Ballot Casting Assurance

Ben Adida
MIT

C. Andrew Neff
VoteHere

Abstract

We propose that voting protocols be judged in part on *ballot casting assurance*, a property which complements universal verifiability. Some protocols already support ballot casting assurance, though the concept has not received adequate explicit emphasis. Ballot casting assurance captures two well known voting concepts – cast as intended, and recorded as cast – into one end-to-end property: Alice, the voter, should obtain *immediate and direct* assurance that her intended vote has "made it" into the tally.

We review Neff's scheme, MarkPledge, and show that it provides ballot casting assurance. We also briefly show how Chaum's "Punchscan" system also provides ballot casting assurance, though under more complicated deployment assumptions. We show how ballot casting assurance opens the door to realistic failure detection and recovery in the middle of an election, a topic which has been ignored in much of the voting literature.

1 Introduction

The secret ballot, sometimes called the Australian ballot, was introduced in US presidential elections only in 1892. The purpose was to prevent vote buying, which had become rampant [12]. There was notable opposition, as forced ballot secrecy significantly reduces public auditability of an election: election officials have to be trusted to collect ballots, maintain their chain of custody, tally them, and report the results appropriately. With modern voting technologies – optical scanning machines, DREs, etc . . . – that further mediate the process of ballot collection and tallying, some activists are calling for an end to the secret ballot, claiming that "Secret ballots and transparency in government are mutually exclusive concepts" [8].

Universal Verifiability. Since the early 1990s, many cryptographic voting protocols have been proposed to provide *universal verifiability* [11, 6, 1, 9]. In these schemes, any observer can verify that only registered voters cast ballots and that cast ballots are tallied correctly. Universal verifiability uses cryptography to restore the bulletin board of yore. Ballots are encrypted and posted along with the voter's plaintext identity. Universal verifiability thus provides a public tally, patching a large portion of the audit-ability hole caused by the secret ballot.

Ballot Casting Assurance. Universal verifiability does not provide complete audit-ability. In addition to knowing that all votes were correctly tallied, Alice would like *direct verification* that *her vote* was properly cast and recorded into this tally. We define this principle as *ballot casting assurance* and argue that it, too, fills an audit-ability gap left by the secret ballot. Interestingly, to the average voter, ballot casting assurance may be more intuitive than universal verifiability, for it is a simple, individually relevant question: independently of all other issues, is Alice certain that her vote "made it?"

Ballot casting assurance is effectively the combination of two existing voting protocol concepts: that ballots should be *cast as intended*, and *recorded as cast*. The added benefit of ballot casting assurance is two-fold:

- **End-to-end verification:** Alice only cares that the recorded ballot matches her intention. Verifying the in-between "ballot casting" is an artifact of how certain voting schemes are verified, and need not be built into a definition of voting system security.
- **Direct verification:** Alice wants direct, not mediated, verification of the correct recording of her vote into the tally. In particular, if Alice must trust election officials to eventually record her vote, the scheme does *not* provide ballot casting assurance.

The Possibility of Failure Recovery. Once ballot casting assurance enables this direct and immediate verification, a new possibility arises: if Alice determines that her vote has not been correctly recorded, she can immediately complain and rectify the situation. Thus, ballot casting assurance is fully achieved when Alice can vote, verify, and revote until verification succeeds.

MarkPledge **Scheme.** We review Neff's scheme for high-probability voter intent verification [2], which we refer to, from now on, as MarkPledge. This is a review of existing material, presented here to clearly illustrate the concept of ballot casting assurance.

At a high level, MarkPledge lets a voting machine prove to the voter that it correctly encrypted the voter's choice, even if the voter's computational ability is highly constrained. Alice's experience inside the voting booth unfolds as follows (for presentation simplicity, we assume a single race):

1. We assume that Alice, the voter, can compare short strings (4 alphanumeric characters.)

2. Alice selects her candidate of choice, candidate $\#j$, using the voting machine's interface.

3. The voting machine produces a specially-formed encryption of Alice's choice, which it prints on a scrolling receipt.

4. The voting machine commits to Alice's candidate of choice by displaying a short string on screen. Alice then enters a short-string challenge. The voting machine then prints out, on the receipt, Alice's challenge, the list of candidates, each with a corresponding short string, and some additional data that effectively completes the proof that Alice's choice was correctly encrypted.

