Journal of Information Assurance and Security 3 (2008) 270-278

Vote Verification using Hard AI Problems

Rahul Simha and Poorvi L. Vora
Department of Computer Science
The George Washington University
Washington, D. C. 20052, USA
Contact author: poorvi@gwu.edu.

Abstract: Recently proposed end-to-end independently-verifiable (E2E) voting schemes provide encrypted paper receipts to voters, who may later check that these receipts are in the electronic ballot box. Unfortunately, few voters are likely to follow up on the voting process after leaving the voting site; as a result, few receipts will be checked. This paper describes an enhancement to E2E schemes that does not require the voter to perform a task outside the polling booth. It enables the voter to electronically transmit her receipt, from the polling booth, to a trusted external verifier. This is done through the use of a human-verifiable digital signature primitive whose (short-lived) security depends on the hardness of an AI problem. The primitive enables the voter to be certain—without access to trusted computational power in the voting booth—that the receipt has been securely deposited with the external verifier. The approach presents several advantages: the voter is not required to do anything outside the polling booth, no receipts are needed after polling, all receipts generated by the polling machine can be checked, and any classical digital signatures on receipts can be checked instantaneously by the trusted verifier. Additionally, an audio-based format is an easy extension for those with visual disabilities. The cost of these benefits is the introduction of the verifier, who needs to be trusted not to launch a denial-of-service attack.

Keywords: vote verification, human-verifiable digital signatures, end-to-end independently-verifiable, hard AI problems

I. Introduction

The past few years have witnessed a number of end-to-end independently-verifiable (E2E) voting schemes (for example: [2, 6, 5]) that can convince a voter that (a) her vote was cast as intended, and (b) all votes were counted as cast. A unique aspect of these schemes is a paper *receipt* received by the voter that contains her vote in encrypted form. The voter may check that her encrypted receipt is in the public electronic bulletin board that forms the virtual ballot box, and anyone may thereafter check that the tally is correctly computed. Because the receipt is encrypted, it contains no information about the vote.

There are some drawbacks to the E2E schemes. First, the verifiability of these schemes requires voter participation outside
the polling booth: if voters choose not to check the presence of their receipts on the bulletin board, it is not possible to catch a cheating or defective polling machine. Second, some of the schemes use digital signatures as a means of authenticating the receipts; however the voter does not have access to a trusted computational device in the polling booth to check the signature. Finally, allowing a voter to walk out with a receipt connected to her vote, even though encrypted, and requiring that the voter follow up with the checking of the vote, even if helped by someone else, is distinct enough from the current voting process to pose a challenge to public acceptance and widespread use. This paper presents an enhancement to E2E schemes that addresses these problems; in particular, it does not require the voter to do anything outside the polling booth.

The enhancement presented in this paper is based on the secure electronic transmission of the receipt, from a *receipt machine* in the polling booth, to a third-party *verifier*. The verifier will immediately check classical digital signatures and commitments on the receipt, and, later, will check that the receipt is in the virtual ballot box. The enhancement uses a *human-verifiable digital signature primitive* that serves to authenticate the transaction between the voter and the verifier, without requiring that the voter have access to trusted computation. The transactions between the receipt machine and the verifier are also signed using classical digital signatures which are used to resolve any communication-related disputes. The introduction of a verifier requires continuous maintenance of secure connections between receipt machines and verifiers, not required in the typical E2E scheme. However, it enables the voter to leave the polling booth with no voting-related task left unfinished.

It may be noted that a verifier needs to be particularly reliable: a defective or malicious verifier can interfere with the process by sending back incorrect responses, thus holding up the process, or by refusing to respond, thus performing an out-and-out denial-of-service attack.

Received September 18, 2008

1554-1010 $ 03.50 Dynamic Publishers, Inc.

Thus verifiers need to be carefully chosen; however it should be possible to find a few parties with an interest in a fair election. It may also be noted that the electronic transmission of a receipt does not preclude the issuing of a paper receipt as well. If a county wishes to provide paper receipts, it may do so; the electronic receipts issued will all be checked, while it is likely that only a fraction of voters with paper receipts will make the effort to check their presence in the virtual ballot box. Further, voters should also be given the option of not sending the receipt to any verifier at all.

