

Cryptographic Voting Protocols: A Systems Perspective

Chris Karlof Naveen Sastry David Wagner
{ckarlof, nks, daw}@cs.berkeley.edu
University of California, Berkeley

Abstract

Cryptographic voting protocols offer the promise of verifiable voting without needing to trust the integrity of any software in the system. However, these cryptographic protocols are only one part of a larger system composed of voting machines, software implementations, and election procedures, and we must analyze their security by considering the system in its entirety. In this paper, we analyze the security properties of two different cryptographic protocols, one proposed by Andrew Neff and another by David Chaum. We discovered several potential weaknesses in these voting protocols which only became apparent when considered in the context of an entire voting system. These weaknesses include: subliminal channels in the encrypted ballots, problems resulting from human unreliability in cryptographic protocols, and denial of service. These attacks could compromise election integrity, erode voter privacy, and enable vote coercion. Whether our attacks succeed or not will depend on how these ambiguities are resolved in a full implementation of a voting system, but we expect that a well designed implementation and deployment may be able to mitigate or even eliminate the impact of these weaknesses. However, these protocols must be analyzed in the context of a complete specification of the system and surrounding procedures before they are deployed in any large-scale public election.

1 Introduction

Democracies are built on their population’s consent, and a trustworthy voting system is crucial to this consent. Recently, “Direct Recording Electronic” voting machines (DREs) have come under fire for failing to meet this standard. The problem with paperless DREs is that the voting public has no good way to tell whether votes were recorded or counted correctly, and many experts have argued that, without other defenses, these systems are not trustworthy [15, 20].

Andrew Neff and David Chaum have recently proposed revolutionary schemes for DRE-based electronic voting [5, 23, 24]. The centerpiece of these schemes consists of novel and sophisticated cryptographic protocols that allow voters to verify their votes are cast and counted correctly. Voting companies Voteegrity and Vote-Here have implemented Chaum’s and Neff’s schemes, respectively. These schemes represent a significant advance over previous DRE-based voting systems: voters can verify that their votes have been accurately recorded, and everyone can verify that the tallying procedure is correct, preserving privacy and coercion resistance in the process. The ability for anyone to verify that votes are counted correctly is particularly exciting, as no prior system has offered this feature.

This paper presents a first step towards a security analysis of these schemes. Our goal is to determine whether these new DRE-based cryptographic voting systems are trustworthy for use in public elections. We approach this question from a systems perspective. Neff’s and Chaum’s schemes consist of the composition of many different cryptographic and security subsystems. Composing security mechanisms is not simple, since it can lead to subtle new vulnerabilities [10, 18, 25]. Consequently, it is not enough to simply analyze a protocol or subsystem in isolation, as some attacks only become apparent when looking at an entire system. Instead, we perform a whole-system security analysis.

In our analysis of these cryptographic schemes, we found two weaknesses: subliminal channels in the encrypted ballots and problems resulting from human unreliability in cryptographic protocols. These attacks could potentially compromise election integrity, erode voter privacy, and enable vote coercion. In addition, we found several detectable but unrecoverable denial of service attacks. We note that these weaknesses only became apparent when examining the system as a whole, underlining the importance of a security analysis that looks at cryptographic protocols in their larger systems context.

Weakness	Protocols	Threat Model	Affects
Random subliminal channels	Neff	Malicious DRE colluding with outsider	Voter privacy, coercion resistance
Semantic subliminal channels	Chaum	Malicious DRE colluding with outsider	Voter privacy, coercion resistance
Message reordering attacks	Neff	Malicious DRE and human error	Election integrity, public verifiability
Social engineering attacks	Neff, Chaum	Malicious DRE and human error	Election integrity, public verifiability
Discarded receipts	Neff, Chaum	Malicious DRE or bulletin board	Election integrity
Other human factor attacks	Neff, Chaum	Malicious DRE	Ability of voter to prove DRE is cheating
Denial of service attacks	Neff, Chaum	Malicious DRE or tallying software	Voter confidence, election integrity

Table 1: Summary of weaknesses we found in Neff’s and Chaum’s voting schemes.

The true severity of the weaknesses depends on how these schemes are finally implemented. During our security analysis, one challenge we had to deal with was the lack of a complete system to analyze. Although Neff and Chaum present fully specified cryptographic protocols, many implementation details—such as human interfaces, systems design, and election procedures—are not available for analysis. Given the underspecification, it is impossible to predict with any confidence what the practical impact of these weaknesses may be. Consequently, we are not yet ready to endorse these systems for widespread use in public elections. Still, we expect that it may be possible to mitigate some of these risks with procedural or technical defenses, and we present countermeasures for some of the weaknesses we found and identify some areas where further research is needed. Our results are summarized in Table 1.

2 Preliminaries

David Chaum and Andrew Neff have each proposed a cryptographic voting protocol for use in DRE machines [4, 5, 23, 24, 29]. Although these protocols differ in the details of their operation, they are structurally similar. Both protocols consist of four stages: *election initialization*, *ballot preparation*, *ballot tabulation*, and *election verification*.

Before the election, we select a set of *election trustees* with competing interests, chosen such that it is unlikely that all trustees will collude. During election initialization, the trustees interact amongst themselves before the election to choose parameters and produce key material used throughout the protocol. The trustees should represent a broad set of interest groups and governmental agencies to guarantee sufficient separation of privilege

and discourage collusion among the trustees.

Ballot preparation begins when a voter visits a polling station to cast her vote on election day, and ends when that ballot is cast. To cast her vote, the voter interacts with a DRE machine in a private voting booth to select her ballot choices. The DRE then produces an electronic ballot representing the voter’s choices and posts this to a public bulletin board. This public bulletin board serves as the ballot box. At the same time, the DRE interacts with the voter to provide a *receipt*. Receipts are designed to resist vote buying and coercion, and do not allow the voter to prove to a third party how she voted. Also, each voter’s ballot is assigned a unique *ballot sequence number* (BSN). BSNs ease auditing and verification procedures, without compromising voter privacy.

After all ballots have been posted to the bulletin board, the ballot tabulation stage begins. In ballot tabulation, the election trustees execute a publicly verifiable multistage mix net, where each trustee privately executes a particular stage of the mix net [12, 24]. To maintain anonymity, the trustees strip each ballot of its BSN before it enters the mix net. Each stage of the mix net takes as input a set of encrypted ballots, partially decrypts or re-encrypts them (depending on the style of mix net), and randomly permutes them. The final result of the mix net is a set of plaintext ballots which can be publicly counted but which cannot be linked to the encrypted ballots or to voter identities. In cryptographic voting protocols, the mix net is designed to be universally verifiable: the trustee provides a proof which any observer can use to confirm that the protocol has been followed correctly. This means a corrupt trustee cannot surreptitiously add, delete, or alter ballots.

At various points during this process, voters and observers may engage in election verification. After her

ballot has been recorded on the public bulletin board, the voter may use her receipt to verify her vote was cast as intended and will be accurately represented in the election results. Note that the receipt does not serve as an official record of the voter's selections; it is only intended for convincing the voter that her ballot was cast correctly. Election observers (e.g., the League of Women Voters) can verify certain properties about ballots on the public bulletin board, such as, that all ballots are well-formed or that the mix net procedure was performed correctly.

Both the Chaum and Neff protocols require DREs to contain special printing devices for providing receipts. The security requirements for the printer are: 1) the voter can inspect its output, and 2) neither the DRE nor the printer can erase, change, or overwrite anything already printed without the voter immediately detecting it. There are some differences in the tasks these devices perform and additional security requirements they must meet, which we will discuss later.

2.1 Security Goals

Neff's and Chaum's voting schemes strive to achieve the following goals:

Cast-as-intended: A voter's ballot on the bulletin board should accurately represent her choices.

Counted-as-cast: The final tally should be an accurate count of the ballots on the bulletin board.

Verifiability: The previous two properties should be verifiable. *Verifiably cast-as-intended* means each voter should be able to verify her ballot on the bulletin board accurately represents the vote she cast. *Verifiably counted-as-cast* means everyone should be able to verify that the final tally is an accurate count of the ballots contained on the bulletin board.

One voter/one vote: Ballots on the bulletin board should exactly represent the votes cast by legitimate voters. Malicious parties should not be able to add, duplicate, or delete ballots.

Coercion resistance: A voter should not be able to prove how she voted to a third party not present in the voting booth.

Privacy: Ballots should be secret.

2.2 Threat Models

We must consider a strong threat model for voting protocols. In national elections, billions of dollars are at stake, and even in local elections, controlling the appropriation

of municipal funding in a large city can be sufficient motivation to compromise significant portions of the election system [14]. We consider threats from three separate sources: DREs, talliers, and outside coercive parties. To make matters worse, malicious parties might collude together. For example, malicious DREs might collude with outside coercers to buy votes.

Malicious DREs can take many forms [3]. A programmer at the manufacturer could insert Trojan code, or a night janitor at the polling station could install malicious code the night before the election. We must assume malicious DREs behave arbitrarily. Verification of all the DRE software in an election is hard, and one goal of Neff's and Chaum's schemes is to eliminate the need to verify that the DRE software is free from Trojan horses.

