

Single Layer Optical-scan Voting with Fully Distributed Trust^{*}

Aleksander Essex¹, Christian Henrich², and Urs Hengartner¹

¹ Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada N2L 2G1
{aessex, uhengart}@cs.uwaterloo.ca
² Institut für Kryptographie und Sicherheit/EISS
Karlsruhe Institute of Technology
76128 Karlsruhe, Germany
christian.henrich@kit.edu

Abstract

We present a new approach for cryptographic end-to-end verifiable optical-scan voting. Ours is the first that does not rely on a single point of trust to protect ballot secrecy while *simultaneously* offering a conventional single layer ballot form and unencrypted paper trail. We present two systems following this approach. The first system uses ballots with randomized confirmation codes and a physical in-person dispute resolution procedure. The second system improves upon the first by offering an informational dispute resolution procedure and a public paper audit trail through the use of self-blanking invisible ink confirmation codes. We then present a security analysis of the improved system.

1 Introduction

Research into cryptographically “end-to-end” verifiable *optical-scan* voting systems has come a long way toward practicality. This progress has not come easily: academics and election administrators often struggle to agree on a vast and often orthogonal set of core system properties. Similar in spirit to Benaloh [2], we advocate the *coexistence* of modern cryptographic proofs of correctness and conventional, lower-tech, methods for auditing elections. In this paper we tackle a long standing trade-off of properties in the voting literature: distributed trust versus a conventional optical-scan paper ballot form.

Typically cryptographic voting schemes allow the voter to construct a receipt of their vote enabling each voter to confirm the inclusion of their ballot in the election tally. In order to protect ballot secrecy, the association between a receipt and the corresponding (clear-text) vote must be kept hidden at all times. Many proposals have relied on trusted entities or hardware to enforce this, especially with regards to ballot printing. Other proposals distribute trust among multiple entities through the use of specialized multi layer ballot forms.

Our Proposal We consider a list of requirements for end-to-end verifiable optical scan voting that factors a diverse set of stakeholders (i.e., cryptographers, election officials, legislators, democracy groups, etc). This list is by no means exhaustive and does not encompass challenges faced by other voting methods (e.g., internet, mail-in, etc). Our list is as follows:

1. **Distributed trust:** No single party, *including* the ballot printer(s), gains an advantage in deducing how a voter voted or in linking a receipt to its corresponding clear-text vote. This is a vital requirement of any secret ballot election employing the receipt paradigm.
2. **Single layer ballot form:** A ballot is a single sheet of paper with a *fixed order* candidate list³ and the voter marks the optical scan ovals *directly beside* their chosen candidate. Multi layer ballots are an artifact of cryptographic voting, requiring voters to re-learn how to cast a ballot. Our experience in running real-world cryptographic

^{*} An extended abstract of this paper appeared at the 3rd international conference on E-voting and Identity (VoteID 2011).

³ There are also potential advantages to using ballots with randomized candidate lists. Our system can accommodate this approach with minor protocol changes.

elections—both with single layer and with multi layer ballot forms—has indicated to us that multi layer ballots are more cumbersome for voters and more difficult to administer for election officials [15, 5, 44].

3. **Human-readable paper audit trail:** Pursuant to the legal requirements of many jurisdictions voting, voting intent remains plainly evident on cast ballot forms. Such an audit trail also allows for recoverability in the event of lost or forgotten cryptographic keys or other unforeseen errors.
4. **Public paper audit trail:** The collection of cast ballot forms (i.e., the *paper audit trail*) can be made public without revealing the link between receipt and clear-text vote. A public audit paper trail may also be a legal requirement and is critical in protecting ballot secrecy during a manual recount.

In this paper we propose two novel end-to-end verifiable optical scan voting systems that meet all four of these requirements. Some of these properties have been examined in the literature, but no proposal has achieved all of them. Scantegrity achieves 2 and 3 [9, 7]. Prêt à Voter and Scratch & Vote achieve 2 and 4 [11, 40, 1, 45], two Punchscan variants achieve only 4 [20, 24], and each of Split-Ballot Voting, ClearVote and Kusters *et al.* achieve 1 and 4 [31, 37, 25]. A proposal due to Benaloh [2] achieves 2, 3, and 4. See the section on related work for additional discussion.

Contributions We present two novel systems for single layer optical-scan voting with distributed trust based respectively on the ballot styles used by Scantegrity [9] and Scantegrity II [7].

Basic System: We propose a basic two-party system for creating ballot forms with randomized confirmation codes that meets properties 1, 2, and 3. It relies on a private paper audit trail and an in-person physical dispute-resolution procedure.

Improved System: We then propose an improved two-party system that uses ‘self-blanking’ invisible ink confirmation codes. It improves on the basic system by allowing the paper audit trail to be made public, thereby achieving all four properties. In addition it offers an *informational* dispute-resolution procedure allowing disputes to be resolved based on knowledge of a confirmation code (as opposed to physical possession of a receipt).

2 Preliminaries

2.1 Physical Primitives

End-to-end verifiable ballots often employ physical security methods as part of the receipt creation process. The use of physical security mechanisms can be contentious due to inherent questions regarding their cost, feasibility, and real-world security properties. However, there is precedent for protocols built around *ideal* physical security mechanisms (c.f. [19, 30]). Throughout the rest of this paper we assume that all physical security mechanisms function ideally. Broadly speaking the ballot secrecy properties of our systems reduce to those of Scantegrity’s when the physical security mechanism fail.

Physical Security Mechanisms We briefly summarize the physical security mechanisms employed by our systems.

Invisible ink as its name implies is initially invisible when printed and becomes visible only after *activation*. It was proposed for use in the Scantegrity II system [7], and has been implemented and fielded in a live municipal election in the United States [5]. For the improved system presented in Section 5 we additionally make use of a ‘slow’ developing ink,

Scratch-off coating is a convenient, cost-effective and widely available method for concealing (and subsequently revealing) printed information. It has been employed in several voting schemes (cf. [1, 42]) to protect ballot secrecy,

Visual cryptography [32] is a well known technique for visually implementing a logical exclusive disjunction (i.e., an *xor*) built from a physical medium acting as a logical disjunction (i.e., an *or*). A message or graphical image can be split into two or more information-theoretically secure shares. When the shares are combined (i.e., overlaid) the message becomes visually perceptible.

Physical Security Sub-protocols We briefly summarize the physical security sub-protocols used by our systems.

Document Authenticity: We require a method for determining a document’s authenticity. Classical methods for anti-counterfeiting (e.g., watermarks, holographic foil, embedded magnetic strips, etc) can be cost-prohibitive. Paper fibre analysis (cf. [13]) using commercial-grade scanners is possible⁴. For the sake of our description we assume that there exists an efficient physical scheme for determining a ballot’s authenticity,

Private Printing: we make use of private printing techniques to pick and print *human-readable* confirmation codes on ballots without either printer individually knowing which codes were printed. A proposal for two-party private printing was made in [16]. Private printing is used in the improved system.

2.2 Cryptographic Primitives

We briefly outline the main cryptographic primitives used by our systems. We note that these primitives are standard across the cryptographic voting literature.

Homomorphic Encryption Let $\langle \text{DKG}, \text{Enc}, \text{DDec} \rangle$ be a *distributed public-key encryption* scheme. Without loss of generality, DKG generates two private key shares x_1 and x_2 for parties \mathcal{P}_1 and \mathcal{P}_2 respectively and a joint public key Y . Encryption $\llbracket m \rrbracket = \text{Enc}_Y(m, r)$ is semantically secure and homomorphic in at least one operation. Decryption $m = \text{DDec}_{(x_1, x_2)}(\llbracket m \rrbracket)$ requires both key shares. Specifically we will make use of exponential Elgamal [14] with distributed decryption [36]. For simplicity we will omit the public-key when implied. We additionally require a *partially-homomorphic xor* operation $\tilde{\oplus}$ such that, for a pair of messages $m_1, m_2 \in \{0, 1\}$, $\llbracket m_1 \rrbracket \tilde{\oplus} m_2$ produces a ciphertext that encrypts the bitwise xor of the associated plaintext bits, i.e., $\llbracket m_1 \oplus m_2 \rrbracket$. We now present a bit encryption scheme based on exponential Elgamal though there is more than one way to accomplish this (cf. [22, 33]).

A Partially Homomorphic bitwise XOR with Exponential Elgamal For two bits $m_1, m_2 \in \{0, 1\}$ and their associated encryptions, we describe a method based on exponential Elgamal to implement a *partially homomorphic* operation $\tilde{\oplus}$ for which $\llbracket m_1 \rrbracket \tilde{\oplus} m_2$ produces a ciphertext that encrypts the bitwise xor of the associated plaintext bits, i.e., $\llbracket m_1 \oplus m_2 \rrbracket$.