The (short-lived) security of the proposed human-verifiable primitive is based on the hardness of an AI problem, and it works as follows: the receipt received by the verifier is returned using a format and images agreed upon ahead of time by the voter and the verifier, and is easily and immediately validated by the voter with little effort. The shared information, as we will see, is both reasonably assumed to be known only to the verifier, and hard to reverse-engineer by the polling machine (without solving a hard AI problem). Hard problems in AI have been used as captchas, to successfully distinguish between bots and humans [8]; these hard problems typically involve the recognition of a string hidden in a noisy image or speech, and it is necessary that a human be able to determine the solution in a very short time. In this paper, the hard problem is the recognition of a font or presentation format. It is required that (a) a computer be able to compute the correct solution only with knowledge of a secret or the aid of a human, and (b) a human be able to verify a correct solution, if it is presented, without the aid of a computer. It is not required that a human be able to compute the correct solution without the aid of a computer. Further, it is expected that a human and a computer together would be able to solve the problem in some time, however, it is anticipated that this time is not short. While the paper that initially described this work [7] was independent of [3, 4], which use a similar approach to secure human user interfaces and to sign documents respectively, this paper uses some of the framework of [4].

This paper provides a formal description of the human-verifiable digital signature primitive, and describes its use with two well-known example E2E voting schemes: Punchscan [5] and Prêt à Voter [6]. It also proves integrity properties of the enhanced protocols. The paper is organized as follows. Section III provides descriptions of Punchscan and Prêt à Voter, Section IV provides an informal description of the approach, Section V contains formal statements of protocol properties with proofs, and concluding remarks are presented in Section VI.

II. Related Work

Several new E2E voting schemes allow the voter to encrypt her own vote by filling out specially-designed paper ballots [5, 6]. The voter then casts her encrypted ballot and can also take home a copy. All encrypted ballots are posted on a public website, which forms the virtual ballot box. Interested voters may check the presence of their encrypted ballots (also referred to as *receipts*)) in the virtual ballot box. Thereafter, votes are decrypted in a publicly verifiable manner such that the decryption process and the verification process do not reveal individual votes. These schemes hence allow voters to check the tally without requiring voters to trust the voting machines. However, voters are required to take the receipt out of the polling place and the responsibility of voting does not end at the polling booth. It would be easier if the voter could transmit the receipt to a trusted entity from the polling booth, but the polling machines are not trusted, and hence cannot authenticate the trusted entity.

We use a human-verifiable digital signature primitive, similar to one used in [3] for the purpose of document authentication. It is based on a hard AI problem, much like a *CAPTCHA*, which is also a security primitive whose hardness assumption is based on a problem in Artificial Intelligence [8]. A popular use of a CAPTCHA is to prevent bots from logging onto sites or accessing certain types of online services. In this application, a string of text is converted, by a program, into an image from which a human may recognize the text, but a program not knowing the text may not. Before being allowed to log in, a user is required to obtain the string from the image – an easy task for a human, but difficult for a bot. We do not use our secure primitive for the purposes CAPTCHAS have typically been used for. In the commonly used bot-defeating application, the CAPTCHA is used to encrypt a number so that any human can decrypt it, but no machine can, without solving a hard problem in AI in real time. Also, it is required that the hard problem in AI not be hard for humans. In this paper, however, the primitive is used to provide a secret-key digital signature that a human with possession of a visual representation of the secret key can verify, but that a computer not knowing the secret key cannot forge without solving a hard problem in AI in real time.

Keyed hard problems (KHAPs) were first described in [4], which describes the development of a human-verifiable authentication primitive similar to the one presented in this paper.

III. Preliminaries

In this section, we provide a brief overview of two of the most popular E2E schemes, Prêt à Voter and Punchscan. In later sections, we will describe our enhancements of these schemes.

A. The Prêt à Voting Scheme

The Prêt à Voter ballot consists of two halves printed on a single sheet of paper, side-by-side, separated by a perforation. The left half contains the names of the candidates, in a pseudo-random order. The right half contains spaces for marks against each candidate, as well as an "onion", whose function will be described later (see Figure 1). The voter

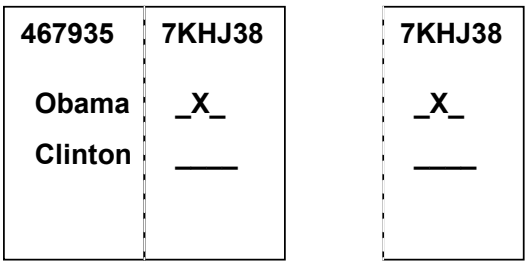

Figure. 1: Left: A Prêt à Voter Ballot; Right: A Prêt à Voter Receipt

marks the space next to the candidate of her choice—in Figure 1 her choice is the candidate Obama—tears the ballot along the perforation, and destroys the left half. The right half is scanned into the polling machine, and the information in it displayed in the virtual ballot box (the exact image is not shown in the virtual ballot box, to prevent the leakage of information through messages written on the ballot by the voter, for example). The right half is also the receipt. As the candidate order is unknown, the receipt reveals no information about the vote.