5. Alice immediately checks that

   - the printed challenge on the receipt matches her challenge, and that
   - the string next to her candidate of choice matches the on-screen commitment.

6. Alice keeps the receipt and can later check, using a computer, that the additional data printed by the machine are consistent.

7. No coercer can tell which random short string is the real one, and Alice can't provide any convincing indication either.

Adapting MarkPledge**.** Recall that ballot casting assurance requires that Alice get *direct and immediate* verification that her ballot was correctly recorded. For this purpose, we introduce *helper organizations* at the polling location. These organizations may be political parties or any other organization. The voting process is then augmented as follows:

1. Alice obtains her challenge from a helper organization of her choice.

2. Alice prepares her ballot in the voting booth, obtaining an encrypted receipt at the end.

3. The voting machine immediately posts Alice's encrypted ballot and plaintext identity on the public bulletin board.

4. Alice lends her receipt to any number of helper organizations, who can confirm the validity of the ballot (without learning its plaintext content) and the presence of the corresponding encrypted vote on the bulletin board.

5. If Alice is unsatisfied by the receipt, or if a helper organization finds an error in the receipt, Alice may simply revote, just as she would today in an optical-scan voting system. In fact, this verification is useful both for integrity of the process and for detecting voters' normal, human errors. As Alice's identity is part of the bulletin board post, only her last vote is eventually tabulated.

6. Once Alice is satisfied, she checks out of the polling location, taking her receipt with her for continued verification. The combination of her in-booth experience and the out-of-booth verification by helper organizations provides strong evidence that her intent was correctly recorded. During a prescribed complaint period, Alice may ask to rectify her vote if she discovers that her ballot has disappeared from the bulletin board.

This process is diagrammed in Figure 1.

This Paper. In Section 2, we review the concept of universal verifiability for election auditing, and introduce the ballot casting assurance concept. In Section 3, we review MarkPledge, including cryptographic details not presented in this introduction. In Section 4, we show how to implement ballot casting assurance on top of this scheme and consider the new threat model. In Section 5, we briefly consider ballot casting assurance for other systems, including Chaum's PunchScan, before concluding in Section 6.

2 Auditing an Election

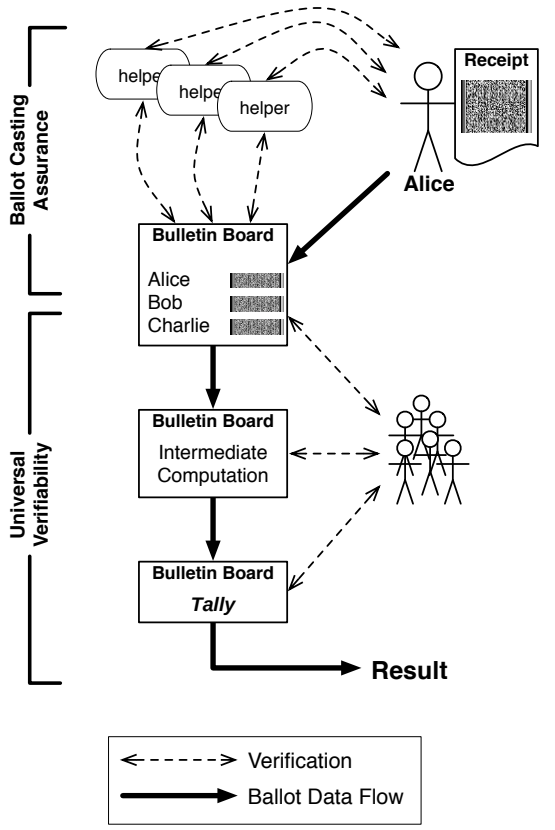

Figure 1: Auditing an Election – Ballot Casting Assurance describes how Alice can be certain that her vote was recorded and posted on the bulletin board as she intended, while Universal Verifiability pertains to the tallying process.

With the introduction of the secret ballot, the public lost the ability to directly audit an election. Once Alice casts her ballot, she also hands off all verification ability to the election administrators. Given the complete dissociation of voter identity and ballot content, even election administrators are left with only crude mechanisms for verification: the dominant measure of election reliability is the residual vote, which indicates only the difference between the total number of votes cast and tallied.

A number of high-level voting system verification goals have been explored in prior literature:

1. **Cast as intended:** the ballot is cast at the polling station as the voter intended.

2. **Recorded as cast:** cast ballots are preserved with integrity through the ballot collection process.