The first party constructs a ciphertext $c = \langle c_1, c_2 \rangle = \langle g^r, g^m y^r \rangle$ for $m \in \{0, 1\}$ and transmits c to the second party. The second party will select their bit $m' \in \{0, 1\}$ and compute the partially-homomorphic xor, $c \tilde{\oplus} m' = \text{PHX}(c, m')$ where,

$$\text{PHX}(c, m) = \begin{cases} \text{ReRand}(c) & m = 0 \\ \text{ReRand}(\langle c_1^{-1}, g^1 c_2^{-1} \rangle) & m = 1 \end{cases}$$

This scheme is essentially the same the inversion scheme used by Neff in [33]. Importantly, this approach alone only provides security against passive adversaries: the first party could construct a malformed ciphertext, while the second party could throw away the first party’s contribution all together. The simplest way to provide integrity would be to run a cut-and-choose protocol whereby both parties output numerous instances of $\langle c_1, c_2 \rangle = \text{PHX}(c_1, m_2)$ and conduct a coin-toss protocol to select some instances to challenge. Parties would open the challenged instances (revealing their respective message bits and random factors), and retain the unopened ones for further use.

Mixnets Mixnets have long been a fixture in cryptographic voting. We make use of a simple re-encryption mixnet (cf. [34]) structure to create our proofs (we do not utilize a separate proof of correct mixing, as it is provided by other parts of our system). *Re-randomization* (a.k.a., re-encryption) of a ciphertext c is accomplished by computing $c' = \text{ReRand}(c, r) = c \cdot \text{Enc}(0, r)$ ⁵. By rerandomizing and shuffling a batch of ciphertexts we implement a simple *reencryption mixnet*, Mix. In this paper, when applying Mix to a matrix of ciphertexts, we describe mixing as occurring on *tuples* of ciphertexts grouped by columns and shuffled by rows.

⁴ In general there are privacy threats due to fingerprinting documents however this is not a threat to ballot secrecy assuming non-collusion.

⁵ Replace 0 with the *identity* element for other groups.

Commitments We use a cryptographic *commitment* scheme to commit to permutations as part of a cut-and-choose proof of shuffle. The dispute resolution procedure in the improved system requires the prover to either unveil (i.e., *de-commit*) the code, or alternatively to issue a *non-interactive proof of plaintext inequality*. A commitment inherent to IND-CPA secure encryption fits this dual role. Here a sender *commits* to a message m by posting its encryption $\llbracket m \rrbracket = \text{Enc}(m, r)$. Later the commitment can be unveiled when the sender reveals an m', r' , allowing anyone to verify $\text{Enc}(m', r') = \llbracket m \rrbracket$, and hence $m' = m$. This approach is commonly used in several voting schemes (e.g., [3, 1, 43]).

Non-interactive Challenges As part of our cut-and-choose correctness proof we require a method for fairly generating random challenge bits. Loosely speaking, *fairness*, requires that no one is able to predict, or controllably influence the output with non-negligible advantage. Furthermore, the fairness of the method should be *convincing* to voters. Both the heuristic due to Fiat and Shamir [18], and the notion of a *random beacon* (cf. [39, 12]) are possibilities.

2.3 Participants

There are several entities that participate in the election.

- A set of **voters** with the authority to cast a ballot in the election, optionally construct a privacy-preserving receipt of their vote, and optionally participate in an election audit,
- An **election operations commission** \mathcal{C} with the capability and authority to organize and run an election, operate a polling place, optically scan ballots, report results, act as a custodian of the cast ballot record, and participate in an in-person dispute resolution procedure,
- Two independent **ballot printers** $\mathcal{P}_1, \mathcal{P}_2$ who possess the capability and authority to print documents in the untrusted printing model and participate in a secure (cryptographic) two-party computation,
- An election **scrutineer** \mathcal{S} with the authority to audit the correctness of printed ballots relative to their cryptographic representation. Additionally \mathcal{S} acts as a proxy for voters during disputes with \mathcal{C} to protect their identity. In practice there might be any number of election auditors, representing the candidates or other democracy groups.

As a fundamental requirement of our security model, we assume that neither printer nor election commission collude with one another.

3 The Basic System

The basic system produces a public and universally verifiable cryptographic proof attesting to the correctness of the election’s outcome. This correctness proof is based on standard cut-and-choose techniques (cf. [9, 7, 8]). Without loss of generality we consider a single-contest election involving n ballots⁶ and m candidates. The basic system involves several protocols. The protocols `generateBallots`, `preElectionPrep`, `postElectionPrep` encompass the preparation for the public election audits. Note that each of these protocols taken individually is only secure in an *honest-but-curious* setting. To make them robust against an active adversary we make use of a set of *audit* protocols `proveScan`, `proveReceipt`, `provePrinting` and `resolveDispute`. A summary of notations used is presented in Table 1.

The Ballot The basic optical-scan paper ballot form has a pre-printed, fixed-order candidate list $L = \{l_1 \dots l_m\}$. Adjacent to each candidate is an optical scan oval with a *mark state* $\mu \in \{0, 1\}$ corresponding respectively to whether the oval was unmarked or marked. The ballot form is separated into two regions by a perforation. The top constitutes the *ballot portion*, and the bottom is the *receipt portion*. An alphabet Σ of m confirmation codes is defined. Each optical scan oval (and hence each candidate) is associated with a confirmation code drawn independently at random, and without replacement, from Σ . A *ballot-id* b is a d -bit⁷ vector printed on the ballot portion. An independent *receipt-id* r is printed on the receipt portion. The first printer prints the receipt-ids under a scratch-off coating and the second

⁶ The number of ballots printed is the total number of voters times a *heuristically* chosen expansion factor to account for audited and spoiled ballots.

⁷ Since in the basic scheme ballot-ids are the xor of random bit vectors, d is chosen to be large enough so as to make duplicate ballot-ids highly unlikely.

n	Number of ballots to print	T	List of all ballot-tuples
m	Number of candidates	BallotTable	Table of ballot information
d	Bit-length of ballot-id	ReceiptTable	Table of receipt information
L	List of candidate names	MP_1/MP_2	Printer 1/2's master permutation
Σ	Confirmation code alphabet	π/ρ	Random perm'n's composing to MP_1
α	Soundness parameter	σ/τ	Random perm'n's composing to MP_2
b/B	Ballot-id/list of ...	MidMarks	Intermediate mark state list
r/R	Receipt-id/list of ...	MidMarksP1	\mathcal{P}_1 's intermediate mark state list
c/C	Confirmation code/list of ...	MidMarksP2	\mathcal{P}_2 's intermediate mark state list
μ	Mark-state of opscan oval	eid	Election-unique identifier

Table 1: Notations

prints the confirmation codes. Both printers will jointly print the ballot-id in invisible ink. Printing of the ballot- and receipt-ids is done such that each printer only knows what *it* prints (and not what its counterpart prints). The basic ballot is depicted in Figure 1(a).

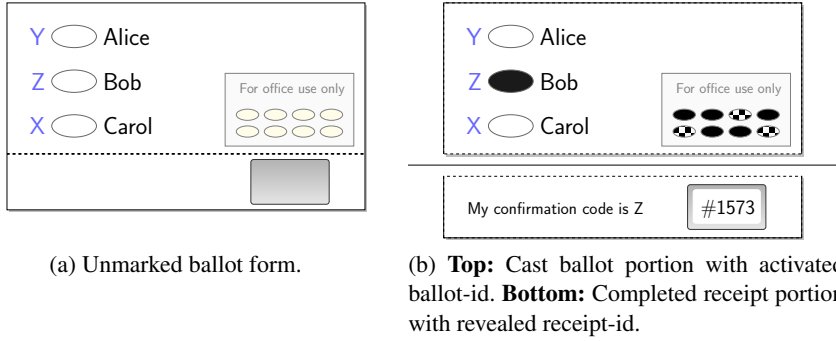


Fig. 1: **Basic ballot**: Optical-scan ballot form with *ballot portion* (top) and tear-off *receipt portion* (bottom) depicting a randomized confirmation code list, a unique ballot-id printed in *invisible ink visual-crypto* and a unique receipt-id beneath a scratch-off coating. Ballot printing is distributed between two printers such that neither can match receipts with cast ballots.

Ballot tuple A ballot is fully specified by the tuple $\{b, r, c\}$, which denotes the association between a unique *ballot-id* bit vector $b \in \{0, 1\}^d$, a unique *receipt-id* $r \in \{1 \dots n\}$, and a random permutation of *confirmation codes* $c = \pi(\Sigma)$ for a permutation π drawn independently and uniformly at random from the set of possible permutations of Σ .

3.1 Election Preparation

The election is initialized as follows: election commission \mathcal{C} initializes a *public bulletin board* \mathcal{BB} ⁸ and a unique election identifier eid . Printers \mathcal{P}_1 and \mathcal{P}_2 jointly run DKG. They post the public key Y to public bulletin board \mathcal{BB} and retain their respective private key shares x_1, x_2 . This list of public parameters $\text{pubParam} = \{n, m, d, L, \Sigma, \alpha, eid, Y\}$ is posted to \mathcal{BB} . All functions/protocols accept pubParam as input.

⁸ Typically modelled as an append-only broadcast channel with state (cf. [4]).

Ballot Tuple Creation The printers now jointly generate encrypted ballot tuples by running `generateBallots`. This protocol is given in Algorithm 1.

