Stamp-It: A Method for Enhancing the Universal Verifiability of E2E Voting Systems

Mridul Nandi[1], Stefan Popoveniuc[2], and Poorvi L. Vora[3]

[1] Department of Computer Science, C.R. Rao AIMSCS Institute
mridul.nandi@gmail.com
[2] KT Consulting
poste@gwu.edu
[3] Department of Computer Science, The George Washington University
poorvi@gwu.edu

Abstract. Existing proposals for end-to-end independently-verifiable (E2E) voting systems require that voters check the presence of a "receipt" on a secure bulletin board. The tally is then computed from all the receipts. Anyone can determine that the computation is correct—that is, the computation of the tally from the receipts is universally-verifiable. The fraud detection probability depends on the number of voters checking their receipts and the number of votes modified. This paper proposes an enhancement, *Stamp-It*, that does not require voters to check published receipts. It allows anyone to determine whether the tally is correctly computed, with probability independent of the number of voters who checked their receipt, extending the universal verifiability of the process. It does not require any additional computations to be performed during the election, and is hence very well-suited for use with the paper-ballot-based E2E systems. Finally, as an add-on, the enhancement does not degrade the original scheme.

1   Introduction

A large number of recent cryptographic voting system proposals [4,7,6,8,2,1,5] enable voters to audit an election outcome without requiring that they trust the voting system or election officials. This paper describes an enhancement that increases the robustness of the verifiability guarantees of the systems.

The general approach of the end-to-end independently verifiable (E2E) voting system proposals is to give each voter a receipt containing an encryption of the vote he or she cast. Through encryption audits, voters may determine that, with high probability, the encryption is correct. The voting system posts all the receipts on a public bulletin board, and each voter can check that his or her receipt is correctly posted. If the receipt is incorrectly posted, or if is not posted at all, the voter can bring his or her receipt as proof of malfeasance. It is assumed that a large enough number of voters do check the correct posting of their receipts, and that the voting system cannot accurately predict which voters will not check their receipts. All the receipts are collectively transformed into a

tally in a manner that (a) allows the general public to check that the tally is indeed derived from the set of receipts while (b) protecting the confidentiality of the ballot. Thus, the presence and correct representation of votes on the public bulletin board is *voter-verifiable*. The correctness of the tally computation—given the correctness of the encrypted votes on the secure bulletin board—is *universally verifiable.*

1.1   Limitations of Current Systems

Note that the ability to verify tally computation is meaningless if the public bulletin board does not contain correct receipts. Hence, the ability to detect fraud requires that (a) a large enough number of voters check their receipts, (b) the voting system cannot predict which voters will not check their receipts, and (c) the set of receipts checked by the voters is the same as the set of receipts used to produce the final tally. These requirements may not always be met. For example, while 30% of the voters checked their receipts in the student election at Université Catholique de Louvain [1], under 4% checked in the 2009 Takoma Park, MD municipal election [3]. Further, individual voters might provide a number of hints to the voting system to indicate whether they are likely to check receipts; for example, paper receipts might be discarded in on site trash cans. Finally, if the bulletin board can distinguish between a voter and an auditor, voters and auditors can be provided different sets of receipts. This may not be difficult, because the voter typically asks for a single ballot and provides its serial number, while an auditor need only ask for the entire set of receipts.

Voters may not be interested in performing any extra steps other than casting the ballot. This should not mean, however, that the election is not verified. Currently, candidates and non-voting observers—some of whom may not be physically present—are left with no means of assessing the validity of an election outcome if voters do not to check their receipts.

1.2   Stamp-It

In this paper, we propose an enhancement, *Stamp-It*, that allows anyone to check that all the receipts have been correctly posted. It assumes that at least one of several election associates does not collude with the voting system, and the integrity adversary is computationally bounded. Stamp-It is an add-on: it does not degrade the voter-verifiability and universal verifiability properties of the original E2E proposals, allowing those properties to hold without requiring that the election associates be trusted, and without requiring that the adversary be computationally bounded. Additionally, for those voting systems that provide computational privacy, Stamp-It does not degrade this property.

The additional functionality of Stamp-It comes about from its use of verification codes, generated by election associates who are trusted to not all collude with the voting system (which may wish to present an incorrect tally without being caught). The verification codes behave as signatures, by the election associates, on the receipts. When used with paper ballot systems, Stamp-It can

be implemented using invisible ink so that the associates are not required to be online during the election, when the codes are provided to voters.

