Paperless Independently-Verifiable Voting

David Chaum, Alex Florescu[1], Mridul Nandi[3], Stefan Popoveniuc, Jan Rubio[1], Poorvi L. Vora[1,2], and Filip Zagórski[1]

[1] The George Washington University, Washington D.C.**
[2] Indian Institute of Technology, Bombay
[3] Indian Statistical Institute, Kolkata

Abstract. We present a new model for polling-booth voting: the voter enters the polling booth with a computational assistant which helps her verify that her vote is correctly recorded. The assistant interacts with the voting system while the voter votes on the machine in the polling booth. We present an independently-verifiable, coercion-resistant protocol based on this model. Unlike all other independently-verifiable protocols, this one is completely paperless and does not require the voter to perform any tasks outside the polling booth. We provide property definitions, rigorous claims and a description of a prototype.

1 Introduction

Independently-verifiable protocols were first proposed almost a decade ago, and have been tried in binding elections, including one small governmental election. All of the secure independently-verifiable protocols require the use of paper, however, and also require the voter to perform checks outside the polling booth. This has perhaps slowed down the adoption of these protocols, which have particularly strong verifiability properties. Additionally, Direct Recording Electronic (DRE) voting machines—with all their flaws—enable voters with disabilities to vote independently for the first time ever. The use of paper hence presents a step backwards for this category of voter. In a first attempt towards rectifying some of these problems, this paper presents an independently-verifiable polling-booth protocol which is completely paperless.

The protocol is based on a new model for independently-verifiable polling booth voting: the voter enters the polling booth with a computational assistant that she has brought with her. This could be, say, a smartphone, or special-purpose hardware. She puts it into a special docking station where she can see its output behind a transparent cover (much like the screen used for VVPATs), but cannot provide any input to it. A blind voter may obtain output through the use of headphones, and be prevented from providing input by disabling the microphone; again, specially-designed hardware could be useful. We assume that the voter takes no pictures and votes alone so no one can watch her vote. An adversary may, however, query her through alternate channels (such as the scratch-off cards proposed in [13]).

** This work was done while Nandi was at The George Washington University. Florescu, Rubio and Vora were supported in part by NSF Award No. 0831149, Nandi and Zagórski by NSF Award No. 0937267. Zagórski was also supported in part by the Polish Ministry of Science and Higher Education scientific project - grant N N206 369839

In the protocol, while the voter votes on the machine in the polling booth (this could be a DRE machine that is capable of performing cryptographic operations), the assistant interacts with the machine. It performs computations based on the interaction, and provides information to the voter to help her determine if her vote is recorded correctly. The role of the assistant is to provide the voter the capability to perform digital signatures and commitments, and to perform the random challenges on behalf of the voter. The assistant cannot determine the vote from the information it obtains. The data communicated in both directions between assistant and voting machine is signed by the sender and verified by the receiver, and made public immediately after the protocol ends. Hence, as with other independently-verifiable protocols, the encryption-correctness proof—which is based on the interaction between the computational assistant and the voting machine—can be checked by anyone. In contrast with existing independently-verifiable protocols, the voter's tasks are completed inside the polling booth, and the process is entirely paperless. We do not assume the existence of a randomness beacon.

We present a coercion-resistant independently-verifiable fully-electronic protocol using this model. We are not aware of any other such protocols that do not require an external independent randomness beacon.

1.1 Existing Paperless Protocols and Their Limitations

It may be argued that the design of an independently-verifiable paperless protocol has been the goal of cryptographers since the invention of secure electronic voting in the early eighties. However, all the existing all-electronic protocols have vulnerabilities.

The classical protocols are vulnerable because the vote is entered through the voter's machine which is trusted to keep the vote secret. Malware on the voter's machine knows the vote, and can also change it.

The newer, all-electronic independently-verifiable voting systems may be represented by *polling-booth-Helios*. While Helios is a remote voting system, it is an all-electronic simplification of polling-booth protocols Simple-Verifiable-Voting [3] or Voter-Initiated Poll Station Auditing [4] which provide paper receipts. It is hence easily modified for polling booth use, and we use it for illustrative purposes. When we refer to polling-booth-Helios in this paper, we will mean an all-electronic polling-booth protocol where the voter communicates an electronic vote to the untrusted voting machine, obtains an electronic hash of her receipt as a commitment from the machine, communicates electronically whether she wants to cast (or audit) the receipt, and receives an electronic receipt (or proof of encryption-correctness). We use it to represent a simple all-electronic independently-verifiable polling-booth system.

Polling-booth-Helios is vulnerable to two types of attacks:

1. Coercive attack: the adversary coerces the voter to make a challenge that is a function of the receipt hash. Because the voter does not know whether her vote will be cast or audit when she is entering it, and she knows that there is a possibility it will be audited, or exposed to the adversary, she has an incentive to vote as directed by the adversary. Further, if this adversary colludes with the voting system, the latter's

attempts at cheating cannot be detected because it will know what challenge to expect. This attack can be avoided if the voter is required to commit to her challenge bit before she sees any information on the ballot, however a voter cannot commit to the bit on an all-electronic system without access to trusted computation[4].
2. Challenge correctness: the voting system can choose to ignore the challenge and proceed to cast or audit as it wishes. While the voter knows the system is cheating, she cannot prove it. The inability to resolve a dispute about the correctness of the challenge can also allow a voter to falsely claim a voting system is cheating.

1.2 Our Contributions

We address the problem of making and checking commitments in an all-electronic protocol by having the computational assistant perform a more active role than in the current independently-verifiable voting systems. The assistant makes and verifies cryptographic commitments and digital signatures on behalf of the voter. We avoid the coercion attacks possible in the classical cryptographic protocols—where too the voter's computer performed an active role—by preventing the voter from providing any input to the assistant. Finally, we make fewer assumptions than in the classical model where the voter's computer is trusted to follow instructions. We also do not assume the availability of a beacon of randomness. Our contributions are as follow:

- ◇ We model an *Actively-Assisted-Human Interactive Proof (AAHIP)*. In this model the voter votes on the voting machine and does not provide any input to the assistant. The assistant interacts with the voting machine and provides information to the voter. The interaction between assistant and voting machine takes place over an authenticated channel, where both assistant and voting machine sign messages and check signatures. All the data sent over the channel, in both directions, is made public immediately after the protocol ends. The assistant is allowed to deviate from protocol, and its memory and logs can be examined by anyone once the vote is cast. We also allow the assistant to instruct the voter before the protocol begins, and to query her at any time.
- ◇ We provide a rigorous description of a *paperless* vote-casting protocol based on a cut-and-choose AAHIP. We provide definitions and rigorous statements of our results. Space limitations prevent us from providing proofs. We assume the model of an AAHIP and that either the voting machine or the assistant is honest. We define *soundness* and *coercion-resistance* and demonstrate that the protocol achieves both given the assumptions. We are not aware of other all-electronic protocols that have these properties. In particular, note that when the assistant is dishonest the protocol

[4] Paper-based systems can enable the voter to make a commitment without using computation– for example, poll workers may mark the voter's choice of challenge on her paper ballot before handing it to her. In an all-electronic voting system, however, without access to any other computation while voting, the voter is unable to make the commitment. In order to do so, the commitment would need to be digital and the voter would need access to a trusted computer. This computer could change the challenge as it wished without informing the voter. An audit challenge would result in an audit of both the voter and the voting machine. In trying to avoid one coercive adversary, the Helios voter is exposed to another.

is still coercion-resistant. Classical all-electronic protocols are coercion-resistant only when the voter's computer is trusted.

- ◇ We describe a simpler challenge-response AAHIP for vote-casting, which is much like polling-booth-Helios with an assistant. We show that it has weaker coercion-resistance. Thus the vulnerabilities of polling-booth-Helios are not overcome by naturally extending it to the AAHIP model. We are not aware of any other work that observes a distinction between cut-and-choose and challenge-response protocols.
- ◇ We describe our prototypes for both challenge-response and cut-and-choose protocols. An Android-based smartphone performs the role of the computational assistant in the prototypes. Note, of course, that the voter's device can be a special hardware device and not a smartphone; at this time, however, we do not have access to special hardware.

1.3 Comparison With Other Approaches

Our protocol provides the following security improvements over polling-booth-Helios (note that the remote version also possesses these vulnerabilities, but they may not be of as much consequence in the elections Helios was designed for). Again, polling-booth-Helios is being used only for illustrative purposes, as a representative simple all-electronic system:

- ◇ In contrast with polling-booth-Helios, the voter can prove that a challenge bit was changed if the assistant is honest.
- ◇ Further, in contrast with polling-booth-Helios, the voter cannot be coerced if the voting machine is honest.

We are able to achieve the above two properties because we have constructed a protocol where the challenge bit is issued by the assistant and not by the voter. One cannot simply add digital signatures to the challenge bit in Helios to achieve the first property, because the voter cannot check digital signatures without an assistant. Further, the second property is not achieved by naturally extending Helios to use a computational assistant to issue the challenge bit and to make and verify digital signatures. We show that a protocol that is a natural extension of Helios with a computational assistant is not coercion resistant, even if it is assumed that the voter does not provide input to the assistant. That is, it is not possible to achieve the properties achieved by our main protocol by simply extending Helios to the AAHIP model.

On the other hand, our protocol also shares some of the limitations of polling-booth-Helios and does not achieve all the security properties of paper-ballot-based voting systems. If the voting machine is dishonest, an honest assistant can help the voter detect the cheating, but neither can prove it. In a paper-ballot-based system, the voter and one of the computational assistants she uses can provide the proof.

Note that one may fault our protocol because, if the assistant and voting machine are both dishonest and colluding, the voter will not detect an attempt to change the tally. While this is possibly a consequence of more general results on protocols in which more than half of the parties are dishonest, note that polling-booth-Helios also has a similar problem. Consider an adversary colluding with the voting system to change the tally

in our protocol. The adversary prevents the voter from using her chosen assistant and requires her to use one the adversary provides. This assistant can provide pre-prepared challenges known to the voting system. This will result in a fraudulent encryption-correctness proof. The same adversary can use the coercion channel present in polling-booth-Helios to force the polling-booth-Helios voter to execute a challenge that is a function of the receipt-hash. In collusion with this adversary, the polling-booth-Helios voting system can also predict challenges, also resulting in a fraudulent encryption-correctness proof. However, in our protocol, the coercive adversary cannot coerce the voter if the voting machine is honest; this is not true with polling-booth-Helios.

1.4 Organization

The paper is organized as follows. Section 2 reviews related work. Section 3 provides an informal description of the model. Section 4 describes the challenge-response protocol which is a natural extension of polling-booth-Helios to the AAHIP model, and describes a simple coercion attack on it. The main protocol is described informally in 5. Section 6 provides a rigorous description of the model and the main protocol. Section 7 provides a rigorous set of property definitions with theorem statements, and section 8 describes the prototypes. Section 9 presents our conclusions.

2 Related Work

The first description of the notion of coercion-resistance is due to Benaloh and Tuinstra [5]. Their protocol assumes the existence of a randomness beacon, avoiding the coercion avenue created when voters are allowed to issue the challenges. Later protocols in the classical model, such as that of Juels, Catalano and Jakobsson [11] assume that the computer used to encrypt the vote may be trusted to do so correctly (or that voters are Interactive Turing Machines — ITMs). This is clearly not a valid model for polling-place voting. The first description of a secure voting protocol where voters are not ITMs is due to Chaum [7]; however this protocol requires the use of visual cryptography and two transparent layers stacked one on top of the other with very good registration and is highly impractical. These protocols were quickly followed by several others, such as MarkPledge [14], Prêt à Voter [16], Punchscan [15], Simple-Voter-Verifiable [3], Votebox [17], Helios [2] and Scantegrity [8].

Of the independently-verifiable protocols, most are paper-based. Votebox and Helios, based on Simple-Voter-Verifiable, are fully-electronic, but limit themselves to elections where coercion is not a concern.

The protocols we describe and study in this paper were first informally proposed by Chaum, Popoveniuc and Vora [9]. We present slightly modified versions in this paper. The original paper does not contain rigorous descriptions or proofs and they do not present any information on prototypes. Additionally, the original paper was not presented at a venue with proceedings.

Our notion of an AAHIP extends Adida's work on AHIPs [1].

3 The Informal Model

In our protocols, we use the notion of a commitment, and that of oblivious transfer. A cryptographic *commitment scheme* enables a sender to "commit" to a value M without revealing it. It does so by providing a "commitment", com_M, to the receiver. At a later stage, the sender can "open" the commitment and reveal M to the receiver. The receiver can verify that M is the original value committed to. *1-out-of-2 oblivious transfer* is a protocol between a sender and a receiver by which the receiver can obtain only one of two secret elements K_0 and K_1 from the sender. The sender is oblivious of which of the two values the receiver received.