3. **Counted as recorded:** recorded ballots are counted correctly.

4. **Eligible voter verification:** only eligible voters can cast a ballot in the first place.

With classic security processes and methods, it is unclear how to achieve all of these verifiability tasks directly. Instead, current voting practices rely on delegating these tasks to election officials.

Universal Verifiability. Over the last twenty years, numerous cryptographic schemes have been developed to address the *counted-as-recorded* and *eligible voter verification* verification tasks. These schemes are generally said to implement *universal verifiability*, as any observer can verify that all collected ballots have been correctly anonymized, preserved, and tallied.

Generally, these schemes use an authenticated bulletin board, where ballots are posted as ciphertexts together with the voter's plaintext identity. Individual voters can verify that their encrypted ballot is correctly posted on the bulletin board. All observers can check that only eligible voters cast a ballot and watch the tallying process on encrypted votes, which is usually implemented as an anonymizing mixnet or a homomorphic aggregation.

Ballot Casting Assurance. Ballot casting assurance fills the remaining audit-ability gap: *cast as intended* and *recorded as cast*. How can Alice obtain verification that her intent has been recorded appropriately? If she determines that her vote has been recorded incorrectly, how can she rectify the situation? Were it not for the secret ballot, these questions could be answered trivially.

Two important properties of ballot casting assurance should be emphasized:

- **End-to-end verification**: typical voting security analyses distinguish the properties "cast as intended" and "recorded as cast." This distinction is an artifact of a chain-of-custody approach to verification, where each step must be independently verified. Ballot casting assurance need not concern itself with this specific mechanism for verification. The requirement is end-to-end, from the voter's brain to the bulletin board.

- **Direct verification**: Alice, the voter, should get *direct* and *immediate* verification that her vote was correctly recorded. Mediating the verification via election officials, or delaying the verification until it is too late for Alice to rectify the situation, is insufficient.

A number of recent proposals, including Chaum's visual cryptography ballot [3], its variants [4] including PunchScan, and Neff's encrypted receipt scheme

MarkPledge, offer solutions that address these requirements: it is possible to give Alice direct verification of her vote without providing so much information that she can prove her vote to a third party, all given conservative assumptions of the voter's computational ability – i.e. we cannot expect a voter to perform complex math. These schemes provide the cryptographic basis for ballot casting assurance in a secret-ballot setting.

3 The MarkPledge Scheme

In this section, we review the Neff ballot casting scheme [2], which we refer to as MarkPledge, so that we can illustrate how it can be adapted to achieve ballot casting assurance.

3.1 Assumptions and Goals

A voter cannot be expected to do complicated math to verify her vote. Thus, voter expectations are reduced to a realistic minimum: they need only be able to compare short strings, e.g. 4 alphanumeric characters. The goal of the scheme is to achieve optimal correctness according to this computational ability: the voting machine can cheat the voter only by randomly guessing a single correct string out of all possible short strings the voter is assumed to be able to compare.

In other words, if the voter is assumed capable of comparing strings of 4 alphanumeric characters, then the voter can be certain that, with probability $(1 - \frac{1}{35^4}) > (1 - \frac{1}{10^6})$, her ballot was correctly produced. For the rest of this description, we denote the string length in bits as α. (We assume 35 alphanumeric characters, with the 0 removed to prevent ambiguity with O.)

It is worth noting that this level of voter intent verification is high: competing schemes like Chaum's [3] and Ryan's [4] offer only $\frac{1}{2}$ verification probability per voter.

3.2 Special Form of Bit Encryption

Recall that the Exponential El Gamal encryption of m is simply the normal El Gamal encryption of g^m [7]. Thus, the Exponential El Gamal encryptions of 0 and 1 are the El Gamal encryptions of 1 and g, respectively. Note, in particular, that 0 is now in the plaintext domain. We denote Exponential El Gamal encryption and decryption as Enc_{pk} and Dec_{sk}, respectively.

The central component of the scheme is a special form of bit encryption, which we denote BitEnc_{pk}. A single bit b is encoded as an α-length sequence of pairs of ciphertexts, each an encryption using Exponential El Gamal. The special bit encryption function is defined as:

$$\mathsf{BitEnc}_{pk}(b) = \left\{[u_i, v_i]\right\}_{i \in [0,\alpha-1]}$$

where, $\forall i, \mathsf{Dec}_{sk}(u_i) \oplus \mathsf{Dec}_{sk}(v_i) = \bar{b}$. In other words, if $b = 1$, then each pair encodes either $[0, 0]$ or $[1, 1]$, and if $b = 0$, then each pair encodes either $[0, 1]$ or $[1, 0]$.