After the election, the voting system posts the verification codes along with the receipts on the bulletin board. Election associates are not required to participate when verification codes and receipts are checked, because associates also provide self-opening commitments to the codes. Anyone can check the self-opening commitments to determine that the codes are associated with the receipts, and, hence, that the receipts are correctly posted. Thus, voters need not later check that the receipt is correctly posted; they may, however, perform this task if they wish to.

From the voter's perspective, the process of obtaining the receipt does not change for electronic-ballot systems. For paper-ballot systems, however, the process gets slightly more complicated with the use of Stamp-It. With respect to the adversarial model, access to the complete ballot before the election affects both the privacy and the integrity guarantees (it affects only the (computational) privacy guarantees of other paper-based protocols). The enhancement does not address the issues of stuffing ballot boxes with additional ballots or of election officials spoiling paper ballots by marking additional votes if these are not addressed by the original system. Finally, when used with a paper-ballot system Stamp-It assumes certain properties of invisible ink (such as those made by the Scantegrity II system, used by the City of Takoma Park for the municipal election of November 2009). In particular, it assumes that the symbols printed with the invisible ink are not visible unless explicitly exposed.

Section 2 presents related work. Section 3 presents an informal description of the voting ceremony and section 4 a formal description of the front end of the system. Section 5 presents the security analysis (universal verifiability and the add-on property) of the enhancement. Section 6 addresses different types of attacks, and modifications that allow for the satisfaction of other properties. Section 7 presents conclusions.

2   Related Work

In this section we provide a brief review of E2E voting systems, and alternate solutions to the problem of receipt checking.

2.1   E2E Voting Systems

The general approach of the end-to-end independently verifiable (E2E) voting systems is to give each voter a receipt containing an encryption of the vote he or she cast. A voter may choose to challenge the encryption [2]; the voting system responds with a proof that it did encrypt the correct vote. For example, the voting machine may reveal the random numbers used for an asymmetric-key encryption. For another example, it might open a commitment made by it before the election, to the substitution cipher used for the particular ballot. If it cannot prove it correctly encrypted the vote cast by the voter, the voter knows it has been cheating. The vote-encryption phase of a cryptographic voting protocol is
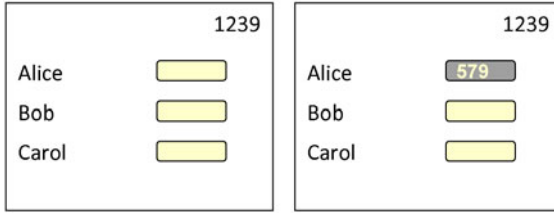
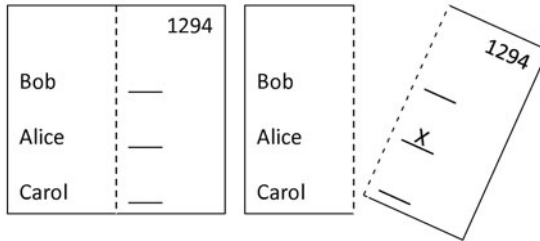**Fig. 1.** Unmarked and Marked Scantegrity Ballots

Fig. 2. Unmarked and marked Prêt à Voter Ballots Showing Receipt

thus an interactive phase: the voter provides the voting system with a choice, the system responds with an encryption of the choice, the voter then communicates whether to cast the encrypted vote or to audit it, and the system responds by casting the vote or providing a proof respectively.

In paper-ballot E2E systems the ballots are designed so as to enable the interaction of the vote-encryption phase without the presence of a computer. Voters are able to encrypt their votes by simply filling out a paper ballot. For example, the Scantegrity ballot looks like a regular optical scan ballot. Printed in invisible ink in the ovals are *confirmation codes* corresponding to the encryption of that choice. When a voter marks an oval, he obtains the encryption of that choice in the form of the confirmation code, see Figure 1. He notes the encryption along with the serial number of the ballot. To audit the encryption the voter fills up all ovals and checks the confirmation codes against the commitments (to the correspondence between confirmation numbers and candidates) provided by the voting system before the election and opened after the election (thus the result of the audit is known after the election). (The voting system opens commitments for only uncast and unspoiled ballots).

In another example, the Prêt à Voter voting system presents the voter with a permuted list of candidates. The position of the voter's mark is an encryption of the vote, and, with the ballot serial number, forms the receipt. (See Figure 2.) A ballot is audited by challenging the system to open commitments corresponding to that ballot, revealing the ordering of the candidates and checking it against that printed.

