Scantegrity II: End-to-End Verifiability by Voters of Optical Scan Elections Through Confirmation Codes

David Chaum, Richard T. Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, Alan T. Sherman, and Poorvi L. Vora

***Abstract*—Scantegrity II is an enhancement for existing paper ballot systems. It allows voters to verify election integrity—from their selections on the ballot all the way to the final tally—by noting codes and checking for them online. Voters mark Scantegrity II ballots just as with conventional optical scan, but using a special ballot marking pen. Marking a selection with this pen makes legible an otherwise invisible preprinted confirmation code. Confirmation codes are independent and random for each potential selection on each ballot. To verify that their individual votes are recorded correctly, voters can look up their ballot serial numbers online and verify that their confirmation codes are posted correctly. The confirmation codes do not allow voters to prove how they voted. However, the confirmation codes constitute convincing evidence of error or malfeasance in the event that incorrect codes are posted online. Correctness of the final tally with respect to the published codes is proven by election officials in a manner that can be verified by any interested party. Thus, compromise of either ballot chain of custody or the software systems cannot undetectably affect election integrity. Scantegrity II has been implemented and tested in small elections in which ballots were scanned either at the polling place or centrally. Preparations for its use in a public sector election have commenced.**

***Index Terms*—Cryptography, electronic voting, end-to-end verifiability, privacy.**

I. Introduction

PAPER ballots dominate elections globally, apart from a few exceptions such as Brazil and India. In the United States, optical scan systems and direct recording electronic (DRE) voting machines began to replace paper ballots and lever systems in about 1980 [2]. More recently, however, due to reliability failures and security vulnerabilities, the trend has been toward replacing DREs with paper ballot systems, including optical scan systems [30]. Optical scan, however, is not without its own demonstrated and inherent integrity vulnerabilities (see, for example, [29]). Undetected errors, unintentional or malicious, in the scanning or tallying software can cause undetected errors in the electronic tally. Improperly printed ballots enable a variety of attacks on integrity. Misplaced ballots or breaches in chain-of-custody render even expensive manual recounts ineffective. Further, the transparency offered by manual recounts is at best limited to those officials and observers in attendance. Scantegrity II is an enhancement for optical scan voting systems that addresses the above deficiencies, while also providing ballot secrecy guarantees under reasonable assumptions.

In Scantegrity II, voters mark ballots using a special ballot-marking pen, which makes legible preprinted confirmation codes corresponding to voter selections. The link between confirmation codes and voter selections is cryptographically protected, with the key(s) being shared by election officials. Voters may note down their confirmation codes onto a chit that is detachable from the ballot. After the election, all voted confirmation codes are posted online, where voters may check them. The final tally is computed in a verifiable manner from the posted confirmation codes.

The functionality of Scantegrity II is enabled by the use of several types of ink with special properties, in the following ways.

1) Confirmation codes and ballot ovals are printed with a special ink that darkens when it reacts with the ink in the ballot-marking pen; the confirmation code ink reacts more slowly than the ballot oval ink, and hence darkens several minutes after the oval does. Thus, the code is visible for several minutes after being marked, during which the voter may note it on the chit. On the other hand, the confirmation code may be assumed to be indistinguishable from its background in an unmarked oval. This allows the Scantegrity II system to provide a confirmation code to the voter only after the voter has made the corresponding ballot selection.
2) The Scantegrity II chit bears two serial numbers that are required of the voter in order to check the confirmation codes online. These serial numbers are also indistinguishable from the background until made legible through the use of a decoding pen. The ink in the decoding pens is different from the ink in the ballot-marking pens. Poll workers reveal the serial numbers using a decoding pen after the

Manuscript received February 23, 2009; revised October 07, 2009. First published October 20, 2009; current version published November 18, 2009. The work of J. Clark and A. Essex was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC). The work of S. Popoveniuc and P. L. Vora was supported by NSF-CNS-0831149. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Bart Preneel.

D. Chaum is with the Voting Systems Institute, Los Angeles, CA 90064 USA (e-mail: info@chaum.com).

R. T. Carback and A. T. Sherman are with the Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Baltimore, MD 21250 USA (e-mail: carback1@umbc.edu; sherman@umbc.edu).

J. Clark is with the David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, N2L 3G1, Canada (e-mail: j5clark@cs.uwaterloo.ca).

A. Essex is with the School of Information Technology and Engineering, University of Ottawa, ON, K1N 6N5, Canada (e-mail: aesse083@site.uottawa.ca).

S. Popoveniuc and P. L. Vora are with the National Institute of Standards and Technology, Gaithersburg, MD 20899 USA (e-mail: poste@gwu.edu; poorvi@gwu.edu).

R. L. Rivest and E. Shen are with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: rivest@mit.edu; eshen@csail.mit.edu).

R. Y. A. Ryan is with the Faculte des Sciences, de la Techologie et de la Communication, University of Luxembourg, L-1359, Luxembourg (e-mail: peter.ryan@uni.lu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2009.2034919

ballot is cast. This prevents voters from falsely claiming that a valid confirmation code, obtained from an uncast ballot, came from a cast ballot. When it is not possible to use the different inks required for chit serial numbers and decoder pens, it is possible to achieve a similar end, though with weaker integrity guarantees, by requiring that a record be kept, by polling officials and observers, of serial numbers of spoiled ballots.

Scantegrity II has implemented procedures for printing with the inks to make it virtually impossible to read unexposed numbers and codes with the human eye. Further, it is reasonable to assume that voters do not have access to ballots outside the polling booth, and that they do not have access to specialized equipment inside the polling booth. The inks thus enable the Scantegrity II voting system to provide voters with confirmation codes that correspond exactly to their selections, and serial numbers that correspond exactly to valid cast ballots.

Scantegrity II, like several other systems (such as *Prêt à Voter* [11], Punchscan [15], [16], [26], Scratch & Vote [1]) provides a mechanism for end-to-end verifiability of election integrity: voters may verify that their selections are included unmodified in the collection of selections; additionally, anyone may verify that the tally is computed correctly from the collection of selections. Voters and authorized observers may "audit" ballots by requiring the voting system to expose all confirmation codes and corresponding selections on the audited ballots, and checking that these correspond to those printed on the ballots. Audited ballots may not be used for voting.

The verifiability property of Scantegrity II is independent of voting system software correctness and ballot chain-of-custody after ballots are cast. The proof of correctness made by those running an election is based only on 1) the inability of the voting system to change values once they have been committed to, and 2) the unpredictability of choices made by voters and election auditors—to verify confirmation codes online, to audit ballots, and to audit the data provided by the voting system regarding the processing of confirmation codes to obtain the tally. The ability of the system to expose false charges of election fraud is based on the ink properties described above.

Paper ballot systems possess inherent weaknesses with respect to the requirement of ballot secrecy [20]. For example, a voter can be identified by a distinctive manner of making marks. The process of polling place scanning also introduces privacy vulnerabilities; for example, the timing of voters scanning ballots can be analyzed to improve an estimate of the voter's selections. Further, advances in forensic technology make it possible to examine fingerprints on ballots. Finally, the miniaturization of cameras poses challenges to the secrecy of voter selections in all types of voting systems, whether paper-based or not.

Scantegrity II is an overlay on paper ballot systems, and cannot remove the inherent ballot secrecy limitations of the underlying system. It does, however, attempt to limit any additional ballot secrecy vulnerabilities. For example, the linking of confirmation codes to votes requires the collusion of a set of election officials, or the breaking of the security of cryptographic techniques used. Further, the use of a slow-reacting ink and a modification to the voting procedure can ensure that information linking confirmation codes and ballot serial numbers to voter selections can be removed from ballots a few minutes after they are marked. As with regular optical scan, forensic attacks are possible—coercive adversaries could, for example, use specialized equipment to attempt to read the codes on the ballots. We assume these are too time-consuming and unwieldy to be very practical, for two reasons. First, we have instituted printing procedures to minimize the effectiveness of such ballot analyses; these are described in Section V. Second, simpler attacks, based on the fingerprinting of the underlying paper using commodity scanners [12], are possible against perforated paper-ballot-based end-to-end voting systems in general (including Scantegrity and *Prêt à Voter*).

A. Contributions

Scantegrity II and its predecessor Scantegrity [9] have the following characteristics that distinguish them from other systems that provide end-to-end verifiability:

1) Compatibility with optical scan equipment: Scantegrity and Scantegrity II do not require the replacement of any optical scan polling place equipment. Both systems interface cleanly with the underlying optical scan system, requiring only a modified ballot and access to the results from the scanners.
2) Familiar ballot-marking procedure: The ballot-marking procedure is very similar to that for a conventional optical scan ballot. Opting into verification of election integrity is up to the individual voter.

Two properties of Scantegrity II distinguish it from Scantegrity.

1) Scantegrity did not use invisible ink; all confirmation codes were visible on the ballot. This allowed voters to file spurious disputes concerning which codes appear on the website, and required a tedious dispute resolution process to resolve such issues. If voters cannot guess confirmation codes or chit serial numbers, a dispute regarding the correct recording of confirmation codes can be resolved in Scantegrity II without the cumbersome physical proof required by Scantegrity.
2) Scantegrity II makes commitments to multiple Scantegrity back-ends and uses a new audit procedure.
   a) While the Scantegrity audit procedure reveals some information about individual votes, the Scantegrity II audits reveal no additional information if the cryptographic techniques used are secure, and election officials do not collude to violate ballot secrecy.
   b) In Scantegrity, the probability that a cheating voting system is undetected decreases exponentially with the number of modified votes. In Scantegrity II, this probability is independent of the number of modified votes, but decreases exponentially with the number of back-ends audited.

Scantegrity was described in [9]. Scantegrity II was first described at EVT 2008 [8]. The present paper provides a more detailed description; additionally, the use of chit serial numbers to improve dispute resolution, the use of multiple back-end instances, the new audit procedure, and a proposal for accessibility are original to this paper.

B. Organization

In Section II, we provide a nontechnical sketch of the protocol as viewed by the voters, poll workers, and election administrators. A complete technical specification of the entire protocol is provided in Section III. Our security assumptions and an analysis of the integrity and privacy provided by Scantegrity II follows in Section IV. We also offer a discussion of the use of invisible ink in Section V and the accessibility of Scantegrity II to voters with disabilities in Section VI.

II. Scantegrity II Procedures

Scantegrity II provides integrity guarantees through the use of a confirmation code provided to each voter for each ballot selection. All confirmation codes are posted on a website after the election, and all results are obtained through the processing of these codes. The Scantegrity II protocol defines the manner in which participants in the election—voters, election administrators, and observers—interact with the voting system in order to ensure that 1) confirmation codes are correctly present on the ballots, 2) marked confirmation codes are correctly present on the website, and 3) confirmation codes are correctly processed to obtain the final tally. The protocol is designed to enable the detection of election fraud if it has occurred, as well as to prevent false charges of election fraud. This section provides an (intentionally) informal description of the protocol; its purpose is to provide a description that is somewhat accessible to voters, poll workers, and election administrators, and to prepare the reader for the more formal description in the next section.

A. Vote Casting Procedure

This section describes the vote-casting procedure, which is very similar to that of a regular optical scan ballot. The slight differences between the two are as follows. First, the unmarked ballot itself looks slightly different: it bears a detachable chit that can be used to note confirmation codes. Second, while marking the ballot, voters will notice the appearance of confirmation codes, which will also disappear after a few minutes. Third, voters or observers may *audit* ballots to determine whether printed confirmation codes correctly reflect voter selections; such ballots may not then be cast. While we have simplified the ballot audit procedure considerably, it does not have a corresponding equivalent in the regular optical scan protocol, and might appear complicated to voters and officials. Similarly, spoiled ballots are discarded using a procedure that is more complex than that used for optical scan. Fourth, voters interact with a polling official after the vote is successfully cast, in order to expose serial numbers on the receipt chit.

1) Scantegrity II Ballot: The Scantegrity II ballot consists of two parts: the *main body* and the *chit* (see Fig. 1). Similar to an optical scan ballot, the main body of a Scantegrity II ballot contains, for each contest, a list of valid selections printed in a canonical order predetermined by polling place procedures (e.g., alphabetical, rotated across precincts, etc.). Next to each possible selection is a markable region, oval in shape.