Proof of $b = 1$. This type of bit encryption then exhibits a particular property, whereby one can prove, in zero knowledge and with soundness $1 - \frac{1}{2^\alpha}$, that a particular $c = \mathsf{BitEnc}_{pk}(b)$ is the bit encryption of 1. This proof protocol is particularly interesting because both the prover's first message and the challenge are short, α-bit-length strings.

Assuming a protocol input $(b, c = \mathsf{BitEnc}_{pk}(b))$ to Prover $\mathcal{P}$ and Verifier $\mathcal{V}$, the proof unfolds as follows:

1. $\mathcal{V}$ sends $commit(\mathsf{chal})$ to $\mathcal{P}$, $|\mathsf{chal}| = \alpha$.
2. $\mathcal{P}$ sends ChosenString to $\mathcal{V}$, where bit i of ChosenString corresponds to the bit encrypted within both elements of pair i in input c. ChosenString is thus α bits long.
3. $\mathcal{V}$ sends chal.
4. For each bit i of chal, $\mathcal{P}$ reveals the randomness for u_i if $\mathsf{chal}_i = 0$ or for v_i if $\mathsf{chal}_i = 1$. This reveals an α-bit string, one bit for each pair within c.
5. $\mathcal{V}$ checks that this string matches ChosenString.

This protocol is clearly complete. Soundness derives from the randomness of chal, which ensures that, if $b = 0$, then $\mathcal{P}$ must guess the single α-bit string that will be revealed by chal. Zero-knowledge follows from the straight-forward simulation: knowing chal in advance, it's trivial to produce the appropriate ChosenString.

Selecting option j out of k. It is then relatively straight-forward to encode option j out of k possible options. For each $i \in [0, k-1]$, create a bit encryption $c_i = \mathsf{BitEnc}_{pk}(b_i)$, where $b_i = 0$ for all i except $b_j = 1$.

3.3 Voting Process

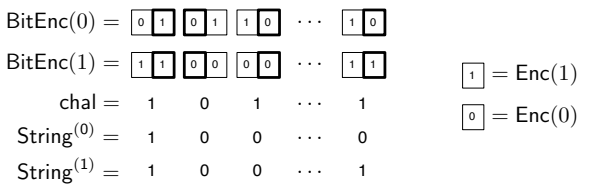

Figure 2: The voting process with bit encryptions.

Alice, the voter, selects option j out of k possible candidates. Then, the voting machine produces k bit encryptions, one for each index, and proves to Alice that it has

correctly encoded her choice j. To keep this explanation simple, we assume that, of the k bit encryptions, only one encodes a 1, and all others encode 0 – in other words that the ballot is well-formed. There are techniques to prove well-formedness, but they are secondary to the goal of this protocol review. The protocol is thus:

1. Alice enters $commit(\mathsf{chal})$.
2. The voting machine displays ChosenString, which matches the encrypted string for b_j, the bit corresponding to Alice's candidate choice. Recall that ChosenString is short (4 alphanumeric characters).
3. Alice enters chal, which must match $commit(\mathsf{chal})$.
4. The voting machine completes the proof that $b_j = 1$. Then, the machine produces simulated transcripts for all other b_i using the same challenge chal, effectively simulating proofs that they, too, are 1. This involves revealing $\mathsf{String}^{(i)}$ for all indices i, which are the would-be ChosenString values for each bit. These strings are printed on Alice's receipt, which already includes the bit encryptions $c_0, c_1, \ldots, c_{k-1}$.
5. Alice verifies that ChosenString appears next to her candidate of choice.
6. The receipt does not reveal which of the $\mathsf{String}^{(i)}$ was the real ChosenString displayed on screen. Note also that Alice cannot convincingly prove which string is the real one.

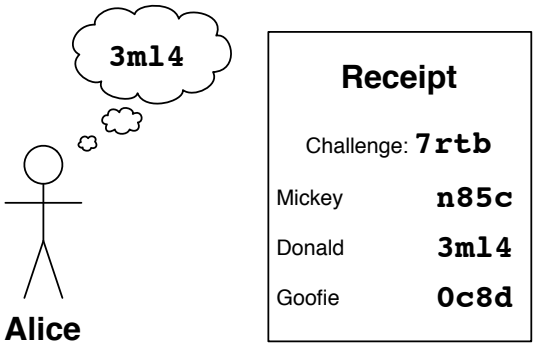

Figure 3: Alice's receipt does not reveal her vote, though it matches her memory of the screen in the voting booth.