If an encryption fails an audit; that is, an incorrect vote is encrypted, this is easily proven because the voter's choice is noted on the paper ballot. Electronic ballot systems are considerably weaker from this perspective. If, on being challenged, the voting system reveals a proof of encryption of an incorrect vote, the voter has no way of proving that it is indeed incorrect because he has no record of what he communicated. A paper ballot behaves as a write-once tape recording the protocol interaction between the voter and the voting system and provides proof of the nature of the interaction during an audit. Such a proof cannot be provided by electronic ballots when the encrypting computer is not trusted. (This problem is addressed through the use of procedures for electronic ballot systems).

2.2   Other Approaches to the Problem of Receipt Checking

There are simple approaches to the problem of receipt checking that are not completely satisfactory. We note a few here.

A set of election associates who are trusted to not all collude with the voting system are assigned to the task of checking receipts. Multiple copies of receipts may be made at the polling booth and handed one each to the associates, who then check the receipts online. This trivial solution, however, requires election associates to do all their work after the election, and is hence vulnerable to denial-of-service style attacks. Further, it requires election associates to be present at the poll site, or requires a secure chain of custody on the receipts (the problem of maintaining a secure chain of custody of voted ballots is exactly the kind of issue end-to-end verifiable voting systems seek to address). Stamp-It also uses election associates, but does not require them to be involved after the election.

If it is possible to perform computations at the time of voting, receipt values can be electronically signed by the election associates and the signatures posted on the secure bulletin board with the receipts. Anyone can then check the signatures and verify the presence of receipts on the board without voters checking their own receipts. This, however, requires all election associates to be online during the election, and requires that it be possible for voters to check digital signatures in the polling booth or just outside it. Stamp-It provides similar properties to this approach when used with electronic ballots.

In a manner similar to that used by VoteBox [10] to prevent ballot stuffing, the receipt value can contain, in addition to the encrypted vote, a hash of several previous receipt values. The checking of a single receipt corresponds to the checking of all receipts before it in the chain[1]. In order for this approach to be successful, voters should have a means of verifying the hash values on their receipts while in the polling booth, or, practically speaking, just outside.

When Stamp-It is used with paper-ballot systems, it has properties similar to those of the second solution above. Additionally, Stamp-It does not require any

---

[1] None of the authors were present at EVT/WOTE 2009 rump session, but their understanding is that this approach was proposed by Josh Benaloh at that time.

computations to be performed at the time of voting, and allows for an interactive protocol between the voter and each election associate even though the associates do not need to be online during the election.

3   Informal Description of Stamp-It

3.1   Actors

For each election, we assume the following actors: voters, election officials, election associates, designated printing auditors, public auditors, a public bulletin board. Stamp-It makes the following assumptions about the actors: The election officials do not collude with the voters or the election associates or the designated printing auditors. The election associates do not collude with the printing auditors. The bulletin board is secure and append-only.
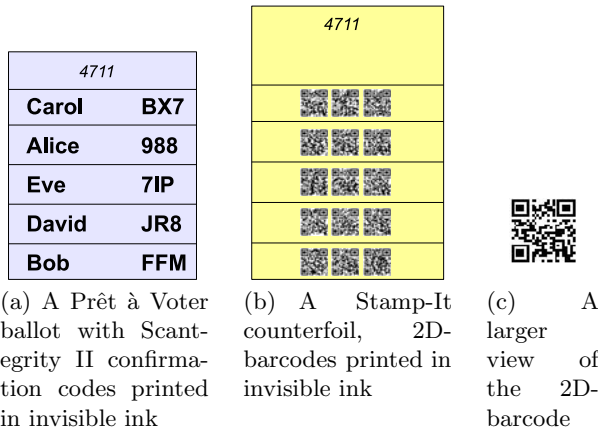
3.2   The Modified Paper Ballot

While the use of Stamp-It is not very complicated when ballots are electronic, when used with paper ballots the paper ballots need to be modified. We exemplify its use with a Prêt à Voter [6] ballot combined with Scantegrity II [5] confirmation numbers. However, other ballot styles can be used, e.g. PunchScan [8]. Electronic ballot styles can also be used if the set of possible (unsigned) receipts is small.

Stamp-It uses a counterfoil for the ballot style of the corresponding end-to-end system. A sample ballot is presented in Figure 3(a) and the corresponding Stamp-It counterfoil is presented in Figure 3(b). The Scantegrity II confirmation numbers shown on the ballot are printed in invisible ink, which is developed when the voter applies a rubber stamp dabbed with developing ink. The voter then sees the confirmation number for the candidate she marked and for no other candidate. Care must be taken to ensure that the rubber stamp has enough ink on it so that the number is exposed.