The information required to obtain the vote from the encrypted receipt is contained, in encrypted form, in the onion. (In Figure 1, the value of the onion is 7KHJ38.) That is, the onion contains information on the candidate ordering contained in the (destroyed) left half of the ballot. The entire virtual ballot box, containing all the receipts, is passed serially through a set of trusted parties. Each trusted party decrypts part of the onion on each vote, uses the information to perform an operation on the corresponding vote, leaves the rest of the onion with the processed vote, shuffles all the votes, and passes them on. The composition of all the operations performed by the trusted parties results in the decryption of the ballots. The decrypted ballots may be tallied by anyone. The input and output to each trusted party is made available on a public bulletin board.

The system also provides proof—to a group formed of the election authority, the candidates, and election observers—that it followed the encryption and decryption processes described above. We do not provide a description of the proof here; the interested reader may refer to [6, 1]. If at least some voters check that their receipts are in the virtual ballot box, a cheating trusted party or election system is caught with high probability. If the receipts are unforgeable, it is not possible to disrupt an election by falsely claiming that the election system is cheating.

B. The Punchscan Voting Scheme

The Punchscan ballot consists of two layers, one below the other. The upper layer contains a one-to-one map from the candidates to a set of dummy variables, such as letters of the alphabet. The lower layer contains another map, from the dummy variables to a position in a list – such as left and right (see Figure 2). A voter marks the position (and dummy variable) of the candidate of her choice. Because of a hole in the upper layer, the mark appears on both layers. Thus, both layers contain information on the vote, however, neither, by itself, provides information on the choice of candidate.

A voter chooses a single layer as the record of her vote. The other layer is destroyed. The single layer is scanned into the polling machine, and the information in it displayed in the virtual ballot box. It is also the voter's receipt.

The EA is able to decrypt the ballots as it (the EA) possesses the mappings from position to candidate for each serial number; the decrypted ballots are displayed on the public website. The original set of ballots is shuffled, and the serial numbers stripped, to preserve anonymity. As with Prêt à Voter, the decrypted ballots may be tallied by anyone, and the system provides a proof that the votes were correctly encrypted and decrypted; details of this may be found in [5].

The integrity of the casting stage of Punchscan or Prêt à Voter depends on (a) at least some voters requesting paper receipts and at least some of these voters checking them and (b) the unforgeability of the paper receipts. The casting stages of the enhanced versions presented in this paper—E-Punchscan and E-Prêt-à-Voter—on the other hand, make it possible to electronically check the presence of *all requested* receipts in the virtual ballot box without any voter follow-up. The integrity of E-Punchscan and E-Prêt-à-Voter depends on the unforgeability of regular digital signatures and of the human-verifiable digital signature primitive. The privacy of Punchscan and Prêt à Voter depends on the security of the encryption and commitment schemes used. Because E-Punchscan and E-Prêt-à-Voter do not reveal any more information in the receipt than revealed by the original schemes, the use of the human-verifiable digital signature primitive does not affect the privacy properties of the schemes. Finally, the enhancement—because it depends entirely on the receipt and does not affect any stage other than the casting of the vote and the verification of its presence in the virtual ballot box—does not affect the tallying stage or the audits.

IV. The Enhanced Protocols: A Sketch

In this section, we provide an informal description of the enhanced protocols. Section V provides a formal description of the protocol and the human-verifiable digital signature primitive, and also provides formal statement of properties.

We use the term Election Authority (EA) in the usual manner to mean the organization that oversees the polling, the voting machines, and the counting. Our enhancements of Punchscan and Prêt à Voter, which we term E-PunchScan and E-Prêt-à-Voter respectively, have the following additional requirements:

- *Verifier*. A *verifier* is an entity to whom an electronic version of the voter's receipt is sent from the polling booth. There will typically be several verifiers associated with an election. A voter using a verifier should