In all the protocols described here, there are three participants: voter, voting machine and computational assistant. The voter is human. Its computational capability is limited to the ability to compare small strings. The voting machine and computational assistant are Probabilistic Polynomial Time (PPT) Interactive Turing Machines (ITMs) with authenticated write access to a secure append-only bulletin board that can be read by anyone.

The general purpose of all the participants is as in other independently-verifiable voting systems, except for two important distinctions: the computational assistant is an interactive participant in the protocol and does not receive any input from the voter. All its input is provided by the voting machine.

We provide more detail on the similarities. As before, the voting machine presents the voter with a ballot. The voter votes on the voting machine, which provides a string which it claims is the encryption of the vote. The voting machine provides an interactive proof supporting this claim. The computational assistant checks the correctness of the proof transcript without knowing how the voter voted. The voter performs a check inside the booth. The voting machine is said to provide a correct encryption only when both checks are passed.

Details on the distinctions are as follow.

- ◇ The computational assistant (and not the voter) obtains the vote encryption and provides challenges to the voting machine to obtain a proof that the encryption is correct.
- ◇ The assistant provides information to the voter based on how the voting system responds to the challenges. This information may consist of more than the binary outcome of its checks. It may also include information on what should be on the ballot (for example, the correct ordering of candidates, or the correct association of candidates with dummy variables). The voter compares this information to that on the ballot presented to her by the voting machine. If the information matches, she accepts that the encryption is correct.
- ◇ The voter does not provide any information to the assistant and only receives information from it.
- ◇ The voter has to choose a single assistant to participate in the protocol (in an AHIP, the voter can present her receipt to several distinct assistants asking each to check for her).

All interaction between the two ITMs is signed and verified. If a signature does not verify, the recipient party requests a resend and aborts after a pre-determined fixed num-

ber of failed attempts to verify. Similarly, when a commitment is not opened correctly, the aggrieved ITM can abort the protocol. Thus, either party can perform a denial-of-service. As this possibility exists in the use of all computational devices, we do not consider it any further. Note that, because information between the two ITMs is signed and published after the protocol ends, the channel between the two parties is much like a public append-only channel. The data across this channel will demonstrate which party has cheated.

The voter can also abort the protocol if she catches a party behaving dishonestly. However, because he is not an ITM, she cannot sign and verify signatures and hence does not share a verifiable tape with any party. Hence she is typically not able to prove certain types of dishonest behavior.

The assistant would typically be provided by an individual or organization the voter chose. This is not to say the individual or organization is not malicious or dishonest and will not attempt to coerce the voter to vote in a certain manner. Our main protocol is coercion-resistant if at least one of the voting machine or assistant is honest and the voter does not provide any input to the assistant during the protocol. This is true even if the adversary can query the voter and examine the memory and logs of the assistant after the vote is cast and if the assistant is allowed to deviate from protocol.

4 The Challenge-Response Protocol and a Weakness

In this section we present a slight generalization of the protocol named eTegrity in [9]—a challenge-response-style protocol which is the natural extension of Helios to the AAHIP model. The voter familiar with the ballot audits of Prêt à Voter [16] or Scantegrity [6] will notice the similarity with ballot audits. We name the generalized protocol eChallengeResponse. The protocol uses the assistant to make and verify commitments and digital signatures in a straightforward manner. Its simplicity makes it easy to use. Space restrictions prevent us from going into more detail. We describe a coercive attack on this protocol, illustrating the difficulty of designing a coercion-resistant all-electronic protocol.

In this protocol, the assistant first commits to the challenge bit on behalf of the voter. It sends the challenge bit to the voter, and the commitment to the voting machine. The voter enters her vote and the voting machine sends the vote-encryption to the assistant. The assistant opens its commitment and the voting machine audits or casts the encryption based on what the challenge bit is. The assistant checks encryption-correctness if the challenge corresponds to an audit. After describing the protocol we describe a simple coercive attack, thus motivating the somewhat more complicated protocol which is our main contribution. The challenge-response protocol and the coercive attack are interesting because they illustrate the subtle problems involved in designing an incoercible protocol.

4.1 eChallengeResponse: Informal Description

In general, an independently-verifiable voting system will provide a receipt which functions as an encryption of the vote. We will denote by $\mathcal{E}_s$ the cipher for the ballot with

serial number s. That is, whether $\mathcal{E}_s$ is implemented as a cipher or a look-up table, it has the properties of a secure cipher. In order to vote, a voter walks into a polling booth with a computational assistant.

To begin the protocol, the assistant and the voting machine set up a communication link. Before the voter and voting machine interact, the assistant performs a cryptographic commitment to a uniformly-distributed bit b representing whether or not it will audit the upcoming encryption. It sends b to the human voter and the commitment c_b to the voting machine. Note that it has not been possible to perform this step electronically in an AHIP, and independently-verifiable voting systems have required a combination of paper and procedures for this step [5].

The voting machine sends to the assistant the serial number s of the ballot it will use. After the assistant checks that s is fresh, the voting machine presents the fresh ballot with serial number s to the voter. The voter selects her candidate, v, on the voting machine and confirms her vote. The voting machine then presents the signed pair of the serial number and encrypted vote, $(serial, encryption)$, to the assistant. In correct instances of the protocol, $(serial, encryption) := (s, \mathcal{E}_s(v))$.

The assistant opens c_b to reveal b to the voting machine.

- ◇ If the commitment verifies and corresponds to a choice of "cast", the vote is cast. Both voting machine and assistant post the signed pair $(serial, encryption)$ in the list of verified votes on the secure bulletin board. The protocol ends.
- ◇ If the commitment verifies and corresponds to a choice of "audit", the voting machine reveals the vote it encrypted, $vote$, and the encryption parameters. Both voting machine and assistant post the signed pair $(serial, encryption)$ in the list of audited votes on the secure bulletin board. The assistant provides the value of $serial$ and $vote$ to the voter. If $vote$ is the vote she cast, i.e. if $vote = v$, and $serial$ is the serial number of the ballot presented to her, i.e. if $s = serial$, the voter knows the encryption was correctly performed.