The Stamp-It counterfoil has a series of verification numbers underneath each candidate, encoded in 2D barcodes also printed in invisible ink—see Figure 3(c). In order to facilitate the exposure of the barcodes corresponding to the chosen candidate, the ballot is printed on "carbon" paper—the ink on the back of the ballot also reacts with the invisible ink, developing it and making it visible. The election authority gets to see the Stamp-It verification barcodes exposed for each ballot, and no other verification barcodes printed on the counterfoil. It is required to publish the exposed barcodes on a public bulletin board along with the Scantegrity confirmation code. The ballot, as the voter sees it, is presented in Figure 4(a). The Stamp-It counterfoil, as the election authority sees it, is presented in Figure 4(b).

3.3   Voting Ceremony as the Voter Sees It

To vote, the voter dabs the rubber stamp into the ink pad, finds his or her favorite candidate and marks the entire rectangle in which the candidate is printed. As in Scantegrity II, a confirmation number will appear next to the marked candidate.

(a) A Prêt à Voter ballot with Scantegrity II confirmation codes printed in invisible ink

(b) A Stamp-It counterfoil, 2D-barcodes printed in invisible ink

(c) A larger view of the 2D-barcode

Fig. 3. The ballot is stacked on top of the Stamp-It counterfoil

(a) A marked ballot, as the voter sees it

(b) A marked Stamp-It counterfoil, as the election authority sees it
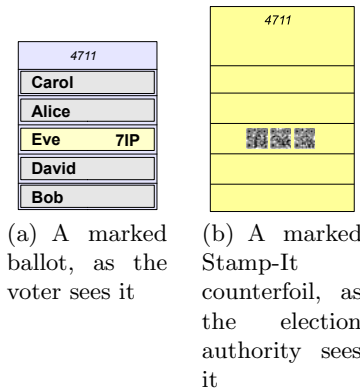
Fig. 4. Marked ballot and Stamp-It counterfoil

The voter then casts her ballot into an optical scan machine or a ballot box, and turns her counterfoil over to an election official. If the voter wishes, she may write down the confirmation number along with the serial number of her ballot. She may later check that these are correctly posted on a public bulletin board, as in Scantegrity II. However, even if no voter does this, Stamp-It provides a secondary mechanism which ensures, given the assumptions mentioned earlier, that the election officials correctly post all the necessary information for the checking of the tally.

Except for the random order of candidates, the voter's experience is largely the same as in currently-used optical scan systems, if he or she does not keep receipts. Additionally, the voter need not do anything after leaving the polling place. If the voter chooses to keep a receipt, the voter's experience is similar to that with Scantegrity (except for the random ordering of candidates).

3.4   What Actually Happens during the Voting Ceremony

The main idea behind Stamp-It is that the election officials only get to see the Stamp-It verification codes for the positions marked by the voters. Thus they can only publish on the public bulletin board these Stamp-It verification codes. Anyone can check, through checking verification code commitments, that the verification code does correspond to the posted position (the Prêt à Voter receipt). While the voter may check that the public bulletin board contains correct receipts, this is no longer necessary.

We describe the steps which are introduced by the use of the Stamp-It counterfoil as an enhancement to an end-to-end verifiable system:

1. Each designated Stamp-It election associate publishes commitments which tie a verification code to a position and a ballot (e.g. designated Stamp-It associate 3, ballot 4711, position 2, verification code 77484888277735910). The verification code is long enough so that it cannot be easily guessed and someone with knowledge of a valid confirmation number would be able to check the commitment without the need for any additional information.
2. The Stamp-It counterfoils are printed collectively by the election associates and given to the election officials as close to the election as possible (this might be as late as just before the election). The verification codes are printed as 2D barcodes in invisible ink on the Stamp-It counterfoils.
3. The election officials prepare the end-to-end setup, and assemble two-page ballots, the bottom page being a Stamp-It counterfoil and the top page being printed on "carbon" paper.
4. Before the election, each designated printing auditor is allowed to choose a number of random ballots (and corresponding counterfoils) and check that the printing on them is according to the commitments.
5. In the process of voting, the voter reveals the Stamp-It verification codes for the position she marks.
6. The ballot is deposited in the ballot box, while the Stamp-It counterfoil is scanned and the revealed Stamp-It verification codes and positions are immediately published by the election officials on the public bulletin board.
7. The Stamp-It counterfoil is shredded.
8. At the end of election day, the election officials produce a public report containing the serial numbers of the voted ballots (and thus used Stamp-It counterfoils). The report is signed and checked by public observers and candidate representatives present at the polling place.
9. The positions that were marked represent the encoded votes (Prêt à Voter receipts) which serve as input to a publicly-verifiable tallying scheme. As these positions are tied to the Stamp-It verification codes through the commitments posted by the election associates, anyone can check their correctness, along with the correctness of the reported tally.