Differing from an optical scan ballot, the background of each oval is printed with a reacting ink. The confirmation code corresponding to the selection for the particular ballot is printed inside the oval. The ink used to print the confirmation code is similar to that used for the oval background, but is slow-reacting. Both inks look the same when printed on the ballot paper; they may be assumed to be indistinguishable to the human eye before the oval is marked with the ballot-marking pen (see Section V for details on the validity of this assumption). Further, we assume that voters will not be able to take expensive spectral analysis equipment into the polling booth; such equipment might aid in the ability to distinguish between background and confirmation number. Thus, we assume that, before marking, the oval has a single color, and confirmation codes are indistinguishable from the background of the oval; that is, confirmation codes are invisible. Additionally, a Scantegrity II ballot contains a ballot serial number that is machine-readable but not easily read or memorized by a human (e.g., a two-dimensional barcode).

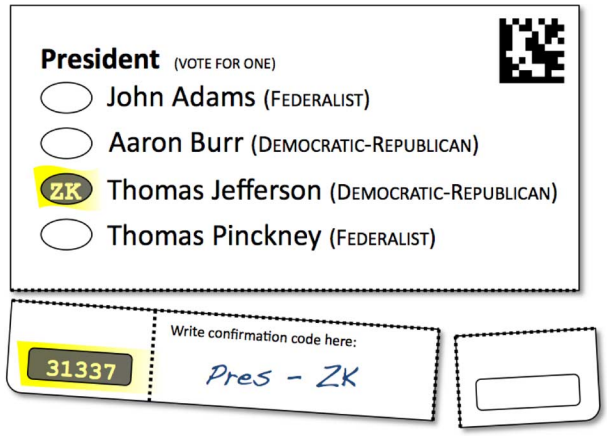


Fig. 1. Scantegrity II ballot showing the main body (top) with one marked position and machine-readable serial number; left chit (bottom left) with a developed chit serial number and confirmation code written in; and right chit (bottom right) with an undeveloped chit serial number. This figure is meant to demonstrate the parts of the ballot and does not represent the actual final state of the portions after voting.

The chit is attached to the bottom of the ballot via a perforation, such that it can be easily detached. It has two halves, left and right; the halves can be detached from each other using a pair of scissors. On each half is a chit serial number: the left chit serial number and the right chit serial number. These chit serial numbers are distinct from each other and from the ballot serial number; we describe later how they are used to ensure that voters cannot make false claims regarding confirmation codes on uncast ballots. Both the left and right chit serial numbers are printed in invisible ink such that they are neither human nor machine readable before being decoded using a special decoder pen. Both the left and right chit serial numbers are assumed to appear after they are marked with the decoder pen.

2) Ballot Marking: Upon arrival, a voter is authorized to cast a ballot, and is handed the next one in the pile; it is enclosed in a privacy sleeve. At this time, she may choose to audit a ballot, which she may choose from the existing ballot pile. For details on the ballot audit procedure, see Section II-B1.

In order to vote for a particular selection, the voter fills in the corresponding oval using a ballot-marking pen. In accordance with the invisible ink printed on the ballot, the background of the oval will immediately turn dark, leaving a confirmation code visible in the foreground. The relative darkness of any marked

ovals to unmarked ones will allow an optical scanner employing dark mark logic to register the oval as marked. The foreground of the oval will be human-readable and a voter interested in verifying that her vote is in the virtual collection of votes to be tallied may record the code on the chit portion of the ballot. Uninterested voters may disregard the codes.

The link between a confirmation code and the corresponding selection on a particular ballot is protected cryptographically. We omit the details underlying the generation and protection of the data until the next section. At this stage, however, we do note the following. The disclosure of a confirmation code does not reveal the selection, if the cryptographic techniques used are assumed secure, and election officials are assumed not to collude to determine the selection.

Although not apparent to the voter, the confirmation code is printed in a slow-reacting invisible ink that will also turn dark, but only after the passage of several minutes (e.g., five to seven minutes). At this time, the oval will be completely dark and the code will no longer be visible, leaving no human-readable unique information on the ballot.

As an option, the two-dimensional bar-coded serial number could also have slow reacting ink in its background such that if a voter marked it, it would turn solid black.

Section V describes how a masking ink and appropriate printing techniques may be used to reduce the ability to distinguish between the inks, even with the use of microscopes and spectral equipment. Indeed, it may be assumed that the slow and fast-reacting invisible inks are, for all practical purposes, indistinguishable a) before exposure and b) within T seconds after both have been exposed, where T is the response time of the slow-reacting ink. After a period long enough to include reaction times, a filled-in Scantegrity II ballot provides, for all practical purposes, an amount of information that is similar to that on an optical scan ballot, and can be used in a manual recount with a level of privacy very similar to that of optical scan.

3) Spoiling the Ballot: If the voter makes an error in marking a ballot or wishes to register a protest vote through spoiling the ballot, it is returned to the poll worker. Without seeing the contents of the ballot, the poll worker removes the ballot from the privacy sleeve and detaches the right side of the chit from the ballot. The main body and left chit are shredded in view of the voter. The right chit is retained by the poll worker and used to verify that the number of ballots issued is identical to the sum of the number of ballots tallied, print-audited, and spoiled. The number of spoiled ballots allowed per voter is typically limited by predetermined polling place procedures.

4) Casting the Ballot: When the voter has satisfactorily marked a ballot, it is returned to the poll worker. As previously, the poll worker detaches the chit from the ballot. Further, with the choices on the ballot still concealed, the poll worker places the main body of the ballot into the scanner, which records the ballot serial number and the marked choices. In the preferred version of the protocol, voters are not allowed to cast undervoted or overvoted ballots. If a voter does not wish to vote for a particular candidate, she must make a selection of "none of the above." If the scanner detects an undervote or overvote, the voter is returned her ballot, and will spoil it and re-enter the issuance procedure. Note that, in the U.S., the requirement that a voter be notified of undervotes or overvotes is not uncommon; in fact, the Help America Vote Act requires that voters be notified of overvotes if electronic equipment is used. However, requiring that undervoted or overvoted ballots not be cast is considerably stricter, and decreases the usability of the voting system. The alternative version of the protocol does not ban undervotes or overvotes in cast ballots. However, in this version, a secure chain of custody is required to ensure that unvoted races were not changed to voted ones, nor voted races overvoted. Research on requiring neither the restriction on undervoted and overvoted ballots, nor a secure chain of custody, is underway.

In order for the scanner to read the serial number, it must be encoded in a two-dimensional barcode as the scanner can only recognize marked or unmarked regions.

After a successful scan, the two serial numbers on the chit are developed by the poll worker. The voter may leave with the chit. It is expected that public interest groups will make available the possibility of creating a copy of chits to alleviate the need for concerned but time-constrained voters to personally participate in auditing the election.

5) Casting Without Automation: For polling places without adequate voting technology or in the event of a power failure, Scantegrity II may still proceed with the voter being issued the chit in the same manner. The main body of the ballot will, instead of being scanned, be placed into a sealed ballot box that has been certified as being empty prior to sealing. If scanning technology is unavailable at the polling place, the ballots may be transported to a central scanning location.

6) Accounting for Ballots: At the end of the day, poll workers and official observers make a note of the numbers of spoiled, voted and audited ballots, and ensure that their sum is equal to the number of used ballots. These numbers are made publicly available; this prevents ballot stuffing. Further, they note down the exposed chit serial numbers of voted, spoiled, and audited ballots, so these cannot be changed after the election.

B. Election Audit Procedures

A voter may participate in auditing the election in several ways. In addition to checking the confirmation numbers on her ballot, she may audit a printed ballot, and check the processing of confirmation codes. Election observers may also participate in the latter processing check.

1) Auditing a Printed Ballot: Voters wishing to audit a printed ballot may choose one from the ballot pile; we refer to the process of auditing the ballot as the *print audit*. They will each be issued a ballot main body and the left or right half of the chit, with the serial number activated using the decoder pen; which half is chosen may be determined by a flipped coin. The other half of the chit is removed and retained by the pollworker in a clear box on the poll worker table. At her leisure, the voter fully marks the ballot to reveal all the confirmation codes, which she may check using the procedure in the following section.

2) Checking Confirmation Numbers: At a prearranged time after the polls close, voters who recorded the confirmation codes associated with the candidates they voted for, or those who wish

to check the confirmation codes on a print-audited ballot, may visit a website where they will be prompted for the serial number on the chit. In the case of voted ballots, the voter will have two serial numbers—left and right; either is suitable to identify the ballot uniquely. Upon entering a serial number, the website will report the confirmation codes in the positions it believes were marked for voted ballots, but will not report the candidates associated with these codes. For this reason, providing a copy of the confirmation codes in no way undermines the secrecy of the ballot. Voters are encouraged to share their confirmation codes, share photographs of their chits, or post screen-captures of the results. In the case of an audited ballot, entering the serial number will similarly report the confirmation codes that should appear on the ballot and, only in this case, also reveal the candidates associated with each code.

All confirmation codes and their associated candidates are committed to prior to the election to ensure the values or associations cannot be changed. Thus, the audited ballots provide probabilistic evidence that the confirmation codes were correctly printed on the ballots. The correct and full inclusion of confirmation codes from a voted ballot provides probabilistic evidence that the votes were properly scanned and not maliciously altered. Full details are provided in Section III, and the strength of this evidence is quantified in Section IV.

3) Checking the Processing of Confirmation Numbers: Due to the commitments to confirmation codes and candidates before the beginning of the election, it is known that candidates are mapped to confirmation codes and that this mapping cannot be changed. Further, through the print audits, voters are assured that this mapping has been faithfully transposed to the printed ballots they marked. By checking the inclusion of their confirmation codes, they are further assured that the marks they made for candidates have been faithfully transposed to confirmation codes consistent with those on the ballot. The final step is to check that the confirmation codes are properly mapped back to the correct candidates.

The protocol for achieving this check will be based on an open specification. Voters may either obtain software from a software provider they trust, or write their own software, to check the processing of the confirmation numbers. All required information for writing the software (such as the format of the data and what the data are) is provided by Scantegrity II to all interested parties. Those administering the election are encouraged to appoint an independent auditor to perform this check so as to provide at least one audit of the tally computation from confirmation codes. The details of this check are also provided in Section III.

C. Dispute Resolution Process

If any voters discover incorrect confirmation codes or ballots that are incorrectly designated as voted, print-audited, or spoiled, they may file disputes. In the case of a confirmation code being incorrect, they may provide the confirmation code they believe should be on the ballot. A voter's knowledge of a valid confirmation code on the ballot, that is not present on the website, suggests an error or malfeasance; the validity of the code can be established since the codes are committed to, and the likelihood of guessing a correct code can be made low through the use of longer codes (exact quantification to follow in Section IV). If a voted ballot is incorrectly designated, the voter can provide both chit serial numbers to prove that it was voted. Similarly, if a print-audited ballot is incorrectly designated, the voter or independent auditor can provide all the confirmation codes on the ballot to prove that it was print-audited. In the case when the voter knows all confirmation codes in an overvoted ballot, this ballot's designation cannot be changed to print-audited as the voter knows both serial numbers. In order to ensure that unvoted races are not voted, and that properly voted ballots are not changed to overvoted ones, a restriction of not allowing undervotes or overvotes on cast ballots is required.

III. Cryptographic Proof of Tally

The following describes the method used for proving the correctness of an election outcome while simultaneously maintaining voter anonymity. It is based on the protocol introduced in [9] and [8], adapted to the enhanced polling procedures described in Section II.

A. Ballot Definition

For simplicity, we consider a notation based on a single contest ballot. Generalization to ballots containing multiple races, as well as elections containing multiple ballot styles, should be viewed as multiple independent executions of the single contest case described herein. Let $L = (s_0, \ldots, s_{n-1})$ define a list of n ballot selections (e.g., candidates, choices, etc).

B. Roles

We consider three categories of entities participating in the election with the acknowledgement that the entities are role-based and thus an individual might possibly assume any or all roles.

Voters: Voters are those with the authority to cast a ballot in the election. We assume that voter authentication (external to this discussion) is undertaken prior to ballot issuing and that only authenticated voters are issued ballots. In this section, we will refer to a particular voter as V.

Election Trustees: Let T be the set of t election trustees, $T_1, \ldots, T_t \in T$. The trustees engage in the cryptographic protocol to setup and generate the correctness proofs of the election. T would generally consist of public officials and, optionally, candidate representatives. The protocol is intended to proceed when a minimum number of trustees are present—not requiring the presence of all so as to mitigate the disruption caused by any individual trustee's absence at various stages of the protocol.

Verifier: The set of verifiers A consists of all agents verifying the correctness proofs herein. The intention is that the tally-correctness be "universally verifiable" as defined in [28]—meaning that any voter, citizen, or observer can participate either directly, or through delegation, in the verification of the tally if they so choose.