Algorithm 1: `generateBallots`

Participants: Printers $\mathcal{P}_1, \mathcal{P}_2$

1 **Printer \mathcal{P}_1 should:**

2 **for** $i \in \{1 \dots n\}$ **do**

3 Encrypt vectors of random bits:

4 $B'(i) \leftarrow (\text{Enc}(\text{randBit}), \dots, \text{Enc}(\text{randBit}))$

5 Post a non-malleable commitment to each `randBit` along with the random factor used to encrypt it.

6 Encrypt and shuffle receipt-ids:

7 $R \leftarrow \text{Shuffle}(\text{Enc}(1) \dots \text{Enc}(n))$

8 **end**

9 **Printer \mathcal{P}_2 should:**

10 **for** $i \in \{1 \dots n\}$ **do**

11 Randomly shuffle and encrypt code confirmation codes:

12 $C(i) \leftarrow \text{Shuffle}(\text{Enc}(\Sigma(1)) \dots \text{Enc}(\Sigma(m)))$

13 **end**

14 **Both Printers should:**

15 Simultaneously and respectively output B', R and C to BB .

16 **end**

17 **Printer \mathcal{P}_2 should:**

18 **for** $i \in \{1 \dots n\}; j \in \{1 \dots d\}$ **do**

19 Homomorphically xor random bits:

20 $b'_1 \dots b'_d \leftarrow B'(i)$

21 $B(i) \leftarrow (b'_1 \tilde{\oplus} \text{randBit}, \dots, b'_d \tilde{\oplus} \text{randBit})$

22 Post a non-malleable commitment to each `randBit` along with the random factor used in computing the xor.

23 Output B to \mathcal{P}_1

24 **end**

// Remark: `Shuffle(X)` applies a permutation to a list X , drawn independently and uniformly randomly from the set of permutations of size $|X|$. `randBit` returns a single bit drawn independently and uniformly at random. It is possible that \mathcal{P}_2 might attempt to maliciously select its bits as a function of \mathcal{P}_1 's. However \mathcal{P}_2 will not know (beyond a guess) what to print on the ballot, and will be caught in `ProvePrinting` with statistical certainty.

Ballot Printing The n ballot forms are printed in three steps. For each ballot-tuple a paper ballot is prepared in the following order:

- **Static background:** directions, candidate names, etc, printed in black ink,
- \mathcal{P}_1 's **share:** the receipt-id is printed and concealed under scratch-off coating, \mathcal{P}_1 's share of the ballot-id printed in invisible ink visual-crypto,
- \mathcal{P}_1 's **share:** the confirmation codes are printed in regular ink, \mathcal{P}_2 's share of the ballot-id printed in invisible ink visual-crypto over \mathcal{P}_1 's share.

The completed ballot forms are then randomly shuffled and delivered into the custody of the election commission \mathcal{C} . Throughout the ballot printing and voting phases the printers will conduct random audits of ballot forms to ensure their authenticity and to look for signs of tampering (e.g, to catch if someone reveals the secret information then replaces the ballot with a replica). Note that if either printer prints something *other* than their contribution in `generateBallots` (e.g., if a printer prints an all-black VC pixel), this will be caught in `provePrinting` with statistical confidence dependent on the number of audited ballots.

Pre-Election Proof Preparation The printers initialize the public audit dataset and cut-and-choose correctness proofs by running `preElectionPrep`. This protocol is given in Algorithm 2.

Voting and Receipt Creation An individual wishing to vote shall attend the polling place and authenticate themselves to \mathcal{C} . All qualified and authenticated individuals (i.e., voters) are then eligible to receive a ballot. The voter selects a ballot form at random from a stack of unmarked ballot forms and takes it, a regular (black) marking pen, and a privacy sleeve into a private voting booth. The voter marks the oval next to their preferred candidate l_i on the ballot portion. Then, if they so choose, the voter creates a receipt of their vote by noting the code letter c_i and writes it in the appropriate space on the receipt portion. The voter then places the marked ballot form into the privacy sleeve and returns it to the poll worker. The poll worker confirms the receipt-id's scratch-off coating is still intact and the ballot-id has not been activated (rejecting the ballot in such a case), then detaches the receipt portion and places it on a table in view of the voter. The ballot portion is then fed into the optical scanner. If the ballot is accepted the receipt portion is retained by the poll worker. If the ballot portion is successfully cast, the receipt portion is returned to the voter and the voting process is complete. A diagram showing completed ballot and receipt portions is depicted in Figure 1(b).

A Note about Timing Attacks In some jurisdictions, poll workers keep a poll book of voter identities in the *order they voted*. If the scanner were to likewise maintain the order of cast ballots it, taken along with the poll book, would compromise ballot secrecy. Since in our case the ballot is drawn at random from the pile, and the poll worker does not see the ballot- or receipt-ids, this threat can be mitigated by having voters cast ballots into a ballot box at the polling place and then scanning them later at a central location.

Post-Election Proof Preparation After the election \mathcal{C} populates the `BallotTable` with the mark state information collected by the optical scanners. With this data the printers and can now finalize the cut-and-choose correctness proof by running `postElectionPrep`. This protocol is given in Algorithm 3.

4 Verifying the election

4.1 Election Audits

There are three simultaneous properties that must be proven in order for the overall results to be proven correct. These audits include,

- **Proving correct mark-state reporting by \mathcal{C} :** Using their receipt, a voter \mathcal{V} checks whether \mathcal{C} correctly registered their vote by running `proveScan`,
- **Proving mark-state propagation by $\mathcal{P}_1, \mathcal{P}_2$:** The printers prove to any interested party that they honestly applied their master permutations to mark state information in `BallotTable` by running `proveReceipt`,
- **Proving printed ballot forms match \mathcal{BB} :** A scrutineer \mathcal{S}^9 runs `provePrinting` with the printers to verify that the ballot tuple information conveyed by the paper ballot forms *matches* the ballot tuple representation in \mathcal{BB} . Audited ballots are *spoiled* and not counted.

The audit protocols `proveScan`, `provePrinting` and `proveReceipt` employed by the basic scheme are below in listed in Algorithms 4, 5 and 6 respectively.

4.2 Dispute Resolution

Because the receipt generation process is unsupervised, a number of possibilities for disputes may arise between the bulletin board and the voters. A simple dispute resolution procedure was proposed for Scantegrity I in [9]. Briefly, this involved a two-stage physical protocol using special privacy sleeves.

⁹ A scrutineer is not strictly necessary. Voters themselves may choose to initiate this audit, although in our experience they rarely do!

Algorithm 2: preElectionPrep

Participants: Printers $\mathcal{P}_1, \mathcal{P}_2$

Public Input: Candidate list L

Private Input: Lists of encrypted ballot-ids B , receipt-ids R , and code shuffles C

1 Both Printers should:

//Expand the n ballot tuples into a table of mn rows (one for every candidate on every ballot):

2 for $i \in \{0 \dots n - 1\}$ **do**

3 $c_1 \dots c_m \leftarrow C(i)$

4 **for** $0 \leq j \leq m - 1$ **do**

5 $T(1, mi + j) \leftarrow B(i)$

6 $T(2, mi + j) \leftarrow \text{Enc}(L(j + 1))$

7 $T(3, mi + j) \leftarrow R(i)$

8 $T(4, mi + j) \leftarrow c_j$

// \mathcal{P}_1 followed by \mathcal{P}_2 using master permutations MP_1 and MP_2 respectively:

9 $T' \leftarrow \text{Mix}(T)$

//Create ballot and receipt tables:

10 BallotTable $\leftarrow \text{DDec}(T'(1 \dots 2, :))$

11 ReceiptTable $\leftarrow \text{DDec}(\text{Mix}(T'(3 \dots 4, :)))$

12 Post BallotTable, ReceiptTable to \mathcal{BB}

13 end

//Prepare cut-and-choose proof of correspondence between elements in the ballot and receipt tables:

14 Printer \mathcal{P}_1 should:

15 for $i \in \{1 \dots \alpha\}$ **do**

16 Choose $\pi_i \in_R \Pi_{mn}$

17 Set ρ_i such that $\rho_i \circ \pi_i = \text{MP}_1$

18 Post Commit(π_i), Commit(ρ_i) to \mathcal{BB}

19 end

20 Printer \mathcal{P}_2 should:

21 for $i \in \{1 \dots \alpha\}$ **do**

22 Choose $\sigma_i \in_R \Pi_{mn}$

23 Set τ_i such that $\tau_i \circ \sigma_i = \text{MP}_2$

24 Post Commit(σ_i), Commit(τ_i) to \mathcal{BB}

25 end

//Remark: Let $x \in_r \Pi_y$ denote a permutation function x drawn independently and uniformly at random from the set of permutations of list of y elements. Let $\text{MP}_1, \text{MP}_2 \in_R \Pi_{mn}$. Then for $i \in \{1 \dots \alpha\}$, we have

$\tau_i \circ \sigma_i \circ \rho_i \circ \pi_i = \text{MP}_2 \circ \text{MP}_1$.