The voter may repeat the above protocol many times if she desires, at different times during the day, and at different voting machines.

4.2 A Coercive Attack

We now describe a simple coercion attack when the voting machine is honest but the assistant is not. This motivates the use of the more complicated cut-and-choose protocol we describe next, which is not vulnerable to this attack. The coercer tells the voter to vote for candidate i and provides her with the assistant. An examination of the assistant after the vote is cast will demonstrate to the coercer whether the voter used it or not,

[5] As in an AHIP, the voter may choose b and communicate it to the assistant. However, because the assistant commits to b, any security analysis must assume that the assistant chooses b. Further, the coercion-resistance analysis also requires that the voter not be allowed to provide input to the assistant, so, if the voter were allowed to choose this bit, she should not be allowed to provide any other information to the assistant. This bit itself does not provide any information because it could as well have been communicated outside the booth. (It is obtained from the voter before she herself has been provided any information in the polling booth).

so the voter cannot use another assistant. The assistant is programmed to perform the correct protocol, except for the fact that it provides no information to the voter in the first step (when it is supposed to send her the challenge bit). The voter will always vote for candidate i because she does not know when *her* commitment to her vote (the encrypted vote) will be audited by the coercer (assistant).

We now propose the following, more complicated protocol, based on a cut-and-choose approach which we implement with the use of 1-out-of-2 Oblivious Transfer.

5 The Cut-and-Choose Protocol

In this section we informally describe the protocol named ePunchscan in [9] with a few modifications; our protocol is named eCutAndChoose. It is a cut-and-choose protocol for a commitment-based cryptographic voting system. It uses a mix of ideas from Prêt à Voter, Punchscan and Scantegrity. Its most striking aspect, the use of oblivious transfer, is, however, common only to Punchscan among the independently-verifiable protocols that have been used in real elections.

The ballot consists of a serial number s and two ballot parts: (i) $Part_0$ contains a permutation π_s of the c candidates, reflecting the order in which candidates will be presented to the voter (ii) $Part_1$ consists of a list of Scantegrity codes, one for each of c candidate positions. Each part bears the serial number s. $Part_0$ also bears another serial number, s_L.

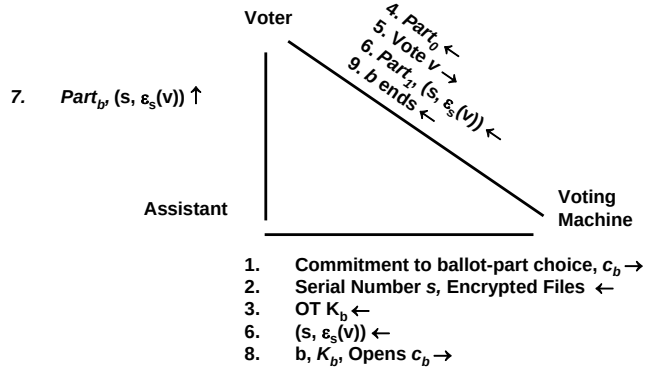

Fig. 1. The Cut-And-Choose Protocol. Arrows denote direction of communication.

5.1 Protocol Description

Before the election starts, the voting system makes cryptographic commitments to the two ballot parts separately, as well as to the commitments required by a Scantegrity

back-end that will obtain vote tallies from confirmation codes. To begin, a voter walks into a polling booth with a computational assistant. The assistant and the voting machine set up a communication link.

Commitment to Choice Of Ballot Part: [Step 1 in Figure 1] The assistant performs a cryptographic commitment to a bit b representing the ballot part it will choose to get by Oblivious Transfer. It sends b to the human voter and the commitment c_b to the voting machine. Let us say the voter gets bit. In correct instances of the protocol, $bit := b$. Note that, in AHIP-based Punchscan, this step is performed using a combination of paper and polling procedures. Polling officials note down whether a voter will take home the top or bottom layer before the voter is allowed to see the ballot[6].

Ballot Preparation: [Step 2] The voting machine sends to the assistant the serial number s of the ballot it will use, and the assistant checks that s is fresh. The voting machine creates two ciphertexts. The i^{th} ciphertext, $i = 0, 1$ is a symmetric-key encryption of $File_i$, which consists of $Part_i$ and the information required to verify its published commitments. $File_i$ is encrypted with symmetric-key encryption using key K_i for $i = 0, 1$. Each key is pseudo-randomly generated afresh for each ballot.

Oblivious Transfer: [Step 3] The assistant obtains K_b from the voting machine by oblivious transfer, obtains $File_b$ by decrypting the appropriate ciphertext, and checks the commitments to $Part_i$.

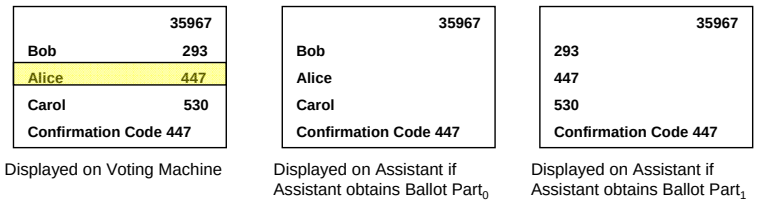

Fig. 2. Voter's View of the Ballot After Casting a Vote for Alice

Ballot Presentation: [Step 4] The voting machine presents $Part_0$ of the ballot with serial number s to the voter. This looks much like a Prêt à Voter ballot.

Voting: [Steps 5] The voter enters her vote v on the voting machine and confirms it.

Receipt: [Step 6] The voting machine now presents $Part_1$ and the traditional Scantegrity confirmation number, $\mathcal{E}_s(v)$, to the voter. It also presents the signed confirmation number to the assistant.

Receipt: [Step 7] The assistant presents $Part_b$ and the confirmation code to the voter. The voter checks that the ballot part and confirmation number presented by the assis-

[6] As in an AHIP, the voter may choose b and communicate it to the assistant. However, because it is the assistant that performs the cryptographic commitment to b, any security analysis must assume that the assistant chooses b. Further, as in eChallengeResponse, the voter must not be allowed to provide any other information to the assistant during the protocol.

tant matches the corresponding part and confirmation number presented by the voting machine. (See Figure 2.)

