On Subliminal Channels in Encrypt-on-Cast Voting Systems

Ariel J. Feldman
Princeton University
ajfeldma@cs.princeton.edu

Josh Benaloh
Microsoft Research
benaloh@microsoft.com

Abstract

Ballot secrecy, while essential, is difficult to achieve with any voting system cryptographic or otherwise. Moreover, the majority of cryptographic voting systems introduce new ballot secrecy problems. In encrypt-on-cast voting systems, like that of Benaloh [1, 2], a malicious voting machine can use the encrypted votes that it posts to the public bulletin board as a subliminal channel to convey information about voters' choices to a coercer. Although it was known that a machine could manipulate the randomness used to encrypt the votes to leak information [14], we show that this threat is more severe than previously recognized and that existing mitigations may be ineffective. A compromised machine may only need to leak a few bits and modify only a handful of ballots in order to coerce most of the voters in a polling place. In light of this threat, we propose an extension to the Benaloh scheme that allows anyone to verify that every ciphertext on the bulletin board uses the right randomness. Finally, we show that even without manipulating the randomness, a machine can still use the ciphertexts to leak a small, but potentially dangerous, number of bits by strategically flipping a few votes. Overall, we show that while subliminal channels in encrypt-on-cast voting systems can be partially mitigated, they cannot yet be eliminated completely.

1 Introduction

Ballot secrecy is essential because without it, vote-buying and voter coercion are possible, and elections cannot be trusted to reflect the will of the voters. Even if a voter's choices may only be disclosed to a coercer with a small probability, that may be enough to influence his or her behavior. Unfortunately, ballot secrecy is very difficult to achieve with any voting system. Ubiquitous cell-phone cameras allow voters to capture themselves in the act of voting, absentee voting allows coercers to be present while voters cast their ballots, and paper ballots can be uniquely identified by distinguishing marks added by the voter or by paper fingerprinting techniques [9]. In addition, any voting machine such as a DRE or an optical scanner that is given the plaintext votes in the order that they are cast can record this information and leak it to a malicious poll worker or to a malicious voter in response to a "secret knock."

End-to-end cryptographic voting systems promise to improve the accuracy of elections considerably by making the vote-tallying process publicly verifiable. But not only do such systems generally suffer from many of the ballot secrecy problems listed above, the majority of them also introduce new ones of their own.

Most recent cryptographic voting systems fall into two categories: those that use pre-encrypted paper ballots and those which encrypt each vote as it is cast. In systems that employ pre-encrypted paper ballots, such as Prêt à Voter [8], PunchScan [19], and Scantegrity I [7] and II [6], every ballot has a secret mapping between the ovals that the voter marks and the candidates in each race. Unfortunately, anyone with access to these mappings, including the election trustees who initially chose them, anyone involved in printing the ballots, and, in some cases, anyone who handled the ballots before the election, can reconstruct how every voter voted.

In encrypt-on-cast systems such as those proposed by Benaloh [1, 2], Neff [17], and Sandler *et al.* [22], the voting machines in the polling place accept voters' choices either through a touchscreen interface or though an optical scanner, encrypt them, and then post the ciphertexts to a public bulletin board. The problem is that a malicious voting machine can use the ciphertexts it posts to the bulletin board as a subliminal channel to leak information about voters' choices [14]. It can do this by repeatedly running the encryption function with different inputs until the resulting ciphertext has the desired properties (e.g. that the low order bits of the HMAC of the ciphertext have certain values).

In this paper, we examine the threat posed by subliminal channels to encrypt-on-cast voting systems and how it can be partially mitigated. Our primary contributions are:

- We discuss the possibility that a compromised voting machine could undermine ballot secrecy by maliciously choosing the random values used in the randomized encryption of the votes. Although this attack was first recognized in [14], we show that it is more severe than previously recognized. An attacker may only need to leak a small number of bits and only modify a handful of ballots in order to coerce most of the voters in a polling place. As a result, we demonstrate that an existing proposal [11] for mitigating this problem may be insufficient.
- We present an enhancement to the Benaloh [1, 2] voting scheme that mitigates the threat posed by maliciously chosen randomness. In our protocol, as in [11], all of the randomness used to encrypt the votes is generated prior to the election by a set of mutually-independent trustees. But, unlike previous approaches, our protocol makes it possible for any interested party to verify that the right random values were used to encrypt every vote on the bulletin board.
- Finally, we show that even if a malicious voting machine cannot control the randomness used in the encryption of the votes, it can still use the ciphertexts on the bulletin board to leak information by manipulating the plaintext votes themselves on a small number of ballots. Although the subliminal channel created by such vote-flipping is low bandwidth, it is still potentially dangerous and difficult to mitigate.

