voter is first presented with a list of n candidates from which to choose. After the voter conveys her selection $i \in [1, n]$ to the voting machine, the machine encrypts an array of bit-pairs for each of the n candidate choices. The bit-pairs are set in a way to encode her vote for candidate i , and ElGamal is used for the encryption of each bit. The correctness of this encoding can later be verified by the voter through a probabilistic interactive proof with the machine. In particular, the voter is prompted to choose one of the two encrypted bits of the encrypted bit pairs, which will be opened by the machine. The voter later verifies that the expected bit was properly encrypted.

Because ElGamal encryptions are randomized, Neff’s scheme is also vulnerable to subliminal channels, as was observed by Karlof *et al.* [30]. Our technique can be applied to Neff’s scheme to address this problem. Each bit encryption can be randomized with pseudorandomness deterministically obtained from the keyed smart cards. We can associate a unique public seed with each bit of each ballot to seed the verifiable random function. During the interactive proof step, each bit encryption that is opened should be accompanied with a proof that the “randomness” was correctly, deterministically formed. (Notice that this does not change the receipt since the proof can simply be included with the rest of the verification information on the electronic public bulletin board.)¹¹

5.2 Riva-Ta-Shma Scheme

In the scheme proposed by Riva and Ta-Shma [43], the voter prepares two encrypted ballots at home B_0, B_1 , which she brings with her to the polls on election day. The ballots are prepared such that each contains an unlabeled encrypted vote for each candidate $c_1 = E(v_1), \dots, c_n = E(v_n)$. (That

¹⁰using the ISO 7816 EXTERNAL-AUTHENTICATE command

¹¹The option in Neff’s scheme to let the voting machine randomly generate some of the challenges would also have to be removed to avoid other potential subliminal channels.

is the correspondence between encrypted votes and candidates is not indicated on the ballot.) In the precinct, the voter provides a proof that ballots are correctly formed. She also proves that she knows the correspondence between candidates and encrypted votes by opening the encryptions on one of the ballots B_a , where bit a is randomly chosen by the poll worker. The voter then chooses an encrypted vote c_i to be cast, corresponding to the candidate of her choice on the remaining ballot. The voting machine creates two secret re-encrypted permutations of the votes for each candidate $\{\mathcal{P}_0(c_1), \dots, \mathcal{P}_0(c_n)\}, \{\mathcal{P}_1(c_1), \dots, \mathcal{P}_1(c_n)\}$, and it states which re-encrypted vote corresponds to the voter’s selected vote c_i in each of the two permutations $r_0 = \mathcal{P}_0(c_i), r_1 = \mathcal{P}_1(c_i)$. The voter then chooses one of the re-encrypted permutations \mathcal{P}_b to be opened by the machine, and she verifies that re-encryption r_b did in fact re-encrypt her chosen encrypted vote c_i . The unopened $r_{\bar{b}}$ is then posted on the public bulletin board as the voter’s re-encrypted vote.

The Riva-Ta-Shma scheme differs from previous work in the sense that voters’ candidate choices are kept completely secret from the voting machine. Thus, subliminal channels cannot be exploited to directly reveal a voter’s choices. Unverifiable randomization is still problematic in the scheme, however. Secret re-encryptions of the ballots are necessary to prevent a coercer from mapping posted, re-encrypted votes to encrypted votes generated by voters. Such mapping enables coercion because voters can, of course, open any of their generated encrypted votes to a coercer. If the “randomness” used for re-encryptions were actually deterministic and known by a coercer (for example, by compromise of a machine), the re-encryptions would no longer be secret. Note that coercion, in this case, could occur as simply as a coercer providing printable ballots for voters online with instructions and then paying voters when appropriate values appeared on the public bulletin board.

Our technique can be applied to generate the pseudorandomness used for the re-encryptions and permutations. Each ordered encrypted vote of each ballot can be associated with a unique seed for the verifiable random function. A proof of correctly formed pseudorandomness can be provided for all the re-encryptions of the permutation that the voter requests to open. (The verifiable randomness technique could also be used to choose which of the voter’s original two ballots should be opened although this might also be done with something as simple as a coin flip.)

6 Conclusion and Future Work

We present a new approach for eliminating non-determinism from cryptographic voting protocols. Our technique provides voters with a probabilistic guarantee that deterministically generated pseudorandomness is used in the randomized operations of the protocol. We achieve this guarantee in exchange for trust in one of the many candidate parties competing in the election. We discuss ways that our technique may be applied to eliminate the non-determinism in protocols by Benaloh [3, 4], Neff [40] and Riva and Ta-Shma [43]. Our construction does not increase the burden on the voter in terms of her interaction with the machine. In general, it simply includes additional proof information to the bulletin board to be verified by voters.