4   Formal Description of Stamp-It

In this section we describe formally the Stamp-It ballot and the protocol. We also provide an analysis of the protocol.

4.1 Formal Description

Convention: Each of a set of c candidates is associated with a member of the set $\mathbb{Z}_c$ based on its rank in an alphabetically-ordered list. (Note that the algebraic structure of $\mathbb{Z}_c$ plays no role here). For example, Alice, Bob and Carol are assigned to be 0, 1 and 2 respectively when they are the only three candidates. One may include "no choice" as one candidate, which will always be last in the list.

   In addition to the election officials, c' independent election associates participate in the voting protocol. Designated print auditors audit the printed ballots and Stamp-It counterfoils. Independently, voters may also optionally audit these.

Publishing public parameters. Before the election, election officials (EOs) make the following parameters public.

1. The number of ballots, N.
2. Specifications (including public keys if any) of cryptographic algorithms such as commitment schemes, digital signature schemes, hash functions etc. used in the protocol.
3. Two code spaces $\mathcal{C}$ (of small size, e.g. set of all strings of three alpha-numeric characters) and $\mathcal{C}'$ (large enough that an exhaustive search is infeasible, e.g. the set of all strings of twenty hexadecimal characters).

Publishing commitments. Election officials (EOs) and the c' election associates make the following commitments and publish them.

1. The ith election associate generates independent random verification codes $C'_{S,i}(p)$ from the code space $\mathcal{C}'$, $1 \leqslant S \leqslant N$, $p \in \mathbb{Z}_c$. All associates publish the hash-outputs of the codes for a pre-selected secure[2] hash function H.
2. For each serial number S, EOs secretly generate random confirmation codes $C_S(p), 0 \leqslant p < c$ and a shuffle or permutation π_S over $\mathbb{Z}_c$. The EOs publish commitments to all confirmation codes and the candidate shuffles π_S. The EOs also publish any other commitments that might be required for the computation of the tally from the confirmation codes (this would depend on the back-end used). Finally, the EOs publish commitments to the entire set of confirmation codes used per ballot, arranged in ascending order.

All the commitments are signed. Commitments to verification codes are self-opening commitments, so that election associates are not required to participate after the election, and are not required for the opening of the commitments to verification codes. Commitments to confirmation codes are not self-opening. This is because we attempt to retain as much as possible of the original Scantegrity design as possible; Scantegrity confirmation codes are short so as to be easy to remember, and commitments to the short codes cannot be self-opening and pre-image secure.

---

[2] The hash function H restricted on $\mathcal{C}'$ is preimage resistant.

Definition 1. Ballot and Stamp-It Counterfoil *A ballot paper associated with serial number S consists of a permutation π_S on $\mathbb{Z}_c$ (candidate shuffle) and a tuple of random codes $\langle C_S(p) \rangle_{p \in \mathbb{Z}_c}$, where $C_S(p) \in \mathcal{C}$. For each position $p \in \mathbb{Z}_c$, $\pi_S(p)$ and $C_S(p)$ denote a candidate and the corresponding confirmation code respectively. Each ballot is associated with a Stamp-It counterfoil printed by a set of c' election associates. The Stamp-It counterfoil associated with serial number S is the tuple of randomly-generated verification codes $\langle C'_{S,i}(p) \rangle_{i,p}$ where $1 \leqslant i \leqslant c'$, $0 \leqslant p < c$ and $C'_{S,i}(p) \in \mathcal{C}'$.*

We describe how a voter votes for a candidate $x \in \mathbb{Z}_c$ on a ballot with serial number S. The verification codes are printed in invisible ink on the Stamp-It counterfoil. That is, verification are not available to the EO before the election.

1. Voter finds $p \in \mathbb{Z}_c$ such that $\pi_S(p) = v$, the candidate of his or her choice. (p is the position of the vote v on the ballot with serial number S)
2. Voter obtains the pth verification codes $C'_{S,i}(p)$, $1 \leqslant i \leqslant c'$ of each election associate from the Stamp-It counterfoil. He also obtains the confirmation code $C_S(p)$ from the ballot.
3. Voter gives the obtained verification codes $C'_S(p) := C'_{S,1}(p) \| \ldots \| C'_{S,c'}(p)$ along with (S, p) to the EO and destroys the Stamp-It counterfoil so that EOs cannot obtain any verification codes that were not exposed.

Step 2 above is implemented by using invisible ink as described in Section 3. When polls close, EOs publish $(S, p, C'_S(p))$ for each cast ballot with serial number S. Knowing π_S, EOs can compute the vote for the serial number S. The back-end would eventually provide a proof that the final result is correctly computed from the all positions p_S.