We also must consider malicious parties in the tallying process, such as a malicious bulletin board or malicious trustees. These parties wield significant power, and can cause large problems if they are malicious. For example, if the bulletin board is malicious, it can erase all the ballots. If all the software used by the trustees is malicious, it could erase the private portions of the trustees' keys, making ballot decryption impossible.

To evaluate a voting system's coercion resistance, we must consider outside coercive parties colluding with malicious voters. We assume the coercer is not present in the voting booth. Attacks where the coercer is physically present are outside the scope of voting protocols and can only be countered with physical security mechanisms. Similarly, attacks where a voter records her actions in the poll booth (e.g., with a video or cell phone camera) are also outside the scope of voting protocols, and we do not consider them here.

Finally, we must consider honest but unreliable participants. For example, voters and poll workers might not fully understand the voting technology or utilize its verification properties, and a malicious party might be able to take advantage of this ignorance, apathy, or fallibility to affect the outcome of the election.

3 Two Voting Protocols

In this section, we describe Neff's and Chaum's voting protocols in detail.

3.1 Neff's Scheme

Andrew Neff has proposed a publicly verifiable cryptographic voting protocol for use in DREs [23, 24]. During election initialization, the trustees perform a distributed key generation protocol to compute a master public key; decryption will only be possible through the cooperation of all trustees in a threshold decryption operation. Also,



Figure 1: This is an example of a detailed receipt for Neff's scheme, taken from the VoteHere website, <http://www.votehere.com>.

there is a security parameter ℓ . A DRE can surreptitiously cheat with a probability of $2^{-\ell}$. Neff suggests $10 \leq \ell \leq 15$.

Neff's scheme is easily extensible to elections with multiple races, but for the sake of simplicity assume there is a single race with candidates C_1, \dots, C_n . After a voter communicates her choice C_i to the DRE, the DRE constructs an encrypted electronic ballot representing her choice and commits to it. Each ballot is assigned a unique BSN. The voter is then given the option of interacting with the DRE further to obtain a receipt. In Figure 1, we show an example of a receipt taken from the VoteHere website. This receipt enables the voter to verify with high probability that her vote is accurately represented in the tallying process.

After the voter communicates her intended choice C_i to the DRE, it constructs a *verifiable choice* (VC) for C_i . A VC is essentially an encrypted electronic ballot representing the voter's choice C_i (see Figure 2). A VC is a $n \times \ell$ matrix of *ballot mark pairs* (BMPs), one row per candidate (recall that ℓ is a security parameter). Each BMP is a pair of El Gamal ciphertexts. Each ciphertext is an encryption of 0 or 1 under the trustees' joint public key, written $\boxed{0}$ or $\boxed{1}$ for short. Thus, each BMP is a pair $\boxed{b_1} \boxed{b_2}$, an encryption of (b_1, b_2) .

The format of the plaintexts in the BMPs differs between the row corresponding to the chosen candidate C_i (i.e., row i) and the other ("unchosen") rows. Every BMP in row i should take the form $\boxed{0} \boxed{0}$ or $\boxed{1} \boxed{1}$. In contrast, the BMPs in the unchosen rows should be of the form $\boxed{0} \boxed{1}$ or $\boxed{1} \boxed{0}$. Any other configuration is an

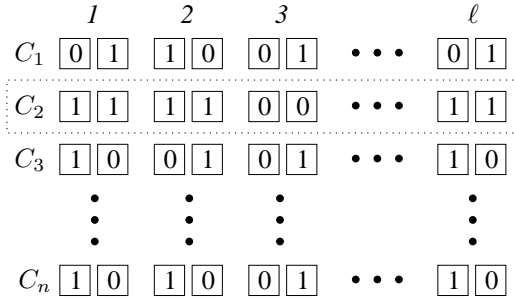


Figure 2: A verifiable choice (VC) in Neff's scheme. \boxed{b} represents an encryption of bit b . This VC represents a choice of candidate C_2 . Note the second row contains encryptions of $(0, 0)$ and $(1, 1)$, and the unchosen rows contain encryptions of $(0, 1)$ and $(1, 0)$.

indication of a cheating or malfunctioning DRE. More precisely, there is a $n \times \ell$ matrix x so that the k -th BMP in unchosen row j is $\boxed{x_{j,k}} \boxed{\sim x_{j,k}}$, and the k -th BMP in the choice row i is $\boxed{x_{i,k}} \boxed{x_{i,k}}$.

Consider the idealized scenario where all DREs are honest. The trustees can tally the votes by decrypting each ballot and looking for the one row consisting of $(0, 0)$ and $(1, 1)$ plaintexts. If decrypted row i consists of $(0, 0)$ and $(1, 1)$ pairs, then the trustees count the ballot as a vote for candidate C_i .¹

In the real world, we must consider cheating DREs. Up to this point in the protocol, the DRE has constructed a VC supposedly representing the voter's choice C_i , but the voter has no assurance this VC accurately represents her vote. How can we detect a dishonest DRE?

Neff's scheme prints the pair (BSN, $\text{hash}(VC)$) on the receipt and then splits verification into two parts: 1) at the polling booth, the DRE will provide an interactive proof of correct construction of the VC to the voter; 2) later, the voter can compare her receipt to what is posted on the bulletin board to verify that her ballot will be properly counted. At a minimum, this interactive protocol should convince the voter that row i (corresponding to her intended selection) does indeed contain a set of BMPs that will be interpreted during tallying as a vote for C_i , or in other words, each BMP in her chosen row is of the form $\boxed{b} \boxed{b}$. Neff introduces a simple protocol for this: for each such BMP, the DRE provides a *pledge* bit p ; then the voter randomly selects the left or right position and asks the DRE to provide a proof that the ciphertext in that position indeed decrypts to p ; and the DRE does so by revealing the randomness used in the encryption. Here we are viewing the ciphertext \boxed{b} as a commitment

¹This is a simplified view of how the trustees tally votes in Neff's scheme, but it captures the main idea.

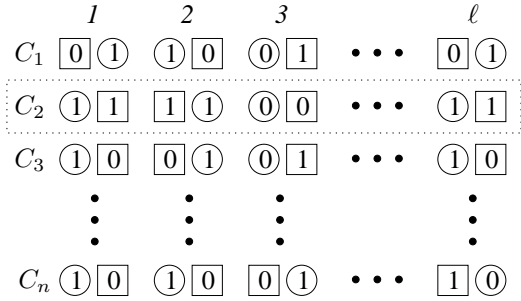


Figure 3: An opened verifiable choice (OVC) in Neff’s scheme. \boxed{b} represents an encryption of bit b , and \textcircled{b} represents an opened encryption of bit b . An opened encryption of b contains both b and the randomness ω used to encrypt b in the VC.

to b , and \boxed{b} is *opened* by revealing b along with the randomness used during encryption. If this BMP has been correctly formed as $\boxed{b} \boxed{b}$, the DRE can always convince the voter by using the value b as a pledge; however, if the BMP contains either $\boxed{0} \boxed{1}$ or $\boxed{1} \boxed{0}$, the voter has a $\frac{1}{2}$ probability of detecting this. By repeating the protocol for each of the ℓ BMPs in row i , the probability that a malformed row escapes detection is reduced to $(\frac{1}{2})^\ell$. The role of the interactive protocol is to ensure that the receipt will be convincing for the person who was in the voting booth but useless to anyone else.

In practice, it is unrealistic to assume the average voter will be able to parse the VC and carry out this protocol unassisted within the polling station. Instead, Neff’s scheme enables the voter to execute it later with the assistance of a trusted software program. The DRE first prints the pledges on the receipt, and then receives and prints the voter’s challenge. The challenge c_i for the row i is represented as a bit string where the k -th bit equal to 0 means open the left element of the k -th BMP and 1 means open the right element.

The DRE then constructs an *opened verifiable choice* (OVC) according to the voter’s challenge and submits it to the bulletin board. In Figure 3, we show an example of an OVC constructed from the VC in Figure 2. We represent an opened encryption of bit b in an half-opened BMP by \textcircled{b} . In the OVC, the opened BMPs in row i are opened according to c_i , so that each half-opened BMP contains a pair of the form $\textcircled{b} \boxed{b'}$ (if $c_{i,k} = 0$) or $\boxed{b} \textcircled{b'}$ (if $c_{i,k} = 1$). To ensure that the OVC does not reveal which candidate was selected, the BMPs in the unchosen rows are also half-opened. In unchosen row j , the DRE selects an ℓ -bit challenge c_j uniformly at random and then opens this row according to c_j . Thus, an OVC consists of an $n \times \ell$ matrix of half-opened BMPs. Consequently, the usual invocation of the receipt formation

protocol is as follows:

1. Voter \rightarrow DRE : i
2. DRE \rightarrow Printer : BSN, hash(VC)
3. DRE \rightarrow Printer : $\text{commit}(p_1, \dots, p_n)$
4. Voter \rightarrow DRE : c_i
5. DRE \rightarrow Printer : c_1, \dots, c_n
6. DRE \rightarrow B. Board : OVC

Here we define $p_{i,k} = x_{i,k}$ and $p_{j,k} = x_{j,k} \oplus c_{j,k}$ ($j \neq i$). While at the voting booth, the voter only has to check that the challenge c_i she specified does indeed appear on the printed receipt in the i -th position (i.e., next to the name of her selected candidate). Later, the voter can check that the OVC printed in step 5 does appear on the bulletin board and matches the hash printed in step 2 (and that the candidates’ names are printed in the correct order), and that the OVC contains valid openings of all the values pledged to in step 3 in the locations indicated by the challenges printed in step 5. Note that the VC can be reconstructed from the OVC, so there is no need to print the VC on the receipt or to post it on the bulletin board.