Other Entities: Poll workers are responsible for administering the voting process, instructing and assisting voters, as well as enforcing the registration, ballot issuing, marking, and casting procedures outlined in Section II.

Finally, we require the existence of a public bulletin board $\mathcal{BB}$ which implements an append-only public record. In practice, it might be implemented as a mirrored public website.

C. Functions

In this section, we outline the main functions used in the protocol. For a positive integer len, we use $[\mathrm{len}]$ to denote the set of integers $[0, 1, \ldots, \mathrm{len} - 1]$.

The functions consist of the following:

1) A parameter initialization function that, given a security parameter, provides an election-specific nonce and minimum key lengths.
2) A trustee threshold-key generation function that produces individual trustee keys for trustees and a master key that can be reconstructed from a minimum threshold τ number of trustee keys. This function takes as input the election-specific nonce, the value of τ, and input bit strings from the trustees, the entropy of which provides the entropy of the keys generated.
3) A master-key reconstruction function that, given a set of τ or more trustee keys, reconstructs the master key.
4) A subkey generation function that is a cryptographic one-way function, accepts a master key and an identifier, and outputs another key.
5) A keyed permutation function that, given a key and the value len, generates a pseudorandom permutation of integers in the range $[\mathrm{len}]$.
6) A cryptographic commitment function that is computationally hiding and computationally binding.
7) A ballot generation function that, given the candidate list, the confirmation code alphabet and length, the election master key, and the number of ballots required, generates the master list of ballots.

Details of each of these functions follow.

Parameter Initialization: $P \leftarrow \mathsf{Parameters}(1^p)$ accepts a security parameter p and outputs a set of functional parameters P including a unique election-specific nonce λ selected in accordance with a public convention (not considered here), and a specification of cryptographic algorithms used to realize certain cryptographic one-way and trapdoor functions, as well as specifying their enforced minimum key lengths. For brevity, we will omit continual reference to P by assuming all following functions accept it as input.

Trustee Threshold-Key Generation: $(k_1, \ldots, k_t, K) \leftarrow \mathsf{TrusteeKeys}(\omega_1, \ldots, \omega_t, \tau, \lambda)$ accepts an arbitrary-length random bit string, denoted $\omega_i \in \{0,1\}^*$, from each trustee T_i, as well as a threshold $1 \leq \tau \leq t$, specifying the number of trustees needed to reconstruct a unique election master key K. It outputs a distinct key for each of the trustees, $k_1, \ldots, k_t$, as well as a master key K. We do not consider the policy guidelines for selecting trustees or τ in this section. K is such that, if at least one ω_i is uniformly distributed across all possibilities, K will be as well. K is also dependent on the election nonce λ (so if the same value of ω_i were supplied in a different election, K would be different). K is only used as private input to other functions. Each output key k_i is transmitted over an authenticated and physically untappable channel to the corresponding trustee T_i.

Election Master Key Reconstruction: $\emptyset/K \leftarrow \mathsf{ElectionKey}(\{j_1, \ldots, j_n\})$ accepts as input a set $\{j_1, \ldots, j_n\}$ of keys and outputs the unique election master key K if and only if $|\{j_1, \ldots, j_n\} \cap \{k_1, \ldots, k_t\}| \geq \tau$. Otherwise it returns a symbol (denoted by $\emptyset$) indicating the function failed to reconstruct the key.

The assumption for the two preceding algorithms is briefly stated: given any unbounded adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ (over a random guess) in guessing K, given any set containing fewer than τ keys from $k_1, \ldots, k_t$, is exactly zero. One suitable construction is due to Pedersen [25], and has been suggested for use in voting by Benaloh [3]. A suitable notion of an untappable channel is the one due to Sako and Killian [28].

Sub-key Generation: $\kappa_{\mathrm{ID}} \leftarrow \mathsf{SubKey}(K, \mathrm{ID})$ is a cryptographic one-way function that accepts a master key K and identifier ID and outputs another key κ_{ID}, where ID defines what key is to be generated.

Keyed Permutation: $\pi \leftarrow \mathsf{Perm}(\kappa, \mathrm{len})$ accepts a key κ and list length len, and outputs $\pi : [\mathrm{len}] \rightarrow [\mathrm{len}]$, where π is a permutation selected pseudorandomly from the set of possible permutations of len elements Π_{len}. The function π depends on κ. We use the notation $X'(i) \leftarrow X(\pi(i))$ to denote the element-wise shuffle of a len-element set X for $0 \leq i < \mathrm{len}$. Finally, we define a special-case null index, denoted $\emptyset$, in which $\pi(\emptyset) = \emptyset$ for all $\pi \in \Pi_{\mathrm{len}}$.

Cryptographic Commitment: We consider a cryptographic commitment protocol as including the following pair of functions: $\bar{m} \leftarrow \mathsf{Commit}(\kappa, m)$ accepts a key κ and an arbitrary length message m to obtain a commitment $\bar{m}$. $0/1 \leftarrow \mathsf{Decommit}(\kappa', m', x)$ accepts a commitment x, key κ', and message m', and outputs 1 if $\mathsf{Commit}(\kappa', m') = x$. Otherwise it outputs 0.

The cryptographic assumptions for these algorithms are briefly stated: given any probabilistic polynomial time-bounded $\mathcal{A}$ producing messages m and m', and keys κ and κ', the probability that $m \neq m'$ and $\mathsf{Commit}(\kappa', m') = \mathsf{Commit}(\kappa, m)$ is a negligible quantity in the security parameter p. That is, $\mathcal{A}$ cannot find two distinct messages that produce the same commitment. This is an informal definition of the computationally binding property of a commitment. Additionally, given any probabilistic polynomial time-bounded $\mathcal{A}$

$$|Pr[\mathcal{A}(\mathsf{Commit}(\kappa, m)) = 1] - Pr[\mathcal{A}(\mathsf{Commit}(\kappa, m')) = 1]|$$

is a negligible quantity in the security parameter p. That is, $\mathcal{A}$ cannot distinguish between a commitment to m and one to m', if the commitments use the same key. This is an informal definition of the computationally hiding property of commitment functions.

Generate Ballots: $\mathbf{P} \leftarrow \mathsf{GenerateBallots}(L, \Sigma, l, K, b)$ accepts ballot selection/candidate list L, confirmation-code alphabet Σ (typically the set of alphanumeric characters), confirmation-code length l, election master key K, and the overall number of ballots to be generated b. $\mathbf{P}$ contains three lists. The first is a list of b ballots,

j	α	β	γ	q
0	0000	7973	4630	7LH
1	0000	7973	4630	WT9
2	0001	2567	1490	J3K
3	0001	2567	1490	TC3
4	0002	4900	7891	9JH
5	0002	4900	7891	J3K
6	0003	1631	5275	KWK
7	0003	1631	5275	H7T

(a)

j	r
0	0
1	1
2	0
3	1
4	1
5	0
6	0
7	1

(b)

j	s
0	Alice
1	Bob
2	Alice
3	Bob
4	Alice
5	Bob
6	Alice
7	Bob

(c)

Fig. 2. Tables $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{S}$ when there are two candidates, $s_0 =$ Alice and $s_1 =$ Bob. For example, a vote for Alice on Ballot 0000 would reveal the confirmation code 7LH; however, one for Bob would reveal WT9. Note that, for purposes of illustration, we show one way in which the $\mathbf{R}$ table may be populated based on votes cast during the election. The function GenerateBallots() however, initializes all these values to zero. In this example, the votes cast on Ballots 0000, 0001, 0002, and 0003 were for Bob, Bob, Alice, and Bob, respectively, and would reveal confirmation codes WT9, TC3, 9JH, and H7T, respectively. The publicly published versions of tables $\mathbf{Q}$ and $\mathbf{S}$ will contain commitments to the information shown above; this detail is provided in Step 6c of the setup phase in Section III-E. There is no information in Table $\mathbf{R}$ before votes are cast, and there is no information made public about this table before the election. (a) Table $\mathbf{Q}$; (b) Table $\mathbf{R}$; (c) Table $\mathbf{S}$.

sorted by serial number, each with n selections, each selection associated with a confirmation code in Σ^l. In addition to this list, $\mathbf{P}$ also bears space for the voters' choices after ballots are filled, and a third list which bears the corresponding candidates.

We deviate slightly from the notation introduced in [8]. Let $\mathbf{P}$ denote the canonical "master" list associating codes, candidates, and voter-made marks, which we define as the triple of $(b \cdot n)$-element lists $\mathbf{P} = \{\mathbf{Q}, \mathbf{R}, \mathbf{S}\}$. For all $0 \leq j < bn$,

1) $\mathbf{Q}$ is a list of serial numbers and confirmation codes, including serial numbers (α, β, γ) for each ballot, and confirmation codes q for each selection in a ballot. Let $\mathbf{Q}(j) = \{\alpha_j, \beta_j, \gamma_j, q_j\}$.
2) $\mathbf{R}$ will eventually represent the list of scanned voter-made marks $r \in \{0, 1\}$ indicating the absence or presence of a mark (i.e., vote) made for an associated selection. Let $\mathbf{R}(j) = r_j$, and let all r_j be initialized to 0.
3) $\mathbf{S}$ is a list consisting of b repetitions of selection/candidate list $L = (s_0, \ldots, s_{n-1})$. Let $\mathbf{S}(j) = s_{(j \bmod n)}$.

For notational convenience throughout the rest of this paper, we will use the index g to refer to a given ballot B_g, and its associated voter-receipt V_g, where $g = \alpha_j = \lfloor j/b \rfloor$. For any $j \neq j'$, let $\alpha_j = \alpha_{j'}, \beta_j = \beta_{j'}, \gamma_j = \gamma_{j'}$, if $\lfloor j/n \rfloor = \lfloor j'/n \rfloor$.

Serial numbers β, γ shall be selected independently (without replacement) by a secure pseudorandom number generator seeded by the election master key K. These numbers shall be selected from range defined by p, such that correctly guessing an unknown β or γ would occur with a small (but not cryptographically negligible) probability.

Finally, confirmation codes q will be independently selected by a pseudorandom generator such that confirmation codes are not repeated across a given ballot B_g, namely $q_j \neq q_{j'}$ if $\lfloor j/n \rfloor = \lfloor j'/n \rfloor$, for distinct j, j'.

See Fig. 2 for an example of a list of four ballots when there are two candidates on the ballot, and confirmation codes consist of three alphanumeric symbols.

D. Trusted Computation Platform

The protocol assumes the existence of a hardware device, referred to as the *trusted computation platform*, which the trustees use to evaluate the various functions described above. This device relies on the following assumptions related to the preservation of ballot secrecy:

- **Private and authenticated input**: the ability to receive input from authenticated trustees via a physically untappable channel;
- **Private evaluation**: the ability to evaluate a function such that the intermediate values cannot be recovered by passive or active attack of the hardware or software components; and
- **Correctness**: the ability to attest that the functions being evaluated are equivalent to available and predefined source code.

Note that the correctness assumption enables the trustees to be certain that the required computations are being computed correctly, and hence increases the reliability of the computation from the perspective of the honest trustee. It does not affect the ability of the voter or the auditor to detect a cheating trustee.

With the failure of any of these trust assumptions, it may become possible for a malicious subset of trustees to recover information related to the association between voting intent and ballot serial number. For example, this can be accomplished by observing a sufficient number of trustee keys, observing intermediate state, or altering the functions to overtly or covertly leak this information.

None of these assumptions, including the correctness assumption, dictate the soundness of the tally. In the event that any or all of these assumptions are subverted (or any cryptographic assumption is found not to hold), the correctness of the final tally can still be ascertained through the independent verification mechanism described in this section.

E. Protocol

Setup Phase: The trustees in set T generate their threshold trustee keys and initialize the bulletin board $\mathcal{BB}$ using candidate list L, security parameter p, number of ballots to be generated b, valid trustee threshold list τ, code alphabet Σ, code length l, and a heuristic security parameter I, where $\{L, p, b, \Sigma, l, \tau, I\}$ is issued to T by an external entity not considered herein. The audit described in this paper, which is different from that described in previous publications on Scantegrity or Scantegrity II, requires the commitment of the voting system to several consistent back-ends, each of which can be used to tally votes from the confirmation codes. I is the number of back-ends constructed by the system.

Let the notation $X_i(j) = x_{i,j}$ denote the jth element in the of the ith instance of a shuffled list X_i. Additionally, let the notation X_i', X_i'', and X_i''' denote list X shuffled by the composition of permutations $(\pi_{(i,1)})$, $(\pi_{(i,2)} \circ \pi_{(i,1)})$, and $(\pi_{(i,3)} \circ \pi_{(i,2)} \circ \pi_{(i,1)})$, respectively.