2 Ciphertexts as a Subliminal Channel

To achieve semantic security, encrypt-on-cast voting machines use a randomized encryption scheme, often El Gamal, of the form $E_{pk}(r, v)$ where pk is the public key of the election, v is the vote, and r is a random value ordinarily chosen by the machine. As Karlof *et al.* [14] explain, a malicious voting machine can leak b bits for each ciphertext with expected $O(2^b)$ work simply by trying to encrypt the plaintext multiple times with different values of r until the low order b bits of the hash[1] of the ciphertext have the desired values.

2.1 Exploiting the Subliminal Channel

An attacker trying to coerce voters by compromising ballot secrecy can simplify his task considerably by making two simple observations. First, he does not necessarily need to leak information about every race on the ballot. To compromise the secrecy of a single ballot, he may only need to leak a single bit indicating whether the voter has complied fully with his requirements. Second, he does not need to compromise the secrecy of every ballot in order to coerce most of the voters. He need only violate the privacy of enough voters to scare the majority into compliance. As a result, he may only need to leak a small number of bits in order to accomplish his objective.

Consider a polling place with 1000 voters and a single precinct-count optical scan voting machine that produces ciphertexts along the lines of the system described in [2]. A coercer could compromise the privacy of a fraction p of the voters as follows. Prior to the election, the coercer would share a seed s with the compromised voting machine. Then, as each ballot was cast, the machine would decide to violate the current voter's privacy with probability p using a pseudorandom generator seeded with s. Next, the machine would manipulate a small number of the ciphertexts that it provided to the bulletin board in order to leak a bit vector containing a bit for each of the selected voters. A one would indicate that the voter complied with the coercer, and a zero would indicate that the voter did not. After the election, since the coercer would know s, he would be able to replicate the pseudorandom choices that the machine made and connect each bit of the vector to an individual voter. Using this strategy, a coercer could make it so that there is a 10% chance that any voter's vote would be revealed by leaking just 100 bits.

Alternatively, a coercer could make it known that, after the election, he would reveal the identity of one noncompliant voter who would then be punished severely. He could accomplish this by having the voting machine leak an integer corresponding to the target voter's place in the line of voters who voted on the malicious voting machine. In general, this strategy can identify k voters out of a total of n using $k \log n$ bits. Thus, for the example polling place, it would require as little as 10 bits.

Which strategy is preferable depends on the percentage of voters who attempt to defy the coercer. If a large percentage try, the first is preferable, whereas if only a small percentage try and the coercer merely has to maintain obedience by singling out the few resisters, the second is preferable. Of course, the malicious voting machine can reserve a bit of leaked data to indicate which strategy it is using and pick the one that will reveal the most resisters.

2.2 Avoiding Detection

In an attempt to eliminate the subliminal channel provided by ciphertexts on the bulletin board, Gardner *et al.* [11] propose a scheme in which the choice of the random values used in the encryption of the votes is taken away from the voting machine. In their scheme, a group of mutually-

[1] To better conceal the attack, a malicious machine could use an HMAC with a key known only to the adversary instead of a hash.

independent election trustees each provide every voting machine with a tamper-resistant smart card containing a secret key. Every time that a voting machine encrypts a vote, the pseudorandom value that it uses is derived deterministically from the secret keys of all of the trustees and the ballot serial number. The serial number is public and chosen by a means outside of the machine's control (e.g. preprinted on the paper that will become the voter's receipt).

To check whether the machine is using the right random values, the Gardner proposal relies on voter-initiated audits. All voting schemes that encrypt votes as they are cast offer each voter the option to audit her ballot after it has been encrypted in order to verify that the machine has encrypted the votes that she intended. In the Benaloh voting system, a voter audits her ballot by having the machine open the encryptions of her votes. In so doing, she spoils her ballot, but she is allowed to vote again in order to cast a ballot that counts. In the Gardner proposal, which builds on the Benaloh scheme, auditing a ballot not only opens the encryption of each of the votes, it also reveals the components of the pseudorandom value used in the encryption that came from each trustee. The voter, or her chosen representative, can then use the trustees' public keys to verify that the components are correct and that, when combined, they form the right pseudorandom value.

The problem with relying on voter-initiated audits is that if the percentage of voters who currently check the paper audit trails produced by today's DREs is any guide, the percentage of voters who would audit their ballots in a cryptographic voting system is likely to be very low (perhaps less than 5%)[2]. Such low audit rates can be sufficient to catch a malicious voting machine that is trying to directly alter the votes because, in order to have a meaningful impact on the vote total in all but the closest elections, the machine would have to corrupt enough ballots that even a very low audit rate would catch it with high probability [16]. But, given how few bits need to be leaked in order to carry out the coercion attacks described in previous subsection, voter-initiated audits may be insufficient.