To prevent vote buying and coercion, the voter is optionally allowed to specify challenges for the unchosen rows between steps 2 and 3, overriding the DRE’s default random selection of c_j ($j \neq i$). If this were omitted, a vote buyer could tell the voter in advance to vote for candidate C_i and to use some fixed value for the challenge c_i , and the voter could later prove how she voted by presenting a receipt with this prespecified value appearing as the i -th challenge.

After the election is closed, the trustees apply a universally verifiable mix net to the collection of posted ballots. Neff has designed a mix net for El Gamal pairs [21, 24], and it is used here.

In VoteHere’s implementation of Neff’s scheme, voters are given the option of taking either a *detailed* or *basic* receipt. The detailed receipt contains all the information described in this section (Figure 1), but a basic receipt contains only the pair (BSN, hash(VC)). This decision is made separately for each race on a ballot, and for each race that a voter selects a detailed receipt she must independently choose the choice and unchosen challenges for that race.

A basic receipt affords a voter only limited verification capabilities. Since a basic receipt foregoes the pledge/challenge stage of Neff’s scheme, a voter cannot verify her ballot was recorded accurately. However, a basic receipt does have some value. It enables the voter to verify that the ballot the DRE committed to in the poll booth is the same one that appears on the bulletin board. Since the DRE must commit to the VC before it knows whether the voter wants a detailed or basic receipt, a DRE committing a VC that does not accurately represent the voter’s selection is risking detection if the

1. Voter \rightarrow DRE : i
2. DRE \rightarrow Printer : BSN, hash(VC)
3. DRE \rightarrow Voter : basic or detailed?
4. Voter \rightarrow DRE : r , where $r \in \{\text{basic}, \text{detailed}\}$
- 5a. DRE \rightarrow Printer : $\text{commit}(p_1, \dots, p_n)$
- 5b. Voter \rightarrow DRE : c_i
- 5c. DRE \rightarrow Printer : c_1, \dots, c_n
6. DRE \rightarrow B. Board : OVC

Figure 4: Summary of receipt generation in Neff’s scheme with the option of basic or detailed receipts. Steps 5a, 5b, and 5c happen only if $r = \text{detailed}$.

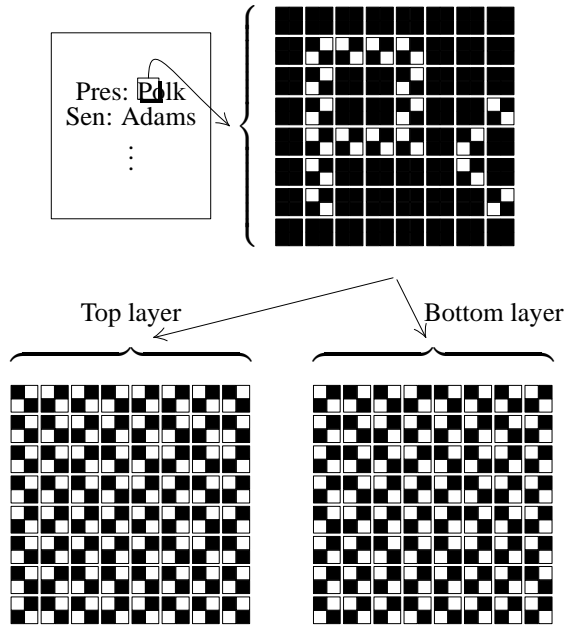


Figure 5: Representation of the printed ballot and transparencies. The top two images show the ballot as well as a zoomed in portion of the two overlaid transparencies portrayed below.

voter chooses a detailed receipt. The receipt protocol augmented with this additional choice is summarized in Figure 4.

3.2 Chaum’s Visual Crypto Scheme

David Chaum uses a two-layer receipt based on transparent sheets for his verifiable voting scheme [4, 5, 29]. A voter interacts with a DRE machine to generate a ballot image \mathcal{B} that represents the voter’s choices. The DRE then prints a special image on each transparency layer. The ballot bitmaps are constructed so that overlaying the top and bottom transparencies (T and B) reveals the voter’s original ballot image. On its own, however, each

Encoding for Transparency	1:	0:	
Encoding for Overlay	$\hat{1}$:	$\hat{0}$:	or

\oplus_v Truth Table	$0 \oplus_v 1 = \hat{1}$		\oplus_v		$=$	
	$0 \oplus_v 0 = \hat{0}$		\oplus_v		$=$	
	$1 \oplus_v 1 = \hat{0}$		\oplus_v		$=$	
	$1 \oplus_v 0 = \hat{1}$		\oplus_v		$=$	

Figure 6: Visual Cryptography. A printed pixel on a single transparency has a value in $\{0, 1\}$, encoded as shown in the first row. We apply the visual xor operator \oplus_v by stacking two transparencies so that light can shine through areas where the subpixels are clear. The pixels in the overlay take values from $\{\hat{0}, \hat{1}\}$. The bottom table shows the truth table for the visual xor operator and its parallels to the binary xor operator.

layer is indistinguishable from a random dot image and therefore reveals nothing about the voter’s choices (see Figure 5).



The DRE prints cryptographic material on each layer so that the trustees can recover the original ballot image during the tabulation phase. The voter selects either the top or bottom layer, and keeps it as her receipt. A copy of the retained layer is posted on the bulletin board, and the other layer is destroyed. The voter can later verify the integrity of their receipt by checking that it appears on the bulletin board and that the cryptographic material is well formed.

Visual cryptography exploits the physical properties of transparencies to allow humans to compute the xor of two quantities without relying on untrusted software. Each transparency is composed of a uniform grid of *pixels*. Pixels are square and take values in $\{0, 1\}$. We print for a 0-valued pixel and for a 1-valued pixel. We refer to each of the four smaller squares within a pixel as *subpixels*. Overlaying two transparencies allows light to shine through only in locations where both subpixels are clear, and the above encoding exploits this so that overlaying performs a sort of xor operation. Pixels in the overlay take values in $\{\hat{0}, \hat{1}\}$. Pixels in the overlay have a different appearance than those in the individual transparency layer: $\hat{0}$ appears as or , while $\hat{1}$ appears as . Using \oplus_v to represent the visual overlay operation, we see that $0 \oplus_v 0 = \hat{0}$, $0 \oplus_v 1 = \hat{1}$, and in general if $a \oplus b = c$ then $a \oplus_v b = \hat{c}$ (see Figure 6).

Chaum’s protocol satisfies three properties:

1. **Visual Check:** Given the desired ballot image \mathcal{B} , the DRE must produce two transparencies T and B so that $T \oplus_v B = \mathcal{B}$. This property allows the voter to verify the correct formation of the two transparencies.

2. **Recovery:** Given a single transparency T or B and the trustee keys, it must be possible to recover the original ballot image \mathcal{B} .
3. **Integrity:** T and B contain a commitment. There is a way to open T or B and to verify the opening so that for all other top and bottom pairs T' and B' such that $T' \oplus_v B' \approx \mathcal{B}$ and T' (or B') does not decrypt to \mathcal{B} , then B' (or T') is unopenable. In other words, for a pair of transparencies that overlay to form \mathcal{B} (or a close enough approximation for the voter to accept it as \mathcal{B}), the DRE should only be able to generate a witness for a transparency if the other transparency decrypts to \mathcal{B} .

We will consider each pixel to have a type $\in \{\boxed{P}, \boxed{E}\}$ in addition to its value $\in \{0, 1\}$. The pixel's type will determine how we compute the value. We label pixels on the transparency so that no pixels of the same type are adjacent to each other, forming a repeating grid of alternating pixel types. Additionally, when the two transparencies are stacked, we require that \boxed{P} -pixels are only atop \boxed{E} -pixels and \boxed{E} -pixels are only atop \boxed{P} -pixels. The upper left corner of the top transparency looks like:  and the upper left corner of the bottom transparency looks like: . The \boxed{P} -pixels in a layer come from a pseudorandom stream. The stream is composed of n separate streams, one from each trustee. Each of these trustee streams is based on the trustee number and the voter's BSN; the seed will be encrypted using each trustee's public key requiring the trustee to participate in the decryption process. The value of the \boxed{E} -pixel is set so that overlaying it with the corresponding \boxed{P} -pixel in the other layer yields a ballot pixel. An \boxed{E} -pixel alone reveals no information: it is the xor of a \boxed{P} -pixel and the ballot image.