Algorithm 3: postElectionPrep

Participants: Election Commission \mathcal{C} , Printers $\mathcal{P}_1, \mathcal{P}_2$

Private Input: Secret Master permutations MP_1, MP_2 , Scanned Cast Ballots

//Populate BallotTable with scanner data

1 Election commission \mathcal{C} should:

2 foreach $\{b, s, \mu\}$ recorded by scanner **do**

3 Find i for which $\text{ballotTable}(1, i) = b$

4 and $\text{ballotTable}(2, i) = s$

5 ballotTable $(3, i) \leftarrow \mu$

6 Post $\text{ballotTable}(3, :)$ to \mathcal{BB} .

7 end

//Propagate marks from BallotTable to ReceiptTable

8 Printer \mathcal{P}_1 should:

9 MidMarks $\leftarrow MP_1(\text{BallotTable}(3, :))$

10 Post MidMarks to \mathcal{BB} **for** $i \in \{1 \dots \alpha\}$ **do**

11 MidMarksP1 _{i} $\leftarrow \pi_i(\text{BallotTable}(3, :))$

12 Post MidMarksP1 _{i} to \mathcal{BB}

13 end

14 Printer \mathcal{P}_2 should:

15 ReceiptTable $(3, :) \leftarrow MP_2(\text{MidMarks})$

16 Post ReceiptTable $(3, :)$ to \mathcal{BB} . **for** $i \in \{1 \dots \alpha\}$ **do**

17 MidMarksP2 _{i} $\leftarrow \sigma_i(\text{MidMarks})$

18 Post MidMarksP2 _{i} to \mathcal{BB}

19 end

Background Initially the election commission \mathcal{C} demonstrates to the voter that their ballot portion is present in a set of ballot portions as follows:

- \mathcal{C} retrieves the voter’s ballot portion and gathers additional ballot portions such that they *all* show a mark beside an oval showing the voter’s asserted code, but each in a *different position*. The voter checks that all the ballot portions indicate a mark for the same code letter therefore suggesting what the ballot’s entry in the receipt table should be—if one of those ballot portions matches the receipt portion. Because the ballot portions each register a vote for different candidates, any other entity present for this procedure will not know how the voter voted.
- \mathcal{C} places each of these ballot portions into a separate privacy sleeve that hides everything except the bottom perforation line. The ballots are shuffled. \mathcal{C} then demonstrates to the voter that the paper fibre pattern of the receipt physically matches *one of* the ballot portions in the set. This proves that the ballot portion the voter cast is present in the group.

The two steps taken together prove which code the voter marked. The downside of this approach, of course, is that \mathcal{C} must know the association between the receipt portion and ballot portion (and therefore between receipt and vote) in order to perform dispute resolution.

The procedure We now describe the dispute resolution procedure for the basic scheme presented in Section 3. In order to conceal the identity of the voter from \mathcal{C} (who has their ballot) we assume there exists a scrutineer \mathcal{S} that will function as a proxy for the voter during the procedure. Note that the voter will have to trust the \mathcal{S} to honestly follow the protocol.

Note however \mathcal{S} does not learn how the voter voted. The dispute resolution procedure for the *basic scheme* is as follows:

1. The a scrutineer \mathcal{S} , acting on behalf of a voter, transmits the voter’s receipt-id r_v to \mathcal{P}_1 ,

2. \mathcal{P}_1 finds the row in R that contains the encryption r and transmits this index i to \mathcal{P}_2 ,
3. \mathcal{P}_1 and \mathcal{P}_2 each (privately) send to \mathcal{C} the bit vectors they used to construct $B(i)$,
4. \mathcal{C} computes the bitwise xor of the received bit vectors and locates the ballot portion with the resultant ballot-id,
5. S places their receipt portion in a privacy sleeve that hides the receipt-id,
6. S and \mathcal{C} continue with the Scantegrity dispute resolution procedure as defined in [9].

A possible dilemma In many cases the voter will be found to have made a transcription error. However a major dilemma arises from this procedure when it is that case that \mathcal{C} has misreported the code: the voter must give up ballot secrecy to prove to the public that \mathcal{C} is in error.

One of our fundamental requirements for ballot secrecy is that no one be permitted to know the association between ballot-id and receipt-id. If the voter wrote down their code incorrectly, this is not a problem: \mathcal{C} can prove it without needing to know the receipt-id. However if \mathcal{C} reported the code incorrectly, then the receipt-id would need to be made public to prove the discrepancy between the physical and electronic records. This violates ballot secrecy as we have defined it. Still, in the case that the ballot-id/receipt-id association needs to be revealed, it may still suffice if the association between *voter identity* and receipt-id is suppressed. This however would essentially require the voter to never show their receipt to anyone. We leave solving this dilemma to future work, noting that it is mooted by the use of an informational dispute resolution process. We now present an improved system with such an informational dispute resolution procedure.

5 Improved System

In this section we present a system that improves upon the basic system in two ways: First, it replaces the physical dispute resolution procedure with an *informational* dispute procedure. Second, the collection of cast ballots (i.e., the paper audit trail) can be viewed publicly without compromising ballot secrecy.

Informational Dispute Resolution The dispute resolution procedure of the basic system is inefficient and time consuming. Chaum et al. proposed the notion of invisible ink confirmation codes in Scantegrity II [7] as an *informational* means of resolving dispute. Under this approach, codes are printed in invisible ink, and only revealed to the voter if marked. Assuming the code space is sufficiently large so as to make successful random guess unlikely, then knowledge of *any* valid code can be taken as evidence that a voter correctly created their receipt. Any discrepancy found between a receipt and the ReceiptTable can then be attributed to \mathcal{C} (assuming the other correctness proofs are valid). In the improved system, we create and print the codes using a *private printing* protocol. Thus the role of invisible ink is twofold: it restricts the voter's knowledge of unmarked codes *and* it prevents the printers from linking receipts to votes.

Algorithm 4: proveScan

Participants: Any voter \mathcal{V} who created a receipt
Public Input: ReceiptTable, \mathcal{V} 's receipt $\{r, c_v\}$
// Check receipt against ReceiptTable:

- 1 **Voter \mathcal{V} should:**
- 2 Find row i for which $\text{ReceiptTable}(1, i) = r$
- 3 and $\text{ReceiptTable}(2, i) = c_v$
- 4 **if** $\text{ReceiptTable}(3, i) = I$ **then**
- 5 ACCEPT
- 6 **else**
- 7 Run `resolveDispute`
- 8 **end**

Algorithm 5: proveReceipt

Participants: Printers $\mathcal{P}_1, \mathcal{P}_2$ and any interested party

Public Input: A vector Challenge of α challenge bits.

```
1 Both Printers should:
2   for  $i \in \{1 \dots \alpha\}$  do
3     if Challenge( $i$ ) = 0 then
4        $\mathcal{P}_1$  unveils commitment to  $\pi_i$ 
5        $\mathcal{P}_2$  unveils commitment to  $\sigma_i$ 
6     if Challenge( $i$ ) = 1 then
7        $\mathcal{P}_1$  unveils commitment to  $\rho_i$ 
8        $\mathcal{P}_2$  unveils commitment to  $\tau_i$ 
9     All decommitment information is posted to  $\mathcal{BB}$ .
10 end
11 Anyone can:
12   Run verifyCommit on all of the unveiled commitments
13   for  $i \in \{1 \dots \alpha\}$  do
14     if Challenge( $i$ ) = 0 then
15       Check:  $\pi_i(\text{BallotTable}(3, :)) = \text{MidMarksP1}_i$ 
16       Check:  $\sigma_i(\text{MidMarksP2}_i) = \text{ReceiptTable}(3, :)$ 
17     if Challenge( $i$ ) = 1 then
18       Check:  $\rho_i(\text{MidMarksP1}_i) = \text{MidMarks}$ 
19       Check:  $\tau_i(\text{MidMarks}) = \text{MidMarksP2}_i$ 
20 end
```

// Remark: Challenge is generated by a public coin toss, or the Fiat-Shamir heuristic when appropriate (e.g., for $\alpha > 80$.)

Public Paper Trail Invisible ink confirmation codes require a code space that makes random guessing statistically unlikely. For example Scantegrity II proposes a 3-digit code (making a random guess successful 0.1% of the time on average). However in the presence of unique (or semi-unique) codes, access to cast ballots coupled with the public audit dataset is sufficient (or nearly sufficient) to allow *any* observer to link receipts to clear-text votes. This not only means that the paper ballot record must be kept *secret*, but further that the custodian of the ballot record (i.e., \mathcal{C}) is trusted with knowledge of how voters voted. This is one of the major limitations of Scantegrity II. To address this privacy weak-spot, we require a method for not only privately printing a confirmation code, but for displaying it *only while the voter is in the booth*. In the presence of “disappearing” codes, not only can we offer distributed trust with respect to $\mathcal{P}_1, \mathcal{P}_2$ and \mathcal{C} , but we can also make the paper ballot record public.