Returning to the example of the 1000 voter polling place, suppose that a coercer wanted to intimidate 10% of the voters by leaking 100 bits, that there are 10 races on the ballot, and that each voter's choice in each race is encrypted separately[3]. A malicious voting machine could leak all 100 bits through a single pseudorandomly chosen ballot by leaking 10 bits through the ciphertext of each race with expected $O(2^{10})$ work per race (an easily manageable amount). Thus, if 5% of voters chose to audit their ballots, the coercer would have a 95% chance of avoiding detection.

To further reduce the chances of detection, a coercer could program the compromised voting machine to accept a "secret knock" so that when a malicious voter chooses a certain pattern of candidates, the machine knows that the voter will not audit the ballot and that it can safely use the ballot's ciphertexts to leak bits. Of course, in many cases, a corrupted machine could just leak information directly to the malicious voter via the machine's display or printer.

In sum, an attacker who is willing to risk detection by corrupting some fraction of the ballots may be more likely to succeed if his goal is to leak bits and perform a coercion attack than if his goal is to directly change the tally because he does not need to alter nearly as many ballots.

3 Minimizing the Subliminal Channel

In this section we describe our approach to preventing a malicious voting machine from compromising ballot secrecy via its choice of the random values used to encrypt the votes. Like Gardner *et al.* [11], we propose to take the choice of the randomness out of the hands of a potentially compromised voting machine, and instead have it be generated by a set of mutually-independent trustees such as political parties or non-governmental organizations. But, in light of the attacks that we describe in Section 2, our approach differs in that it allows any interested member of the public to verify that the right random values (i.e. those that came from the trustees) were used to encrypt every vote, not just those that individual voters decided to audit.

3.1 Definition

More formally, the ciphertexts that our protocol produces have the following property:

Definition 1. The ciphertexts that a voting system produces are *subliminal channel minimizing* if, for a given election public key pk, for any sequence of cast, audited,

[2]In a study conducted by Selker and Cohen [23], only 3 out of 108 voters in a simulated election noticed a discrepancy between the paper trail and the DRE's screen. Although Norden *et al.* [18] think that this figure is an underestimate, we believe that the audit rate with a cryptographic voting system would remain quite low because voter-initiated audits are more cumbersome than checking a DRE's paper trail. The Neff scheme requires the voter to engage in a relatively complex interactive proof, while the Benaloh scheme requires the voter to fill out a second ballot.

[3]Encrypting and tallying each race separately prevents a different kind of coercion attack: a voter can make her ballot identifiable to a coercer by voting in a certain pattern in the races on the ballot [20]. If a voter's choices in all of the races were encrypted together in a single ciphertext, the number of bits that a malicious voting machine could leak per ballot would be lower, but would still be enough to carry out the attacks we describe.

and canceled ballots $B = b_0, b_1, \ldots, b_m$ comprised of any sequence of votes $V = v_0, v_1, \ldots, v_n$, there is exactly one valid sequence of ciphertexts $C = c_0, c_1, \ldots, c_\ell$.

Letting a voting machine choose the random parameters to the encryptions of the votes creates a subliminal channel because it allows there to be many valid ciphertexts corresponding to each plaintext vote, and the machine's choice of which of these ciphertexts to publish can convey information. Intuitively, a voting system that produces subliminal channel minimizing ciphertexts prevents this attack by ensuring that for any given sequence of votes, there is only one valid way to encrypt it. This definition does not rule out the use of a semantically secure encryption scheme because it only requires that the there be a single combined ciphertext for each possible sequence of votes. It does not require the same vote on two different ballots to have the same encryption.

This property is called subliminal channel *minimizing* rather than subliminal channel *eliminating* because, as explained in Section 5, even without control of the randomness, a malicious voting machine can still leak some bits.

3.2 Description of our Protocol

Our protocol builds on the Benaloh [1, 2] voting scheme. But although the Benaloh scheme allows votes to be encrypted using a variety of encryption schemes and allows tallying to be done using either homomorphic encryption or a reencryption mixnet, our protocol requires the use El Gamal encryption and a reencryption mixnet.

Our protocol has three phases. Prior to the election, the trustees generate the random values that all of the voting machines will use, and the random values are probabilistically checked for validity. Then, during the election, each voting machine uses the random values that it was given in the proper order to encrypt the votes. Finally, after polls close and the encrypted cast votes are put through a reencryption mixnet and decrypted, interested parties check to make sure that the right randomness was used for every encryption.

3.2.1 Definitions

- Let G be a multiplicative cyclic group of prime order q with generator g.
- Let h be the public key of the election that is used to encrypt the votes posted to the bulletin board.
- Let $T = t_0, t_1, \ldots, t_k$ be the set of trustees that supply shares of the randomness that will be used for encryption. These may be the same as or different from the trustees who hold shares of the key needed to decrypt the votes.
- Let each trustee $t \in T$ have a public key pk_t and a secret key sk_t that it uses to sign the random values that it produces. Many widely-used signature schemes could be used for this purpose including DSA and RSA.
- Let M be the set of voting machines.
- Let each voting machine $m \in M$ have a public key $h_m = g^{x_m}$ where x_m is the corresponding secret key chosen randomly from $[0, q-1]$.