3.2.1 Details on Transparency Formation

Next, we present the details on transparency formation for the interested reader. This section may be safely skipped on first reading.

The pseudorandom stream for a given transparency is composed of n pseudorandom streams, each of which is seeded by a different value. For each of the top and bottom transparencies, there is one stream per trustee. The i^{th} trustee's seed for the top is

$$st_i \triangleq h(\text{sign}_{k_t}(\text{BSN}), i) \quad (1)$$

where BSN represents the unique ballot sequence number assigned to the voter and $\text{sign}_{k_t}(\cdot)$ is a signature using

k_t , a key specific to the DRE, and $h(\cdot)$ is a hash function. The i^{th} trustee's seed for the bottom is

$$sb_i \triangleq h(\text{sign}_{k_b}(\text{BSN}), i) \quad (2)$$

The hash expansion function $h'(\cdot)$ is used to generate the trustee stream. Trustee streams are xored together to produce the pseudorandom stream for the top layer:

$$PT \triangleq \bigoplus_{i=1}^n h'(st_i) \quad (3)$$

The corresponding bottom stream uses the bottom seeds:

$$PB \triangleq \bigoplus_{i=1}^n h'(sb_i) \quad (4)$$

We can now define each pixel's value. We view the ballot as a stream of pixels \mathcal{B} , and $\mathcal{B}[i]$ denotes the i^{th} pixel. A \boxed{P} -pixel i on the top transparency is assigned the value $PT[i]$. The \boxed{E} -pixel i on the bottom transparency is defined to have value $PT[i] \oplus \mathcal{B}[i]$. When viewing the two transparencies in alignment, then, the voter sees the original ballot stream \mathcal{B} because $PT[i] \oplus_v (PT[i] \oplus \mathcal{B}[i]) = PT[i] \oplus (PT[i] \oplus \mathcal{B}[i]) = \mathcal{B}[i]$. When taken alone, neither transparency reveals any information since each pixel is either pseudorandomly generated or the xor of a pseudorandom quantity and the original ballot.

After constructing the two layers, the DRE appends an onion encryption of the seeds so the trustees can jointly recover PT or PB . The DRE adds

$$\begin{aligned} DB &\triangleq e_{k_n}(sb_n || e_{k_{n-1}}(\dots || e_{k_2}(sb_2 || e_{k_1}(sb_1)))) \\ DT &\triangleq e_{k_n}(st_n || e_{k_{n-1}}(\dots || e_{k_2}(st_2 || e_{k_1}(st_1)))) \end{aligned} \quad (5)$$

to each transparency. DT and DB are known as dolls. $e_{k_i}(\cdot)$ is a public-key encryption function that uses the i^{th} trustee's public key, k_i .

The voter is then presented a choice to either choose the top or bottom transparency as a receipt. After the voter chooses a receipt layer, the DRE appends signatures committing to the voter's and its choices. Without loss of generality, assume the voter keeps the top transparency as a receipt. The DRE then prints $\text{sign}_{k_t}(\text{BSN})$ as an opening for the top layer (see the integrity requirement of the previous section). This opening allows the voter to verify that the DRE properly formed st_i and that the DRE printed the \boxed{P} -pixels on the chosen layer as it should. By recreating the onion encryption, the voter can verify that DT is properly formed. Finally, the DRE appends a copy of the chosen layer to the bulletin board. We show a summary of Chaum's protocol in Figure 7.

When the voter performs these checks, a malicious DRE has only a $1/2$ chance of evading detection. By

1. Voter \rightarrow DRE : candidate choices
2. DRE \rightarrow Printer : transparency images
3. DRE \rightarrow Printer : BSN, DB , DT
4. Voter \rightarrow Printer : c where $c \in \{\text{top, bottom}\}$
5. DRE \rightarrow Printer : $\text{sign}_{k_c}(\text{BSN})$,
 $\text{sign}_{k_{\text{DRE}}}(\text{BSN}, DT, DB, \text{chosen transparency})$

Figure 7: Summary of Chaum’s protocol.

extension, its chance of changing a significant number of ballots without being caught is exponentially small. For instance, a DRE can cheat by forming the \boxed{P} -pixels incorrectly so the voter will see what they expect in the overlay yet the ballot will decrypt to some other image. However, the voter will detect cheating if her receipt transparency contains incorrectly formed \boxed{P} -pixels. Therefore, a malicious DRE must commit to cheating on either the top or bottom transparency (not both, or else it will surely be caught) and hope the voter does not choose that layer as a receipt.

3.2.2 Tabulation & Verification

Chaum uses a Jakobsson et al. style mix net to decode the transparency chosen by the voter and recover their choices from \mathcal{B} in the tallying phase [12]. The values of the pseudorandom pixels do not contain any information, while the encrypted pixels contain the ballot image xor-ed with the pseudorandom pixels from the other transparency. For each ballot that a trustee in the mix net receives, trustee i in the mix net recovers its portion of the pseudorandom stream. Let’s assume the voter chose a top transparency. In the case, trustee i will first decrypt the doll provided by the DRE (Equation (5)) to obtain sb_i and then xor $h'(sb_i)$ into the \boxed{E} -pixels in the encrypted ballot. This trustee next permutes all of the modified ballots and passes the collection to the next trustee. When the ballots exit the mix net, the \boxed{P} -pixels still contain pseudorandom data, but the encrypted pixels will contain the voter’s ballot pixels from \mathcal{B} .

4 Subliminal Channels

Subliminal channels, also known as covert communication channels, arise in electronic ballots when there are multiple valid representations of a voter’s choices. If the DRE can choose which representation to submit to the bulletin board, then the choice of the representation can serve as a subliminal channel. Subliminal channels are particularly powerful because of the use of public bulletin boards in voting protocols. A subliminal channel in

ballots on the bulletin board could be read by anyone (if the decoding algorithm is public) or only by a select few (if the decoding algorithm is secret).

A subliminal channel in an encrypted ballot carrying the voter’s choices and identifying information about the voter threatens voter privacy and enables vote coercion. For example, as Keller et al. note, a DRE could embed in each encrypted ballot the time when the ballot was cast and who the voter chose for president [13]. Then, a malicious observer present in the polling place could record when each person voted and later correlate that with the data stored in the subliminal channel to recover each person’s vote. Alternatively, if a malicious poll worker learns a voter’s BSN, she can learn how a person voted since each encrypted ballot includes the BSN in plaintext. Detecting such attacks can be quite difficult: without specific knowledge of how to decode the subliminal channel, the encrypted ballots may look completely normal. The difficulty of detection, combined with the enormous number of voters who could be affected by such an attack, makes the subliminal channel threat troubling.

The above scenarios illustrate how an adversary can authentically learn how someone voted. Coercion then becomes simple: the coercer requires the voter to reveal their BSN or the time at which they voted, then later verifies whether there exists a ballot with that identifying information and the desired votes.

The threat model we consider for subliminal channel attacks is a malicious DRE colluding with an external party. For example, a malicious programmer could introduce Trojan code into DREs and then sell instructions on how to access the subliminal channel to a coercer.

Neither Neff’s nor Chaum’s protocol completely address subliminal channels in ballots. In this section, we present subliminal channel vulnerabilities in these protocols and some possible mitigation strategies.

One interesting observation is that subliminal channels are a new problem created by these protocols. Subliminal channels only become a serious problem because the bulletin board’s contents are published for all to see. Since all the ballots are public and anonymously accessible, decoding the channel does not require any special access to the ballots. Subliminal channels are not a significant problem with current non-cryptographic DREs because electronic ballots are not public.

4.1 Randomness

Several cryptographic primitives in Neff’s scheme require random values, and subliminal channel vulnerabilities arise if a malicious DRE is free to choose these random values.² These primitives use randomness to

²Chaum’s scheme, as originally published, does not specify which encryption primitives should be used to construct the onion encryp-

achieve semantic security [8], a strong notion of security for encryption schemes which guarantees that it is infeasible for adversaries to infer even partial information about the messages being encrypted (except maybe their length). Each choice for the random number allows a different valid ballot, which creates opportunities for subliminal channels.

Subliminal channels are easy to build in protocols or encryption schemes that use randomness. If a cryptographic protocol requests the DRE to choose a random number r and then publish it, the DRE can encode $|r|$ bits through judicious selection of r . Alternatively, given any randomized encryption scheme $e_k(\cdot, \cdot)$, the DRE can hide a bit b in an encryption of a message m by computing $c = e_k(m, r)$ repeatedly using a new random number r each time until the least significant bit of $h(c)$ is b . More generally, a malicious DRE can use this technique to hide ℓ bits in c with expected $O(2^\ell)$ work. Thus, all randomized encryption schemes contain subliminal channels.

Random subliminal channel attack. Neff’s scheme uses randomness extensively. Each BMP consists of a pair of El Gamal ciphertexts, and the El Gamal encryptions are randomized. In forming the OVC, the DRE reveals half of the random values ω used in the encryptions (Figure 3).