4.2   Stamp-It as an Enhancement

The following generalization of Stamp-It may be used to enhance other voting protocols. Let $\mathcal{P}$ be a paper-based E2E voting protocol. (The generalization of Stamp-It for an electronic voting protocol is straightforward, where online election associates sign, or provide self-opening commitments to, the voter's receipt.) Let $E : \mathcal{V} \times \mathcal{K} \to \mathcal{E}$ be the encryption algorithm used by the voting protocol, where K, $\mathcal{V}$ and $\mathcal{E}$ denote the key-space, the set of all votes and the set of encrypted votes, respectively. Let $K_1, \ldots, K_N$ be the encryption keys for the encryption algorithm; K_i corresponding to ballot serial number i. The encryption keys are usually implicitly present in the paper-based ballots; see, for example, [9] for a description how several E2E system ballots are of this form. Consider a voting protocol which publishes the encrypted votes where the subset $\mathcal{E}' = E(\mathcal{V}, K_S) \subset |\mathcal{E}|$ corresponding to a ballot S is small (a small multiple of the number of options in a race).

The Stamp-It counterfoil associated with serial number S is the tuple of randomly-generated verification codes $\langle C'_{S,i}(e) \rangle_{i,e}$ where $1 \leqslant i \leqslant c'$, $e \in \mathcal{E}'$ and $C'_{S,i}(e) \in \mathcal{C}'$. The tuple is constructed by c' election associates before the election. (It is assumed that the associates do not all collude with the voting

system to change the election outcome). The associates also commit to the verification codes they generate before the election. For example, the commitments could be based on hash functions. During the casting phase, the voter obtains encrypted vote e corresponding to his vote v and ballot number S. In the enhanced protocol, the voter provides the values of $C'_{S,i}(e)$ for all i to the election official. In addition to all other information required to be posted on the secure bulletin board for $\mathcal{P}$, the election officials also provide these values. Anyone can verify that the supplied verification codes are correct with respect to the commitments posted before the election. When the commitments are simply hashes of the verification codes, the EA need not reveal any additional information after the election.

5   Security Analysis of Stamp-It

5.1   Universal Verifiability Analysis

We assume that the number of ballots cast is exactly the number of receipts on the bulletin board. This may be ensured through a simple procedure, see section 6.

The voter-verifiability property of the enhanced system is clearly no different from that of the original system. As long as voters check confirmation codes uniformly at random, the voter-0verifiability property of the original system holds.

Now we demonstrate that universal verifiability is extended; that is, that anyone can check the correctness of the tally even if voters do not check the bulletin board for receipts. (Voters do perform some verification at the time of voting, however. For example, by casting the vote they verify that the encrypted value was obtained by following the process of casting a vote). For simplicity we assume that the hash function is a random oracle. One may obtain a similar result for a pre-image resistant hash function. We choose $\mathcal{C}'$ large enough such that $N/|\mathcal{C}'| = \delta$ is negligible. The following result holds even if not a single voter checks his or her confirmation code.

Theorem 1. (Universal Verifiability) *If (a) at least one of the c' election associates does not collude with the EOs; (b) $H : \mathcal{C}' \rightarrow \{0,1\}^{hSize}$ is a random oracle; and (c) EOs do not have access to the invisible codes on the Stamp-It counterfoils, then the probability of an undetected mismatch between true cast positions and corresponding published positions is approximately $q \times c \times \delta$ where $\delta = \frac{N}{|\mathcal{C}'|}$ and q is the number of random oracle queries made by the EOs.*

Proof. We have a total of c verification codes per election associate for each of the N ballots, resulting in a total of $M = c \times N$ verification codes whose values are committed to by each election associate. In order for an undetected mismatch between true cast positions and corresponding published positions, the EOs must successfully replace a true verification code, corresponding to a true cast position, with a valid verification code for an uncast but published

position, for all election associates. Suppose all but one of the election associates colludes with the EOs and makes available their printed verification numbers for all positions on all ballots. Then the EOs must guess correctly a valid verification code for the non-colluding associate, for at least one of the ballots. If the EOs make a query to H, the probability that this query corresponds to one of the verification codes is $c \times \delta$. If the EOs make q queries, the probability that one corresponds to a verification code is approximately $q \times c \times \delta$ for q small compared to $(c \times \delta)^{-1}$. (As H is a random oracle, each queried hash value is independent of any other, and of the verification codes). □

5.2   The Stamp-It Enhancement as an Add-on That Does Not Degrade the Security of the Original Scheme

Now we show that the enhancement is simply an add-on that does not degrade the original protocol. In particular, all the security notions present in $\mathcal{P}$ are present in the enhancement. Because we have already demonstrated the verifiability properties of the enhancement, we now provide a short sketch of the fact that the privacy/incoercibility properties of the original scheme are also retained. In order to do so, we show that, given the appropriately-defined inputs, the information obtained from an enhanced protocol can be simulated by an adversary from the information obtained from the original protocol. We assume that the physical process of constructing the ballot does not reveal information about the position of candidates on the receipt layer.