Using a trusted computing platform, the trustees perform the following computations:

1) Initialize security parameters: $P \leftarrow$ Parameters(1^p).
2) Initialize bulletin board: Post $\{P, L, p, b, \Sigma, l, \tau, I\}$, and the specification of all functions to $\mathcal{BB}$.

j	α	β	γ	q
0	0000	7973	4630	WT9
1	0001	2567	1490	J3K
2	0001	2567	1490	TC3
3	0002	4900	7891	9JH
4	0002	4900	7891	J3K
5	0003	1631	5275	KWK
6	0003	1631	5275	H7T
7	0000	7973	4630	7LH

(a) Table $\mathbf{Q}'_1$

j	s
0	Bob
1	Alice
2	Bob
3	Alice
4	Bob
5	Alice
6	Alice
7	Bob

(b) Table $\mathbf{S}'''_1$

Fig. 3. Tables $\mathbf{Q}'$ and $\mathbf{S}'''$ for the example of Fig. 2. $\mathbf{Q}'$ is $\mathbf{Q}$ permuted by $\pi_{(1,1)}$, which is an upward circular shift of one unit, and $\mathbf{S}'''$ is $\mathbf{S}$ permuted by $\pi_{(1,3)} \circ \pi_{(1,2)} \circ \pi_{(1,1)}$, where $\pi_{(1,2)}$ corresponds to an upward circular shift of two units, and $\pi_{(1,3)}$ swaps the last two elements in the list. Note that the confirmation numbers of $\mathbf{Q}'$ can be made to match up with the correct candidates in $\mathbf{S}'''$ if the permutation $\pi_{(1,3)} \circ \pi_{(1,2)}$ is applied to $\mathbf{Q}'$. Note also that we use simple permutations such as these merely for the purposes of illustration. For the system itself, we advocate that each permutation be chosen pseudorandomly from the set of all possible permutations, without restricting this set to the set of simple permutations such as cyclic permutations or swaps. (a) Table $\mathbf{Q}'_1$; (b) Table $\mathbf{S}'''_1$.

3) Generate trustee keys: Each trustee T_i contributes entropy ω_i and is issued corresponding trustee key k_i via an untappable channel with the trusted computing platform $(k_1, k_2, \ldots, k_t) \leftarrow \mathsf{TrusteeKeys}(\omega_1, \ldots, \omega_t, \tau, \lambda)$.
4) Generate election key: assuming the trusted platform is stateful during this phase, the election master key K is generated by the previous step. (Note that key K must not leave or be leaked from the trusted platform during computation, nor should the trusted platform be stateful between the setup, result declaration, and audit response phases. A minimum of τ keys from $\{k_1, k_2, \ldots, k_t\}$ can regenerate all the information required for the result declaration and audit response phases.)
5) Generate ballots: the trusted platform computes $\mathbf{P} = \{\mathbf{Q}, \mathbf{R}, \mathbf{S}\} \leftarrow \mathsf{GenerateBallots}(L, \Sigma, l, K, b)$, and transmits $\mathbf{P}$ via a private channel to a trusted printing service which produces paper ballots with corresponding serial numbers and confirmation codes in invisible ink. Note that initially the recorded voter marks table $\mathbf{R}$ is empty.
6) Shuffle P and cryptographically commit to the shuffles: The following mixnet-like construction shuffles the two lists $\mathbf{Q}$ and $\mathbf{S}$ and posts commitments to the two shuffled lists and to the shuffles. The shuffles are constructed in a manner that will make the tally-verification audit simple to implement, as will be seen later. See Fig. 3 for an illustration on the example of Fig. 2. Note that, *in this example only*, we use cyclic permutations and a swap *merely* in an attempt to illustrate the mixnet-like construction in as simple a manner as possible. We *do not* advocate the restriction of permutations to a set of a few permutations, but, as mentioned below, require that each permutation be chosen in a pseudorandom manner from the set of all possible permutations of the respective tables.
   a) Generate permutations: For each back-end, the trusted platform computes three permutations. That is, for $0 \le i < I$, the trusted platform computes:
      i) $\pi_{(i,1)} \leftarrow \mathsf{Perm}(\mathsf{Subkey}(K, \{\text{“1st”}, i\}), (b \cdot n))$
      ii) $\pi_{(i,2)} \leftarrow \mathsf{Perm}(\mathsf{Subkey}(K, \{\text{“2nd”}, i\}), (b \cdot n))$
      iii) $\pi_{(i,3)} \leftarrow \mathsf{Perm}(\mathsf{Subkey}(K, \{\text{“3rd”}, i\}), (b \cdot n))$.
   b) Shuffle lists: For each back-end the trusted platform computes a single-shuffled instance $\mathbf{Q}'_i$ of $\mathbf{Q}$ and a triple-shuffled instance $\mathbf{S}'''_i$ of $\mathbf{S}$. (Note that the number of apostrophes denotes the number of shuffles the list has gone through.) That is, for $0 \le i < I$ and $0 \le j < (b \cdot n)$, the trusted platform computes:
      i) $\mathbf{Q}'_i(j) \leftarrow \mathbf{Q}(\pi_{(i,1)}(j))$
      ii) $\mathbf{S}'''_i(j) \leftarrow \mathbf{S}_i(\pi_{(i,3)}(\pi_{(i,2)}(\pi_{(i,1)}(j))))$.
   c) Commitments: The trusted platform commits to each back-end—the shuffled confirmation code numbers, the corresponding candidate lists, and the permutation values—on an element-by-element basis. For each single-shuffled code list $\mathbf{Q}'_i(j) = \{\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j}, q_{i,j}\}$, triple-shuffled candidate list $\mathbf{S}'''_i(j) = s_{i,j}$, and the corresponding elements of permutations $\pi_{(i,h)}(j)$, the trusted platform computes commitments as follows.
      For $1 \le h \le 3$, $0 \le i < I$, and $0 \le j < (b \cdot n)$
      i) $\bar{\alpha}_{i,j} \leftarrow \mathsf{Commit}(\mathsf{Subkey}(K, \{\text{“}\alpha\text{”}, i, j\}), \alpha_{i,j})$
      ii) $\bar{\beta}_{i,j} \leftarrow \mathsf{Commit}(\mathsf{Subkey}(K, \{\text{“}\beta\text{”}, i, j\}), \beta_{i,j})$
      iii) $\bar{\gamma}_{i,j} \leftarrow \mathsf{Commit}(\mathsf{Subkey}(K, \{\text{“}\gamma\text{”}, i, j\}), \gamma_{i,j})$
      iv) $\bar{q}_{i,j} \leftarrow \mathsf{Commit}(\mathsf{Subkey}(K, \{\text{“}q\text{”}, i, j\}), q_{i,j})$
      v) $\bar{s}_{i,j} \leftarrow \mathsf{Commit}(\mathsf{Subkey}(K, \{\text{“}s\text{”}, i, j\}), s_{i,j})$
      vi) $\bar{\mathbf{Q}}'_i(j) \leftarrow \{\bar{\alpha}_{i,j}, \bar{\beta}_{i,j}, \bar{\gamma}_{i,j}, \bar{q}_{i,j}\}$
      vii) $\bar{\mathbf{S}}'''_i(j) \leftarrow \bar{s}_{i,j}$
      viii) $\bar{\pi}_{(i,h)}(j) \leftarrow \mathsf{Commit}(\mathsf{Subkey}(K, \{\text{“}\pi\text{”}, h, i, j\}), \pi_{(i,h)}(j))$.
   d) The trusted platform publishes all $\bar{\mathbf{Q}}'_i$, $\bar{\mathbf{S}}'''_i$, and $\bar{\pi}_{(i,h)}$ to $\mathcal{BB}$.
7) The trusted platform's internal state is purged.

The mixnet-like construction described in step 6 of the setup phase is similar to the initial version of Scantegrity II presented in [8], with a slight functional simplification. The mixnet-like processing in the initial version was equivalent to the application of two permutations, $\pi_{\mathbf{Q}}$ and $\pi_{\mathbf{S}}$, to $\mathbf{Q}$ and $\mathbf{S}$, respectively. The first permutation shuffled the order of the confirmation numbers, and the second one was such that positions corresponding to a single candidate appeared in a single block of consecutive positions in $\mathbf{S}$. The difference we propose in this paper is likely to be easier to implement.

Voting Phase: A voter V, upon being successfully authenticated by poll workers, is given a ballot $B_g = \{\alpha_g, \beta_g, \gamma_g, q_{gn}, q_{gn+1} \ldots, q_{gn+n-1}, s_0, s_1, \ldots, s_{n-1}\}$ consisting of a serial number α_g printed in an optical-scan readable "barcode" and selection/candidate list $s_0, \ldots, s_{n-1}$ printed in normal ink. Serial numbers β_g, γ_g and the corresponding confirmation codes $q_{gn}, q_{gn+1}, \ldots, q_{gn+n-1}$ are printed in invisible ink.

To vote, V marks the optical scan bubble beside the desired selection s_d using the decoder pen, which reveals the confirmation code q_{gn+d}.

Upon scanning ballot B_g, the optical scanner shall produce an "electronic ballot image" $\mathrm{EBI}_g = \{\alpha_g, r_{gn}, r_{gn+1}, \ldots, r_{gn+n-1}\}$, where $r_{gn+d} = 1$ if and only if a darkened region (i.e., a mark) was detected inside the optical scan bubble beside the dth selection s_d. All other

$\hat{d} \neq d$ shall register $r_{gn+\hat{d}} = 0$. The specific electoral system in use would dictate how many marks (i.e., distinct d's) are permissible on a single ballot. V can then choose to construct a vote receipt $\mathrm{VR}_g = \{\beta_g, \gamma_g, q_{gn+d}\}$ for each s_d marked.

Instead of voting on a particular ballot, V can select it to be "print-audited" in accordance with the procedures specified in Section II. All confirmation codes are revealed, and **one of** $\{\beta_g, \gamma_g\} = \delta_g$ is revealed. The print-audited ballot becomes $PA_g = \{\delta_g, q_{gn}, q_{gn+1}, \ldots, q_{gn+n-1}, s_0, s_1 \ldots, s_{n-1}\}$. For example, if Ballot 0001 of Fig. 2 were not voted but were print-audited, and β_1 revealed, the print-audited ballot would be $PA_{0001} = \{1490, \texttt{J3K}, \texttt{TC3}, \texttt{Alice}, \texttt{Bob}\}$. (Note that print-audited ballots cannot be voted.)

Declaring Results: After the polling concludes, a valid subset of trustees (as defined by τ) assemble to tally and declare the results. The trustees also make available data regrading the tally processing that will be audited in the audit phase. Given the set of all EBIs recorded during the election, the trustees proceed using the trusted platform as follows.

1) Regenerate election master key: Each trustee T_i transmits their trustee key k_i over an untappable channel to the trusted platform. The election master key K is reconstructed by calling $\mathsf{ElectionKey}(\{j_1, \ldots, j_n\})$ if at least τ trustees supply correct keys (where $\{j_1, j_2, \ldots, j_n\}$ are n keys provided by n trustees).
2) Regenerate ballot list: Ballot list $\mathbf{P}$ is reconstructed by re-running step 5) of the setup phase.
3) Construct list of recorded marks: For each EBI_g recorded during the election, populate $\mathbf{R}$ by setting $\mathbf{R}(gn + j) = r_{gn+j}$ for $0 \leq j < n$ and $r_{gn+j} \in \mathrm{EBI}_g$. Any unused, spoiled or print-audited ballot B_g inherently constitutes an EBI_g with all $r_{gn+j} = 0$.
4) Post voted codes: During the dispute resolution period (described in Section II), all voted codes shall be published. For all $\mathbf{R}(gn + d) = 1$ post $\{\beta_{gn+d}, \gamma_{gn+d}, q_{gn+d}\}$ and corresponding commitment keys.
5) Post results: Using $\mathbf{P}$, tabulate the election results and post them to $\mathcal{BB}$,
6) Post double-shuffled marks list for audit purposes:
   a) Regenerate permutations: For $0 \leq i < I$ and $0 \leq j < (b \cdot n)$ the trusted platform recomputes permutations:
      i) $\pi_{(i,1)} \leftarrow \mathsf{Perm}(\mathsf{Subkey}(K, \{\text{“1st"}, i\}), (n \cdot b))$
      ii) $\pi_{(i,2)} \leftarrow \mathsf{Perm}(\mathsf{Subkey}(K, \{\text{“2nd"}, i\}), (n \cdot b))$.
   b) Shuffle lists: For $0 \leq i < I$ and $0 \leq j < (b \cdot n)$, the trusted platform computes I independent double-shuffled instances $\mathbf{R}''_i$ of $\mathbf{R}_i$
      i) $\mathbf{R}''_i(j) \leftarrow \mathbf{R}(\pi_{(i,2)}(\pi_{(i,1)}(j)))$.
   c) The trusted platform publishes all $\mathbf{R}''_i$ to $\mathcal{BB}$ and purges its internal state.