For each BMP, one of the encryption pairs will be opened, revealing the random encryption parameter ω . This presents a subliminal channel opportunity.³ Although the DRE must commit to the ballot before the voter chooses which side of the BMP to open, a malicious DRE can still embed $|\omega|$ bits of data for each BMP by using the same ω for both encryptions in the BMP. In this way ω is guaranteed to be revealed in the ballot.

This attack enables a high bandwidth subliminal channel in each voter’s encrypted ballot. For example, in an election with 8 races and 5 candidates per race, there will be $40 \cdot \ell$ ballot mark pairs, where Neff suggests $\ell \geq 10$. A reasonable value of $|\omega|$ is 1024 bits. The total channel, then, can carry 128 bytes in each of the 400 BMPs, for a total of 51200 bytes of information per ballot. This is more than enough to leak the voter’s choices and identifying information about the voter.

tion in Equation 5 [5]. Subsequently, Chaum has related to us that he intended the encryption to use a deterministic encryption scheme [6] precisely to avoid using random values and the associated subliminal channel vulnerability. There is some risk in using this non-standard construction since the widely accepted minimum notion of security for public key encryption is IND-CPA, which requires a source of randomness.

³Another way a malicious DRE could embed a subliminal channel in Neff’s scheme is if the voter doesn’t choose all her unchoice challenges (i.e., the DRE is free to choose some of them). However, Neff outlines a variant of his proposal that solves this using two printers [23].

4.2 Mitigating Random Subliminal Channels

Eschew randomness. One approach to prevent subliminal channels is to design protocols that don’t require randomness. Designing secure protocols that do not use randomness is tricky, since so many proven cryptographic primitives rely on randomness for their security. Proposals relying on innovative uses of deterministic primitives, including Chaum’s, deserve extra attention to ensure that forgoing randomness does not introduce any security vulnerabilities. Ideally, they would be accompanied by a proof of security.

Random tapes and their implementation. In a personal communication, Neff suggested that DREs could be provided with pre-generated tapes containing the random bits to use for all of their non-deterministic choices, instead of allowing them to choose their own randomness [22]. With a random tape for each BSN, the ballot becomes a deterministic function of the voter’s choices and the random tape for that BSN. As long as the BSN is assigned externally before the voter selects her candidates, the ballots will be uniquely represented. This will eliminate the threat of random subliminal channels in encrypted ballots.

It is not enough for the intended computation to be deterministic; it must be verifiably so. Thus, we need a way to verify that the DRE has used the bits specified on the random tape, not some other bits. We present one possible approach to this problem using zero-knowledge (ZK) proofs [9] which allows everyone to verify that each DRE constructed ballots using the random numbers from its tape. We imagine that there are several optimizations to this approach which improve efficiency.

Suppose before the election, the trustees generate a series $r_{s,1}, r_{s,2}, \dots$ of random values for each BSN s , and post commitments $C(r_{s,1}), C(r_{s,2}), \dots$ on a public bulletin board. The election officials then load the random values $r_{s,1}, r_{s,2}, \dots$ on the DRE which will use BSN s .

During the election, for each randomized function evaluation $f(r, \cdot)$, the DRE uses the next random value in the series and furnishes a ZK proof proving it used the next random value in the series. For example, in Neff’s scheme, along with each \boxed{b} , which is an El Gamal encryption $e(r, b)$, the DRE includes a non-interactive zero knowledge proof of knowledge proving that 1) it knows a value $r_{s,i}$ which is a valid opening of the commitment $C(r_{s,i})$ and 2) $e(r_{s,i}, b) = \boxed{b}$. Verifying that each $r_{s,i}$ is used sequentially within a ballot enables any observer to verify that the encryption is deterministic, so there can be no random subliminal channels in \boxed{b} or its opening \textcircled{b} .

However, there is a wrinkle to the above solution: under most schemes, constructing the zero-knowledge proof itself requires randomness, which creates its own opportunities of subliminal channels. It may be possible to determinize the ZK proof using research on unique zero-knowledge proofs (uniZK) [16, 17].

This approach may require further analysis to determine whether it is able to satisfy the necessary security properties.

Trusted hardware. Utilizing trusted hardware in DREs can also help eliminate subliminal channels. In this approach, the trusted hardware performs all computations that require random inputs and signs the encrypted ballot it generates. The signature enables everyone to verify the ballot was generated inside the trusted hardware. As long as trustees verify the DRE’s trusted hardware is running the correct software and the trusted hardware isn’t compromised, DREs will not be able to embed a random subliminal channel.

4.3 Multiple Visual and Semantic Representations

A tabulator that accepts multiple equivalent visual or semantic representations of the voter’s choice creates another subliminal channel opportunity. For example, if the tabulator accepts both James Polk and James K. Polk as the same person, then a DRE can choose which version to print based on the subliminal channel bit it wants to embed.

Semantic subliminal channel attack. Chaum’s scheme is vulnerable to multiple visual representations. A malicious DRE can create alternate ballot images for the same candidate that a voter will be unlikely to detect. Recall that Chaum’s scheme encrypts an image of the ballot, and not an ASCII version of the voter’s choices. The voter examines two transparencies together to ensure that the resulting image accurately represents their vote. A DRE could choose to use different fonts to embed subliminal channel information; the choice of font is the subliminal channel. To embed a higher bandwidth subliminal channel, the DRE could make minor modifications to the pixels of the ballot image that do not affect its legibility. Unless the voter is exceptionally fastidious, these minor deviations would escape scrutiny as the voter verifies the receipt. After mixing, the subliminal channel information would be present in the resulting plaintext ballots.

There is no computational cost for the DRE to embed a bit of information in the font. It can use a simple policy, such as toggling a pixel at the top of a character to encode

a one, and a pixel at the bottom to encode a zero. On a 10 race ballot, using such a policy just once per word could embed 30 bits of information.

There is a qualitative difference between the semantic subliminal channels and the random subliminal channels. The information in the semantic channels will only become apparent after the mix net decrypts the ballot since the channel is embedded in the plaintext of the ballot. In contrast, the random subliminal channels leak information when the ballots are made available on the bulletin board.

Mitigation. To prevent the semantic subliminal channel attack, election officials must establish official unambiguous formats for ballots, and must check all ballots for conformance to this approved format. Any deviation indicates a ballot produced by a malicious DRE. Such non-conforming ballots should not be allowed to appear on the bulletin board, since posting even a single suspicious ballot on the bulletin board could compromise the privacy of all voters who used that DRE. Unfortunately, the redaction of such deviant ballots means that such ballots will not be able to be verified by the voter through normal channels.

An even more serious problem is that this policy violates assumptions made by the mix net. One would need to ensure the mix net security properties still hold when a subset of the plaintexts are never released.

The order in which ballots appear will also need to be standardized. Otherwise, a DRE can choose a specific ordering of ballots on the public bulletin board as a low bandwidth subliminal channel [15]. Fortunately, it is easy to sort or otherwise canonicalize the order of ballots before posting them publicly.

4.4 Discussion

Subliminal channels pose troubling privacy and voter coercion risks. In the presence of such attacks, we are barely better off than if we had simply posted the plaintext ballots on the bulletin board in unencrypted form for all to see. The primary difference is that subliminal channel data may be readable only by the malicious parties. This situation seems problematic, and we urge protocol designers to design voting schemes that are provably and verifiably free of subliminal channels.

5 Human Unreliability in Crypto Protocols

Previous studies have shown that non-cryptographers have a limited understanding of cryptography and how to use it [28, 30]. In these voting protocols, we are not just asking humans to use cryptography, but asking them to

become an active participant in a cryptographic protocol. Participating in an interactive cryptographic protocol is tricky and error-prone, and humans may not notice if the DRE makes subtle deviations from the protocol which dramatically affect security. To make matters worse, the voter has no trustworthy computer to aid her; she can only rely on a potentially compromised DRE.

The security of Neff’s and Chaum’s schemes relies on 1) the DRE not knowing how the voter will make future decisions, 2) the interactions between the DRE and voter happening in a particular order, and 3) the voter carefully monitoring the DRE’s output. For example, in Chaum’s scheme, the DRE must print (i.e., commit) the transparencies and dolls DB and DT before the voter chooses which transparency to take. If the DRE knows which transparency the voter will select before it commits to the dolls and transparency image, it can construct the dolls in a way such that the ballot will decrypt to an arbitrary image (see Section 3.2.1).

In this section we show three types of attacks which leverage human ignorance of the intricacies of cryptographic protocols. First, we present *message reordering attacks* and *social engineering attacks*, which enable DREs to undetectably replace voters’ ballots with votes of its own choosing. Next, we discuss attacks which take advantage of voters who apathetically discard their receipts at the polling station. Finally, we show attacks where a voter may be able to detect wrongdoing but cannot authoritatively prove DRE misbehavior to election officials. We follow with some mitigation strategies.