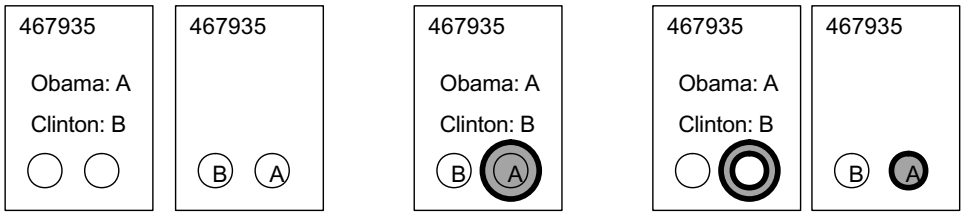

Figure. 2: A Punchscan Ballot. From left to right: upper layer of unmarked ballot; lower layer of unmarked ballot; a marked ballot for Candidate Obama with layers superimposed; upper layer of marked ballot; lower layer of marked ballot

trust the verifier to perform receipt checks on her behalf. The EA, the public and the individual voter should trust the verifier to be able to maintain a secure connection with the receipt machine, and to not wilfully hold up the process by sending false messages. (Though these will typically be detected, they would delay the voting process for an individual voter, resulting in an overall denial-of-service). Possible verifiers include candidate representatives, election observers, and public service organizations such as the League of Women Voters in the US.

- *Receipt machine*. A *receipt machine* is a machine in the polling booth or precinct used to send receipts. It is not used for polling. To enable communication with the verifier, our approach requires a receipt machine to be able to (a) display an image (and play audio for the visually-impaired), and (b) set up a secure connection with servers maintained by the verifiers.

A. The Human-Verifiable Digital Signature

In this section we provide an example of the use of the human-verifiable digital signature in the enhanced protocols. Consider the paper ticket in Figure 3. A voter picks up such a ticket from a shuffled box of such receipts maintained by the verifier of her choice, at the precinct, outside the polling booth. There will be several verifiers there. It depicts a ticket number shown in a particular captcha-style font; in the ticket shown in Figure 3, the number is 5417832. A voter with this ticket can assume that only the verifier knows the association between the font parameters (required to express any message using this font) and the ticket number. She may also assume that, while any human viewing the ticket may check that another message is in the same font, the font parameters are difficult to reverse-engineer from either the ticket or another message in the font. We request the reader to reserve judgement on the breakability of these particular examples – they are merely for illustration, and were not generated using captcha software.

Suppose the voter uses Prêt-à-Voter to vote as shown in Figure 1. The receipt is scanned into the receipt machine, and the voter enters the ticket number, 5417832 in this case. The information that would be stored in the the ballot box is sent to the verifier, with the ticket number; in this case, it

Figure. 3: Human-Verifiable Signature Ticket

Figure. 4: E-Prêt-à-Voter Returned Receipt

is that the first choice was marked, and that the onion value is 7KHJ38. The verifier then returns the image displayed in Figure 4. One can see that it contains all the information contained in the receipt.

Suppose the voter uses Punchscan to vote as shown in Figure 2. Suppose the voter chooses the top layer. It is scanned into the receipt machine and the information sent to the verifier by the receipt machine, which returns the image shown in Figure 5. One can see that the image is a replica of Punchscan's top layer: it contains the ticket number, the mapping from candidates to dummy variables, and the position of the encircled vote. Likewise, if the voter instead chose to keep the bottom layer, the image displayed in Figure 6 is returned by the verifier, showing the ticket number and the selection made.

It is assumed that a human can tell with high probability if two images consist of messages using the same font/format, and that a human can read both messages. It is also assumed that, given a message in a particular font/format, the probability that the adversary can construct, in real time, an-

Figure. 5: E-Punchscan Returned Receipt: Top Layer

Figure. 6: E-Punchscan Returned Receipt: Bottom Layer

other message that appears to a human as being in the same font/format, is small. Section V provides more formal statements.

B. The Protocol

For the following, the words *digital signature*, when not preceded by the term *human-verifiable*, represent the classical digital signature.

3.2.1 The Enhanced Protocols: The Main Additional Features

We now briefly describe the main additions, to the E2E schemes, that enable the secure communication of the receipt to the verifier.

Verifier Signs Receipt with Human-Verifiable Signature: The voter scans her receipt into the untrusted receipt machine, which sends a discrete electronic representation of it to the verifier. That is, the receipt machine does not send an image or audio signal, but simply the information contained in the receipt – the choice "right" in Punchscan, for example, or "first choice" and "7KHJ38" for Prêt à Voter. The verifier responds by sending the receipt back, signed using a human-verifiable signature. Thus the human can determine that the correct receipt was transmitted to the correct entity.