See Fig. 4 for an illustration using the example of Figs. 2 and 3.

Note that, at this stage, the list $\mathbf{R}''_i$ is such that, if permuted by $\pi_{(i,3)}$, the votes will be listed as obtained for the candidate list $\mathbf{S}'''_i$. Further, if list $\mathbf{Q}'_i$ is permuted by $\pi_{(i,2)}$, the confirmation codes will be listed in the order of the votes $\mathbf{R}''_i$.

Audit Challenge and Response: In order to ensure robust, correct behavior by the trustees and in turn, the correctness of the election outcome, two audits are carried out. We first describe the tally computation audit. For each back-end committed to by the trustee, a coin flip determines whether the trustees will demonstrate that the ballot marks of the corresponding public table $\mathbf{R}''_i$ correspond correctly to (a) the announced tally or (b) the public confirmation codes for voted ballots. This is done by opening the commitments to the permutation $\pi_{(i,3)}(j) \forall j$ or to the permutation $\pi_{(i,2)}(j) \forall j$, respectively. Second, we describe the print audit. For values of j, in the original ballot list $\mathbf{Q}$, corresponding to print-audited ballots, permutation values $\pi_{(i,2)}(j)$ and $\pi_{(i,3)}(j)$ are opened $\forall i$. We now describe these audits in more detail.

1) Public challenge of trustees: some time after the trustees have completed declaring the results and posting the shuffled marks lists, each instance of $\mathbf{Q}'_i, \mathbf{R}''_i, \mathbf{S}'''_i$ is challenged to be partially revealed for the purposes of auditing. A fair public coin $\mathcal{C}$ is tossed I times providing a series of audit challenges $\mathcal{C} \in \{0, 1\}^I$, which are posted to $\mathcal{BB}$.
2) For the tally computation audit. For $0 \leq i < I$ and $0 \leq j < (b \cdot n)$, the trusted platform performs the following actions:
   a) If $\mathcal{C}(i) = 0$, regenerate and publish the confirmation codes $\mathbf{Q}'_i$ and the association between $\mathbf{Q}'_i$ and $\mathbf{R}''_i$. That is, regenerate and publish the following:
      i) the second permutation $\pi_{(i,2)}$;
      ii) the commitment subkeys of $\pi_{(i,2)}$: $\kappa_{\pi_{(i,2)}}(j) \leftarrow \mathsf{Subkey}(K, \{\text{“}\pi\text{"}, 2, i, j\}) \forall j$;
      iii) the commitment subkeys to all elements of $\mathbf{Q}'_i$: $\kappa_{x_{i,j}} \leftarrow \mathsf{Subkey}(K, \{\text{“}x\text{"}, i, j\})$, where $x = \{\alpha, \beta, \gamma, q\} \forall j$.
   b) If $\mathcal{C}(i) = 1$, regenerate and publish the permuted candidate list $\mathbf{S}'''_i$, as well as the association between $\mathbf{R}''_i$ and $\mathbf{S}'''_i$. That is, regenerate and publish the following:
      i) the third permutation $\pi_{(i,3)}$;
      ii) the commitment subkeys of $\pi_{(i,3)}$: $\kappa_{\pi_{(i,3)}}(j) \leftarrow \mathsf{Subkey}(K, \{\text{“}\pi\text{"}, 3, i, j\}) \forall j$;
      iii) the commitment subkeys to all elements of $\mathbf{S}'''_i$: $\kappa_{s_{i,j}} \leftarrow \mathsf{Subkey}(K, \{\text{“}s\text{"}, i, j\}) \forall j$.

α	β	γ	q
0000	7973	4630	WT9
0001	2567	1490	TC3
0002	4900	7891	9JH
0003	1631	5275	H7T

(a)

j	r
0	1
1	1
2	0
3	0
4	1
5	0
6	1
7	0

(b)

Fig. 4. The revealed confirmation numbers (entries in $\mathbf{Q}$) and revealed table $\mathbf{R}'_1$, which is a shuffled version of $\mathbf{R}$, using the permutation $\pi_{(1,2)} \circ \pi_{(1,1)}$, where $\pi_{(1,1)}$ is an upward circular shift of one unit, and $\pi_{(1,2)}$ corresponds to an upward circular shift of two units. The tally will be "three votes for Alice and one vote for Bob." The permutations used are secret. Note that, if $\mathbf{R}'_1$ is permuted by $\pi_{(1,3)}$, the votes will be listed wrt candidate list $\mathbf{S}'''$ of Fig. 3. If list $\mathbf{Q}'_i$ of the same figure is permuted by $\pi_{(1,2)}$, the confirmation codes will be listed as corresponding to the choices of $\mathbf{R}'_1$ above. Note also that we use simple permutations for the purposes of illustration. For the system itself, we advocate that each permutation be chosen pseudorandomly from the set of all possible permutations, without restricting this set to the set of simple permutations such as cyclic permutations or swaps. (a) Revealed values of Table $\mathbf{Q}$; (b) Table $\mathbf{R}'_1$.

3) For the ballot audit, compute all permutation elements and commitment keys not computed in tally audit and required for the purposes of demonstrating the entire path of the ballot through the mixnet-like construction. That is, the trusted computing platform does the following for $0 \leq i < I$.
   a) If $\mathcal{C}(i) = 0$:
      - Regenerate $\pi_{(i,3)}$ (do not publish it).
      - For each ballot $PA_g = \{\delta_g, q_{gn}, q_{gn+1}, \ldots, q_{gn+n-1}, s_0, s_1 \ldots, s_{n-1}\}$ that is print-audited:
        i) Search for all elements in $\mathbf{Q}'_i$ such that the second component is δ_g. If there is no such element, search for all elements in $\mathbf{Q}'_i$ such that the third component is δ_g. That is, find all j' such that $\mathbf{Q}'_i(j') = \{*, \delta_g, *, *\}$ failing which find all j' such that $\mathbf{Q}'_i(j') = \{*, *, \delta_g, *\}$. For all such j':
        — Compute $j''' \leftarrow \pi(i,3)(j'')$, where $j'' \leftarrow \pi(i,2)(j')$ has already been computed in the tally computation audit.
        — Publish $\pi(i,3)(j'')$. Compute and publish the subkey used to commit to $\pi_{(i,3)}(j'')$: $\kappa_{\pi_{(i,3)}}(j'') \leftarrow \mathsf{Subkey}(K, \{\text{“}\pi\text{"}, 3, i, j''\})$.
        — Publish $\mathbf{S}'''_i(j''')$. Compute and publish the commitment subkey for this value: $\kappa_{s_{i,j'''}} \leftarrow \mathsf{Subkey}(K, \{\text{“}s\text{"}, i, j'''\})$.
   b) If $\mathcal{C}(i) = 1$:
      - Regenerate $\pi_{(i,2)}$ (do not publish it).
      - For each ballot $PA_g = \{\delta_g, q_{gn}, q_{gn+1}, \ldots, q_{gn+n-1}, s_0, s_1 \ldots, s_{n-1}\}$ that is print-audited:
        i) Search for all elements in $\mathbf{Q}'_i$ such that the second component is δ_g. If there is no such element, search for all elements in $\mathbf{Q}'_i$ such that the third component is δ_g. That is, find all j' such that $\mathbf{Q}'_i(j') = \{*, \delta_g, *, *\}$ failing which find all j' such that $\mathbf{Q}'_i(j') = \{*, *, \delta_g, *\}$. For all such j':
        — Compute $j'' \leftarrow \pi(i,2)(j')$. Note that $j''' \leftarrow \pi(i,3)(j'')$ has already been computed in the tally computation audit.
        — Publish $\pi_{(i,2)}(j')$. Compute and publish the subkey used to commit to $\pi_{(i,2)}(j')$: $\kappa_{\pi_{(i,2)}}(j') \leftarrow \mathsf{Subkey}(K, \{\text{“}\pi\text{"}, 2, i, j'\})$.
        — Publish $\mathbf{Q}'_i(j')$. Compute and publish the commitment subkey for this value: $\kappa_{x_{i,j'}} \leftarrow \mathsf{Subkey}(K, \{\text{“}x\text{"}, i, j'\})$ where $x = \{\alpha_g, \beta_g, \gamma_g, q\}$.

F. Correctness Proofs

We summarize the proofs of correctness that verifying agents A can perform and explicitly state conditions under which the proof completes successfully. Note that in general the best practice response to proofs that do not complete successfully (i.e., fail) is an open policy question, and not considered here. Specifically in case of voter receipts, however, a failed receipt check has a dispute resolution process described in Section II-C.

Note that, for the print audit and tally check correctness proofs, A will verify commitments. In particular, A will confirm that all commitment keys that were challenged as a result of the challenge coin-tosses and the print-audit were responded to (i.e., published on $\mathcal{BB}$) during steps 2 and 3 in the previous section. For all commitment keys κ_x to message x posted to $\mathcal{BB}$ during the audit, A searches $\mathcal{BB}$ for the corresponding message x and commitment value $\bar{x}$, and tests whether $\mathsf{Decommit}(\kappa_x, x, \bar{x})$ outputs 1 (valid). This verification step is successful if and only if all of A's executions of $\mathsf{Decommit}()$ output 1.

Receipt Check: For all challenges $\mathcal{C}(i) = 0$ and $0 \leq j < (b \cdot n)$, A locates permutations $\pi_{(i,2)}$, code lists $\mathbf{Q}'_i$, and recorded mark lists $\mathbf{R}''_i$ on $\mathcal{BB}$. A reconstructs the assertion of the voting system, that $\mathbf{Q}'_i(j)$ is marked or not marked as indicated by the mark value $\mathbf{R}''_i(\pi_{(i,2)}(j))$.

This verification step successfully verifies voter-receipt $\mathrm{VR}_g = \{\beta_g, \gamma_g, q_{gn+d}\}$ if and only if A is able to conclude that all reconstructed assertions agree with VR_g. Specifically for $0 \leq i < I$, VR_g is said to agree with the assertions if $\{\beta_g, \gamma_g, q_{gn+d}\}$ exists at position j in $\mathbf{Q}'_i$, if $\mathbf{R}''_i(\pi_{(i,2)}(j)) = 1$, and if all other occurrences of β_g and γ_g (that is, all $n-1$ other values of tuples $\{\beta_g, \gamma_g, q_{gn+\hat{d}}\}$ found at positions $\mathbf{Q}'_i(\hat{j})$ correspondingly show recorded mark $\mathbf{R}''_i(\pi_{(i,2)}(\hat{j})) = 0$.

Print Audit: A reconstructs assertions of the code-candidate associations of each print-audited ballot. For all i, $0 \leq i \leq I$, and for each print-audited ballot $PA_g = \{\delta_g, q_{gn}, q_{gn+1}, \ldots, q_{gn+n-1}, s_0, s_1 \ldots, s_{n-1}\}$

1) A searches for all elements in $\mathbf{Q}'_i$ such that the second component is δ_g. If there is no such element, A searches for all elements in $\mathbf{Q}'_i$ such that the third component is δ_g. That is, A finds all j' such that $\mathbf{Q}'_i(j') = \{*, \delta_g, *, *\}$ failing which A finds all j' such that $\mathbf{Q}'_i(j') = \{*, *, \delta_g, *\}$. For all such j':
   a) A locates permutation element $\pi_{(i,2)}(j')$, computes $j'' \leftarrow \pi_{(i,2)}(j')$, locates permutation element $\pi_{(i,3)}(j'')$, computes $j''' \leftarrow \pi_{(i,3)}(j'')$;
   b) A locates $\mathbf{R}''_i(j'')$ and $\mathbf{S}'''_i(j''')$;
   c) The assertion is that $\mathbf{Q}'_i(j')$ is the confirmation number corresponding to the candidate $\mathbf{S}'''_i(j''')$ and that the unique ballot with one serial number δ_g has not been voted.
2) This verification step successfully verifies the print-audited ballot if it agrees with the assertions. That is, if A is able to obtain n values of j' and conclude that, $\forall j'$:
   a) the corresponding commitments were opened correctly;
   b) $\{\delta_g, q_{gn+\iota}\} \in \mathbf{Q}'_i(j')$ for some ι such that $0 \leq \iota \leq n-1$ and that each value of ι corresponds to exactly one value of j';
   c) $\mathbf{R}''_i(j'') = 0$;
   d) $\mathbf{S}'''_i(j''') = s_\iota$.