3.2.2 Prior to the Election

1. For each voting machine $m \in M$, each trustee $t \in T$ generates a long list of random values $r_{t,m,0}, r_{t,m.1}, \ldots, r_{t,m,n}$ such that $0 \leq r_{t,m,i} \leq q-1$. The number of values should be more than *twice* the number of encryptions that any single voting machine will ever need to perform in the course of a single election.
2. For each random value $r_{t,m,i}$, the trustee t publishes the triple $a_{t,m,i} = (i, g^{r_{t,m,i}}, \sigma_{t,m,i})$ where i is the index of the random value and $\sigma_{t,m,i}$ is t's signature on the first two elements of the triple with sk_t.
3. Also for each random value $r_{t,m,i}$, the trustee t publishes the triple $b_{t,m,i} = (i, E(r_{t,m,i}), \sigma'_{t,m,i})$. $E(.)$ is El Gamal encryption with the machine's public key h_m and with the random value $w_{t,m,i}$ and $\sigma'_{t,m,i}$ is t's signature on the first two elements of the triple with sk_t.
4. For each voting machine $m \in M$, each trustee $t \in T$ is challenged in public to open the encryptions of half of the random values. The encryptions to be opened can be selected using the Fiat-Shamir heuristic [10] by computing the hash of $b_{t,m,0}, \ldots, b_{t,m,n}$ where the hash function is modeled as a random oracle and then using the hash to seed a pseudorandom generator that is used to pick the indices of the encryptions to be opened. A trustee opens the encryption in the triple $b_{t,m,i}$ by revealing the randomness $w_{t,m,i}$ used to encrypt it.
5. Any interested party can verify the signature on every triple $a_{t,m,i}$ and $b_{t,m,i}$. He can also verify that every opened triple $b_{t,m,i}$ matches its corresponding triple $a_{t,m,i}$. Let (α, β) be the El Gamal encrypted random value in $b_{t,m,i}$. The interested party first computes $r = \frac{\beta}{h_m^{w_{t,m,i}}}$ and verifies that $\alpha = g^{w_{t,m,i}}$. Then, he can verify that g^r is equal to the second element of the triple $a_{t,m,i}$.
6. Once every trustee has produced its output and been challenged, for each voting machine $m \in M$, each

trustee $t \in T$ gives each of its remaining unopened triples $b_{t,m,i}$ to m along with the corresponding triple $a_{t,m,i}$.

3.2.3 During the Election

Every time a voting machine m encrypts a vote it does the following:

1. For each trustee $t \in T$, m selects the next triples $a_{t,m,i}$ and $b_{t,m,i}$ from the list that it received from t. The lists of triples are ordered by increasing values of i, and m must use the next triples on the lists.

2. For each triple $a_{t,m,i}$ and $b_{t,m,i}$, m verifies that it has been signed by trustee t and then decrypts the encrypted random value $r_{t,m,i}$ in $b_{t,m,i}$ using its private key.

3. To encrypt a vote v, m first creates a message $\mu = (v, f(v))$ where f is a hash function modeled as a random oracle. Then, m produces the El Gamal ciphertext pair $y = (\alpha, \beta) = (g^{r_{0,m,i}} \cdot g^{r_{1,m,i}} \cdot \ldots \cdot g^{r_{k,m,i}}, h^{r_{0,m,i}+r_{1,m,i}+\ldots+r_{k,m,i}} \cdot \mu)$. Finally, m produces the ciphertext $c = (y, z)$ where z is a hash chain value. For the first El Gamal ciphertext y_0, the corresponding hash chain value $z_0 = h(y_0)$ where h is a collision-resistant hash function. For a subsequent El Gamal ciphertext y_j, $z_j = h(h(y_j), z_{j-1})$.

4. All encrypted votes, whether they are ultimately cast, audited, or canceled, are posted to the bulletin board in the order that they are generated by m. As far the bulletin board is concerned, the only difference between cast and audited and canceled votes, is that for audited and canceled votes, the vote v and the random values $r_{0,m,i}, \ldots, r_{k,m,i}$ are posted in addition to the ciphertext c.