Self-blanking Confirmation Codes We propose a method for printing of confirmation codes that is self-blanking (i.e., the message is only temporarily visible). The standard invisible ink described by Scantegrity II activates *instantaneously*. That is to say, the chemical reaction responsible for the ink’s pigmentation completes on the order of milliseconds. It was suggested in [7] that a *slower* reacting ink might be the addition of an *anti-catalyst*. This substance, if present, can slow down pigmentation by seconds or minutes (depending on design needs). Combining the technique of visual cryptography with such a ‘slow’ invisible ink, we can construct a self-blanking pixel (see Table 2). Finally, combining self-blanking pixels with the private printing protocol of [16], we can print confirmation codes that are both distributed between two-parties and self-blanking.

The Improved Ballot The improved ballot differs from the basic ballot in that it makes use of *self-blanking invisible ink* confirmation codes. The codes are printed inside the optical scan ovals in *self-blanking invisible ink*. When the voter marks an oval using the specially provided activator pen, the confirmation code is revealed allowing the voter (finite) opportunity to write down the code on their receipt. Eventually the oval darkens completely indicating *that* the oval was chosen by the voter, but not what the confirmation code was (see Figure 2).

Algorithm 6: provePrinting

Participants: Printers $\mathcal{P}_1, \mathcal{P}_2$, Scrutineer \mathcal{S} **Public Input:** A printed ballot chosen at random by \mathcal{S}

```
1 Scrutineer  $\mathcal{S}$  should:
2   Activate/scratch-off hidden areas on ballot form to reveal ballot tuple
3   Post ballot tuple  $\{b, L, r, c\}$  to  $\mathcal{BB}$ 
4 end
5 Printer  $\mathcal{P}_1$  should:
6   foreach BallotTable(1,  $i$ ) =  $b$  do
7     Unveil the commitments to  $\mathcal{P}_1$ 's share of  $b$  (i.e., ballot id bits and associated random factors).
8     foreach  $j \in \{1 \dots \alpha\}$  do
9       Post  $i_{\pi_j} \leftarrow \pi_j(i)$ 
10      Post  $i_{\rho_j} \leftarrow \rho_j(i_{\pi_j})$ 
11 end
12 Printer  $\mathcal{P}_2$  should:
13   foreach  $i_{\rho_j}$  do
14     Unveil the commitments to  $\mathcal{P}_2$ 's share of  $b$  (i.e., ballot id bits and associated random factors used to compute the
15     xor with  $\mathcal{P}_1$ 's share).
16     foreach  $j \in \{1 \dots \alpha\}$  do
17       Post  $i_{\sigma_j} \leftarrow \sigma_j(i_{\rho_j})$ 
18       Post  $i_{\tau_j} \leftarrow \tau_j(i_{\sigma_j})$ 
19 end
20 Anyone can:
21   Run verifyCommit on all of the unveiled commitments,
22   Recompute ballot-id  $b$  using the unveiled id bits and associated random factors and ensure it matches both the electronic
23   and printed versions,
24   foreach BallotTable(1,  $i$ ) =  $b$  do
25     foreach  $j \in \{1 \dots \alpha\}$  do
26       Output an error and exit if the following does not hold:,
27       BallotTable(3,  $i$ ) = MidMarksP1 $_j(i_{\pi_j})$  = MidMarks( $i_{\rho_j}$ ) = MidMarksP2 $_j(i_{\rho_j})$  = ReceiptTable(3,  $i_{\tau_j}$ )
28   Output 1
29 end
```

a	b	VC(a)	VC(b)	Result when activated		
				$t = 0$	$t > 0$	$t \gg 0$
0	0	■ ∅	∅ ■	■ ■	■ ■	■ ■
0	1	■ ∅	■ ∅	■ ■	■ ■	■ ■
1	0	∅ ■	∅ ■	■ ■	■ ■	■ ■
1	1	∅ ■	■ ∅	■ ■	■ ■	■ ■

Table 2: **Self-blanking VC Pixel.** Two sub-pixels contain invisible ink. Each party applies an anti-catalyst (cyan) to *one* sub-pixel. Sub-pixels containing this substance darken more slowly than those without ($t = 0$ is the moment of activation). Eventually all sub-pixels darken “blanking” the pixel’s value.

Changes to the protocols The addition of self-blanking invisible-ink confirmation codes induces some changes the protocols presented in Section 3 which is summarized as follows:

- **Ballot tuples:** \mathcal{P}_2 generates ballot-ids. Both printers run a *private printing* protocol to select a confirmation code and distribute it to VC shares,

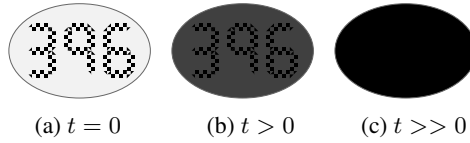


Fig. 2: **Optical-scan oval with self-blanking confirmation code** after being marked with an activator pen ($t = 0$ is the moment of activation).

- **Ballot printing:** \mathcal{P}_2 prints ballot-ids in invisible ink. Both printers print their shares of the confirmation codes using self-blanking visual crypto pixels,
- **Informational dispute resolution:** As in Scantegrity II, the printers only publish the confirmation code corresponding to the voted candidate. In the case of a dispute, the printers jointly issue a non-interactive proof of plaintext inequality between all remaining (unencrypted) codes on the disputed ballot.

5.1 Changes to generateBallots

The generateBallots algorithm of the basic system is adjusted as follows: \mathcal{P}_1 is still responsible for generating and printing a list of unique receipt-ids R . \mathcal{P}_2 now *solely* generates and prints the ballot-ids B . Both printers collaborate to privately generate and print the confirmation codes C . This is accomplished by a *private printing* protocol to select a confirmation code and distribute it to visual crypto shares.

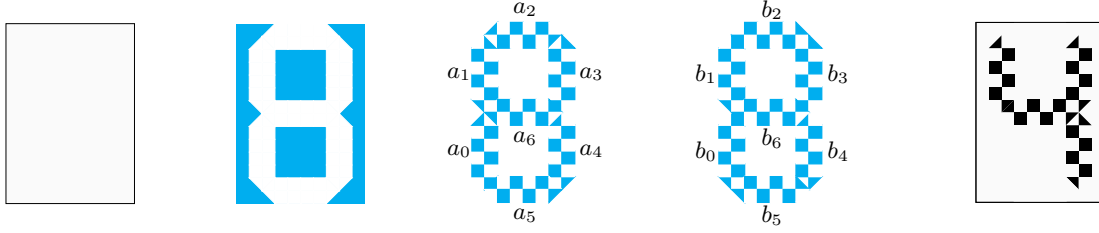
For each confirmation code, the printers engage in the following pre-protocol. We briefly sketch how this is accomplished, referring the reader to [16] for a concrete example. First the printers run a two-party (dealerless) protocol to privately select a single message (i.e., confirmation code) from a set of valid messages, distribute the message to visual crypto shares, and privately output each share to the respective printer. Using these resultant bit-vectors as private input to generateBallots, they homomorphically xor their shares together, repeating for each candidate across each ballot to create confirmation code list C . To make the protocol more efficient, the confirmation code digits can be represented by a 7-segment display, allowing each digit to be represented by 7-bits (and hence 7 ciphertexts).

5.2 Changes to ballot printing

As in the basic ballot, \mathcal{P}_1 prints the receipt-ids $R(i)$ and conceals them under scratch-off coating. For each optical-scan bubble, \mathcal{P}_1 applies a solid background of invisible ink and overlays its visual crypto share using the inhibitor substance. Maintaining the ordering, \mathcal{P}_1 transfers the ballots to \mathcal{P}_2 , who prints the ballot-ids $B(i)$ in invisible ink (n.b., without visual crypto). It then applies its VC shares to the corresponding optical-scan bubbles. Printing self-blanking confirmation codes is depicted in Table 3.

Changes to preElectionPrep For the most part, the improved system has the same overall structure in terms of the cut-and-choose proof. The preElectionPrep is executed in the same way, obviously with the minor difference that the elements of ballot-id list B are now single ciphertexts (as opposed lists of d encrypted bits), and the elements of receipt-id list R are now encrypted as bit-vectors. When ReceiptTable is generated, the receipt-ids are decoded from their 7-segment bit-vector representation into a single integer. However before ReceiptTable is posted to \mathcal{BB} , the printers will *encrypt* each of the confirmation codes (now in integer form) to facilitate the informational dispute resolution procedure. They will save the random factors used in these encryptions for later use.

Changes to proveScan After the election is complete and the mark state information has been updated, the printers will decrypt and post every code $\text{ReceiptTable}(2, i)$, for which $\text{ReceiptTable}(3, i) = 1$. They will additionally post each associated random factor as proof of decryption.



Invisible ink background Background Mask $\mathbf{a} = \{0, 1, 1, 1, 0, 1, 0\}$ $\mathbf{b} = \{0, 0, 1, 0, 1, 1, 1\}$ $\mathbf{a} \oplus \mathbf{b} = \{0, 1, 0, 1, 1, 0, 1\}$

Table 3: **Printing a self-blanking invisible ink confirmation code:** \mathcal{P}_1 and \mathcal{P}_2 run a two-party protocol to select a code and distribute it to VC shares (\mathbf{a} , \mathbf{b} respectively). They respectively apply an anti-catalyst (cyan) over a solid invisible ink background. When activated the code becomes temporally visible (eventually darkening to all black).