Receipt Machine and Verifier Communicate Using Classical Digital Signatures: The communication between the receipt machine and the verifier – who is assumed to always have access to trusted computational power – is signed using classical digital signatures. Hence any dispute between the receipt machine and the verifier, who acts on behalf of the voter, can be settled using classical digital signatures.

Timeliness and Challenges can be Added: The protocol can be enhanced through the appropriate use of challenges and nonces.

Simple Key Establishment: The human-verifiable digitally signed message is simply a representation of the original message as a visual signal, using a particular format. The format is the secret key of the signer (the verifier). An important feature of the human-verifiable digital signature is that a human can tell correctly whether two distinct messages have been human-verifiably signed with the same key. Because of this property, key establishment is straightforward:
– The human is provided with a ticket which consists of a number printed using a particular font and format (see Figure 3). This can be done, for example, by dividing the polling site into a polling area and a verifier area, where all verifiers set up booths providing the paper tickets, which are randomly picked by the voters.
– The number on the ticket, 5417832 in Figure 3, identifies the format to the trusted entity, so the voter provides it when she transmits her receipt using the receipt machine. Each verifier does not repeat ticket numbers or fonts/formats, so each number is used only once.
– When she gets back her human-verifiable signed receipt, the voter can tell if it has been human-verifiably signed with the same font/format.

Post-poll checking and counting: Each trusted party checks that each receipt is on the poll website. Any discrepancies are resolved through the checking of digital signatures. Vote tallying and post-counting audits proceed according to the original Prêt à Voter/Punchscan protocols.

3.2.2 The Enhanced Protocol

Our protocol, described in general for both E-Punchscan and E-Prêt-à-Voter, proceeds as follows:

Step 1: Prior to election.

- The EA posts information about candidates and verifiers, polling sites and the election schedule.
- The receipt machines are programmed to open secure connections to verifiers.
- Each verifier creates and maintains a secret *injective* mapping g between a large set $\mathcal{V}$ of verification numbers and a set $\mathcal{F}$ of internally-generated formats and image sets that the verifier will use. For simplicity, we refer to $g(v)$, $v \in \mathcal{V}$, as a *format*.

- The polling site is divided into two sections – the verifier area, and the voting area.
- Each verifier contributes several tickets, each ticket corresponding to a single value $v \in \mathcal{V}$. Each ticket contains printed on it the value v and sufficient information for a human to recognize a receipt image in format $g(v)$.
- The tickets are loosely placed in a box as would raffle tickets prior to a drawing; the tickets for each verifier are placed in separate boxes. The tickets are in sealed envelopes so that a voter may not choose v or $g(v)$.

Step 2: The voting procedure.

- A voter enters the polling site where the verifiers are located and draws a ticket from the ticket box of any one verifier *of her choice*.
- The voter is given a paper ballot in much the same way as with the original protocols, and directed to a voting booth where she will cast her vote.
- The voter makes her selections. In the case of Punchscan, she chooses her receipt layer. In the case of Prêt à Voter, she tears off the left half and destroys it. To cast her ballot, the voter scans in the receipt into the polling machine. If she wishes to send it to the independent verifier, she then scans it into a separate receipt machine.
- The receipt machine presents a textfield where the voter can enter her ticket number v. A function (not necessarily one-way) of the ticket number identifies the verifier to the polling machine.
- The voter enters the ticket number present on her ticket.
- The receipt machine then digitally signs the receipt and sends it to the associated verifier using the secure connection.
- The verifier server checks the signature of the polling machine on the receipt. It then constructs a *composite image* of the receipt using the format $g(v)$, and transmits that back to the voting machine. The server also digitally signs the composite image.
- The machine displays the received image to the voter, along with an option to "confirm" the receipt transmission.
- The voter sees her receipt in the image returned *in the corresponding format*, $g(v)$, and ends the voting process.

Note that a disgruntled verifier could hold up this protocol by sending an incorrect composite image. A disagreement of this kind can be resolved on-the-spot through human viewing of the ticket, receipt and composite image, and the checking of digital signatures. An uncooperative verifier could also hold up this protocol by simply refusing to respond; this is the price of inserting a new entity into the protocol, as all entities can carry out denial-of-service attacks. Note also that, either the verifier can be trusted to not provide two receipts with the same value of v (else the machine can learn the value of $g(v)$), or, if it cannot, the only way it can cheat is through the machine. In the latter case, it does not need to create multiple tickets with the same value of v. The machine can submit an incorrect receipt to the virtual ballot box, and the verifier can simply refrain from pointing out any errors in the corresponding receipt.