3.2.4 After the Election

1. For each voting machine $m \in M$, any interested party can verify that the first half of every El Gamal ciphertext $y = (\alpha, \beta)$ posted to the bulletin board is correct by verifying that $\alpha = g^{r_{0,m,i}} \cdot g^{r_{1,m,i}} \cdot \ldots \cdot g^{r_{k,m,i}}$ for the appropriate values of $g^{r_{t,m,i}}$. He can also recompute every hash chain value z. In addition, he can verify that the correct random values were used for each audited or canceled vote. Let v be the vote and let $s_0, s_1, \ldots, s_k$ be the random values that were posted to bulletin board as part of the opened encryption. Also, let ρ be the race to which v belongs and let V_ρ be the set of possible choices in ρ. The interested party can verify that v is correctly formed by verifying that $v \in V_\rho$. He then can recompute $\mu = (v, f(v))$ and verify that $\frac{\beta}{\mu} = h^{s_0+s_1+\ldots+s_k}$ and that $g^{s_0} \cdot g^{s_1} \cdot \ldots \cdot g^{s_k} = \alpha$.

2. All of the cast votes on the bulletin board are put through a reencryption mixnet and then the outputs of the mixnet are decrypted by the election trustees.

3. For each decrypted message μ that came through the mixnet, any interested party can verify that μ is correctly formed by verifying that $\mu = (v, f(v))$ and that $v \in V_\rho$ where V_ρ is the set of valid values for v.

4 Security Analysis

4.1 Proof Sketch

Claim 1. *The ciphertexts that the protocol presented in Section 3.2 produces will either be subliminal channel minimizing or will be detected as invalid with high probability.*

If the sequence of ciphertexts $C = c_0, \ldots, c_\ell$ on the bulletin board is created according to the protocol, it will be subliminal channel minimizing because the ciphertexts are posted in the order that they are created and every ciphertext $c = (y, z)$ on the bulletin board is derived completely deterministically. The hash chain value z is determined solely by the sequence of ciphertexts that preceded c. Moreover, the El Gamal ciphertext $y = (\alpha, \beta)$ is derived deterministically from the vote v, the election public key h, and the values $r_{0,m,i}, \ldots, r_{k,m,i}$ and $g^{r_{0,m,i}}, \ldots, g^{r_{k,m,i}}$ provided to the voting machine by the trustees.

If a malicious voting machine does not follow the protocol, the invalid ciphertexts that it produces will be detected with high probability by the post-election verification steps described in Section 3.2.4. For every ciphertext, any interested party can verify z and α by just recomputing them from public values. For an audited or canceled vote, he can recompute β as well because v and $r_{0,m,i}, \ldots, r_{k,m,i}$ are available in the clear.

For a cast vote, verifying that the decrypted message μ is well formed is sufficient to verify c. To see why, consider a malicious voting machine that is trying to get away with posting an invalid ciphertext $c' \neq c$. Since the z and α values of every ciphertext on the bulletin board can be easily checked, c' must be of the form $((\alpha, \beta'), z)$ where $\beta' \neq \beta$. But then, $\beta' = h^{r_{0,m,i}+\ldots+r_{k,m,i}} \cdot \mu'$ where $\mu' \neq \mu$. In other words, c' is the encryption of some new message μ'. Thus, when c' is decrypted, there are three possibilities for μ':

1. μ' is not of the form $(v', f(v'))$ for some $v' \neq v$, and the machine will be caught cheating. Since f is

modeled as a random oracle, this is by far the most likely outcome if β' is an arbitrarily-chosen element of G.

2. μ' is of the form $(v', f(v'))$ for some $v' \neq v$, but $v' \notin V_\rho$, and the machine will still be caught cheating.

3. μ' is of the form $(v', f(v'))$ for some $v' \neq v$, and $v' \in V_\rho$. In this case, c' is just encryption of a different valid vote, and what the machine did is equivalent to flipping the vote.

A malicious machine cannot publish ciphertexts out of order because the ciphertext c_j must use the jth set of random values from the trustees. Otherwise, when an interested party tries to recompute the α component of c_j, it will fail. The machine could, however, swap votes between ciphertexts. For example, it could put vote v_j inside ciphertext $c_{j+\delta}$ and $v_{j+\delta}$ inside c_j. In that case, the voter who casts v_j would need to get a receipt with a commitment to $c_{j+\delta}$. But, since the machine would not yet have generated $c_j, c_{j+1}, \ldots, c_{j+\delta-1}$, it would not yet know the correct hash chain value $z_{j+\delta}$, and the commitment on the voter's receipt would be wrong. Thus, if the voter or her chosen representative compared her receipt to the bulletin board, the machine would be caught cheating.

4.2 Threats and Mitigations

4.2.1 Malicious Voting Machine

Since the output of our protocol is subliminal channel minimizing, a voting machine cannot maliciously choose the random values used in the encryption of the votes to leak bits to the bulletin board. A machine could still attempt to flip votes, but the voter-initiated auditing process prevents the machine from cheating on more than a few votes. Vote-flipping does have implications for ballot secrecy, however, as discussed in Section 5.