Changes to provePrinting When \mathcal{A} audits the printing of a ballot, the codes will only be temporarily visible after activation. This is problematic should \mathcal{A} need to use the ballot as evidence in the event of provePrinting fails. There are several ways this might be addressed. One way would be for \mathcal{A} to take a photograph or video of the activated codes however this may not constitute strong enough evidence. Another way would be to require multiple independent auditors to be present when revealing codes. The evidence then would be the testimony of (honest) auditors which may be problematic if a dispute arises over which codes were observed. Finally, we could conjecture the existence of a “fixer” chemical that could halt (or slow) the reaction. This could potentially be accomplished by applying high concentrations of the anti-catalyst substance immediately after activation.

Changes to the dispute resolution procedure The voter can file a dispute by submitting their receipt-id r and the confirmation code c_v they claimed to have seen. Recall in Scantegrity II, \mathcal{C} responds by unveiling all the commitments to confirmation codes on the dispute ballot. This is acceptable within their security assumptions since \mathcal{C} is trusted to protect ballot secrecy. However since in our system \mathcal{C} is untrusted, the printers must prove in *zero-knowledge* that the disputed code is *not equal* to any of the encrypted codes in ReceiptTable, and nothing more. If c_v is a valid confirmation code of receipt-id r , the printers prove the decryption of the associated code in ReceiptTable. If c_v is *not* a valid confirmation code, for each row i for which $\text{ReceiptTable}(1, :) = r$, the printers (jointly acting as a single prover) issue a non-interactive *proof of plaintext inequality* between $\text{ReceiptTable}(2, i)$ and c_v . An algorithm for a proof of plaintext inequality, `plaintextInequalityProof`, is given in Algorithm 7.

6 Security Analysis of the Improved System

To briefly summarize our results, owing to the similarities between systems, we reduce the correctness of the improved system to that of Scantegrity II [7]. Although Scantegrity has been peer reviewed and used in a real election we are not aware of a formal proof of the correctness. A formal security proof for the improved system would include a formal security proof for Scantegrity II and is out of the scope for this paper. A proof of correctness of Eperio, a related system, does offer some insight into how such a proof would proceed [17]. With respect to secrecy we present an argument that the improved system protects voter privacy even when one printer is corrupted. Assumptions regarding the physical primitives can be found there as well.

6.1 Assumptions

For the security analysis we have to consider the properties of several physical components employed by the improved system of Section 5.

Algorithm 7: plaintextInequalityProof

Participants: Printers $\mathcal{P}_1, \mathcal{P}_2$ acting as a single prover \mathcal{P} , Voter \mathcal{V}

Public Input: An asserted confirmation code m' , a encrypted confirmation code $c = E(m)$, public key $\langle g, q, y = g^{sk} \rangle$

1 **Prover \mathcal{P} should:**

 // Encrypt asserted code m'

2 Post $\langle c_1, c_2 \rangle = \langle g^r, my^r \rangle$

 // Blind quotient of m/m'

3 Select $b \in_r \mathbb{G}_q$ and post $c' = \langle c'_1, c'_2 \rangle = \langle (c_1)^b, (\frac{c_2}{m'})^b \rangle$

 // Prove knowledge of b

4 Post proof of conjunction on DDH-tuple $\langle c_1, \frac{c_2}{m'}, c'_1, c'_2 \rangle$

 // Post decryption of c'

5 Post $\langle rb, (\frac{m}{m'})^b \rangle = \langle u, v \rangle$

6 **end**

7 **Verifier \mathcal{V} should:**

8 Verify proof knowledge of b in step 4

9 Verify decryption of c' : $\langle g^u, vy^u \rangle = \langle c'_1, c'_2 \rangle$

10 Verify plaintext inequality of m, m' : $v \neq 1$

11 Output 0 and exit if any the above do not hold, otherwise output 1.

12 **end**

 // Remark: This is essentially the plaintext equality test due to Jakobsson and Jules [21] adapted for a single prover. A proof of conjunction of DDH-tuples is due to Chaum and Pedersen [10].

Tamper Evidence Although scratch-off coating and invisible ink function as a form of *physical commitment* scheme, they do not offer the strong assumptions that govern the unveiling of a cryptographic commitment scheme since *anyone* can open such a physical commitment. We make use of *tamper evidence* in the physical commitment setting as a weaker alternative to the hiding property of a cryptographic commitment (cf. [30]). Instead of the *hiding* property of cryptographic commitment, a physical commitment ideally has the property that an adversary must actively *tamper* with a document to reveal its secret, which then will be *evident* to the intended recipient.

Scratch-Off Coating We use scratch-off coating to reversibly conceal some information printed onto a ballot. First we assume that such coating is secure under passive attack, i.e., the message cannot be read without actively tampering with the coating. However anyone can easily remove the coating, so instead of a hiding property we make the *ideal assumption* that revealing the information under the coating can *only* be done in a way that it is *evident*. By checking the integrity of the scratch-off coating anyone can reliably decide whether or not the physical commitment was opened before. If the coating is intact, anyone can be convinced that the content of the commitment is still hidden. In the presence of tamper evidence this type of physical commitment can viewed as *binding* in the sense that modifying the underlying message would require tampering.

Invisible ink For the use of invisible ink we make similar ideal security assumptions as for scratch-off coating with regards to security to passive attack and tamper evidence in the case of active attack. In contrast to scratch-off coating however, it is possible, and actually *desirable*, to be able to add (but not remove) printed information.

'Slow' Invisible Ink The improved system employs invisible ink with delayed activation or *'slow' invisible ink*. In addition to our assumptions about normal invisible ink we assume that after a certain time after activation the information printed is no longer readable and thus effectively erased.

6.2 Correctness

For the proof of correctness we assume both the voting authority of Scantegrity II, as well as the election commission and both printers of the improved system, are corrupted and under the complete control of an adversary \mathcal{A} . For simplicity we combine the election committee \mathcal{C} and printers \mathcal{P}_1 and \mathcal{P}_2 and denote it as \mathcal{C} . Note that in this case \mathcal{C} knows everything printed in invisible ink or under a scratch-off coating and the physical assumptions only prevent uncorrupted voters from learning information protected that way. We further assume that both voting systems use the same commitment scheme.

Assume there exists an adversary \mathcal{A} able to undetectably cheat in an election run using our improved system. We show how this adversary can then be used to undetectably cheat in a Scantegrity II election by giving a translation from \mathcal{A} to an attack on Scantegrity II. Let $E = \{\mathcal{C}, \mathcal{V}_1 \dots \mathcal{V}_n\}$ be an election system with election committee \mathcal{C} and voters $\mathcal{V}_1 \dots \mathcal{V}_n$. We say an election system is sound, if, for all adversaries \mathcal{A} , the probability of a verifier accepting an invalid correctness proof is negligible in the security parameter.

Technique Toward a contradiction, we will show how to use the existence of \mathcal{A} , which implements a correctness attack on our system, to leverage the equivalent attack on the Scantegrity II system. We will accomplish this with rewindable black-box access to \mathcal{A} . At a high-level, we translate an election being run with Scantegrity II into an election being run with our system. At each phase of the election, we receive output from \mathcal{A} and translate it into the equivalent output in Scantegrity II. We are essentially generating the equivalent election in both systems in parallel. Of course, by using \mathcal{A} , the tally in our system is undetectably incorrect. We show how to translate this into an undetectably incorrect tally in Scantegrity II. Since this should not be possible if Scantegrity II provides correctness, it must be the case that \mathcal{A} cannot exist.

Reduction We generate the public parameters of the Scantegrity II election, translate them into PubParam, and initialize \mathcal{A} with them. This is a direct translation. We then take the output of preElectionPrep as generated by \mathcal{A} and attempt to translate it back into the preelection data for Scantegrity II. This translation is possible to do directly but for convenience we will extract, via rewinding \mathcal{A} , the permutations π, ρ, σ and τ , which are the permutations between BallotTable and ReceiptTable. For the extraction to work, we simulate two successful elections to request two different openings of the cut and choose proof for the correctness of the permutation. As it is a 1-out-of-2 proof this is enough to learn the permutations. With this knowledge we prepare the public information for the Scantegrity II election as follows:

Employing the similarities between BallotTable in our system and the **S** table of Scantegrity II, we initially group all rows in BallotTable with identical ballot-ids, then remove the ballot-id column. This corresponds to the Scantegrity II **S** table. Let this mapping be called M_S . Next we map ReceiptTable to the equivalent **Q** table. These two tables are identical except we relabel receipt-ids in ReceiptTable as “ballot-ids” in the **Q** table. Let this mapping be M_Q .

Under the assumption that both systems use the same commitment scheme, we directly transfer all commitments as-is. From the permutations extracted from \mathcal{A} and the mappings M_Q and M_S we compute the mapping that maps each cell in table **Q** to one in table **S**.

We do this by composing $\{\rho, \tau\}$ and preparing **Q**-pointers that correspond to this resultant permutation. Similarly we generate the **S**-pointers from the composition of $\{\pi, \sigma\}$. Then we publish the tables **Q**, **R** and **S**.