5.1 Message Reordering and Social Engineering Attacks

Message reordering attacks. The security of cryptographic protocols implicitly relies on the participants detecting any reordering of the messages in the protocol. With human participants, this assumption is not necessarily valid; if the deviation is minor, the average voter may not notice. In a *message reordering attack*, a malicious DRE attempts to cheat voters by slightly reordering the steps of the cryptographic voting protocol.

VoteHere’s implementation of Neff’s scheme is vulnerable to message reordering attacks. Recall that VoteHere’s implementation of Neff’s scheme gives voters the option of a basic or detailed receipt after it has committed to the ballot construction (Figure 4). Now suppose that the DRE reorders steps 2, 3, and 4 in Figure 4 in the

following way:

1. Voter \rightarrow DRE : i
3. DRE \rightarrow Voter : basic or detailed?
4. Voter \rightarrow DRE : r , where $r \in \{\text{basic, detailed}\}$
2. DRE \rightarrow Printer : BSN, hash(VC)
- 5a. DRE \rightarrow Printer : commit(p_1, \dots, p_n)
- 5b. Voter \rightarrow DRE : c_i
- 5c. DRE \rightarrow Printer : c_1, \dots, c_n
6. DRE \rightarrow B. Board : OVC

The change to the voter’s experience is probably minor, but the advantage afforded the DRE is large—the DRE learns whether the voter wants a basic or detailed receipt before it must commit to the ballot construction VC . With a basic receipt, a voter gives up her right to later verify her ballot is actually a vote for the desired candidate. A basic receipt only contains (BSN, hash(VC)). This is generally fine—if the DRE doesn’t know whether a voter wants a basic or detailed receipt until after it commits to the VC, it risks detection if the voter chooses a detailed receipt and it constructed a VC for a different candidate. But if the DRE learns the voter wants a basic receipt before it commits to the VC, it can undetectably submit a ballot for any candidate. If the voter wants a detailed receipt, the DRE just behaves honestly and properly constructs the VC.

By making a slightly more noticeable change to the voter experience, a DRE can cheat even if the voter chooses a detailed receipt. Consider the following message reordering attack when a voter chooses a detailed receipt (again, altered from Figure 4):

1. Voter \rightarrow DRE : i
3. DRE \rightarrow Voter : basic or detailed?
4. Voter \rightarrow DRE : detailed
- 5b. Voter \rightarrow DRE : c_i
2. DRE \rightarrow Printer : BSN, hash(VC)
- 5a. DRE \rightarrow Printer : commit(p_1, \dots, p_n)
- 5c. DRE \rightarrow Printer : c_1, \dots, c_n
6. DRE \rightarrow B. Board : OVC

In this attack, the DRE gets all the information from the voter before it prints or commits to anything. The DRE can now undetectably construct the VC to represent a vote for any candidate. Since the DRE knows the voter’s choice challenge c_i before it constructs the VC, it can construct the VC where row i is an unchosen row and select another row as the choice row. Since c_i determines how row i of the VC is opened, the DRE can construct the VC such that the row i of the OVC is consistent with the challenge c_i . This means when the voter checks her ballot later on the bulletin board, it will verify correctly even though it is actually a vote for another candidate.

Social engineering attacks. In a social engineering attack, an adversary attempts to fool a victim into voluntarily revealing a secret [2]. Social engineering is usually discussed in the context of passwords. For example, an adversary posing as tech support calls the CEO of a company to “verify” some things about her computer and tells her that her password is needed to do this. We present two social engineering attacks where the DRE fools the voter into revealing her future actions.

For our first attack, consider Chaum’s scheme. As we see in the summary of Chaum’s protocol in Figure 7, the voter relays her choice of layer to the printer, and not the DRE. The DRE cannot cheat after it commits to the transparency image, BSN, DB , and DT . But nothing prevents a DRE from simply asking the voter which layer she will choose in advance and then hoping that she actually chooses the layer she said she would. In this social engineering attack, the DRE slightly alters the protocol:

1. Voter \rightarrow DRE : candidate choices
- *. DRE \rightarrow Voter : top or bottom?
- *. Voter \rightarrow DRE : c_* , where $c_* \in \{\text{top, bottom}\}$
2. DRE \rightarrow Printer : transparency images
3. DRE \rightarrow Printer : BSN, DB , DT
4. Voter \rightarrow Printer : c , where $c \in \{\text{top, bottom}\}$
5. DRE \rightarrow Printer : $\text{sign}_{k_c}(\text{BSN})$,
 $\text{sign}_{k_{\text{DRE}}}(\text{BSN}, DT, DB, \text{chosen transparency})$

where * represents added steps. In step 5, the DRE can construct the doll DB or DT (depending on the voter’s response c_*) and the two layers such that the ballot decrypts to an arbitrary image of the DRE’s choosing (see Section 3.2.1). This attack is successful and undetectable as long as $c_* = c$ (i.e., the voter doesn’t change her mind or make a mistake). With a sternly worded message, the DRE can strongly encourage the voter to honor her original choice.

Our second attack succeeds since most non-cryptographers probably do not realize that multiple executions of a cryptographic protocol should be independent. For example, random choices in separate executions should be freshly generated. In Neff’s scheme, suppose a DRE triggers a reboot after it learns a voter’s random challenge c_i for her chosen row, and then restarts the protocol, feigning an error. If the voter chooses the same challenge c_i again, the DRE can undetectably forge a ballot for a different candidate by constructing row i as an unchosen row consistent with the challenge c_i . If the voter happens to choose a different challenge, the DRE can escape detection by rebooting again and then behaving normally.

5.2 Discarded Receipts

The security of Neff’s and Chaum’s scheme relies on voters using their receipts to verify their ballots. If an adversary can determine that certain ballots will not be verified, she can undetectably alter or replace these ballots. We expect some voters will be apathetic about the verification process and discard their receipts. Without her receipt, it is unlikely a voter will verify her ballot on the bulletin board. A malicious poll worker could collect receipts discarded in or near the polling station and tell a malicious DRE the BSNs of ballots which are safe to replace or alter.

5.3 Other Human Factor Attacks

In this section, we present other human factor attacks where a voter may be able to detect wrongdoing but cannot authoritatively prove DRE misbehavior to election officials.

Generating an invalid signature. Both Neff’s and Chaum’s scheme require the DRE to sign receipts it produces. The primary purpose of signed receipts is to prevent voters from falsely claiming fraud. The concern is that some voter might try to cast doubt on the election results by forging a receipt to frame a DRE. If after verifying her ballot on the bulletin board a voter claims the DRE cheated her, she must produce a receipt with a valid signature to prove it. Thus, the main purpose of signing receipt is not to prevent voters against cheating DREs, but to prevent election officials against misbehaving voters.

However, signatures create problems for honest voters interacting with malicious DREs. If at some point the DRE realizes it will be caught cheating when the voter later verifies her ballot on the bulletin board, the DRE can produce a receipt with an invalid signature. Although the voter can detect she was cheated, she cannot prove this to the election authorities. The signature on her receipt is invalid, and for all they know, might be forged.

Printing the wrong DRE Machine ID. For auditing purposes, receipts in Neff’s and Chaum’s schemes contain the DRE’s machine ID. However, we cannot trust a malicious DRE to print its correct identifier, since using a false identifier will shift suspicion. For example, a DRE may generate bogus signatures and use an identifier from an honest machine. When election officials receive fraud complaints from voters with the invalid receipts containing the honest machine’s identifier, the legitimate votes from the honest DRE might be called into question. Ultimately, this casts suspicion on the entire election.

Ignoring voter input. DREs can ignore voter input in an attempt to forge ballots. For example, in Neff’s scheme, if a DRE wants to forge a VC for a candidate of its own choosing, it might ignore the voter’s challenges and instead use and print challenges consistent with the forged VC. An observant voter will be able to clearly detect that this has occurred, since the challenges printed on the receipt will not match what the voter typed in. However, it might be difficult for the voter to prove to a third party that the DRE cheated in this way. First, if a voter wants to try to demonstrate the misbehavior to a poll worker, she might have to reveal her voting intentions. Second, if the DRE only ignores voter input every 50 voters, subsequent attempts to replicate the misbehavior will fail.

5.4 Mitigation Strategies

Message reordering and social engineering attacks. Message reordering and social engineering attacks have a high chance of succeeding because the average human voter doesn’t fully understand the importance of message ordering in cryptographic protocols or how a commitment scheme works. Also, humans may be forgiving of or not notice slight deviations from the “canonical” voting experience. Even if the voter notices that the DRE does not print something exactly when it should have, she might ignore the deviation if the rest of the voting experience goes smoothly and all the numbers on her receipt “match up”.

Since message reordering and social engineering attacks only involve slight modifications to a voter’s interaction with the DRE, the only defense we foresee is ultimately unsatisfying: parallel testing. To resist these attacks, election officials must audit DREs throughout the day to verify they do not even slightly deviate from the canonical behavior. Note that pre-election auditing is not sufficient since there are many ways that a malicious DRE could arrange to cheat only on election day. However, live testing of DREs on election day can be risky. For example, we must ensure that a malicious DRE cannot submit audit votes from testers as legitimate votes.