Tally Check:

1) A will check that the corresponding commitments were opened correctly.

2) A will verify $\mathcal{BB}$ self-consistency:
   a) For all challenges $\mathcal{C}(i) = 0$ and $0 \leq j < (b \cdot n)$, A locates permutations $\pi_{(i,2)}$, code lists $\mathbf{Q}'_i$ and recorded mark lists $\mathbf{R}''_i$ on $\mathcal{BB}$. A reconstructs the assertion of the voting system, that $\mathbf{Q}'_i(j)$ is marked or not marked as indicated by the mark value $\mathbf{R}''_i(\pi_{(i,2)}(j))$. This verification step successfully verifies $\mathcal{BB}$ self-consistency if all public voted confirmation numbers correspond exactly to $\mathbf{R}''_i(\pi_{(i,2)}(j)) = 1$.
   b) For all $\mathcal{C}(i) = 1$, A locates $\pi_{(i,3)}$, $\mathbf{R}''_i$, and $\mathbf{S}'''_i$ on $\mathcal{BB}$. For all j, A reconstructs the assertion of recorded mark $\mathbf{R}''_i(j)$ made for candidate $\mathbf{S}'''_i(\pi_{(i,3)}(j))$, and computes the election outcome by tallying each of these assertions. He checks the declared tally against the computed tally. This verification step successfully verifies $\mathcal{BB}$ self-consistency if the two tallies are identical.

IV. Security Analysis

In Section III, we described the verification proofs for receipt checks, the tally check, and print-audited ballots. In this section, we both quantify the effectiveness of the verification and consider the security of the Scantegrity II to additional attacks, most involving a procedural element not easily captured by a cryptographic description. Thus, the goal of this analysis is to sketch the security heuristics underlying the design, and not to rigorously prove security properties in a formal cryptographic model.

We consider three categories of attacks. The first category is *manipulation attacks*, in which the goal of the attacker is to manipulate the final tally so that the election's outcome is more favorable to the attacker's preferred candidate(s). The second is *identification attacks*, where the goal of the attacker is to form a link between voting intent and ballot receipts. The final category is *disruption attacks*, in which the attacker wishes to prevent the completion or certification of the election. Since, in general, disruption attacks are applicable in any voting system, and difficult to prevent, we will only consider a special-case of disruption involving the prevention of certification of any tally in the event the attacker feels the results may be unfavorable.

In order to best frame this discussion we note that, as an enhancement to optical scan, Scantegrity II is inherently constrained by our design goal of noninterference with the underlying optical scan processes. For this reason, Scantegrity II is designed to be a strict improvement over optical scan systems with manual recounts. However, components which cannot be secured without intervening in the underlying processes of optical scan are not pursued.

A. Assumptions

The level of security of Scantegrity II depends on the nature of the attack. Critical components offer probabilistic security that is invariant to the adversary's computational power, while other components premise their security on one or more assumptions, both procedural and cryptographic in nature. The security setting of our analysis includes the following assumptions:

1) the existence of a trusted computing platform for use by election officials (*contra* identification attacks);
2) the set of collusive officials in the election authority does not satisfy the threshold requirement for recovery of the master key (identification and disruption);
3) chain-of-custody over the printed ballots prior to voting day (identification);
4) the inability of voters and others to read codes printed in invisible ink (manipulation, identification);
5) proper balancing of the pollbook (manipulation);
6) the intractability of obtaining information about a message given only its cryptographic commitment (identification);and
7) the intractability of opening a cryptographic commitment of a message differing from that message initially committed to (manipulation).

In our view, most of these assumptions are reasonable and standard in the literature. The trusted platform is a scaled-down computing device, with no external memory, running software attested by the trustees that performs the cryptographic operations. To avoid collusion among trustees, they could be selected from competing political parties. Using a threshold scheme allows the election to proceed even if a group of trustees is unable, or refuses, to supply their key share. Prior to the election, printed ballots must be protected against an adversary revealing codes and reprinting substitute ballots. Assumption 4 is unique to our approach and we provide justification for it in Section V. "Balancing the pollbook" refers to the assumption that the sum of the number of voted, tallied,and spoiled ballots is equal to the number of cast ballots, which is not larger than the number of voters. Assumptions 6 and 7 are referred to as the hiding and binding properties of commitments respectively in the previous section.

B. Manipulation Attacks

1) Printing: An adversary may misprint ballot B_g, so that the code $q_{gn+\hat{d}}$ associated with candidate $s_{\hat{d}}$ in the master list $\mathbf{P}$ is printed beside a different candidate s_d (or all candidates) on the same ballot. If the adversary then modifies any EBI_g associated with such a misprinted ballot such that $r_{gn+\hat{d}} = 1$ and $r_{gn+d} = 0$, the system will count the vote for $s_{\hat{d}}$ and report q_{gn+d} as the confirmation code, which is consistent with what appears on the ballot.

The print audit mechanism, described in Section III, is designed to make such an attack detectable by revealing discrepancies between printed ballots and $\mathbf{Q}'_i$, using commitments $\bar{\mathbf{Q}}'_i$ under assumption 7. If the number of ballots chosen to be print-audited is $0 \leq b_a \leq b$,where b is the number of ballots in the election overall, the probability of detecting at least 1 of $1 \leq b_f \leq b$ misprinted ballots is

$$\mathbf{Pr}[\text{detection}] = 1 - \frac{\binom{b-b_f}{b_a}}{\binom{b}{b_a}} = \frac{(b-b_f)!(b-b_a)!}{b!(b-b_f-b_a)!}. \quad (1)$$

2) Voting: One line of manipulation attack can exist in systems that are not diligent in spoiling ballots [4], [21]. If an attacker has a line of communication with the voter, the voter can

be instructed to mark her ballot and wait for further instruction. The attacker then communicates to the voter to either spoil the ballot or cast it. If the spoiled ballot is not protected or destroyed, the attacker may consult it to see how the voter would have voted had the attacker instructed the voter to cast the ballot. The line of communication can be eliminated by using random material on the ballots to determine the instruction, in a way analogous to the approach of making interactive protocols noninteractive. Scantegrity II avoids this line of attack by having spoiled ballots shredded in front of the voter, without the poll worker seeing the contents of the ballot.

A second line of manipulation attack can exploit the presence of undervoted ballots. An attacker may add additional marks to a contest left empty by the voter during a recount or appropriately modify the digital records.[1] This attack is not introduced by Scantegrity II and exists is any optical scan voting system. One method of prevention is to require each voter to mark a "none of the above" selection when denoting an undervote. Similarly, an attacker might try to prevent a correctly cast ballot from being tallied by overvoting it; this attack is prevented by not allowing any overvoted ballots to be cast.

3) Auditing: Consider a manipulation attack based on swapping voter-made marks in $\mathbf{R}''_i$ from one candidate to the attacker's preferred candidate. To prevent this attack, with probability 1/2, each back-end will be challenged to open the correspondence between the lists $\mathbf{R}''_i$ and $\mathbf{Q}'_i$, and any modified mark states for these instances will be incongruent with the voter receipts. The attacker may gamble, only modifying marks in roughly half of the back-end instances in the hope that exactly these will have challenge $\mathcal{C}(i) = 1$ and thus that, instead, the correspondence between the lists $\mathbf{R}''_i$ and $\mathbf{S}'''_i$ is instead revealed in the modified instances i. The probability of doing so is 2^{-I}. However, if a different subset is revealed, the tallies across the subsets will differ and the attack is detectable. Alternatively, the attacker might modify $\mathbf{R}''_i$ for all instances $1 \leq i \leq I$, which guarantees self-consistent tallies but also guarantees the attack is detectable by the receipt check protocol. At first glance this may seem to be an irrational strategy until one considers the possibility of only a small subset of voters actually checking their receipts. With I instances, $b_r \leq b$ ballots actually cast, b_c ballot receipts checked, and b_m modifications to each $\mathbf{R}''_i$, the probability of detection is $(b_r - b_m)!(b_r - b_c)!/b_r!(b_r - b_m - b_c)!$. The adversary will choose the least detectable of the two strategies, thus,

$$\mathbf{Pr}[\det .] = \min\left(1 - \frac{1}{2^I}, 1 - \frac{(b_r - b_m)!(b_r - b_c)!}{b_r!(b_r - b_m - b_c)!}\right). \quad (2)$$

By estimating b_c and bounding b_m as half of the smallest margin of victory for which we can certify an election, we can use this equation to determine a suitable I for our implementation such that the first term exceeds the estimated value of the second. In most instances, $I = 10$ is suitable.

A second approach to manipulating the tally is to change the final state of the ballots. Ballots can have one of three states: voted, print-audited, or spoiled. Under assumption 5, we assume that modifications must preserve the number of ballots in each state. If a voted ballot is maliciously modified to be spoiled, a spoiled ballot must be converted into a voted ballot. To prevent these transitions, the voter retains positive evidence of ballots being in a voted state: knowledge of both serial numbers, $\{\beta_g, \gamma_g\}$. Alternatively, for a print-audited ballot, the voter retains positive evidence a ballot was print-audited via knowledge of all the confirmation codes on the ballot, $\{q_{gn}, q_{gn+1} \ldots q_{gn+n-1}\}$ but only *one of* $\{\beta_g, \gamma_g\}$. Both pieces of information would be unknown to the voter if the ballot were in any other state when the voter left the polling place.

In the case of spoiled ballots, the voter does not retain anything. However, if a spoiled ballot is maliciously converted into a voted ballot, a voted ballot will need to be spoiled, and the corresponding voter can prove malfeasance through knowledge of both chit serial numbers.

The transition from a spoiled to print-audited state is important for different reasons. This transition does not change the tally directly; however, it is indirectly useful in facilitating the first manipulation attack presented in Section IV-B2. By misreporting a spoiled ballot as print-audited, the confirmation codes on the ballot would be released during the verification process allowing a coercer to see if the ballot matched the conditions of the contract for spoiling the ballot. Under assumption 5, this attack will be detectable as it requires a print-audited ballot to be made into a spoiled ballot. To prevent this attack, the trustees could first publish a list of ostensibly spoiled ballots prior to releasing the print audit confirmation codes. If an auditor discovers her print-audited ballot is in the wrong state, the discrepancy can be caught prior to releasing the codes.

C. Identification Attacks

1) Initialization: The earliest opportunity for identification occurs during the election initialization process. Successfully changing or introducing faults into the initialization protocol could generate a permutation of $\mathbf{P}$ or subsequent lists that is known to the attacker. This is not possible if the protocol is run on a trusted computing platform and assumption 1 holds. Without direct interference with the protocol, the attacker may provide structured data instead of randomness in the protocol. However, under assumption 2 and the construction of the threshold key generation scheme, any amount less than the minimum threshold of shares leaks negligible information for the purposes of determining the key.

2) Printing: After the ballots are printed, a number of identification attacks may be conducted including the addition of revealing marks on the ballots or revealing the codes on the ballots, recording these codes, and reprinting the ballots with unrevealed ink. The prevention of these attacks is based on assumption 3.

3) Auditing: After the election has concluded, the data generated and published for voter-verification of the tally must meet the requisite ballot secrecy. Given no information other than the tally, a certain level of information can be obtained about which candidate a voter selected. The tally provides a probability distribution for the possible selections and may even exclude selections, based, for example, on a candidate receiving zero votes.

[1]Although this issue was previously known to the authors, we acknowledge David Wagner for raising it in private correspondence.

This level of information is often legally required and thus acceptable. If the attacker is provided, in addition, with the information on each voter's receipt, further information is revealed: how many marks the voter made and the codes associated with these marks. Our assertion of ballot secrecy is that no additional information is leaked about the association between a mark and code on a receipt and any element in the set of selections in the tally.

Opening only one of the commitments to either a) the correspondence between confirmation codes $\mathbf{Q}'_i$ and voter marks $\mathbf{R}''_i$ or b) the correspondence between voter marks $\mathbf{R}''_i$ and candidates $\mathbf{S}'''_i$ reveals negligible information about permutations $\pi_{(i,3)}$ or $\pi_{(i,2)}$, respectively. Hence, the association between $\mathbf{Q}$ and $\mathbf{S}$ is always hidden by one cryptographic permutation. The commitment to the permutation key, if binding, uniquely identifies the permutation, however, reversing the commitment is assumed intractable by assumption 6.