Closing: [Steps 8-9] The assistant opens its commitment to bit b for the Voting Machine and also provides K_b. The Voting Machine provides bit b to the voter, so it may know how to lie about the ballot half the assistant does not have if coerced by the assistant. The voting machine casts the vote by posting the signed pair of serial number and vote encryption in the list of cast votes on the secure bulletin board. It informs the voter and assistant that the vote is cast. The assistant posts the signed pair of serial number and vote encryption it received from the voting machine in Step 6 in the list of verified votes on the secure bulletin board. The protocol ends.

5.2 Dispute Resolution

Notice that, if the voter and the voting machine disagree about what transpired between them over their private channel, it is not possible to resolve the dispute except through physical observation. So, for example, the voting machine might constantly behave as though the voter voted $v' \neq v$, or might refuse to abort when instructed to, etc. While the voter would be aware that the voting machine was cheating, she would not be able to prove it as the tape between the two parties is not verifiable and can be easily overwritten by the voting machine. While this is not as bad as the original problem with Direct Recording Electronic voting machines, it demonstrates the difference between the use of electronic and paper tapes (paper ballots) between voting machines and voters. Every interaction between voter and voting system in paper ballot independently-verifiable systems such as Prêt à Voter and Scantegrity is on an append-only write-once tape such as a paper ballot or through physical processes. This allows the voter to provide evidence when the voting system does not follow protocol. Notice that this can be a problem even if the voting system is honest, when the voter is dishonest. A small group of vocal dishonest voters can call into question an honest election by accusing the voting machine of not following protocol.

A possible solution to this problem is to allow the voter to interact with the assistant to obtain a blind signature on her confirmation code and to vote by casting the blinded confirmation code, through the assistant. The voting machine then will not know what the vote is and cannot change it. It is not clear whether this opens the voter up to more coercion-attacks, and whether there are verifiability and security problems with this protocol that are similar to those found in the past with blind-signature-based protocols.

6 Rigorous Description: eCutAndChoose

In this section, we provide a rigorous description of the model and of the informally-described cut-and-choose protocol in [9], where it is referred to as ePunchscan. We describe only set-up and vote-casting; vote tallying proceeds as with Scantegrity.

6.1 The AAHIP Model

We do not describe the most general version of this model here. We focus only on the specific case when a voting machine seeks to prove to a voter that a string x is an

encryption of her vote v using a cipher $\mathcal{E}$. There are three participants in the protocol: the voter $\mathcal{V}$; the voting machine $\mathcal{V}M$ and the assistant $\mathcal{A}$. $\mathcal{V}$ is a human whose computational capability is limited to the comparison of strings of size n bits. $\mathcal{V}M$ and $\mathcal{A}$ are PPT ITMs. There are the following channels among the participants:

1. A two-way private channel between $\mathcal{V}$ and $\mathcal{V}M$ provided by the poll booth. No one other than $\mathcal{V}$ or $\mathcal{V}M$ can read from, or write to, this channel.
2. A two-way authenticated channel between $\mathcal{A}$ and $\mathcal{V}M$ made append-only through the use of digital signatures. This channel can be read by anyone at any time. Only $\mathcal{A}$ and $\mathcal{V}M$ have read-write access to it.
3. A one-way channel from $\mathcal{A}$ to $\mathcal{V}$ which is private during the voting process and public thereafter. Only $\mathcal{A}$ can write to it.

Note that there is no channel from $\mathcal{V}$ to $\mathcal{A}$ during the protocol; however, $\mathcal{A}$ may query $\mathcal{V}$ after the vote is cast.

The view of participant $\mathcal{X}$ is denoted $view_{\mathcal{X}}$. The transcript of the interaction between participants $\mathcal{X}$ and $\mathcal{Y}$ is denoted $\mathtt{Tape}_{\mathcal{X},\mathcal{Y}}$.

The protocol takes as input (private) vote v from $\mathcal{V}$ and (public) challenge bit b from $\mathcal{A}$. We denote it $\mathcal{A}AHIP(\mathcal{V}(v), \mathcal{A}(b))$. It produces the following output:

1. The receipt, public output from $\mathcal{V}M$:

$$(\mathtt{Receipt}_s, \mathtt{Receipt}_E, \mathtt{Receipt}_P) :=$$

$$\mathtt{Receipt}(v, b) = (s, \mathcal{E}_s(v), \mathtt{Proof}(\mathcal{E}_s(v), b))$$

   where s is the serial number of the ballot, $\mathcal{E}_s(v)$ the claimed encryption of the vote, and $\mathtt{Proof}(\mathcal{E}_s(v), b)$ the proof that $\mathcal{E}_s(v)$ is correctly constructed, for challenge b. It consists of the transcript between $\mathcal{V}M$ and $\mathcal{A}$.
2. The check of the assistant, public output (True or False) from $\mathcal{A}$:

$$\mathtt{CheckProof}(\mathtt{Receipt}(v, b))$$

   indicating whether the commitments are correctly opened by $\mathcal{V}M$ in $\mathtt{Receipt}$. This is similar to the output produced by $\mathcal{A}$ in an AHIP.
3. The ballot-part, (private during the protocol but public after the vote is cast) from $\mathcal{A}$ to $\mathcal{V}$:

$$\mathtt{Part}(view_{\mathcal{A}})$$

   which represents what $\mathcal{A}$ has learnt about the manner in which $\mathcal{E}_s(v)$ was constructed by $\mathcal{V}M$. In our cut-and-choose protocol, $\mathtt{Part}(view_{\mathcal{A}})$ is the ballot part obtained by oblivious transfer.
4. The check of the voter, public output (True or False) from $\mathcal{V}$:

$$\mathtt{CheckPart}(\mathtt{Part}(view_{\mathcal{A}}), view_{\mathcal{V}})$$

   which indicates whether $\mathtt{Part}(view_{\mathcal{A}})$ is consistent with what $\mathcal{V}$ observes in the private channel with $\mathcal{V}M$. In our cut-and-choose protocol, $\mathtt{CheckPart}$ indicates whether the chosen part—as determined by the assistant through the opening of the original commitments—matches the corresponding part in the ballot displayed to the voter.

6.2 Initial Set-Up by Election Officials

Let $\mathcal{C}$ define the Scantegrity code-space; that is, it is the set of all possible Scantegrity confirmation codes. It can be any set such that (a) individual elements of the set can be compared by humans (that is, each code is no longer than n bits long) and (b) the size of $\mathcal{C}$ is large enough so that probability $\epsilon := |\mathcal{C}|^{-1}$ is small enough.