Since each voting machine accepts votes as input in the clear, it could disclose them in the order they are cast to a malicious voter or poll worker. But, this vulnerability is present in every voting system in which a machine accepts plaintext votes, whether it is cryptographic or not.

4.2.2 Malicious Trustee

A malicious trustee that is supplying random values to voting machines could disclose its values to an adversary, but if there is at least one honest trustee, the adversary would not be able to gain any information about the votes. To see why, suppose that for a given vote v, there is one honest trustee that has supplied the random value r and suppose that the adversary knows the sum s of the rest of the random values supplied by the other trustees. Then, from adversary's standpoint, $(g^{s+r}, h^{s+r} \cdot v)$ is still uniformly distributed in the group G.

A malicious trustee could also launch a denial of service attack by giving the voting machine the random value r, but publishing $g^{r'}$ where $r \neq r'$. The trustee's corrupt output would cause the voting machine to produce invalid ciphertext and to be blamed for cheating. Step 4 in Section 3.2.2 prevents this attack with high probability by challenging each trustee to reveal half of the random values it has released.

5 Vote-flipping as a Subliminal Channel

As we alluded to in the previous sections, although the protocol we present in Section 3 removes a significant subliminal channel, there is still a way for the machine to leak some bits via the ciphertexts on the bulletin board. Even without controlling the random values used in the encryption of the votes, a malicious voting machine can still control a ciphertext c on the bulletin board to a certain extent by ignoring the voter's choices and trying every possible valid vote until the low order bits of HMAC$_k(c)$ have the desired values where k is a secret key known only to the adversary. Of course, if enough voters choose to audit their ballots, the malicious machine will be caught cheating. But as before, we will show that if the machine only alters a few ballots, its attack can succeed.

Assuming, as before, that each race on a ballot is encrypted separately, the number of bits that a malicious machine can leak through a given ciphertext is not limited by the amount of work that the machine is able to do, but by the number of valid choices there are in the given race. As a result, the subliminal channel created by flipping votes has much lower bandwidth than the channel created by manipulating the randomness, but it still poses a threat. In general, the probability p that a malicious machine can leak b bits using the ciphertext of a race with m candidates is:

$$p = (1 - (1 - \frac{1}{2^b})^{m+1})$$

(The exponent is $m + 1$ as opposed to m because leaving the race blank is generally a valid choice.) Table 1 lists values of p for various values of m and b.

5.1 Exploiting the Vote-flipping Channel

Returning the example of the 1000 voter polling place from Section 2, suppose a coercer wants to leak 10 bits in order to reveal the identity of a single voter who voted the "wrong" way. Once again, the coercer will try to leak these 10 bits by altering one or more pseudorandomly chosen ballots posted to the bulletin board. However, unlike the previous example, the coercer's task is complicated by the fact that, as Table 1 illustrates, for a given ciphertext, there is some probability that the coercer will not be able to leak the bits that he wants. He can minimize the chances of failure by only trying to leak bits in races which have enough choices that the probability of success is sufficiently high. Nevertheless, there will still likely be races where he fails to leak the desired bits. To deal with these cases, he can treat failures as bit errors in his message and employ an error-correcting code.

Suppose that the coercer limits himself to leaking bits on races in which he has $\geq 93.75\%$ chance of success (i.e. leaking 1 bit of races with ≥ 3 choices and leaking 2 bits on races with ≥ 9 choices). He can then encode the 10 bits he wants to leak with an Extended BCH code [24] that is 30 bits long and can correct up to 5 errors $(n = 30, k = 10, d = 11)$ [12]. Given the bit error rate of $\leq 1 - 0.9375 = 0.625$, by the Chernoff bound, the probability that there will be more than 5 errors in the 30 bits is less than 0.1688. Assuming that there are 10 races on the ballot, each with fewer than 9 choices, the coercer would need to alter 3 pseudorandomly selected ballots out of 1000. Thus, if 1% of the voters audit their ballots, the coercer has an over 97% chance of avoiding detection, and if 5% audit, he still has an over 85% chance of success [16].

5.2 Factors that Exacerbate the Threat

Although the subliminal channel created by vote-flipping is low bandwidth, there are several factors that can exacerbate the threat that it poses:

- **Write-in votes.** Races that allow write-in votes greatly increase the number of bits that a malicious voting machine can leak because the write-in field gives the machine much greater latitude in choosing the message to be encrypted. The only limit on the message is that it would have to be a name that a voter might plausibly write.
- **Secret knocks.** As in the previous attack, a "secret knock" from a malicious voter could tell the machine that the voter will not audit the ballot and that it can safely use the ballot's ciphertexts to leak bits.
- **A single ciphertext for all races on the ballot.** If all of the races on the ballot are encrypted as a single ciphertext, the number of possible messages that a malicious machine can choose from is equal to the product of the number choices in each of the races on the ballot. If this number is large, as it often would be, the number of bits that the machine would be able to leak per ballot would be more limited by the amount of work that the machine is able to do than by the size of the message space.