Auditing for message reordering and social engineering attacks is difficult. It requires auditors to be intimately familiar with the details of DRE operation. The changes in the voting experience these attacks impose might appear completely innocuous to a naive observer. Also, to evade random auditing, a clever DRE may choose to only launch message reordering and social engineering attacks intermittently.

Voter education may help mitigate these attacks. Voter education can effectively turn every voter into an auditor. However, if the system requires too many instructions and checks, voters may worry that if they make a mis-

take, they will be cheated. Others may reject the system on principle, arguing that it should not be hard to correctly use a voting system.

Discarded receipts. To address the problems with discarded receipts, we must educate voters that receipts are valuable and should not be frivolously discarded in a public place.

Other human factor attacks. One approach to the problem of machines generating invalid signatures is to have voters verify the signature on their receipts at the polling station. However, since voters cannot verify signatures on their own, this approach requires another set of hardware devices and software that voters must trust. Also, it is not obvious how the signature verification devices receive the public keys from the DREs in the polling station in a trustworthy manner. If the device loads them directly from the DREs, a malicious DRE may give a different public key to the verification device than the one it was loaded with.

DREs cannot be trusted to reliably print their machine IDs on receipts. The best solution is to require receipts and transparencies to be preprinted with machine IDs and loaded onto the corresponding machines at the polling stations. This solution prevents malicious DREs from lying about their machine ID, but it creates logistical hassles. Blank paper and transparency stock is no longer generic—a particular roll can only be used by one machine.

One possible defense against DREs that ignore voter input is parallel testing of live machines on election day. However, as previously discussed, parallel testing has limitations.

6 Denial of Service Attacks and Election Recovery

Although Neff’s and Chaum’s schemes can detect many attacks, recovering legitimate election results in the face of these attacks may be difficult. In this section, we present several detectable but irrecoverable denial of service (DoS) attacks launched at different stages of the voting and tallying process. We consider attacks launched by malicious DREs and attacks launched by malicious tallying software, and discuss different recovery mechanisms to resist these attacks.

6.1 Denial of Service (DoS) Attacks

Launched by malicious DREs. Malicious DREs can launch several DoS attacks which create detectable, but

unrecoverable situations. We present two classes of attacks: ballot deletion and ballot stuffing.

In a ballot deletion attack, a malicious DRE erases voters' ballots or submits random bits in their place. Election officials and voters can detect this attack after the close of polls, but there is little they can do at that point. Since the electronic copy serves as the only record of the election, it is impossible to recover the legitimate ballots voted on that DRE.

DREs can launch more subtle DoS attacks using ballot stuffing. Recall that both Neff's and Chaum's schemes use ballot sequence numbers (BSNs) to uniquely identify ballots. BSNs enable voters to find and verify their ballots on the public bulletin board, and by keeping track of the set of valid BSNs, election officials can track and audit ballots.

In the *BSN duplication attack*, a DRE submits multiple ballots with the same BSN. Election officials will be able to detect this attack after the ballots reach the bulletin board, but recovery is difficult. It is not clear how to count ballots with the same BSN. Suppose a DRE submits 100 valid ballots (i.e., from actual voters) and 100 additional ballots, using the same BSN for all the ballots. How do talliers distinguish the invalid ballots from the valid ones?

In the *BSN stealing attack*, a malicious DRE "steals" BSNs from the set of BSNs it would normally assign to legitimate voters' ballots. For a particular voter, the DRE might submit a vote of its own choosing for the BSN it is supposed to use, and on the voter's receipt print a different (invalid) BSN. Since the voter will not find her ballot on the bulletin board, this attack can be detected, but recovery is tricky: how do election officials identify the injected ballots and remove them from the tally?

Neff's and Chaum's scheme enable voters and/or election officials to detect these attacks, but recovery is non-trivial because 1) the voters' legitimate ballots are missing and 2) it is hard to identify the invalid ballots injected by the DRE.

Launched by malicious tallying software. DoS attacks in the tallying phase can completely ruin an election. For example, malicious tallying softwares can delete the trustees' keys, making decryption and tallying of the encrypted ballots forever impossible. Malicious bulletin board software can erase, insert, or delete ballots.

Selective DoS. An attacker could use DoS attacks to bias the outcome of the election. Rather than ruining the election no matter its outcome, a more subtle adversary might decide whether to mount a DoS attack or not based on who seems to be winning the race. If the adversary's preferred candidate is winning, the adversary need

do nothing. Otherwise, the adversary might try to disrupt or ruin the election, forcing a re-election and giving her preferred candidate a second chance to win the election, or at least raising questions about the winner's mandate and reducing voters' confidence in the process.

There are many ways that selective DoS attacks might be mounted:

- If an outsider has a control channel to malicious DREs, the outsider could look at the polls and communicate a DoS command to the DREs.
- An autonomous DRE could look at the pattern of votes cast during the day, and fail (deleting all votes cast so far at that DRE) if that pattern leans towards the undesired candidate. This would disrupt votes cast only in precincts leaning against the attacker's preferred candidate.
- If trustees' software is malicious, it could collude to see how the election will turn out, then cause DoS if the result is undesirable. Note that if all trustees are running the same tallying software, this attack would require only a single corrupted programmer.

Selective DoS attacks are perhaps the most troubling kind of DoS attack, because they threaten election integrity and because attackers may have a real motive to launch them.

6.2 Mitigation Strategies and Election Recovery

Note that in all these attacks, non-malicious hardware or software failures could cause the same problems. This may make it hard to distinguish purposeful attacks from unintentional failures.

The above attacks create irrecoverable situations because voters' legitimate ballots are lost or corrupted, the bulletin board contains unidentifiable illegitimate ballots submitted by malicious DREs, or both. In this section, we evaluate two recovery mechanisms for these DoS attacks: *revoting* and a *voter verified paper audit trail*.

Revoting. One recovery strategy is to allow cheated voters to revote. Depending on the scope of the attack or failure, this could range from allowing only particular voters to revote to completely scrapping the election and starting over. However, revoting is problematic. Redoing the entire election is the most costly countermeasure. Alternatively, election officials could allow only those voters who have detected cheating to revote. Unfortunately, this is insufficient. Less observant voters who were cheated may not come forward, and it may be hard to identify and remove illegitimate ballots added by

a malicious DRE. Revoting does not help with selective DoS.

Voter verified paper audit trail. A voter verified paper audit trail (VVPAT) system produces a paper record verified by the voter before her electronic ballot is cast [19]. This paper record is cast into a ballot box. The paper trail is an official record of the voter’s vote but is primarily intended for use in recounts and auditing.

It would not be hard to equip cryptographic voting systems with a VVPAT. This would provide a viable mechanism for recovering from DoS attacks. In addition to providing an independent record of all votes cast, VVPAT enables recovery at different granularities. If election officials conclude the entire electronic record is questionable, then the entire VVPAT can be counted. Alternatively, if only a single precinct’s electronic record is suspect, then this precinct’s VVPAT record can be counted in conjunction with the other precincts’ electronic records. This approach enables officials to keep the universal verifiability of the uncorrupted precincts while recovering the legitimate record of the corrupted precinct.

A third benefit of VVPAT is that it provides an independent way to audit that the cryptography is correctly functioning. This would be one way to help all voters, even those who do not understand the mathematics of these cryptographic schemes, to be confident that their vote will be counted correctly.

7 Implementing Secure Cryptographic Voting Protocols

A secure implementation of Neff and Chaum’s protocol will still need to resolve many issues. In this section, we outline important areas that Neff and Chaum have not yet specified. These parts of the system need to be fully designed, implemented, and specified before one can perform a comprehensive security review. Also, we list three open research problems which we feel are important to the viability of these schemes.

7.1 Underspecifications

Bulletin board. Both protocols rely on a public bulletin board to provide anonymous, read only access to the data. The data must be stored robustly, overcoming software and mechanical failures as well as malicious attacks. Further, only authenticated parties should be able to append messages to the bulletin board. An additional requirement is to ensure that the system delivers the same copy of the bulletin board contents to each reader. If the bulletin board were able to discern a voter’s identity, say

by IP address, it could make sure the voter always saw a mix transcript that included a proof that their vote was counted. But, for the official transcript, the mix net and bulletin board could collude to omit the voter’s ballot. In this scenario, the voter would think her vote had been counted but in reality it was not.

Neff and Chaum have not yet elaborated on a proposed bulletin board architecture or the properties they require. We imagine that the principles of distributed storage systems, such as Farsite, CFS, or OceanStore [1, 7, 27], might be applicable in the bulletin board setting. However, without a further specification of exactly which architecture would be used, we cannot evaluate the system’s security.

BSN assignment. Neff’s and Chaum’s schemes do not specify how to assign BSNs to voters’ ballots. BSNs could be assigned externally by a smartcard initializer which authorizes a voter to use a DRE, or be assigned by DREs, say by a monotonically increasing counter prefixed by the DRE machine ID.⁴ Clever BSN assignment combined with careful auditing and sign-in procedures could help limit the scope of some of the DoS attacks in Section 6, but since DREs can always erase or corrupt a voter’s electronic ballot after she casts it, we still must consider recovery mechanisms.