Given a ballot-based voting protocol $\mathcal{P}$, let $\mathcal{P}'$ be the enhanced voting protocol. Let $\mathbf{Cor}'$ be the set of corrupted players in the enhanced protocol $\mathcal{P}'$ and $\mathbf{Cor} = \mathbf{Cor}' \setminus \mathbf{EA}$ the corresponding set of corrupted players in the original protocol $\mathcal{P}$, where $\mathbf{EA}$ is the set of election associates. That is, $\mathbf{Cor}$ and $\mathbf{Cor}'$ are identical except for the possible presence of one or more election associates in the latter. We represent the election associates forming the difference between the two sets by $\mathbf{Cor}_{\mathrm{EA}} = \mathbf{Cor}' \cap \mathbf{EA}$. Let $(x, \mathsf{view}(\mathbf{Cor}))$ and $(x', \mathsf{view}(\mathbf{Cor}'))$ denote the pair of input and view of the corrupted players in $\mathcal{P}$ and $\mathcal{P}'$ respectively. Note that x is a sub-tuple of x' since $\mathbf{Cor} \subseteq \mathbf{Cor}'$.

To prove that $\mathcal{P}'$ is an add-on over $\mathcal{P}$ we need to show that for any corrupted players $\mathbf{Cor}'$ and their inputs x' the view $\mathsf{v}' = \mathsf{view}(\mathbf{Cor}')$ can be efficiently simulated from $\mathsf{v} = \mathsf{view}(\mathbf{Cor})$ and x'. In other words, the view of the corrupted players in the enhanced protocol can be simulated from their inputs and the view of the corresponding corrupted players in the original protocol.

We write $\mathsf{v}' = \mathsf{v} \| \mathsf{v}_{\mathrm{enh}}$ where $\mathsf{v}_{\mathrm{enh}}$ is the additional part of the view due to the enhancement, and is as follows:

- When there is no corrupted election associate, $\mathsf{v}_{\mathrm{enh}}$ consists of the tuple of all published verification codes $\langle C_{S,i}(e_S) \rangle_{S \in \mathcal{S}, i}$ and hash values of all verification codes (i.e. commitments to all verification codes generated by election associates) where $1 \leqslant i \leqslant c'$, S varies over all cast serial numbers and e_S is the encrypted vote corresponding to the ballot number S, i.e. $e_S = E(v_S, K_S)$ where v_S and K_S are the vote and the keys used in the ballot with serial number S.

– When one or more election associates are corrupted, in addition to the above, their verification codes are included as part of their input as well.

We now provide a sketch of the proof that $\langle C_{S,i}(e_S)\rangle_{S \in S, i}$ and the hash values of all verification codes can be simulated from v and x'. Since the inputs of the election associates contain their verification codes, we need to simulate (1) $C_{S,i}(e_S)$ for $i \notin \mathbf{Cor}_{EA}$ and cast ballot serial number S, (2) hash values $\mathrm{Hash}(C_{S,i}(e))$ for all $e \in \mathcal{E}'$, and all serial numbers S. Since $C_{S,i}(e)$'s are generated independently and uniformly (also independently of the keys K_S), $C_{S,i}(E(v_S, K_S))$ is also independent of $E(v_S, K_S)$ for any choice of the vote v_S. Thus $C_{S,i}(e_S)$ are statistically independent of v, which contains $E(v_S, K_S)$ and can be simulated. Similarly, because all codes are independently generated, the simulator can generate all verification codes and output the hashes of these for (2). Thus we have sketched the following useful result for our enhancement Stamp-It.

Theorem 2. *The view of corrupted players in the enhanced protocol can be simulated from their inputs and the view of the corresponding corrupted players in the original scheme.*

6 Discussion

In this section we provide reasoning for some of the choices we made while designing the protocol, and briefly present the alternatives and how they might change system properties. We do not discuss the properties that may have already been discussed by others; for example, the properties of invisible ink.