Step 3: Post-poll checking and counting.

- Each verifier checks that each receipt is on the poll website. Any discrepancies are resolved through the checking of digital signatures.
- Vote tallying and post-counting audits proceed according to the original scheme.

V. Formal Definitions, Descriptions and Properties

In this section, we state more formally the protocol and our assumptions, and prove properties of the enhanced protocol. In this description, Alice is the voter, Vera the verifier, and M the receipt machine.

A. Formal Definition: Human-Verifiable Digital Signature

Let $\mathcal{R}$ represent the set of all numbers that can be signed; the discrete representation of a receipt belongs to $\mathcal{R}$, as do all ticket numbers. Let $\mathfrak{R}$ be the set of all human-verifiably signed messages returned by the verifier, and $\mathfrak{F}$ the set of all possible formats. Let $D_{\mathcal{R}}$ and $D_{\mathfrak{F}}$ be the probability distributions on $\mathcal{R}$ and $\mathfrak{F}$ respectively, and assume that receipts and fonts are independent, much as, in cryptography, messages and keys are chosen independently. If $\mathcal{A}$ and $\mathcal{B}$ are sets with probability distributions $D_{\mathcal{A}}$ and $D_{\mathcal{B}}$, let $D_{\mathcal{A},\mathcal{B}}$ denote the probability distribution induced on $\mathcal{A} \times \mathcal{B}$ when elements are chosen independently from $\mathcal{A}$ and $\mathcal{B}$. Consider the signing function

$$\rho : \mathcal{R} \times \mathfrak{F} \rightarrow \mathfrak{R}$$

which uses a key (format) from $\mathfrak{F}$ to sign a message (receipt) from $\mathcal{R}$ to obtain a signed message (composite image) in $\mathfrak{R}$. Informally, ρ is human-verifiable if a human can verify whether two distinct messages were signed with the same key, and if a human can identify the message signed. It is a digital signature if it is difficult to forge a signed message.

First, we define what it means for ρ to be human-verifiable. In order to do so, we first define a correct response to the question of whether two messages are in the same font/format, and whether one of them represents a particular string. Let

$$H : \mathfrak{R} \times \mathfrak{R} \times \mathcal{R} \rightarrow \{Y, N\}$$

represent the human response (Yes or No) to whether two signed messages from $\mathfrak{R}$ (for example, the ticket and the composite image returned by the verifier) were signed with the same key, and whether the second corresponds to a particular receipt from $\mathcal{R}$. The human response of Yes or No is *correct* if, loosely speaking, it either (a) correctly identifies that the two formats are identical and that the second composite image represents the receipt, (that is, it is a correct Yes) or (b) correctly identifies that the formats are not identical, or that the second composite image does not represent the receipt (or both), that is, it is a correct No.

Definition [CORRECTNESS] A value $H(\rho(r,f),\rho(r',f'),x)$ is said to be *correct* if it is Y and $f = f'$ and $r' = x$, or it is N and either $f \neq f'$ or $r' \neq x$.

We now define as (α,β)-human-verifiable a function ρ that is correctly verified with probability β by a fraction α of humans.

Definition [HUMAN-VERIFIABILITY] ρ is (α,β)-*human-verifiable* iff, for a fraction α of humans,

$$\Pr_{r,f\leftarrow D_{\mathcal{R},\mathfrak{F}}\ v,f'\leftarrow D_{\mathcal{R},\mathfrak{F}}\ x\leftarrow D_{\mathcal{R}}}[H(\rho(r,f),\rho(v,f'),x)\ is\ correct] = \beta$$

Note that the above definition is exactly the application of the definition of a *human-executable test* [8] to our problem. Now, we turn to the security definitions. It is very likely that, given enough time and enough instances of a font, the receipt machine can learn to detect the font of a message and use this knowledge to forge the message. Hence, we recommend that fonts not be reused during an election. Consider a voter sending receipt r to the verifier, using a receipt machine. Suppose the font on her ticket is f. The main task of a cheating receipt machine is to send a different receipt r' to the verifier, and to provide the voter with a human-verifiably-signed version of her receipt, r, using font f. On receiving receipt r', the verifier would return $\rho(r',f)$. To prevent the voter from detecting the fact that the verifier has obtained a different receipt, the machine would have to create $\rho(r,f)$, assuming that f has not been used before. Thus the hard AI problem governing the security of our protocol is that of imitating the font f on a different message, r, after having previously seen exactly one instance of it on a known message, r'. We define this as the solution of a hard AI problem, exactly as in [8].