4 Implementing Casting Assurance

We now augment MarkPledge with a few procedural tweaks to achieve ballot casting assurance. Alice, the voter, obtains immediate and direct verification that her vote was correctly recorded, and individual revoting for failure recovery becomes a possibility.

4.1 Setup

A voting location should be staffed with a handful of *helper organizations*, e.g. political parties and activist organizations. These organizations are not given any privileged access: they are present merely to help the voter verify proper ballot casting. They may be represented simply by verification machines. We assume that some of these organizations may act maliciously, but that, for each voter, at least one is honest.

4.2 Ballot Casting

1. Alice consults a helper organization of her choice to obtain her "challenge ticket," which includes the committed challenge and the challenge itself.
2. In the isolation booth with the voting machine, Alice proceeds with her ballot selection and verification, as per Section 3. She uses her challenge ticket to provide $commit(\mathsf{chal})$ and chal, using an input device such as a bar code scanner. By the end of the process, Alice has a physical receipt in hand, and she has verified that:
   (a) the verification code next to her candidate of choice matches what she saw on screen, and
   (b) the challenge printed on the receipt matches the challenge on her challenge ticket.

   In addition, the voting machine is now expected to digitally sign the receipt.
3. The voting machine immediately posts the encrypted vote along with Alice's name to the digital bulletin board.
4. Alice hands her receipt to a helper organization of her choice. The helper verifies that the ballot has been correctly posted, that it is internally consistent, and that it is correctly signed by the voting machine.

   If the verification is in any way unsatisfactory to Alice, she can simply return to step 1 and vote again. When she does, her new vote replaces her old vote on the bulletin board. The bulletin board maintains a history of all ballots cast by Alice, noting which is the last one to be used for tallying.

   Note that this revoting process is similar to current procedures for optically-scanned ballots when the voter makes a mistake and requests a new ballot. As an added bonus here, the history of revoting is kept, in case it ever becomes a source of fraud.
5. Once Alice is satisfied with her receipt and the verifications performed by various helpers, she leaves the polling location with her receipt in hand.

6. At her discretion, Alice can leave a copy of her voting receipt with any number of helper organization.

Note that, if it is too onerous to post immediately to the bulletin board and verify, live, that the bulletin board has been updated, the process can be modified so that helper organizations only check the voting machine's digital signature. In this case, it might also be useful to witness election officials verifying the signature, as they can be made liable in case the ballot goes missing at a later point.

4.3 Complaint & Correction Process

After having left the polling location, Alice can check, using software she trusts – even software she may have written herself – that her vote has been correctly posted and tallied. We consider what happens if Alice finds that her ballot has gone missing from the public tally. We consider here the possibility of *late revoting*. This process provides significantly increased ballot casting assurance, though we note that it requires significant care to ensure that no new weaknesses are introduced (See Section 4.5).

Before a prescribed complaint deadline (maybe 24 hours after polls close), Alice can follow these steps:

1. Present her receipt and identification to the complaint center.

2. If the receipt's signature is *incorrect*, the complaint fails and no further action is taken.

3. If the receipt's signature is *correct*, the encrypted ballot is compared to the claimed encrypted ballot on the bulletin board. If it is the same ballot, then there was no mistake, and no further action is taken.

4. If the signature is correct and the bulletin board shows a different ballot than the receipt's, election officials should update the bulletin board to reflect Alice's corrected vote.

5. If election officials refuse to update a vote, the voter may turn to a political helper organization that can perform exactly the same checks as the election official and submit a formal complaint (possibly using the press.)

One should assume that most voters will never care to verify and complain on their own. Thus, as described earlier, Alice has the ability to simply hand her ballot to a verification agent who can do all checks on her behalf. If this agent finds that Alice's vote has been mishandled, it may complain on her behalf.

4.4 Trust Assumptions

The tweaks we propose do not significantly alter the trust assumptions of typical cryptographic voting schemes. We review these assumptions here, and explain what we expect of the newly-introduced *helper organizations.* It is important to distinguish the two major classes of malicious behavior: those which affect *tally correctness*, and those which affect *ballot secrecy*.