m \ b	1	2
1	0.7500	0.4375
2	0.8750	0.5781
3	0.9375	0.6836
4	0.9688	0.7627
5	0.9844	0.8220
6	0.9922	0.8665
7	0.9961	0.8999
8	0.9980	0.9249
9	0.9990	0.9437
10	0.9995	0.9578

Table 1: The probability of leaking b bits using the ciphertext of a race with m choices

5.3 Mitigation

Preventing a malicious voting machine from leaking bits by flipping votes seems to be difficult because the general strategy that we employ in Section 3 does not apply. The votes, unlike the randomness, obviously cannot be decided in advance of the election by a set of trustees. As a result, it currently seems that the only available mitigation is to rely on a relatively high rate of voter-initiated audits. But, we are skeptical of assuming a high audit rate because voter-initiated audits are cumbersome even for the Benaloh scheme, which is designed to make them relatively easy.

Benaloh [1] and others have proposed boosting the audit rate artificially by having election officials or representatives of political parties or non-governmental organizations encrypt extra ballots that are never cast for the sole purpose of auditing them. Unfortunately, since the number of corrupted ballots needed for a subliminal channel attack is so small, the non-voting auditors would have to audit an impractically large number of additional ballots in order to have a reasonable chance of detecting an attack. For example, to have a nearly 50% chance of detecting the attack described in Section 5.1 in which 3 ballots are corrupted out of 1000, the audit rate would need to be 20%. If 5% of the voters chose to audit on their own, the non-voting auditors would have to encrypt an extra 188 ballots in order to boost the audit rate to the required level.

In addition, the use of non-voting auditors suffers from many of the same weaknesses as parallel testing [18]. Malicious auditors can use a "secret knock" to tell compromised voting machines to refrain from corrupting ballots while audits are taking place. Moreover, if there is any difference in voting behavior between auditors and real voters (e.g. in the distribution of votes, the time taken to vote, or the time between votes), then a compromised voting machine could use this discrepancy to decide when not to cheat. Thus, finding an effective mitigation to the vote-flipping subliminal channel seems to require future work.

6 Related Work

Although cryptographic voting systems have been researched and refined since Chaum's initial work in 1981 [4], the issue of coercion was not addressed until much later. The early expectation was that voters would simply use their own trusted devices, or trusted devices supplied by others, to encrypt their votes.

Benaloh and Tuinstra [3] developed the first cryptographic voting system that was receipt-free (i.e. it prevented voters from proving how they voted to a coercer). However, their approach was cumbersome and did not scale well to elections with many races and candidates. Moreover, although one variant of their protocol was receipt-free, their full protocol was not [13].

Chaum [5] and Neff [17] proposed voting systems with creative user interfaces that allowed voters to verify that their votes were correctly recorded without the aid of a computer in the voting booth. Karlof *et al.* [14] described several vulnerabilities in these schemes and also explained how the randomness used to encrypt votes could be manipulated in order to compromise ballot secrecy.

More recent systems have user interfaces that are even easier for unaided voters to use. Most of these systems fall into two categories: those that use pre-encrypted paper ballots, such as Prêt à Voter [8], PunchScan [19], and Scantegrity I [7] and II [6], and encrypt-on-cast voting systems, such as those proposed by Benaloh [1, 2], and Sandler *et al.* [22]. Systems that employ pre-encrypted paper ballots have the weakness that anyone involved in creating or possibly handling the ballots before the election can compromise ballot secrecy. Encrypt-on-cast systems have the subliminal channel vulnerabilities discussed in this paper.

Several systems attempt to provide greater privacy at the expense of increased complexity for both voters and election officials. Moran and Naor [15] proposed a paper-based scheme in which every ballot is comprised of two halves, and the halves come from mutually-independent trustees. A coercer cannot compromise a voter's privacy unless he sees both halves of her ballot before the election. Unfortunately, this approach results in a complicated ballot layout and requires voters to make random choices. Riva and Ta-Shma [21] suggested a scheme in which the voter does not reveal her plaintext votes to the voting machine. But, the voter is required to use a computer to encrypt her votes before arriving at the polling place.

Gardner *et al.* [11] proposed a scheme aimed at preventing a malicious voting machine from compromising ballot secrecy by manipulating the random values used to encrypt the votes. But, in Section 2, we argue that their protocol may be insufficient to mitigate the threat. They also provided a definition of coercion resistance. Roughly speaking, they claimed that a coercion resistant system is one in which a coercer cannot tell whether a given ciphertext contains his chosen vote or a different vote as long as the ciphertext is ultimately audited. While this definition has some intuitive appeal, we choose not to use it because it only holds for the minority of ballots that are audited. Moreover, it does not rule out certain attacks that we describe such as using one ciphertext to leak information about the votes contained in other ciphertexts.