We first describe a potential threat to the system that leads to a violation of one of the assumptions of Theorem 1, and ways to address it. The assumption that EOs do not have access to printed invisible ink codes on the counterfoil may not always be valid. In particular, it might be possible to read the invisible codes using hyper-spectral equipment, either before or after the election. In order to prevent EOs from replacing one set of verification codes with another, we require that voted serial numbers are committed to at the end of the election, in the presence of election observers and candidate representatives[3], and that voted counterfoils are shredded. We also require that counterfoils are provided as close to election day as possible, and are stored in tamperproof containers. (Note that it is much easier to ensure secure storage of ballots over a single night than it is for the weeks over which ballots are stored for manual audits). One may require additionally that all voted or non-voted counterfoils be returned to election associates, so they may check that these are not tampered with.

We now provide our reasoning for the use of self-opening commitments. With the use of self-opening commitments, election associates are not required to participate after the election as they need not be involved in the opening of the commitments and the checking of the published positions. This requires, however, that the verification codes be long enough so that even a small fraction

---

[3] This is also a requirement for other paper-ballot-based end-to-end systems, such as Scantegrity.

of the space of all verification codes cannot be tested by trial and error by EOs attempting to change the recorded position of the vote, and the corresponding verification code. It also implies that a computationally unbounded adversary (such as an unbounded EO) can correctly and efficiently guess verification codes and replace these (as well as the corresponding positions and hence the votes) with ones of its choice. If no voters check confirmation codes, this change will not be detected. If election associates could be involved after the election to open commitments, it would be possible to use unconditionally hiding commitments and to thus prevent the unbounded adversary from successfully guessing codes in the absence of voters checking confirmation codes. Note that, if a large enough number of voters checks confirmation codes, an unbounded EO would not be able to change verification codes without being detected, because voters would detect a change in confirmation codes.

7   Conclusions

We have presented an enhancement to end-to-end verifiable systems that provides universal verifiability for the end-to-end process (and not just of tally computation) if (a) a set of election associates will perform certain simple tasks before the election, and can be trusted to not all collude with the voting system (or election officials) to change the tally and (b) the integrity adversary is computationally bounded. When these assumptions are not satisfied, the verifiability properties of the original end-to-end system (voter-verifiability of cast vote representation and universal verifiability of the tally when the adversary is computationally unbounded) still hold. Further, if the original scheme provides computational privacy, the add-on preserves this property. The enhancement is simple and allows for election verification in the absence of voter-verification, without putting much of a burden on election associates. Because associates are not required to perform any tasks after the election, the election is not vulnerable to a denial of service attack by the associates. For this reason, and because it does not require computations to be performed at the time of voting, it is more robust than other simpler proposals.

Acknowledgements

References

1. Adida, B., deMarneffe, O., Pereira, O., Quisquater, J.-J.: Electing a university president using open-audit voting: Analysis of real-world use of helios. In: IAVoSS Workshop On Trustworthy Elections (WOTE 2009), Montreal, Canada (August 2009)

2. Benaloh, J.: Simple verifiable elections. In: Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop, EVT 2006, p. 5. USENIX Association, Berkeley (2006)
3. Carback, R., Chaum, D., Clark, J., Conway, J., Essex, A., Herrnson, P.S., Mayberry, T., Popoveniuc, S., Rivest, R.L., Shen, E., Sherman, A.T., Vora, P.L.: Scantegrity II Municipal Election at Takoma Park: The First E2E Binding Governmental Election with Ballot Privacy. In: Proceedings of the 19th USENIX Security Symposium (August 2010)
4. Chaum, D.: Secret-ballot receipts: True voter-verifiable elections. IEEE Security and Privacy, 38–47 (January/February 2004)
5. Chaum, D., Carback, R., Clark, J., Essex, A., Popoveniuc, S., Rivest, R.L., Ryan, P.Y.A., Shen, E., Sherman, A.T.: Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In: Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop, EVT 2007. USENIX Association (2008)
6. Chaum, D., Ryan, P.Y.A., Schneider, S.: A practical voter-verifiable election scheme. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
7. Andrew Neff, C.: Practical high certainty intent verification for encrypted votes (2004), `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.1006&rep=rep1&type=pdf`
8. Popoveniuc, S., Hosp, B.: An introduction to PunchScan. In: IAVoSS Workshop On Trustworthy Elections (WOTE 2006), Robinson College, Cambridge UK (June 2006)
9. Popoveniuc, S., Vora, P.: A framework for secure electronic voting. In: IAVoSS Workshop On Trustworthy Elections (WOTE 2008), Katholieke Universiteit, Leuven, Belgium (July 2008)
10. Sandler, D., Derr, K., Wallach, D.S.: Votebox: a tamper-evident, verifiable electronic voting system. In: Proceedings of the 17th USENIX Security Symposium, SS 2008, pp. 349–364. USENIX Association, Berkeley (2008)