Within ballot secrecy, it is also important to distinguish *subliminal channel attacks*, where the election equipment and data violate secrecy, and *side-channel attacks*, where some external equipment – e.g. a cell phone camera – is used to violate the privacy of the voting booth. This latter class of attacks is extremely problematic in any election scenario, and we do not attempt to address it here.

Voting Machines. The voting machine is *not trusted* for correctness. The proof protocol ensures that the voting machine cannot cheat a single voter with more than very small probability ($\frac{1}{10^6}$), which rules out a cheating voting machine having any noticeable impact on the election. In the current model, the voting machine is trusted not to perform subliminal channel attacks on ballot secrecy, though it should be noted that methods exist for preventing such malicious actions, too [2].

Bulletin Board. The bulletin board is *not trusted*: it is only a common conduit for authenticated content from the voting machines. It is expected that the various helper organizations will keep an eye on the bulletin board, including ensuring, via regular auditing, that everyone sees the same view of the bulletin board. This can be performed using well-understood auditing techniques, including hash trees [10].

Helper Organizations. We assume that at least one helper organization is honest and running correct software. Some helper organizations may be dishonest for certain voters, though it is expected that these organizations will be mutually distrusting, such that it is highly unlikely that all organizations will be dishonest for a given voter. It is important to note that, if a helper organization incorrectly verifies a ballot, the voter may easily consult another organization, likely a competing political party. Thus, a helper organization takes a big risk if it lies.

For ballot secrecy, the helper organizations are completely *untrusted*, as they may have strong incentive to coerce voters. Note that, even if a helper organization provides the "challenge ticket," it cannot violate ballot secrecy.

Voter's Home Software. For correctness, we assume that a small fraction of voters will run the correct verification software. We assume that it is very difficult for an adversary to target a particular voter at the polling location, corrupt her helper organizations, and corrupt her home computer simultaneously. We assume that, even if an adversary accomplishes this complex attack against one voter, it isn't easily scalable to more voters.

We assume that all voters are potential adversaries when it comes to coercion. We note that, in any scheme, a voter may take a photo of her ballot using a camera-phone. This is the *side-channel attack* previously mentioned, which no current voting system addresses, and which our proposal does not exacerbate.

4.5 Threats

The verification, complaint, and correction processes significantly increase voter confidence. At the same time, they open up new avenues for attack. With the trust assumptions defined in the previous section, we now outline some possible attacks and approaches to countering them. We do not attempt to address every issue: ballot casting assurance requires much continued research, and we cannot expect to quickly solve all open problems. We also note that, while these issues are important, they are generally less worrisome than the potential for abuse in today's unverified elections.

Incorrect Verification. A helper organization might act maliciously in verifying ballots at the polling location, claiming that good ballots are bad. This may be a particularly useful tactic for one political party to use in a precinct that is known to be dominated by another political party. The system presented here attempts to mitigate such issues by ensuring that *anyone* can perform all ballot verification tasks, thereby ensuring *public oversight* of the system. A helper organization can be required to present objective, verifiable proof that a ballot is malformed.

Refusal to Replace. During the complaint period, malicious election officials may delay or even prevent the replacement of a ballot on the bulletin board, even when presented with a valid voter complaint. To address this, the system once again ensures that *anyone* can perform the ballot verification, which should enable a voter to escalate a complaint based purely on publicly available data. One also notes that, with all ballots now encrypted, the complaint process can be performed in the presence of mutually distrusting observers.

Abusive Replacement. An attacker may attempt to replace an honest voter's ballot during the complaint period. One should note that this is an entirely new threat given the complaint-and-replace process. The system provides a first technical line of defense: the bulletin board records all ballots produced by eligible voting machines. Bob cannot simply use one of his overwritten ballots to replace Alice's ballot, as any bulletin board observer would detect this manipulation. The system further mitigates this risk by adding a strong identification requirement to the ballot replacement process.

One might also consider legal and procedural disincentives, such as requesting signed affidavits of ballot replacement by the complainant, as well as notifying Alice by mail that her ballot was changed in order to enable fraud detection and investigation. This last measure is particularly important to prevent a new kind of ballot box stuffing, whereby a malicious election official might replace a number of ballots just before the close of the complaint period.