Definition [FORGERY] A program A is a (δ,τ) forgery if it runs in time at most τ, and,

$$\Pr_{f\leftarrow\mathfrak{F},\ v,x,x'\leftarrow\mathcal{R}}[H(\rho(v,f),A(\rho(x',f)),x) = Y] \geq \delta$$

Finally, we define one-use secure digital signatures.

Definition [ONE-USE SECURE] ρ is (δ,τ)-*one use secure* if there does not exist a (δ,τ) forgery.

B. Formal Protocol Description

Let $Sign(m,k)$ be the classical digital signature of m with public key k, and $Verify(m,s,k)$ the verification of signature s on message m using key k. Let k_V be the verifier Vera's public key, and k_R that of R, the receipt machine. The protocol is as follows.

1. The voter Alice picks up a ticket, $\mathfrak{t}$, such that $\mathfrak{t} = \rho(v,f)$ for some $v \in \mathcal{R}$ and $f \in \mathfrak{F}$.
2. Alice votes and obtains a receipt. She scans in her receipt $r \in \mathcal{R}$ into the receipt machine, R, and enters v.
3. R sends to Vera $(r; v, Sign(r; v, k_R))$ where ; denotes concatenation.
4. Vera receives (m_1, s_1) where $m_1 = r_1, v_1$.
   If $Verify(m_1, s_1, k_R) = YES$, Vera sends back $(\mathfrak{r}, Sign(\mathfrak{r}, k_V))$, where $\mathfrak{r} = \rho(r_1, g(v_1))$, after checking that she has not used $g(v_1)$ to sign a previous message. If $Verify(m_1, s_1, k_R) = NO$, Vera sends back $(e_1, Sign(e_1, k_V))$, where e_1 is an error message.
5. R receives (m_2, s_2). If $Verify(m_2, s_2, k_V) = NO$, or m_2 is error message e_1, R either requests a resend (for the former) or resends $(r; v, Sign(r; v, k_R))$ (in case of the latter). R requests a resend, or resends, a message only a fixed number of times, that is predetermined. If R is not able to obtain a valid receipt without an error message, R displays error message e_2 to Alice. If $Verify(m_2, s_2, k_V) = YES$ and m_2 is not error message e_1, R displays m_2 to Alice.
6. Alice accepts and ends the protocol if the displayed message appears to her to be of the same font as $\mathfrak{t}$, and if it represents receipt r.

   If the displayed message is e_2, Alice knows that the transaction was in error, and that the receipt was not sent to the verifier. She can request an examination of the transaction logs of R and her verifier at day's end to determine if her verifier or R was the cause of the error. She can also obtain a ticket from another verifier to redo the process of sending the receipt, or she can check the receipt herself as is done by voters in the E2E schemes that do not use the human-verifiable signature primitive.

Dispute Resolution

1. If a voter obtains a human-verifiable signed receipt that she does not think matches her vote receipt and/or ticket, she shows it to the verifier representatives (of

several verifiers) in the polling area. Note that, because her vote is encrypted, the receipt cannot leak any information on her vote. If ρ is (α, β)-*human-verifiable*, and α and β are large enough, this can be resolved by a majority of the representatives. *Thus it may be assumed that, after dispute resolution, it is always clear whether a particular human-verifiably-signed receipt is correctly signed or not*. If the signed receipt is, indeed, incorrect, the receipt machine is malfunctioning (see property below).

2. If Alice revotes, or sends another receipt, she should use another ticket/font.

We assume that each ticket number v is used at most once. The following properties hold.

Property 1 [CORRECTNESS] If ρ is (α, β)-human-verifiable, the protocol is correctly executed, no error messages are received, and the number of voters is large enough, the following are true with high probability

1. At least fraction $\alpha\beta$ of the voters will accept and end the protocol.

2. After dispute resolution, all voters will accept their human-verifiably-signed receipts as correct.

Proof: If the protocol is correctly executed, $m_2 = \mathfrak{r} = \rho(r, f)$ where $\mathfrak{t} = \rho(v, f)$. $H(\mathfrak{t}, m_2, r) = H(\rho(v, f), \rho(r, f), r)$. Thus $H(\mathfrak{t}, m_2, r)$ is correct if it is Y. As ρ is (α, β)-human-verifiable, a fraction α of voters provide the correct answer with probability at least β. Voters with the correct answer are exactly those who accept and end the protocol, and, if the number of voters is large enough, the fraction of those accepting and ending the protocol is $\alpha\beta$ with high probability. Further, for those not accepting and ending the protocol, because the receipts are correct, this will be determined after dispute resolution with several humans examining the receipt.