As future work, we consider applying our scheme to reduce the demands of the voter in end-to-end voting schemes. Usability remains one of the greatest challenges in end-to-end schemes since even fairly simple tasks can be quite confusing to voters who are unfamiliar with the process. While Benaloh’s scheme requires very little of the user, it provides a significantly lower guarantee of accuracy than other schemes (depending on the number of people willing to challenge their ballots). We hope to explore the possibility of a verifiable system where the voter walks in and votes almost

as she does today. She simply receives a receipt to allow her to verify the correctness of her cast vote.

As some remaining open issues with end-to-end voting, we note that a voter can always sell her vote to an insider by simply agreeing to accept any value posted on the bulletin board in place of her vote. (This is not true of the Twin protocol suggested by Rivest and Warren [45].) Furthermore, in protocols such as Benaloh's, an overwhelming practical problem arises where voters have no way of proving when their votes have been maliciously encrypted. A cheated voter can simply claim that her intent was inaccurately encrypted. However, we can imagine that this complaint may be a fairly common event even in the case of completely honest machines. Continued work in the area of end-to-end voting schemes remains needed to address these problems before these schemes can become a practical solution to real elections.

Acknowledgments

This work was supported by the National Science Foundation grant CNS-0524252. We thank Susan Hohenberger and Dan Wallach for their insights on this work, and we also thank Josh Mason and Sam Small for their helpful discussions.

References

- [1] B. Adida and R. L. Rivest. Scratch and vote. In *WPES '06: Workshop on Privacy in the Electronic Society*, 2006.
- [2] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93: ACM Conference on Computer and Communications Security*, 1993.
- [3] J. Benaloh. Simple verifiable elections. In *EVT '06:USENIX/ACCURATE Electronic Voting Technology Workshop*, 2006.
- [4] J. Benaloh. Ballot casting assurance via voter-initiated poll station auditing. In *EVT '07:USENIX/ACCURATE Electronic Voting Technology Workshop*, 2007.
- [5] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *STOC '94: ACM Symposium on Theory of Computing*, 1994.
- [6] J. D. C. Benaloh. *Verifiable Secret-ballot Elections*. PhD thesis, Yale University, 1987.
- [7] M. Blaze, A. Cordero, S. Engle, C. Karlof, N. Sastry, M. Sherr, T. Stegers, and K.-P. Yee. Source code review of the Sequoia voting system. Technical report, California Secretary of State, July 2007.
- [8] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *EUROCRYPT '04: Advances in Cryptology*, 2004.
- [9] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO '01: Advances in Cryptology*, 2001.
- [10] J. A. Calandrino, A. J. Feldman, J. A. Halderman, D. Wagner, H. Yu, and W. P. Zeller. Source code review of the Diebold voting system. Technical report, California Secretary of State, July 2007.
- [11] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). In *STOC '98: ACM Symposium on the Theory of Computing*, 1998.
- [12] D. Chaum. Elections with unconditionally secret ballots and disruption equivalent to breaking RSA. In *EUROCRYPT '88: Advances in Cryptology*, 1988.