We consider each race separately. Let c be the number of candidates in the race and N the number of ballots to be generated. Election officials (EOs) perform the following tasks prior to the election:

1. Generate confirmation codes $\mathsf{code}_{s,i}$, for all serial numbers $1 \leq s \leq N$ and candidates $i \in \mathbb{Z}_c$. The c confirmation codes on a single ballot are distinct. That is,

$$\mathsf{code}_{s,i} \neq \mathsf{code}_{s,j}\ i \neq j,\ i,j, \in \mathbb{Z}_c\ 1 \leq s \leq N$$

   Informally speaking, $\mathsf{code}_{s,i}$ should be the secure symmetric-key encryption of i corresponding to ballot s. That is, the function $f : \{0,1,2,..,N\} \times \mathbb{Z}_c \rightarrow \mathcal{C}$ with $f(s,i) = \mathsf{code}_{s,i}$ should have the properties of a symmetric-key encryption (with keyspace $\{0,1,2,..,N\}$, message space $\mathbb{Z}_c$ and ciphertext space $\mathcal{C}$) where the key is a secret function of s. The scheme should be such that the advantage of a PPT adversary in the eavesdropping indistinguishability experiment [12, page 63] is negligible in the length of s.
2. Generate a secret pseudo-random permutation $\pi_s(.)$ of $\mathbb{Z}_c$ (the candidates) which is independent of $f(s,.)\ \forall s$.
3. Generate the ballots, as pairs. For each serial number s, generate the ballot: $(s, Part_0, Part_1)$ where $Part_0 = \pi_s$ and $Part_1 = \pi_s(\langle \mathsf{code}_{s,i} \rangle_{i \in \mathbb{Z}_c})$.
4. Generate commitments to the correspondence between candidate and confirmation code for each ballot and candidate separately, as well as commitments to both parts separately for each ballot using commitment scheme $\mathfrak{C}$ (see Appendix for definition)

$$(com_{s,i}, open_{s,i}) = \mathfrak{C}_{\mathrm{EO}}(s, \mathsf{code}_{s,i})\ \forall\ s, i \in \mathbb{Z}_c$$

$$(com_{s,part_0}, open_{s,part_0}) = \mathfrak{C}_{EO}(s, part_0)\ \forall\ s$$

   and

$$(com_{s,part_0}, open_{s,part_0}) = \mathfrak{C}_{EO}(s, part_0)\ \forall\ s$$

5. Keep a secret record of

$$(open_{s,part_0}, open_{s,part_1}, open_{s,0}, open_{s,1}, ...$$

$$..., open_{s,i}, ..., open_{s,c-1})\ \forall\ s$$

   and publish the values of

$$(com_{s,part_0}, com_{s,part_1}, com_{s,0}, com_{s,1}, ...$$

$$..., com_{s,i}, ..., com_{s,c-1}) \forall\ s$$

   on a secure public bulletin board.

6.3 Casting a Vote

As with eChallengeResponse, this protocol is interactive among three parties: voter $\mathcal{V}$, voting machine $\mathcal{V}M$ and assistant $\mathcal{A}$. The steps below are numbered as in Figure 1. Note that any time a party finds that a commitment is not opened correctly, or finds that a signature is not verified, or notices that another party fails a check, it aborts the protocol.

1. $\mathcal{A}$ chooses $b \in \{0,1\}$ ($b = i$ implies that $\mathcal{A}$ will check $Part_i$). $\mathcal{A}$ sends signed commitment c_b to $\mathcal{V}M$ where $(c_b, o_b) \leftarrow \mathfrak{C}_{\mathcal{A}}(b \| \bar{b})$.
2. $\mathcal{V}M$ verifies signature on c_b. $\mathcal{V}M$ chooses a fresh serial number s at random and generates two secret encryption-keys K_0, K_1 at random. $\mathcal{V}M$ sends a signed message to $\mathcal{A}$ consisting of the serial number s and the ciphertext of two files $C_i = E(K_i, File_i)$, $i = 0, 1$, where $File_i = (Part_i \| open_{s,part_i})$. E denotes a secure symmetric-key encryption scheme.
3. $\mathcal{A}$ verifies the signature and checks on the secure bulletin board that s is unused. $\mathcal{A}$ performs an oblivious transfer with $\mathcal{V}M$ to obtain K_b: $OT_{\mathcal{A},\mathcal{V}M}(b, K_0, K_1)$. $\mathcal{A}$ decrypts $File_b$ with K_b and verifies the opened commitments in $File_b$. $\mathcal{A}$ sends the "ready" signal to $\mathcal{V}$.
4. $\mathcal{V}M$ presents $File_0$ with serial number s to $\mathcal{V}$.
5. $\mathcal{V}$ sends vote v to $\mathcal{V}M$.
6. $\mathcal{V}M$ sends the signed pair $(serial, \mathsf{code}) := (s, \mathsf{code}_{s,v})$—signed serial number and confirmation code—to $\mathcal{A}$ and to $\mathcal{V}$. It also sends $Part_1$ to $\mathcal{V}$.
7. $\mathcal{A}$ verifies the signature and that $serial = s$. $\mathcal{A}$ presents $Part_b$ and the pair $(serial, \mathsf{code})$ to $\mathcal{V}$. $\mathcal{V}$ checks that $Part_b$ as presented by $\mathcal{V}M$ is identical to $Part_b$ as presented by $\mathcal{A}$, and that the confirmation and serial numbers presented by both match. If they match, $\mathcal{V}$ sets $\texttt{CPart} := \texttt{CheckPart}(\texttt{Part}(view_{\mathcal{A}}), view_{\mathcal{V}}) =$ "True". Practically speaking, this is conveyed when $\mathcal{V}$ leaves the polling site without complaint. We denote the two identical parts as seen by the voter as $\texttt{ObservedPart}_b$.
8. $\mathcal{A}$ opens commitment c_b and sends signed values of b and K_b to $\mathcal{V}M$ which verifies both. $\mathcal{V}M$ now knows the value of b.
9. $\mathcal{V}M$ sends b to $\mathcal{V}$. The vote v is cast for the ballot and $\mathcal{V}M$ publishes the signed pair $(s, \mathsf{code}_{s,v})$ in the list of cast ballots on the public bulletin board. $\mathcal{A}$ publishes the signed pair (s, code) in the list of verified ballots on the public bulletin board. Discrepancies in the list are resolved by the examination of the public transcript between the two ITMs.