Property 2 [SECURE DELIVERY] If ρ is (α, β)-human-verifiable and (δ, τ)-one-use-secure, and if R has time τ and cheats on n receipts, the probability that none of these will be detected is $(\delta + (1 - \delta)(1 - \alpha\beta))^n$. If $\delta << 1$, and $\alpha\beta \approx 1$, a cheating R will be detected with high probability for large enough n.
Proof: An incorrect receipt sent to the verifier is not detected if, either, a program is successful in changing the receipt obtained from the verifier to a correct one, or, an incorrect one is presented to the voter who does not detect this. Hence, this probability is $\delta + (1 - \delta)(1 - \alpha\beta)$. The probability that none of n such receipts are detected is $(\delta + (1 - \delta)(1 - \alpha\beta))^n$. As $\delta + (1 - \delta)(1 - \alpha\beta) < \delta + 1 - \alpha\beta << 1$, a cheating R will be detected with high probability for a large enough n.

Property 3 [NON-REPUDIATION] If the classical digital signature scheme used is secure, Ted cannot later deny that it sent $\mathfrak{r}$.
Proof: Follows from the properties of the classical digital signature schemes.

VI. Conclusions and Future Work

The use of hard AI problems in voting is promising because these problems have been widely-used to provide security in other applications involving human-machine interaction. A promising avenue for future work is the incorporation of several different types of problems—such as those based on the sense of hearing—for ease of use for those with visual disabilities. For example, audio-based captchas have been used to increase accessibility. An audio-based digital signature primitive might work as follows. The ticket consists of an MP3 file identifying to the voter a particularly stylized voice (for example, deep female voice with a strong accent). The verifier then returns a description of the ballot-portion in that voice. Thus, the difficulty for the machine is to create a fake vote out of that voice. Because a multitude of background voices and noise can be used, and the main voice does not repeat, the audio snippet cannot be spliced out of previous votes.

Acknowledgments
An anonymous referee at WOTE 2007 suggested that our use of the primitive was as a human-verifiable digital signature. This work was partially supported by NSF Grant ITR-0325207 and NSF Grant SGER-0505510.

References

[1] D. Chaum, P. Y. A. Ryan, and S. A. Schneider. A practical, voter-verifiable election scheme. Technical Report CS-TR: 880, School of Computing Science, Newcastle University, 2004.

[2] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, pages 38–47, January/February 2004.

[3] Igor Fischer and Thorsten Herfet. Visual CAPTCHAs for document authentication. In *Proceedings, International Workshop on Multimedia Signal Processing*, October 2006.

[4] Jeff King, Andre dos Santos, and Chaoting Xuan. KHAP: Using keyed hard AI problems to secure human interfaces. *Scientia*, 15(1):31–40, 2004.

[5] Stefan Popoveniuc and Ben Hosp. An introduction to Punchscan. In *IAVoSS Workshop On Trustworthy Elections (WOTE 2006)*, 2006.

[6] P. Y. A. Ryan. A variant of the Chaum voter-verifiable scheme. Technical Report CS-TR: 864, School of Computing Science, Newcastle University, 2004.

[7] Rahul Simha and Poorvi L. Vora. Vote verification using CAPTCHA-like primitives. In *IAVoSS Workshop On Trustworthy Elections (WOTE 2007)*, 2007.

[8] Luis von Ahn, Manuel Blum, Nick Hopper, and John Langford. CAPTCHA: Using hard AI problems for security. In *Eurocrypt*, 2003.

Author Biographies

Rahul Simha is Professor of Computer Science at The George Washington University. He received his PhD in Computer Science from the University of Massachusetts, Amherst, in 1990. His research interests include embedded systems, compilers, languages, architecture, security and complex systems.

Poorvi L. Vora is Assistant Professor of Computer Science at The George Washington University. She received the B. Tech degree in electrical and electronics engineering from the Indian Institute of Technology, Bombay, India, in 1986, the M. S. and Ph. D degrees in electrical engineering from North Carolina State University, Raleigh, NC, USA, in 1988 and 1993 respectively, and the M. S. degree in mathematics from Cornell University in 1990. Her areas of interest are electronic voting and cryptology.