Subliminal Channels. As detailed by Karlof et. al. [5], the secret receipt's randomness provides a subliminal channel to a malicious voting machine that wish to leak the ballot plaintext. It will be important to develop methods for removing these channels in future work.

5 Other Implementations

Ballot casting assurance is implementable via other techniques than the one described here.

Chaum Receipts. Chaum's visual cryptography receipts and related schemes – e.g. PunchScan– offer similar functionality to MarkPledge. One notable difference lies in the directness of the verification process: if auditing is centralized prior to election day, as Chaum suggests, Alice cannot verify that her specific ballot was correctly recorded, only that, statistically speaking, no more than a handful of ballots can be corrupted. Even then, this verification is indirect, requiring trust in the election officials.

To truly achieve ballot casting assurance, auditing should be performed by individual voter cut-and-choose: Alice receives two ballots, audits one, and votes with the other. This process is fairly challenging to realize with Chaum's ballots, because every precinct must reveal the secret data required to decrypt half of the ballots (selected randomly and "live") while maintaining the other half secret. That said, given this tweak, Chaum's secret receipts do support ballot casting assurance.

Current Systems. Current election systems, be they DRE, optical scan, or paper based, do *not* provide ballot casting assurance, as they completely separate the

voter's identity at the moment of ballot casting, preventing all future direct verification. Even election administrators are limited in the verification tasks they can accomplish. Many types of fraud – a relatively small number of stuffed or lost ballots – may go undetected. When an inconsistency is detected – e.g. a high residual vote rate – the only possible recovery is to rerun the entire election.

The Impact of Cryptography. An interesting conjecture is that direct verification of secret-ballot elections is only possible using cryptographic techniques. This conjecture should be further explored, and, if found to be true, should provide strong motivation for significant continued research in usable cryptographic voting techniques.

6 Conclusion

We suggest that voting research should consider *ballot casting assurance* as a complement to universal verifiability. While universal verifiability provides global auditability that votes are counted correctly, ballot casting assurance provides individual assurance to Alice that her vote "made it" to the tally process as intended. If an error is detected, Alice can revote until verifiable rectification.

We achieve ballot casting assurance with cryptographic schemes that implement secret receipts. A number of questions remain. How usable are these systems in real-world tests? How will revoting really work? In any case, we believe that this issue should be considered as an integral part of voting system design and evaluation.

7 Acknowledgments

The authors would like to thank Ronald L. Rivest for helpful comments and for the "Casting Assurance" name suggestion, Ted Selker, Susan Hohenberger and Stephen Weis for helpful comments. Ben Adida is supported by the Knight Foundation grant to the Caltech/MIT Voting Technology Project.

References

[1] Masayuki Abe. Universally verifiable MIX with verification work independent of the number of MIX servers. In *Proceedings of EUROCRYPT 1998*. Springer-Verlag, LNCS 1403, 1998.

[2] C. Andrew Neff. Practical High Certainty Intent Verification for Encrypted Votes. `http://votehere.com/vhti/documentation/vsv-2.0.3638.pdf`.

[3] David Chaum. Secret-Ballot Receipts: True Voter-Verifiable Elections. *IEEE Security and Privacy*, 02(1):38–47, 2004.

[4] David Chaum, Peter Y. A. Ryan, and Steve A. Schneider. A practical voter-verifiable election scheme. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *ESORICS*, volume 3679 of *Lecture Notes in Computer Science*, pages 118–139. Springer, 2005.

[5] Chris Karlof and Naveen Sastry and David Wagner. Cryptographic Voting Protocols: A Systems Perspective. In *Fourteenth USENIX Security Symposium (USENIX Security 2005)*, pages 33–50, August 2005.

[6] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Proceedings of EUROCRYPT 1997*. Springer-Verlag, LNCS 1233, 1997.

[7] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31:469–472, 1985.

[8] Lynn Landes. Scrap the secret ballot - return to open voting, November 2005. `http://www.opednews.com/articles/opedne_lynn_lan_051104_scrap_the__secret__b.htm`.

[9] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *CCS 2001*, pages 116–125. ACM Press, 2001.

[10] Ralph C. Merkle. *Secrecy, authentication, and public key systems.* PhD thesis, Stanford University, 1979.

[11] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In *EUROCRYPT*, pages 393–403, 1995.

[12] Wikipedia. Secret Ballot. `http://en.wikipedia.org/wiki/Secret_ballot`, viewed on April 3rd, 2006.