During the election phase \mathcal{A} has access to ballot choices made by each voter and returns a confirmation code together with the receipt-id to the voter.

After the election phase we query \mathcal{A} for the mark positions in BallotTable and ReceiptTable. Using M_Q and M_S we translate these positions to the corresponding entries in **Q** and **S** and publish them. We also publish the incorrect tally given by \mathcal{A} . The audit challenges are directly translated from Scantegrity II into our system. \mathcal{A} responds to the challenges by unveiling the corresponding sub-permutations, either $\{\rho, \tau\}$ or $\{\pi, \sigma\}$, which will be accepting by definition of \mathcal{A} . We then translate this accepting proof into Scantegrity II by unveiling the corresponding **Q**- or **S**-pointers. At this point a verifier in Scantegrity II will accept the incorrect tally. This fact contradicts the correctness of Scantegrity II—therefore we conclude \mathcal{A} does not exist.

6.3 Voter Privacy (Sketch)

For space considerations we only sketch the properties of voter privacy. One aspect of our improved system is that voter privacy is still guaranteed if one printer is corrupted. We claim that the additional information an adversary \mathcal{A} gains by corrupting one printer is insufficient to learn anything about the choice of a single voter.

\mathcal{A} corrupts \mathcal{P}_1 When \mathcal{A} corrupts \mathcal{P}_1 the additional information gained is the secret key x_1 , a share of each code and for each $i \in \{1 \dots \alpha\}$ the two permutations π_i and ρ_i . Also \mathcal{A} learns all corresponding receipt-ids. The security properties of the encryption scheme prevent \mathcal{A} from decrypting any ciphertexts by only knowing x_1 . During the postelection proof only one of σ_i or τ_i is ever revealed, therefore \mathcal{A} does not learn the master permutation of \mathcal{P}_2 . Thus no information is revealed about the permutation between BallotTable and ReceiptTable given that the commitment scheme is hiding. Because \mathcal{P}_1 prints its share of ballot tuples first, it does not learn anything about \mathcal{P}_2 's share, which in turn ensures \mathcal{A} learns nothing about the association between BallotTable and ReceiptTable.

\mathcal{A} corrupts \mathcal{P}_2 When \mathcal{A} corrupts \mathcal{P}_2 the additional information gained is the secret key x_2 , a share of each code and for each $i \in \{1 \dots \alpha\}$ the two permutations σ_i and τ_i . For the same reasons as above this does not give \mathcal{A} an advantage in breaking voter privacy. Because \mathcal{P}_1 printed the receipt-id and covered it in scratch-off coating, and printed its share of the confirmation codes in invisible ink, \mathcal{A} learns nothing about the association between receipt-ids, codes, and ballot-ids assuming the security properties of these physical primitives as stated in previous sections.

If \mathcal{P}_2 is able to read the receipt-ids (by breaking the security assumption about the scratch-off coating), or the shares of \mathcal{P}_1 (by breaking the security assumptions about invisible ink), or is even able to replace the ballots printed by \mathcal{P}_1 without leaving evidence, \mathcal{P}_2 learns enough to break voter privacy. In this case the privacy of the improved system reduces to that of Scantegrity II with a corrupted printer.

\mathcal{A} coerces \mathcal{V} When \mathcal{A} coerces \mathcal{V} , we seek assurance that \mathcal{V} cannot prove how he/she voted to \mathcal{A} . This property is known as coercion resistance. It has been shown by Küsters et al. that Scantegrity II achieves coercion resistance [26]. We do not attempt to prove coercion resistance for our system but given the demonstrated similarities between both the cryptography and the ballot, we would expect a proof of coercion resistance would be easy to construct following their result.

We have to assume that \mathcal{A} does not have unlimited access to the public paper trail. Specifically \mathcal{A} must not be able to recognize any ballot, for example by making an in-depth analysis of the fibre structure as used for ensuring document authenticity. If \mathcal{A} is able to identify a ballot and has made a similar fibre analysis while the ballots were in the custody of a corrupted printer \mathcal{A} would be able to pair a receipt of a coerced voter with the retained part of the ballot in the public paper trail. This problem is not specific to our improved system but always occurs when \mathcal{A} gains enough information to link a paper receipt to a plaintext ballot in the public paper trail. A simple countermeasure would be to restrict the access to the public paper trail (not that this would have to prevent anyone from even scanning them).

7 Related Work

We review some work related to verifiable voting systems with optical-scan paper ballots. This literature can be roughly separated into two categories: systems using single layer ballot forms but reliant on trusted parties/hardware and systems with distributed trust but with multi layer ballot forms.

Single layer ballot forms with trusted components The Scantegrity [9] and Scantegrity II [7, 8, 5] systems offer both a simple single layer fixed candidate list and an unencrypted paper trail, but make extensive use of trusted components to protect ballot secrecy including a computer for blackbox construction of the correctness proofs, the polling place scanner, the ballot printer as well as the custodian of cast ballots. Additionally the paper record reveals the link between receipt and clear-text vote making it unsuitable for public viewing. The Prêt-à-Voter [11, 40] system and its variants [1, 45, 41] also offer the voter a single-layer ballot form with randomized candidate list. Although the correctness

proofs are usually described as a multi-party computation, ballot forms are generated by a trusted printer. Cast ballots are generally “encrypted” though variants exist that leave a human readable paper trail [29, 17]. Benaloh [2] proposes that receipts be generated and printed by a special-purpose device connected to the optical scanner. This has the distinct advantage that the ballots contain no identifying information (beyond the vote). However the issue of trusted ballot printing instead becomes a matter of trusted receipt printing.

Distributed trust with multi layer ballot forms Kubiak [24] and Carback et al. [20] propose *mostly* distributed modifications of the Punchscan system [38]. The former still relies on a trusted ballot printer, the later distributes printing but still relies on trusted hardware to generate ballot tuples. Carback and Popoveniuc [37] later propose a *three*-party distributed version of Punchscan in which top- middle- and bottom-sheet permutations are each generated by independent printing authorities. In all cases voters must use an indirect marking procedure. Moran and Naor [31] propose an improved multi layer ballot form that does not rely on indirection and with considerably stronger, provable, security properties. Voters are issued layers in separate sealed envelopes. Once inside the booth the voters are directed to remove each layers from its envelope and stack the layers in a particular order. The resultant candidate list is horizontally offset from the optical scan ovals by a randomized amount. Lundin et al. [28] propose a distributed construction of the Prêt-à-Voter ballot based on a form of dealerless 2-party visual cryptography. The voter must be careful to align the VC shares in the booth in order to reconstruct the candidate list. Most recently Küsters et al. [25] present a version of Prêt-à-Voter system without a trusted printer, physically implementing a re-encryption mixnet using scratch-off coatings. The voter receives a separate ballot for *each* candidate, which can be cumbersome for races involving more than a few candidates.

Other schemes Chaum proposed the first physical receipt based voting system in [6]. It consists of two visual crypto layers showing the name of the voted candidate. A receipt is created by separating the layers and destroying one of them. Paul et al. [35] propose visual crypto for use in voter authentication for (non-cryptographic) remote voting systems. Scratch & Vote [1], Scratch, Click & Vote [27] and Pretty Good Democracy [41] make use of scratch-off coating to conceal encryption random factors and confirmation codes. Finally, Kelsey et al. [23] propose a voter-coercion strategy involving the use of scratch-off cards to direct voter action.

8 Discussion

8.1 Technical Challenges

The formulation of invisible inks has many areas for improvement. Although Carback et al. [5] report progress in the manufacture of invisible inks, they observed their ink chemistry led to rapid degradation in the print-heads causing printers to eventually fail after printing only a fraction of a single precinct’s worth of ballots. It also seems possible that codes could be passively attacked (i.e., read without activation) under laboratory-based forensic analysis. It would be important to the credibility of invisible ink to have a sense of how costly this would be.

Proper alignment (i.e., registration) of shares has long been a limitation of visual cryptography. Assuming an optical scan oval width of 1 cm and the visual crypto pattern depicted in Figure 2, the sub pixels would be on the order of 0.3cm wide. Assuming a quality tolerance of >90% overlap of subpixels between shares, then the printing alignment would require a tolerance on the order of about 3mm (in both horizontal and vertical axes). It seems plausible current consumer printers could achieve this. Printing confirmation codes in the improved scheme would require finer granularity. A rough estimate based on the codes in [5] suggests one VC sub-pixel per millimetre might suffice. This would require a precision on the order of 100 μ m which likely exceeds the capability of consumer printing technology. We envision a device that could *simultaneously* scan positional markers on the document and align the print head relative to them in real-time. The authors are not aware of any existing implementation of this, but note it may be an interesting avenue for future work.

8.2 Usability Questions

Sherman et al. [44] have studied some basic usability questions about invisible ink confirmation codes. There are however several additional and potentially important usability questions that would need to be answered before *self-blanking* invisible ink confirmation codes could be used in a real election.

Perhaps the most important question would be to understand how disappearing ink might interfere with the voter marking the ballot as intended. It is certainly possible that the voter’s mental model of marking a ballot may be affected by the delayed darkening of the oval. Would this delay represent confusing feedback for the voter while they attempt to confirm whether they successfully *marked the ballot as they had intended to*?