-
- [13] D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, 2004.
- [14] D. Chaum, P. Y. Ryan, and S. Schneider. A practical voter-verifiable election scheme. In *ESORICS '05: European Symposium on Research in Computer Security*, 2005.
- [15] L. F. Cranor and R. K. Cytron. Sensus: A security-conscious electronic polling system for the Internet. In *HICSS '97: Hawaii International Conference on System Sciences*, 1997.
- [16] Y. Dodis. Efficient construction of (distributed) verifiable random functions. In *PKC '03: Workshop on Theory and Practice of Public Key Cryptography*, 2003.
- [17] Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In *PKC '05: Workshop on Theory and Practice of Public Key Cryptography*, 2005.
- [18] A. J. Feldman, J. A. Halderman, and E. W. Felten. Security analysis of the Diebold AccuVote-TS voting machine. In *EVT '07: USENIX/ACCURATE Electronic Voting Technology Workshop*, 2007.
- [19] I. O. for Standardization. ISO/IEC 7816-8 standard, 2004.
- [20] S. D. Galbraith. Supersingular curves in cryptography. pages 495–513, 2001.
- [21] R. Gardner, A. Yasinsac, M. Bishop, T. Kohno, Z. Hartley, J. Kerski, D. Gainey, R. Walega, E. Hollander, and M. Gerke. Software review and security analysis of the Diebold voting machine software. Technical report, Florida Department of State, July 2007.
- [22] S. Goldwasser and S. Micali. Probabilistic encryption. *Computer and System Sciences*, 28(2), 1984.
- [23] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
- [24] P. Golle, M. Jakobsson, A. Juels, and P. F. Syverson. Universal re-encryption for mixnets. In *Topics in Cryptology - CT-RSA '04: Cryptographers' Track at RSA*, 2004.
- [25] H. Hursti. Diebold TSx evaluation: Critical security issues with Diebold TSx, May 2006. Available at <http://www.blackboxvoting.org/BBVreportIIunredacted.pdf>.
- [26] S. Inguva, E. Rescorla, H. Shacham, and D. S. Wallach. Source code review of the Hart InterCivic voting system. Technical report, California Secretary of State, July 2007.
- [27] M. Jakobsson and A. Juels. Millimix: Mixing in small batches. Technical Report 99-33, DIMACS, 1999.
- [28] M. Jakobsson, A. Juels, and R. L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *USENIX Security Symposium*, 2002.
- [29] A. Joux and K. Nguyen. Separating decision Diffie-Hellman from computational Diffie-Hellman in cryptographic groups. *Journal of Cryptography*, 16(4):239–247, 2001.
- [30] C. Karlof, N. Sastry, and D. Wagner. Cryptographic voting protocols: A systems perspective. In *USENIX Security Symposium*, 2005.
- [31] J. Kelsey, A. Regenscheid, T. Moran, and D. Chaum. Hacking paper some random attacks on paper-based e2e systems, 2007. Available at <http://kathrin.dagstuhl.de/files/Materials/07/07311/07311.KelseyJohn.Slides.pdf>.
- [32] T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach. Analysis of an electronic voting system. In *IEEE Symposium on Security and Privacy*, 2004.
- [33] R. Laboratories. PKCS #1 v2.1: RSA cryptography standard, 2002.
- [34] A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *CRYPTO '02: Advances in Cryptology*, 2002.

- [35] E. Magkos, M. Burmester, and V. Chrissikopoulos. Receipt-freeness in large-scale elections without untappable channels. In *I3E '01: IFIP Conference on Towards The E-Society*, 2001.
- [36] S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *FOCS '99: IEEE Symposium on Foundations of Computer Science*, 1999.
- [37] S. Microsystems. Java card technology. <http://java.sun.com/products/javacard/>.
- [38] S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. *IEICE Transaction Fundamentals*, E85-A(2):481–484, 2002.
- [39] A. Neff. Election confidence: A comparison of methodologies and their relative effectiveness at achieving it, 2003. Available at <http://www.votehere.com>.
- [40] A. Neff. Practical high certainty intent verification for encrypted votes, 2004. Available at <http://www.votehere.com/vhti/documentation>.
- [41] T. P. Pedersen. A threshold cryptosystem without a trusted party. In *EUROCRYPT '91: Advances in Cryptology*, 1991.
- [42] E. Proebstel, S. Riddle, F. Hsu, J. Cummins, F. Oakley, T. Stanionis, and M. Bishop. An analysis of the Hart Intercivic DAU eSlate. In *EVT '07: USENIX/ACCURATE Electronic Voting Technology Workshop*, 2007.
- [43] B. Riva and A. Ta-Shma. Bare-handed electronic voting with pre-processing. In *EVT '07: USENIX/ACCURATE Electronic Voting Technology Workshop*, 2007.
- [44] R. Rivest and B. Kaliski. RSA problem. In *Encyclopedia of Cryptography and Security (Kluwer)*, 2003.
- [45] R. L. Rivest and W. D. Smith. Three voting protocols: ThreeBallot, VAV, and Twin. In *EVT '07: USENIX/Accurate Electronic Voting Technology Workshop*, 2007.
- [46] P. Y. Ryan and T. Peacock. Prêt à Voter: A systems perspective. Technical report, University of Newcastle, 2005.
- [47] K. Sako and J. Kilian. Receipt-free mix-type voting scheme: A practical solution to the implementation of a voting boot. In *EUROCRYPT '95: Advances in Cryptology*, 1995.
- [48] D. Sandler, K. Derr, and D. S. Wallach. Votebox: A tamper-evident, verifiable electronic voting system. In *USENIX Security Symposium*, 2008.
- [49] A. C. Yao. Protocols for secure computations. In *FOCS '82: IEEE Symposium on Foundations of Computer Science*, 1982.