If a participant observes dishonest behavior by another participant, it aborts. When $\mathcal{V}$ aborts, she sets $\texttt{CPart} :=$ "False" (that is, she makes a complaint to a polling official). When $\mathcal{A}$ aborts, it sets $\texttt{CProof} :=$ "False".

The outputs are as follow:

1. The receipt:

$$\texttt{Receipt}(v, b) =$$
$$(\texttt{Receipt}_s, \texttt{Receipt}_E, \texttt{Receipt}_P)$$
$$:= (s, \mathsf{code}, \texttt{Tape}_{\mathcal{A},\mathcal{V}M})$$

2. The check of the assistant: $\texttt{CProof} := \texttt{CheckProof}(\texttt{Receipt}(v, b))$ is "True" if $\mathcal{A}$ publishes the signed pair $(\texttt{Receipt}_s, \texttt{Receipt}_E)$ in the list of verified votes, and is "False" else.
3. The ballot-part, sent by $\mathcal{A}$ to $\mathcal{V}$: $\texttt{ObservedPart}_b$
4. The check of the voter: if $\texttt{CPart} \neq$ "True" then $\texttt{CPart} :=$ "False".

7 Properties of the eCutAndChoose Protocol

In this section we provide rigorous statements for properties of the eCutAndChoose protocol, proof sketches may be found in the appendix. We model these properties on the more general ones for an AHIP in [1]. We assume that all cryptographic primitives used are secure and all adversaries are PPT.

Theorem 1 (COMPLETENESS). *If all three parties are honest,* $Receipt_E = \mathcal{E}_s(v)$, $\texttt{CProof} =$ *"True" and* $\texttt{CPart} :=$ *"True".*

Lemma 1. *If* $Receipt_E \neq \mathcal{E}_s(v)$ *and* $\mathcal{A}$ *and* $\mathcal{V}$ *are honest,*

$$Pr[\texttt{CProof} = False\, OR\, \texttt{CPart} = False] \geq \frac{1}{2} - \alpha$$

for some small value α

Lemma 2. *There is no means by which* $\mathcal{A}$ *can change a vote without colluding with* $\mathcal{V}M$.

Theorem 2 (SOUNDNESS). *If at least one of* $\mathcal{A}$ *and* $\mathcal{V}M$ *is honest and* $Receipt_E \neq \mathcal{E}_s(v)$,

$$Pr[\texttt{CProof} = False\, OR\, \texttt{CPart} = False] \geq \frac{1}{2} - \alpha$$

Theorem 3 (PRIVACY). *A public transcript for vote* v *is computationally indistinguishable from one for vote* v' *for all pairs* (v, v').

We denote the protocol $\mathcal{P}$. Consider a dishonest $\mathcal{A}$, denoted Adv desiring to coerce voter $\mathcal{V}$ to vote v. Suppose Adv designs a new interaction with $\mathcal{V}$ during the protocol, instructs $\mathcal{V}$ to vote v before or during the protocol, and queries $\mathcal{V}$ after the protocol is over. The purpose of this would be to coerce $\mathcal{V}$ into voting in a certain manner. Let us denote the new protocol $\mathcal{P}'$, and the view of original participant $\mathcal{X}$ in $\mathcal{P}'$ as $view_{\mathcal{P}',\mathcal{X}}$. Note that, from the perspective of $\mathcal{V}M$, the protocol is identical to $\mathcal{P}$. We denote by $\mathcal{V}_{coerced,\mathcal{P}'}$ the voter in $\mathcal{P}'$ obeying the instructions of $\mathcal{A}$. We say a protocol is coercion-resistant if there exists a strategy $\mathcal{V}^*$ for the voter which is indistinguishable from $\mathcal{V}_{coerced,\mathcal{P}'}$ for Adv in protocol $\mathcal{P}'$.

Theorem 4 (COERCION-RESISTANT). *If* $\mathcal{V}M$ *is an honest participant in protocol* $\mathcal{P}$, $\exists$ *PPT* $\mathcal{V}^*$ *such that* $view_{\mathcal{P}',\mathsf{Adv}}$ *when interacting with* $\mathcal{V}^*(v')$ *in protocol* $\mathcal{P}'$ *is computationally indistinguishable from* $view_{\mathcal{P}',\mathsf{Adv}}$ *when interacting with* $\mathcal{V}_{coerced,\mathcal{P}'}(v)$.

8 Prototypes

We have implemented proofs of concept which are fully functional for both eChallengeResponse and eCutAndChoose, and for both the voting machine and the personal assistant. Our implementation uses parts of the open-source Scantegrity II code. We use the Scantegrity II back-end implementation (which is actually a modified version of the Punchscan back-end [6]) to perform a verifiable-tally.

The prototype implements two modules:`DRE` (the voting machine) and `smartphone` (the computational assistant), which handle their own network communication.

The implementation of the `DRE` module is in Java 1.5, and has an interactive voice control system, as well as a synchronized visual interface. The two interfaces can be checked for consistency using observational testing. Full functionality for the `DRE` module has been tested to date on Microsoft Windows XP, Microsoft Windows 7 and Mac OS X Snow Leopard; there are some known problems on Linux systems that currently prevent full functionality. In our implementation, each instance of the `DRE` module posts confirmation codes directly online, to a secure public bulletin board (which we assume exists, but did not implement).

The `smartphone` module is implemented on an Android 1.6 platform, using Java 1.5 and the Bouncy Castle cryptographic library. We have successfully tested it on the HTC Hero and the Nexus One. Our security model assumes that the `smartphone` module take no input from the voter. For research and testing reasons, however, our current implementation allows the voter to choose whether to cast or audit (eChallengeResponse) or whether the left or right ballot half is opened (eCutAndChoose). As we describe in section 5, this has no impact on security results as long as the voter does not provide any other information to the `smartphone` module during the protocol. The application design assumes a touchscreen front-end and does not use any other type of input device (such as a hardware keyboard).

An essential part of the project is communication between `DRE` and `smartphone`. We connected the two using USB and we used the Android Debug Bridge, a tool contained in the Android SDK. Using adb we were able to initiate port forwarding on the host, meaning that all communication aimed at a certain port on the localhost is automatically forwarded to the smartphone through the USB connection. We then implemented sockets on top of this to achieve network communication between the two devices. Newer versions of Android, starting with 2.0, have added better support for networking over bluetooth so this could be a new approach in a future version.