Another important question pertains to the potential pitfall to privacy if the voter leaves the booth too early. The voter would need to be instructed to stay in the booth until the oval has darkened fully (and hence the code has disappeared). The longer it takes for an oval to darken, the more likely it would be that a voter might choose to disregard the instruction and leave the booth anyway. A naïve technical solution would be to dilute the inhibiting substance to speed up the reaction. However this comes at cost of giving the voter a smaller window of opportunity to write down the code on their receipt. This might especially be problematic for multi contest ballots if the voter decides to complete marking the ballot first and *then* records their codes later, as the codes may have disappeared by then.

8.3 Future Work: Toward a Secure Multi-party Protocol

The systems described in this paper are both two-party protocols. Ultimately however it would be desirable to be able to distribute trust among arbitrarily many printers. With some modification the improved system presented in Section 5 could likely be extended to a secure multi-party protocol. With regard to creating the audit dataset this would be mostly a straightforward extension of the two-party approach with each of the $n > 2$ printers generating their own master permutations and issuing their own cut-and-choose proofs. Generating ballot tuples in a multi-party setting should also be a fairly straightforward extension of the two-party setting.

The primary challenge will be to develop an effective approach to distribute the ballot printing among more than printers. This will undoubtedly require a fundamentally different approach from the two-party private printing scheme presented in [16] and is an interesting potential direction for future work.

Conclusion

In this paper we presented two systems for verifiable optical-scan voting with single layer ballots and without trusted components. The basic system based on randomized confirmation codes utilizes existing techniques for invisible ink printing. The improved system proposes a novel self-blanking invisible ink, allowing us to construct a system with more efficient dispute resolution procedure and public paper audit trail.

Acknowledgements

The authors wish to thank Jöern Mueller-Quade and the anonymous reviewers for their helpful feedback. Special thanks go to Jeremy Clark for many helpful discussions throughout the writing of this paper. This research is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC)—the first author through a Postgraduate Scholarship and the third through a Discovery Grant. The second author was supported in part through a foreign exchange scholarship from the Karlsruhe House of Young Scientists (KHYS).

Bibliography

- [1] Ben Adida and Ronald L. Rivest. Scratch & vote: self-contained paper-based cryptographic voting. In *ACM WPES*, pages 29–40, 2006.
- [2] J. Benaloh. Administrative and public verifiability: Can we have both? In *EVT*, 2008.
- [3] Josh Benaloh. Ballot casting assurance via voter-initiated poll station auditing. In *EVT*, 2007.
- [4] Josh D. Benaloh (*né* Cohen) and Michael J. Fisher. A robust and verifiable cryptographically secure election scheme. In *SFCS*, 1985.
- [5] Richard T Carback, David Chaum, Jeremy Clark, John Conway, Aleksander Essex, Paul S. Hernson, Travis Mayberry, Stefan Popoveniuc, Ronald L. Rivest, Emily Shen, Alan T Sherman, and Poorvi L. Vora. Scantegrity II election at takoma park. In *USENIX Security Symposium*, 2010.
- [6] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, 2004.
- [7] David Chaum, Richard Carback, Jeremy Clark, Aleks Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y A Ryan, Emily Shen, and Alan T Sherman. Scantegrity II: end-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In *EVT*, 2008.
- [8] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, Alan T. Sherman, and Poorvi L. Vora. Scantegrity ii: end-to-end verifiability by voters of optical scan elections through confirmation codes. *IEEE Transactions on Information Forensics and Security*, 4(4):611–627, 2009.
- [9] David Chaum, Aleksander Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan T. Sherman, and Poorvi Vora. Scantegrity: End-to-end voter verifiable optical-scan voting. *IEEE Security and Privacy*, 6(3):40–46, May/June 2008.
- [10] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In *CRYPTO*, 1992.
- [11] David Chaum, Peter Y A Ryan, and Steve Schneider. A practical voter-verifiable election scheme. In *ESORICS*, 2005.
- [12] Jeremy Clark and Urs Hengartner. On the use of financial data as a random beacon. In *EVT/WOTE*, 2010.
- [13] William Clarkson, T Weyrich, A Finkelstein, Nadia Heninger, J. Alex Halderman, and Edward W Felten. Fingerprinting blank paper using commodity scanners. In *IEEE Symposium on Security and Privacy*, 2009.
- [14] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *EUROCRYPT*, 1997.
- [15] Aleks Essex, Jeremy Clark, Richard T. Carback, and Stefan Popoveniuc. Punchscan in practice: an e2e election case study. In *WOTE*, 2007.
- [16] Aleks Essex, Jeremy Clark, Urs Hengartner, and Carlisle Adams. How to print a secret. In *HotSec*, 2009.
- [17] Aleks Essex, Jeremy Clark, Urs Hengartner, and Carlisle Adams. Eperio: Mitigating technical complexity in cryptographic election verification. In *EVT/WOTE*, 2010.
- [18] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.
- [19] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. One-time programs. In *CRYPTO*, 2008.
- [20] Richard T. Carback III, Stefan Popoveniuc, Alan T. Sherman, , and David Chaum. Punchscan with independent ballot sheets: Simplifying ballot printing and distribution with independently selected ballot halves. In *WOTE*, 2007.
- [21] Markus Jakobsson and Ari Juels. Mix and match: Secure function evaluation via ciphertexts. In *ASIACRYPT*, 2000.
- [22] Ayman Jarrous and Benny Pinkas. Secure hamming distance based computation and its applications. In *ACNS*, 2009.
- [23] John Kelsey, Andrew Regenscheid, Tal Moran, and David Chaum. Attacking paper-based E2E voting systems. In *Towards Trustworthy Elections*, volume 6000 of *LNCS*, pages 370–387. Springer, 2010.
- [24] Przemyslaw Kubiak. A modification of punchscan: Trust distribution. In *FEE*, 2006.

- [25] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Improving and simplifying a variant of *prêt à voter*. In *VOTE-ID*, 2009.
- [26] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Proving coercion-resistance of Scantegrity II. In *ICICS*, 2010.
- [27] Mirosław Kutylowski and Filip Zagorski. Scratch, click & vote: E2e voting over the internet. In *Towards Trustworthy Elections*. Spr, 2010.
- [28] D. Lundin, H. Treharne, P. Y. A. Ryan, S. Schneider, J. Heather, and Z. Xia. Tear and destroy: Chain voting and destruction problems shared by *prêt à voter* and punchscan and a solution using visual encryption. In *FEE*, 2006.
- [29] David Lundin and Peter Y. Ryan. Human readable paper verification of *prt voter*. In *ESORICS*, 2008.
- [30] Tal Moran and Moni Naor. Basing cryptographic protocols on tamper-evident seals. In *In Proceedings of the 32nd International Colloquium on Automata, Languages and Programming*, pages 285–297, 2005.
- [31] Tal Moran and Moni Naor. Split-ballot voting: Everlasting privacy with distributed trust. In *ACM CCS*, 2007.
- [32] Moni Naor and Adi Shamir. Visual cryptography. In *EUROCRYPT*, 94.
- [33] C. Andrew Neff. Practical high certainty intent verification for encrypted votes. Technical report, VoteHere Whitepaper, 2004.
- [34] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *EUROCRYPT*, 1993.
- [35] Nathanael Paul, David Evans, Aviel D. Rubin, and Dan S. Wallach. Authentication for remote voting. In *HCISS*, 2003.
- [36] Torben Pryds Pedersen. A threshold cryptosystem without a trusted party. In *EUROCRYPT*, 1991.
- [37] Stefan Popoveniuc and Richard Carback. Clearvote: An end-to-end voting system that distributes privacy between printers. In *WPES*, 2010.
- [38] Stefan Popoveniuc and Ben Hosp. An introduction to punchscan. In *WOTE*, 2006.
- [39] Michael Rabin. Transaction protection by beacons. *Journal of Computer and System Sciences*, 27(2), 1983.
- [40] Peter Y A Ryan and Steve A Schneider. *Prêt à voter* with re-encryption mixes. In *ESORICS*, 2006.
- [41] Peter Y A Ryan and Vanessa Teague. Ballot permutations in *prêt à voter*. In *EVT/WOTE*, 2009.
- [42] Peter Y A Ryan and Vanessa Teague. Pretty good democracy. In *Workshop on Security Protocols*, 2009.
- [43] Daniel R. Sandler, Kyle Derr, and Dan S. Wallach. VoteBox: a tamper-evident, verifiable electronic voting system. In *USENIX Security Symposium*, 2008.
- [44] Alan T Sherman, Richard T Carback, David Chaum, Jeremy Clark, Aleksander Essex, Paul S. Herson, Travis Mayberry, Stefan Popoveniuc, Ronald L. Rivest, Emily Shen, Bimal Sinha, and Poorvi L. Vora. Scantegrity mock election at takoma park. In *EVOTE*, 2010.
- [45] Zhe Xia, Steve A Schneider, and James Heather. Analysis, improvement and simplification of *prêt à voter* with paillier encryption. In *EVT*, 2008.