User interfaces. The attacks presented in Section 5 require a malicious programmer to modify the DRE’s software and present a different user interface from the correct version. A related attack would be a malicious DRE neglecting to present a valid candidate to the voter.

Clever user interfaces may be able to overcome such attacks, by making it plainly obvious to the voter that things are amiss. We don’t know of any user interface specification or architecture for Neff’s or Chaum’s voting system.

Tallying software. Both Neff’s and Chaum’s schemes treat the tallying software as a black box. We surmise, that it, too, has stringent requirements on its correct implementation. If all trustees use tallying software from a single source, then this software might collude without the trustees’ knowledge and invalidate the system’s integrity guarantees. Though n -version programming might be able to counter this threat, it makes software development very expensive and requires detailed interface specifications to ensure that all versions of the software will interoperate. We have not seen any details on how to ensure that the tallying software cannot collude.

⁴David Chaum later conveyed to us that he intended his scheme to use a counter to assign BSNs [6].

7.2 Open Research Problems

Subliminal channels. Developing cryptographic protocols that address subliminal channels would help resist privacy and coercion attacks. Subliminal channels in the ballots subvert the confidentiality guarantees provided by encryption. We present some techniques in Section 4 to eliminate subliminal channels in encrypted ballots, but we believe this is still an area for future research.

Mix net security models. We would like to see a definition of security for mix nets that is comprehensive for the voting setting. Such a definition must be natural enough to inspire confidence that it is the correct model. For instance, Jakobsson illustrates a subtle privacy violation if the encryption used in the mixes do not provide non-malleability [11], and others have shown similar results [26]. This illustrates the importance and non-triviality of formulating a correct security model for mix nets. We believe the security of cryptographic voting systems would benefit from a thorough study of the relationship between the mix net requirements and those of the rest of the system.

Humans as protocol participants. These voting protocols require voters to not just use a cryptographic system, but also to participate in a cryptographic protocol. Cryptographic protocols are fragile to deviations and mistakes in their implementation, and humans have been known to make mistakes. A high level understanding of the protocol is not sufficient; to minimize errors, voters often need to understand how the protocol works. Alternatively, voting protocols must be designed to be as foolproof as possible to “faulty” implementations in the average voter. Voter education could help, but this raises an important human-computer interaction problem: how do we educate voters about these issues without discouraging them that these systems are too complicated to securely use?

8 Conclusion

We laud Neff’s and Chaum’s ambitious goal: developing a coercion free, privacy preserving voter-verifiable election system. Their systems represent a significant security improvement over current DRE-based paperless systems. Neff’s and Chaum’s schemes also strive to limit reliance on trusted software and hardware. Most notably, these schemes do not require voters to trust DREs since voters can detect malicious behavior.

Neff’s and Chaum’s schemes are fully specified at the cryptographic protocol level, but they are underspecified from the systems and human interaction level. Due in part to this underspecification, we have discovered a

number of potential weaknesses which only became apparent when considered in the context of an entire voting system. We expect that a well designed implementation and deployment may be able to mitigate or even eliminate the impact of these weaknesses.

We found solutions for some of these weaknesses, but we also identified new challenges and open problems for electronic voting systems. First, subliminal channels have the potential to erode voter privacy and enable voter coercion. Any system that uses a public bulletin board must ensure that the ballots it posts have a unique representation. Second, these voting protocols present a new research challenge by placing human voters directly within an interactive cryptographic protocol. Protocol designers have previously assumed participants are infallible computer agents, but voting protocols must cope with human error and ignorance.

Despite these challenges, we are optimistic about the future prospects of these voting systems.

Acknowledgments

We would like to thank Andrew Neff and David Chaum for helping us understand their voting protocols. Joe Hall, David Molnar, Rob Johnson, Umesh Shankar, and Monica Chew gave invaluable feedback on earlier drafts of this work. This work was supported in part by the NSF under grant CCR-0093337, by the Knight Foundation under a subcontract through the Caltech/MIT Voting Technology Project, and by the US Postal Service.

References

- [1] Atul Adya, William Bolosky, Miguel Castro, Gerald Ceramak, Ronnie Chaiken, John Douceur, Jon Howell, Jacob Lorch, Marvin Theimer, and Roger Wattenhofer. FAR-SITE: Federated, available, and reliable storage for a incompletely trusted environment. In *5th Symposium on Operating System Design and Implementation (OSDI)*, pages 1–14, December 2002.
- [2] Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*, chapter 3, “Passwords”, pages 35–50. John Wiley and Sons, Inc., 2001.
- [3] Jonathan Bannet, David W. Price, Algis Rudys, Justin Singer, and Dan S. Wallach. Hack-a-vote: Demonstrating security issues with electronic voting systems. *IEEE Security and Privacy Magazine*, 2(1):32–37, Jan./Feb. 2004.
- [4] Jeremy Bryans and Peter Ryan. A dependability analysis of the Chaum digital voting scheme. Technical Report CS-TR-809, University of Newcastle upon Tyne, July 2003.
- [5] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy Magazine*, 2(1):38–47, Jan.–Feb. 2004.
- [6] David Chaum, February 2005. Personal Communication.

- [7] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, pages 202–215, October 2001.
- [8] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.
- [9] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. In *SIAM Journal on Computing*, volume 18, pages 270–299, 1984.
- [10] Nevin Heintze and J. D. Tygar. A model for secure protocols and their compositions. *Software Engineering*, 22(1):16–30, January 1996.
- [11] Markus Jakobsson. A practical mix. In *Advances in Cryptology – EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 448–461. Springer-Verlag, May/June 1998.
- [12] Markus Jakobsson, Ari Juels, and Ronald Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *11th USENIX Security Symposium*, pages 339–353, August 2002.
- [13] Arthur Keller, David Mertz, Joseph Hall, and Arnold Urkin. Privacy issues in an electronic voting machine. In *ACM Workshop on Privacy in the Electronic Society*, pages 33–34, October 2004. Full paper available at <http://www.sims.berkeley.edu/~jhall/papers/>.
- [14] Paul Kocker and Bruce Schneier. Insider risks in elections. *Communications of the ACM*, 47(7):104, July 2004.
- [15] Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach. Analysis of an electronic voting system. In *IEEE Symposium on Security and Privacy*, pages 27–40, May 2004.
- [16] Matt Lepinski, Silvio Micali, and abhi shelat. Collusion-free protocols. In *Proceedings of the 37th ACM Symposium on Theory of Computing*, May 2005.
- [17] Matt Lepinski, Silvio Micali, and abhi shelat. Fair zero knowledge. In *Proceedings of the 2nd Theory of Cryptography Conference*, February 2005.
- [18] Heiko Mantel. On the composition of secure systems. In *IEEE Symposium on Security and Privacy*, pages 88–101, May 2002.
- [19] Rebecca Mercuri. A better ballot box? *IEEE Spectrum*, 39(10):46–50, October 2002.
- [20] Deirdre Mulligan and Joseph Hall. Preliminary analysis of e-voting problems highlights need for heightened standards and testing. A whitepaper submission to the NRC’s Committee on Electronic Voting, http://www7.nationalacademies.org/cstb/project_evoting_mulligan.pdf, December 2004.
- [21] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *8th ACM Conference on Computer and Communications Security (CCS 2001)*, pages 116–125, November 2001.
- [22] C. Andrew Neff, October 2004. Personal Communication.
- [23] C. Andrew Neff. Practical high certainty intent verification for encrypted votes. <http://www.votehere.net/vhti/documentation>, October 2004.
- [24] C. Andrew Neff. Verifiable mixing (shuffling) of El-Gamal pairs. <http://www.votehere.net/vhti/documentation>, April 2004.
- [25] Peter G. Neumann. Principled assuredly trustworthy composable architectures. Final report for Task 1 of SRI Project 11459, as part of DARPA’s Composable High-Assurance Trustworthy Systems (CHATS) program, 2004.
- [26] Birgit Pfitzmann and Andreas Pfitzmann. How to break the direct RSA-implementation of MIXes. In *Advances in Cryptology – EUROCRYPT 1989*, volume 434 of *Lecture Notes in Computer Science*, pages 373–381. Springer-Verlag, April 1989.
- [27] Sean Rhea, Patrick Eaton, Dennis Geels, Hakim Weatherspoon, Ben Zhao, and John Kubiatowicz. Pond: the OceanStore prototype. In *2nd USENIX Conference on File and Storage Technologies (FAST '03)*, pages 1–14, March 2003.
- [28] Bruce Schneier. *Secrets and Lies*, chapter 17, “The Human Factor”, pages 255–269. John Wiley and Sons, Inc., 2000.
- [29] Poorvi Vora. David Chaum’s voter verification using encrypted paper receipts. Cryptology ePrint Archive, Report 2005/050, February 2005. <http://eprint.iacr.org/>.
- [30] Alma Whitten and J.D. Tygar. Why Johnny can’t encrypt: A usability evaluation of PGP 5.0. In *8th USENIX Security Symposium*, pages 169–184, August 1999.