In addition to standard libraries, we also used open source code from the Helios and Scantegrity codebases. To commit to a single bit, we use Pedersen bit-commitments, and to commit to larger messages we use Scantegrity commitments. We use the 1-2 oblivious transfer protocol of Even, Goldreich and Lempel [10].

8.1 Network Access

One of the major real-world threats for a voting system like ours is to be online. Voting machines can be hacked, and the leakage of cryptographic keys can lead to unauthorized ballot casting. On the other hand, if we choose to be offline we face two important issues. First, the assistant cannot check if a given ballot is fresh. Second, the assistant

is required to verify the opening of commitments against those published before the election.

A solution to the first problem is to pre-distribute ballots among voting machines. A voting machine that reuses a ballot is caught because it signs the serial number when it offers the ballot to the assistant. A solution to the second problem is for the assistant to maintain a root of a Merkle tree (hash tree) of the commitments so it can verify commitments offline. The voting machine is required not only to open audited commitments but also to show the hash path. This is preferable to storing all commitments locally on the assistant. It allows us to perform ballot casting offline at the cost of increasing the communication complexity between the voting machine and assistant by $O(logN)$ where $N$is the number of ballots.

8.2 Security of the Implemented Protocols

We describe common security concerns in implementations. First, there is the possibility of an attacker tapping the communication between the devices. This is not a concern for us because the transcripts between the two devices are assumed public. Second, there is the possibility of an attacker changing the communication. All messages between the two devices are hence digitally signed. Third, the smartphone is not an output-only device. However, in our implementation, the smartphone will have to be in a special docking station while the protocol is on. In the station, it is is protected by a transparent casing or sheet, so that the voter may obtain information from it, but not provide it any input. The voter can only provide input when it is not in the station, and if it is removed from the station the voting system aborts the protocol.

8.3 Ongoing Work

Scantegrity's back-end does not explicitly commit to the correspondence between candidates and confirmation codes; this correspondence may be deduced from other commitments made by the back-end. We are in the process of writing code to generate these commitments ourselves for use with eChallengeResponse. Since the system uses an electronic voting machine instead of a paper ballot, some sort of voter authorization mechanism should be used to ensure that each voter casts only one ballot, but is allowed to audit multiple ballots in eChallengeResponse. Such functionality has been discussed, but is not yet implemented—the voter would use a one-time password when she wished to cast a vote. An open-source release of the current version of our code will soon be complete.

9 Conclusions

We have described a paperless vote-casting protocol where the voter uses a computational assistant that participates in an interactive protocol with the voting machine. We propose the *Actively-Assisted Human Interactive Proof* model to study these types of protocols. We have rigorously described the AAHIP and demonstrated its security

properties, assuming that the voting system and assistant do not collude and that communication is synchronous. The assistant may be malicious, however, and provide false values or refuse to provide them. Its memory and logs may be examined by anyone after the vote is cast. We also show that a simple challenge-response-style AAHIP has weaker coercion-resistance. We have described our prototypes of both protocols, using a smartphone for the assistant.

Some interesting questions remain open. Is the cut-and-choose protocol secure against stronger adversarial models—for example, if communication is asynchronous? What is the most security one can obtain in the AAHIP model? Does the blind-signature-based protocol solve the problem of dispute resolution in paperless protocols?

References

1. ADIDA, B. *Advances in Cryptographic Voting Systems*. PhD thesis, MIT, 2006.
2. ADIDA, B. Helios: Web-based Open-Audit Voting. In *Usenix Security Symposium* (2008).
3. BENALOH, J. Simple verifiable elections. In *USENIX/Accurate Electronic Voting Technology Workshop* (2006).
4. BENALOH, J. Ballot casting assurance via voter-initiated poll station auditing. In *USENIX/Accurate Electronic Voting Technology Workshop* (2007).
5. BENALOH, J., AND TUINSTRA, D. Receipt-free secret-ballot elections. In *ACM Symposium on Theory of Computing* (1994).
6. CARBACK, R., CHAUM, D., CLARK, J., ESSEX, A., MAYBERRY, T., POPOVENIUC, S., RIVEST, R. L., SHEN, E., SHERMAN, A. T., AND VORA, P. L. Scantegrity II Municipal Election at Takoma Park: The First E2E Binding Governmental Election with Ballot Privacy. In *Usenix Security Symposium* (2010).
7. CHAUM, D. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy* (January/February 2004), 38–47.
8. CHAUM, D., CARBACK, R., CLARK, J., ESSEX, A., POPOVENIUC, S., RIVEST, R. L., RYAN, P. Y. A., SHEN, E., SHERMAN, A. T., AND VORA, P. L. Scantegrity: End-to-end verifiability for optical scan elections. *IEEE Transactions On Information Forensics And Security: Special Issue On Electronic Voting 4*, 4 (2009), 611–627.
9. CHAUM, D., POPOVENIUC, S., AND VORA, P. L. eTegrity and ePunchscan. NIST End-to-End Voting Systems Workshop, October 2009. `http://csrc.nist.gov/groups/ST/e2evoting/documents/papers/Popoveniuc_PaperlessVoting.pdf`.
10. EVEN, S., GOLDREICH, O., AND LEMPEL, A. A randomized protocol for signing contracts. *Communications of the ACM 28*, 6 (1985), 637–647.
11. JUELS, A., CATALANO, D., AND JAKOBSSON, M. Coercion-resistant electronic elections. In *Workshop on Privacy in the Electronic Society WPES* (2005).
12. KATZ, J., AND LINDELL, Y. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2008.
13. KELSEY, J., REGENSCHEID, A., MORAN, T., AND CHAUM, D. Attacking paper-based e2e voting systems. In *Towards Trustworthy Elections* (2010), pp. 370–387.
14. NEFF, C. A. Practical high certainty intent verification for encrypted votes, 2004.
15. POPOVENIUC, S., AND HOSP, B. An introduction to PunchScan. In *IAVoSS Workshop On Trustworthy Elections* (2006).
16. RYAN, P. Y. A. A variant of the Chaum voter-verifiable scheme. Tech. Rep. CS-TR: 864, School of Computing Science, Newcastle University, 2004.
17. SANDLER, D. R., DERR, K., AND WALLACH, D. S. VoteBox: a tamper-evident, verifiable electronic voting system. In *USENIX Security Symposium* (2008).