D. Disruption Attacks

In general, disruption attacks are easy to detect but difficult to prevent. Many of the manipulation attacks could be reconstructed as disruption attacks, and the same mechanisms would detect them. However, as stated, we limit our consideration to disruption for the purpose of preventing the certification of an undesired tally (or an expected undesired tally, if the information is based on exit polls for example).

1) Initialization: During the initialization phase, each trustee in the election authority supplies entropy to seed the random number generator used to generate all the permutation keys and commitment secrets needed in the election. Instead of maintaining state, since the state information would need to remain private, when the tally and audit challenge/response phases are entered, the trustees re-enter their key shares to recreate all the necessary data. To prevent a malicious trustee from withholding their entropy or supplying the wrong entropy, we use a threshold key generation scheme (optionally with robustness to a finite number of errors). Under assumption 2, a suitable threshold will allow the reconstruction of the data despite malicious trustees.

2) Auditing: During the auditing phase, an attacker may file a spurious dispute about the results of a receipt-check. Since the election authority has committed to the confirmation codes that appeared on the ballot, it can rule out any claimed codes that did appear on the ballot. Thus, filing a spurious but plausible dispute reduces to randomly guessing another code on the ballot. The election authority can quantify the probability of this and create an appropriate statistical trigger that predicts actual receipt-check problems. Let n be the number of candidates on a candidate list L for a particular race and let Σ^l be the cardinality of the set of unique confirmation codes. The probability of guessing a plausible code on a voted ballot is $p = (n-1)/(\Sigma^l - 1)$. If D disputes are filed and G are considered plausible, the expected value of G, if disputes are fabricated, is $\mu = D \cdot p$. We set the trigger value τ such that the probability of obtaining at least τ plausible discrepancies if all filed disputes are random guesses is less than 1%. We can use the following bound on the right tail of the binomial distribution [13]. For any $r > \mu$, $\Pr[G - \mu \geq r] \leq (\mu e/r)^r$.

For example, for 5 candidates, 8000 possible codes, and 1000 disputes filed, assuming no scanning error, $p = 4/7999 = 0.0005$ and $\mu = 1000 \cdot 0.0005 = 0.5$. Using $r = 4.5$ we get $\Pr[G \geq 5] \leq (0.5e/4.5)^{4.5} = 0.0046 < 0.01$, so we can set $\tau = 5$. If at least 5 out of the 1000 disputes filed are plausible discrepancies, then an investigation should be instigated. To allow for up to some acceptable rate s of scanning error, we can incorporate s into the probability p of guessing a correct code and compute the statistical trigger as above with the new value of p.

V. Invisible Ink Details

In this section, we describe the main categories of threats that might take advantage of the properties of invisible ink, our assumptions about ink properties, and the procedures for printing the inks on the ballot. Greater detail is available in [5].

A. Threats

Note that the only threats to Scantegrity II that take advantage of the limitations of the ink are those that are based on the following.

1) Distinguishing between confirmation codes and their backgrounds.
   The ability to distinguish would allow:
   a) voters to falsely claim election fraud; and
   b) anyone with access to ballots to violate ballot secrecy by connecting confirmation codes to selections.
2) Distinguishing between chit serial numbers and backgrounds.
   The ability to distinguish would allow
   a) voters to claim that an uncast ballot was cast;
   b) anyone with access to uncast ballots to connect chit serial numbers and confirmation codes with voter selections [in combination with (1)].
3) Distinguishing between the two-dimensional barcode and background.
   The ability to distinguish would allow anyone with access to marked ballots to connect two-dimensional bar-codes with voter selections

B. Assumptions

The main security assumption about the inks is that the slow- and fast-reacting inks used for printing confirmation codes and oval backgrounds, respectively, are not distinguishable before, and sufficiently after, they have been marked with the ballot-marking pen ("sufficiently after" is taken to mean that the time period is long enough to allow the slow-reacting ink to react). We make a similar assumption about the indistinguishability of the chit serial numbers and the two-dimensional barcode from their background.

Note also that the assumptions we make are about physical properties of chemicals, and the detectability of differences. Clearly, most chemicals (if not all) can be distinguished from one another through a sufficiently sophisticated test; our arguments are that, for all practical purposes, our assumption holds, and we describe here our efforts to make it more difficult to distinguish among the inks, particularly by the naked or microscopically aided human eye.

Finally, the ability to distinguish enables voters to make false charges of election fraud, and anyone to connect information about ballot choices with confirmation codes and serial numbers. If voters are assumed to not have access to ballots outside the polling booths, or to specialized equipment (including the decoding pen) inside the booth, the indistinguishability assumption is only required to hold with respect to the human eye in order to prevent false charges of election fraud.

C. Procedures Used for Printing With the Inks

In this section, we describe ways in which the indistinguishability assumptions may be defeated, and our efforts to preserve indistinguishability. Note that the inks proposed for printing on ballots can be used in regular ink-jet printers.

1) To Prevent the Soaking of Paper: Any type of ink used by inkjet printers soaks into the paper. Even if the ink used to print the codes would be completely invisible, the soaked paper would allow the codes to be easily read. To avoid this, we use two types of ink: a reacting ink used to print the background of the oval and a slow-reactive ink used to print the confirmation codes. Both inks have the same color (a light yellow) if printed on the same piece of paper. The reacting ink turns black immediately when it interacts with the ink of the marking pen, while the ink used for the codes undergoes the same reaction at a slower pace. Thus, the immediate result is a yellow confirmation code inside a black oval—the highest contrast color combination. After several minutes, the slow reacting ink will have reacted leaving the oval completely black.

2) To Avoid the Overlapping of Inks: We divide the oval in small square tiles called texels. Each texel is entirely printed with either reactive or slow-reactive ink, but never with a combination of them. A small constant-sized gap is left between any two adjacent texels, such that when two adjacent tiles are printed with different inks, the two inks never overlap even if they diffuse outward as they absorb into the paper. Without such a gap, a border of overlapping types of ink could emerge, under a microscope, for example, making the border easier to detect. Additionally, we ensure that the position of the code in the oval is not fixed; the codes can be shifted left or right.

3) The Addition of Confusing Fluorescence: The use of special types of radiation can expose invisible inks. We apply a third type of ink that we call a masking ink. It is colorless but has high fluorescence. Masking ink is the last ink sprayed onto the paper. We add random amounts of masking ink to all texels of the oval. This is designed to mask the eventual difference in fluorescence between the reactive ink and slow reactive ink used for the codes, as well as a cover to prevent lifting particles from the paper with tape.

4) Ballot-Marking Pens: While this paragraph is not about security properties of the inks used, it is relevant to the discussion of Scantegrity II procedures with respect to inks, and is hence described here. The ballot-marking pens that we use to mark the ovals have a tip that is wider than the height of the oval. A voter can mark the entire oval using a single strike of the pen which is faster than penciling in the mark. Even if the voter pens in more than the oval, the result is a clean, perfectly filled oval. The use of invisible ink also deters stray dark marks that can confuse scanners, although the light yellow hue of the ink could still be visible. The portion of the chit reserved for the voter to record the confirmation codes can also have a solid layer of the same reacting ink, so that the voter may record the codes with the same pen.

VI. Scantegrity II for Voters With Disabilities

In this section, we describe modifications to Scantegrity II to allow its use by those voters who have visual or motor disabilities, and hence cannot mark a Scantegrity II paper ballot. These modifications are inspired by those for Punchscan and *Prêt à Voter* described in [10]. In our approach, the voter is presented with an audio ballot and interacts with the voting system using a microphone and headphones. The voting system prints the vote on a Scantegrity II ballot. The voter with visual disabilities also has access to a trusted interactive device that translates a visual signal into another type of signal, such as an audio signal; this device is used to check a marked ballot. Finally, all voters using the audio interface have access to a personal voice recorder used to record the confirmation number. Details of our approach follow.

Filling a Ballot: The voter is presented the choices for each race through the headphones, and communicates her choice to the voting system through the microphone. The voting system communicates the vote to a printer. The printer prints, on a Scantegrity II ballot, with the ink also used in the Scantegrity II pen, a blob on the corresponding oval, exposing the code as with ballots for other voters. Assistive devices that have been used in the past to help voters with visual or motor disabilities may also be modified for the purpose of filling a Scantegrity II ballot. Examples of such devices include Tactile Ballots which have been used in elections in Rhode Island [17], and the Voting-on-Paper Assistive Devices (Vote-PADs),[2] which consist of a plastic ballot-sleeve, tactile indicators, and an audio tape recording, customized for each election and ballot design.

Checking a Marked Ballot for Correctness: The voter who does not have visual disabilities, but has motor disabilities that make it difficult to mark ballots, may check the correctness of the filled ballot, and dictate the confirmation code into a personal (trusted) voice recorder.

The voter with visual disabilities will use a trusted interactive device—consisting of a trusted scanner with optical character recognition (OCR) and speakers—to check that the ballot is correctly marked. The voter may bring such a device with her to the booth, or may be provided one by a trusted third party, such as a public interest group. With the aid of this device, the voter may translate a marked Scantegrity II ballot into an audio signal, and determine if it has been correctly marked. Additionally, this device would read aloud the confirmation number, which could be taped into a personal (trusted) voice recorder.

Casting the Ballot: Once a ballot has been correctly marked, it may be processed like any other marked Scantegrity II ballot.

Print Audits: If ballots given to voters using this procedure to vote are drawn at random from the pile of ballots for all voters, print audits are applicable to a ballot marked as described above. Further, a voter with visual or motor disabilities can also perform a print audit by marking audit ballots as described above.

[2]Accessible voting without computers. http://www.vote-pad.us/.

Security Properties: This approach does not provide the same security guarantees to voters with visual disabilities as those provided to other voters, who need not rely on a trusted device in the polling booth. A compromise of the trusted device can result in a compromise of the integrity of the vote, as well as in an opportunity for a coercive adversary. On the other hand, the only implemented voting systems that are usable by voters with visual disabilities—DREs, optical scan ballot systems with assistive devices, or Prime III [14]—require that the voter either trusts the voting system itself, or the chain of custody on the ballot box or a paper/audio audit trail. These are stronger assumptions than the requirement that a *personal* device be trusted by a voter. Thus, the above modification of Scantegrity II, like those of Punchscan and *Prêt à Voter* described in [10], provides a much-needed accessible version of voter verifiability—where the voter may determine that her vote is among those tallied, and that the collection of votes is tallied correctly—without having to trust a device provided by election officials.

VII. Related Work

The use of cryptography in voting originates in 1981 with Chaum [6]. The ensuing decades saw the introduction of many electronic voting systems using cryptography to achieve both privacy and integrity. Only more recently have schemes emerged where voters use paper ballots and/or obtain paper receipts; for example, *Prêt à Voter* [11], Punchscan [15], [16], [26], Scratch & Vote [1], ThreeBallot [27], Simple Verifiable Voting [3], Split-Ballot Voting [22], and the protocol of Neff [24]. Public key techniques have dominated the cryptographic verification of tally computation; for example, the *universally verifiable mixnet* of Sako and Kilian [28], and the tally correctness proofs of Furukawa and Sako [18], and Neff [23].

A scheme by Chaum [7] was the first to provide the voter with a receipt for the purposes of verifying the presence of her vote in the vote collection, without requiring her to have access to trusted computational power while casting her vote. The first use of a perforated ballot, where a voter can take a perforated part of the ballot out of the polling booth as a receipt, appears in *Prêt à Voter*. ThreeBallot [27] also uses a perforated ballot. Scratch & Vote was the first to provide a string to the voter that was obtained only after the voter performed an action on the ballot (scratching off a layer), it was also the first to use two-dimensional barcodes. The light use of cryptography is inspired by the verification protocols in previous systems including "Votegrity" [7] and Punchscan, combined with the mixnet auditing technique of randomized partial checking [19]. The approach towards providing accessibility draws from [10].

VIII. Concluding Remarks

We have demonstrated a simple and effective way to dramatically increase the transparency of elections that use optical scan voting systems. It is our hope that its adoption will help prevent the manipulation of election outcomes, and that it may lead to renewed confidence and participation in democracy.

References

[1] B. Adida and R. L. Rivest, "Scratch & vote: Self-contained paper-based cryptographic voting," in *Proc. the 5th ACM Workshop on Privacy in Electronic Society (WPES)*, 2006, pp. 29–40.

[2] S. Ansolabehere and C. Stewart, "Residual votes attributable to technology," *J. Politics*, vol. 67, pp. 365–389, 2005.