7 Conclusion

Among end-to-end cryptographic voting systems, encrypt-on-cast systems like that of Benaloh are appealing because they can provide voters with DRE and optical scan user interfaces that closely resemble those of non-cryptographic voting systems. However, like most other cryptographic voting schemes, they introduce new threats to ballot secrecy. In particular, a malicious voting machine can use the encrypted votes that it posts to the public bulletin board as a subliminal channel to convey information about voters' choices to a coercer. In this paper, we have shown that this threat is more severe than previously recognized, and that voter-initiated audits, while sufficient to prevent significant vote-flipping, may not be effective at mitigating subliminal channels. But, we have presented a protocol that partially mitigates the threat by preventing a compromised voting machine from maliciously manipulating the random values used to encrypt the votes in order to leak information. The protocol allows any interested party to verify that the right randomness was used in the encryption of every vote, not just those on ballots that voters have chosen to audit. Unfortunately, our approach does not completely solve the problem because, as we have shown, just by flipping a handful of votes, a malicious machine can leak a small, but potentially dangerous number of bits to the bulletin board. Consequently, it seems that more work will be necessary to completely mitigate the unique ballot secrecy threats that encrypt-on-cast voting systems face.

Acknowledgments

We thank Edward W. Felten, Nadia Heninger, Sharon Goldberg, and the anonymous reviewers for their insights and helpful comments.

References

[1] J. Benaloh. Ballot casting assurance via voter-initiated poll station auditing. In *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '07)*, Aug. 2007.

[2] J. Benaloh. Administrative and public verifiability: Can we have both? In *Proc. 2008 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '08)*, July 2008.

[3] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *Proc. 26th ACM Symposium on Theory of Computing*, May 1994.

[4] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, Feb. 1981.

[5] D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy*, Feb. 2004.

[6] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A. Ryan, E. Shen, and A. T. Sherman. Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes. In *Proc. 2008 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '08)*, July 2008.

[7] D. Chaum, A. Essex, R. T. C. III, J. Clark, S. Popoveniuc, A. T. Sherman, and P. Vora. Scantegrity: End-to-end voter verifiable optical-scan voting. *IEEE Security & Privacy*, May 2008.

[8] D. Chaum, P. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *10th European Symposium On Research In Computer Security*, Sept. 2005.

[9] W. Clarkson, T. Weyrich, A. Finkelstein, N. Heninger, J. A. Halderman, and E. W. Felten. Fingerprinting blank paper using commodity scanners. In *Proc. IEEE Symposium on Security and Privacy. To Appear*, May 2009.

[10] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology: CRYPTO '86*, 1986.

[11] R. W. Gardner, S. Garera, and A. D. Rubin. Coercion resistant end-to-end voting. In *Proc. Financial Cryptography and Data Security 2009*, Feb. 2009.

[12] M. Grassl. Code tables: Bounds on the parameters of various types of codes, Oct. 2008.

[13] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *Proc. Eurocrypt '00: Advances in Cryptology*, May 2000.

[14] C. Karlof, N. Sastry, and D. Wagner. Cryptographic voting protocols: A systems perspective. In *Proc. 14th USENIX Security Symposium*, Aug. 2005.

[15] T. Moran and M. Naor. Split-ballot voting: Everlasting privacy with distributed trust. In *Proc. ACM conference on Computer and Communications Security (CCS '07)*, Oct. 2007.

[16] C. A. Neff. Election confidence—a comparison of methodologies and their relative effectiveness at achieving it (revision 6), Dec. 2003.

[17] C. A. Neff. Practical high certainty intent verification for encrypted votes, Oct. 2004.

[18] L. Norden et al. The machinery of democracy: Protecting elections in an electronic world: Brennan center task force on voting system security. Brennan Center for Justice at NYU School of Law, Aug. 2006.

[19] S. Popoveniuc and B. Hosp. An introduction to punchscan, Oct. 2006.

[20] S. Popoveniuc and J. Stanton. Undervote and pattern voting: Vulnerability and a mitigation technique. In *Workshop on Trustworthy Elections*, June 2007.

[21] B. Riva and A. Ta-Shma. Bare-handed electronic voting with pre-processing. In *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '07)*, Aug. 2007.

[22] D. Sandler, K. Derr, and D. S. Wallach. Votebox: A tamper-evident, verifiable electronic voting system. In *Proc. 17th USENIX Security Symposium*, Aug. 2008.

[23] T. Selker and S. Cohen. An active approach to voting verification, May 2005.

[24] J. H. van Lint. *Introduction to Coding Theory*, chapter 6.6. Springer-Verlag, 1992.