[3] J. Benaloh, "Simple verifiable elections," in *Proc. 2006 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT)*, Vancouver, Canada, 2006.

[4] J. Benaloh, "Ballot casting assurance via voter-initiated poll station auditing," in *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT)*, Boston, MA, 2007.

[5] R. T. Carback, Printing Secure Automatic Receipts With Activating Ink Tech. Rep., 2009 [Online]. Available: http://scantegrity.org/~carback1/ink/ink.pdf

[6] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[7] D. Chaum, "Secret-ballot receipts: True voter-verifiable elections," *IEEE Security Privacy*, vol. 2, no. 1, pp. 38–47, Jan./Feb. 2004.

[8] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A. Ryan, E. Shen, and A. T. Sherman, "Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes," in *Proc. 2008 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT)*, San Jose, CA, 2008.

[9] D. Chaum, A. Essex, R. Carback, J. Clark, S. Popoveniuc, A. T. Sherman, and P. Vora, "Scantegrity: End-to-end voter verifiable optical-scan voting," *IEEE Security Privacy*, vol. 6, no. 3, pp. 40–46, May/Jun. 2008.

[10] D. Chaum, B. Hosp, S. Popoveniuc, and P. L. Vora, "Accessible voter verifiability," *Cryptologia*, vol. 33, no. 3, pp. 283–291, 2009.

[11] D. Chaum, P. Y. Ryan, and S. A. Schneider, A Practical, Voter-Verifiable, Election Scheme University of Newcastle Upon Tyne, Tech. Rep. Series CS-TR-880, Dec. 2004.

[12] W. Clarkson, T. Weyrich, A. Finkelstein, N. Heninger, J. A. Halderman, and E. W. Felten, "Fingerprinting blank paper using commodity scanners," in *Proc. 30th IEEE Symp. Security and Privacy*, 2009, pp. 301–314.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, 2nd Edition*. Cambridge, MA: MIT Press, McGraw-Hill Book Company, 2000.

[14] E. V. Cross, II, Y. McMillian, P. Gupta, P. Williams, K. Nobles, and J. E. Gilbert, "Prime III: A user centered voting system," in *Proc. 2007 Conf. Human Factors in Computing Systems (CHI)*, 2007, pp. 2351–2356.

[15] A. Essex, J. Clark, R. T. Carback, and S. Popoveniuc, "Punchscan in practice: An E2E election case study," in *Proc. 2007 IAVoSS Workshop on Trustworthy Elections (WOTE)*, Ottawa, Canada, 2007.

[16] K. Fisher, R. Carback, and A. T. Sherman, "Punchscan: Introduction and system definition of a high-integrity election system," in *Proc. 2006 IAVoSS Workshop on Trustworthy Elections (WOTE)*, Cambridge, U.K., 2006.

[17] M. Fresolone, Tactile Ballots: Alternative Voting Method for the Blind Tech. Rep. [Online]. Available: http://www.votersunite.org/info/tactileballots.asp

[18] J. Furukawa and K. Sako, "An efficient scheme for proving a shuffle," in *Proc. 21st Conf. Advances in Cryptology (CRYPTO)*, 2001, vol. LNCS 2139, pp. 368–387.

[19] M. Jakobsson, A. Juels, and R. L. Rivest, "Making mix nets robust for electronic voting by randomized partial checking," in *Proc. 11th USENIX Security Symp.*, 2002, pp. 339–353.

[20] D. Jones, Voting on Paper Ballots [Online]. Available: http://www.cs.uiowa.edu/~jones/voting/paper.html

[21] J. Kelsey, A. Regenscheid, T. Moran, and D. Chaum, "Hacking paper: Some random attacks on paper-based E2E systems," presented at the Frontiers of Electronic Voting Dagstuhl, Germany, 2007.

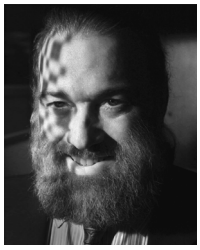
[22] T. Moran and M. Naor, "Split-ballot voting: Everlasting privacy with distributed trust," in *Proc. 14th ACM Conf. Computer and Communications Security (CCS)*, 2007, pp. 246–255.

[23] C. A. Neff, "A verifiable secret shuffle and its application to e-voting," in *Proc. 8th ACM Conf. Computer and Communications Security (CCS)*, 2001, pp. 116–125.

[24] C. A. Neff, Practical High Certainty Intent Verification for Encrypted Votes Tech. Rep., 2004 [Online]. Available: www.votehere.net/old/vhti/documentation/vsv-2.0.3638.pdf

[25] T. P. Pedersen, "A threshold cryptosystem without a trusted party," in *Proc. 1991 Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT)*, 1991, vol. LNCS 547, pp. 522–526.

[26] S. Popoveniuc and B. Hosp, "An introduction to Punchscan," in *Proc. 2006 IAVoSS Workshop on Trustworthy Elections (WOTE)*, Cambridge, U.K., 2006.
[27] R. L. Rivest and W. D. Smith, "Three voting protocols: ThreeBallot, VAV, and Twin," in *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT)*, Boston, MA, 2007.
[28] K. Sako and J. Kilian, "Receipt-free mix-type voting scheme: A practical solution to the implementation of a voting booth," in *Proc. 1991 Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT)*, 1995, vol. LNCS 921, pp. 393–403.
[29] R. Saltman, Accuracy, Integrity, and Security in Computerized Vote-Tallying Tech. Rep. NIST SP 500-158, Aug. 1988.
[30] T. R. Weiss, As Primary Season Ramps Up, An E-Voting Snapshot Computerworld, Jan. 8, 2008.



David Chaum received the M.S. degree and the Ph.D. degree in computer science from the University of California, Berkeley, in 1980 and 1983, respectively.

He founded DigiCash, Inc., where he was CEO from 1993 to 1998. Before that, he built and lead the Cryptography Group at Centrum voor Wiskunde en Informatica (Center for Mathematics and Computer Science), Amsterdam, The Netherlands, from 1985 to 1992. He has also held positions at University of California Santa Barbara and at New York University Graduate School of Business. He has published over 45 original technical articles and received over 17 U.S. patents. He is widely considered to have invented secure electronic voting, with a paper describing a technique for anonymous electronic voting in 1981, and several papers since. He is also generally associated with the invention of electronic money and anonymous credentials. He was the first to propose: mix networks, dining-cryptography networks, blind signatures, untraceable credentials, minimum disclosure, group and undeniable signatures. He has also made early and fundamental contributions to the area of multiparty computations.

Dr. Chaum is founder of the International Association for Cryptographic Research (IACR) and cofounder of Workshop on Trustworthy Elections (WOTE), a series of conferences and its sponsoring organization the International Association for Voting Systems Sciences (IAVOSS).

Richard T. Carback was born in Baltimore, MD. He received the Master's degree in computer science from the University of Maryland, Baltimore County (UMBC), in May 2008. He is currently pursuing the Ph.D. degree in computer science at UMBC.

He is a Research Associate for Convergent Technologies Incorporated, Baltimore, MD, supporting computer security development and training efforts. Previously, he has worked as a Research and Teaching Assistant at UMBC, and a Software Engineer at L-3 GSI, Inc. His research interests include end-to-end election systems, privacy enhancing technologies, virtual systems security, computer network operations, cryptology, and other topics in computer security and information assurance.

Jeremy Clark received the B.E.Sc. in computer engineering from the University of Western Ontario, Canada, in 2004, and the M.A.Sc. degree in electrical engineering from the University of Ottawa in 2007. He is currently pursuing the Ph.D. degree in computer science at the University of Waterloo, Waterloo, ON, Canada.

He is a member of the Centre for Applied Cryptographic Research (CACR) and the Cryptography, Security and Privacy (CrySP) research group. His research interests are in applied cryptography and game theory.

Mr. Clark is a recipient of the Alexander Graham Bell Canada Research Scholarship.

Aleksander Essex received the B.E.Sc. degree in computer engineering from the University of Western Ontario, Canada, in 2004, and the M.A.Sc. degree in electrical engineering from the University of Ottawa in 2008. He is currently pursuing the Ph.D. degree in electrical and computer engineering at the University of Ottawa, Canada.

He is a member of the Information Security Research Group, University of Ottawa. His research interests are in applied cryptography, engineering design, and voting technologies.

Stefan Popoveniuc received the Ph.D. degree from The George Washington University, where he focused on computer security and privacy in general and on electronic voting in particular. His thesis provided a general framework that allows election officials to evaluate and take informed decisions when purchasing end-to-end voting systems.

He is a founding member of the PunchScan team; he has fully implemented a number of voting systems, PunchScan, Scantegrity, and Scantegrity II being just three of them. Currently, he is an election technology consultant in the Washington, D.C., area.

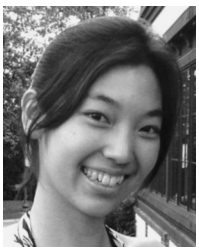
Ronald L. Rivest received the B.A. degree in mathematics from Yale University in 1969, and the Ph.D. degree in computer science from Stanford University in 1974.

He is the Viterbi Professor of Computer Science in the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (MIT). He is a member of MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL), a member of the lab's Theory of Computation Group, and is a leader of its Cryptography and Information Security Group. His research interests are in cryptography, computer and network security, algorithms, and voting systems. He is an inventor of the RSA public-key cryptosystem, and has extensive experience in cryptographic design and cryptanalysis. He is a founder of RSA Data Security and a cofounder of Verisign and of Peppercoin.

Prof. Rivest is a member of the National Academy of Engineering, the National Academy of Sciences, and is a Fellow of the Association for Computing Machinery, the International Association for Cryptographic Research, and the American Academy of Arts and Sciences. He also serves on the EPIC Advisory Board. Together with Prof. Adi Shamir and Prof. Len Adleman, he has been awarded the 2002 ACM Turing Award. He has received an honorary degree (the "laurea honoris causa") from the University of Rome. In 2005, he received the MITX Lifetime Achievement Award; in 2007, he received both the Computers, Freedom and Privacy Conference "Distinguished Innovator" award, and the Marconi Prize. He has served as a Director of the International Association for Cryptologic Research, and of the Financial Cryptography Association. Most recently, he has served on the Technical Guidelines Development Committee, an advisory board to the U.S. Election Assistance Commission.

Peter Y. A. Ryan received the Ph.D. degree in theoretical physics from the University of London, U.K.

Since February 2009, he is Professor of Information Security at the University of Luxembourg. Before moving to Luxembourg, he was a professor at Newcastle University. His research interests include cryptography, modeling and verification of cryptographic protocols and secure systems, verifiable voting systems, and quantum information assurance. He has served on the PC of many security conferences and has served as program chair for several. He was the chair of the Steering Committee of ESORICS from 1999 to 2007.



Emily Shen received the B.S. degree in computer science from Stanford University, in 2006, and the S.M. degree in electrical engineering and computer science from Massachusetts Institute of Technology (MIT), in 2008. She is currently pursuing the Ph.D. degree in electrical engineering and computer science at MIT.

She is a recipient of the Bell Laboratories Graduate Research Fellowship. Her research interests are in cryptography and computer security.

Alan T. Sherman received the Ph.D. degree in computer science from Massachusetts Institute of Technology (MIT) studying under Ronald L. Rivest, the S.M. degree in electrical engineering and computer science from MIT, and the Sc.B. degree in mathematics (*magna cum laude*) from Brown University.

He is an Associate Professor of Computer Science at the University of Maryland, Baltimore County (UMBC) in the CSEE Department, Director of UMBC's Center for Information Security and Assurance, and a member of the National Center for the Study of Elections at UMBC. His main research interest is high-security voting systems. He has carried out research in election systems, algorithm design, cryptanalysis, theoretical foundations for cryptography, and applications of cryptography.

Dr. Sherman is also a private consultant performing security analyses, an editor for Cryptologia, and a member of Phi Beta Kappa and Sigma Xi.

Poorvi L. Vora received the B.Tech. degree in electrical and electronic engineering from IIT Mumbai, in 1986, the M.S. and Ph.D. degrees in electrical engineering from North Carolina State University (NCSU), in 1988 and 1993, and the M.S. in mathematics from Cornell University, in 1990.

She is an Associate Professor in the Department of Computer Science at The George Washington University, Washington, D.C., where she has been on the faculty since 2003. Before 2003, she worked at Hewlett-Packard (HP) Company in various positions in HP Laboratories, as well as in the Imaging and Printing Group (IPG). Her current research interests are in the application of ideas from communication theory and signal processing to problems in security, such as electronic voting, cryptology, and counterfeit deterrence.