# Vote Verification using CAPTCHA-like Primitives

Rahul Simha and Poorvi L. Vora*
Department of computer Science
The George Washington University
Washington, D. C.

June 15, 2007

**Abstract**

Recently proposed voter-verifiable protocols provide encrypted paper receipts to voters, who may later check that these receipts are in the electronic ballot box. This paper describes an enhancement that allows the voter to electronically transmit, from the polling booth, her encrypted receipt to an external verifier, who may perform the check on her behalf. It uses a CAPTCHA-like primitive – whose security depends on the hardness of an AI problem – as a humanly-recognizable digital signature, to enable the voter to be certain that the receipt has been securely deposited with the external verifier. This approach presents several advantages: the voter is not required to do anything outside the polling booth, no receipts are needed after polling, and all receipts generated by the polling machine can be checked. Additionally, an audio-based format is an easy extension for those with visual disabilities, and it is anticipated that a public already familiar with CAPTCHAs will find the approach easy to use.

## 1 Introduction

The past few years have witnessed a number of voter-verifiable voting techniques (for example: [3, 5, 6, 2, 7]) that can convince a voter that (a) her vote was cast as intended, and (b) all votes were counted as cast. These techniques provide a level of integrity and verifiability not present in previous techniques, because they do not require the voter to trust any entity at the polling place – polling machine, election official, or third party. A unique aspect of these protocols 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 ballot box. This receipt is either encrypted [3, 6, 2] or incomplete [7] and therefore maintains vote privacy.

Unfortunately, this idea of letting a voter take her receipt out of the polling booth for individual verification has some drawbacks. First, the verifiability of these schemes requires voter participation: 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, in some of the published schemes, voters are first presented with complete ballots but are asked to leave with only a portion of the ballot for verification, in order to prevent them from being able to prove how they voted. However, it may be physically difficult to force voters to leave with only part of the ballot as instructed. Third, the receipts are themselves susceptible to abuse from malicious voters: even if a forged receipt can be identified, the labor or legal costs of handling false claims can be prohibitive. Finally, the act of allowing a voter to walk out with a receipt connected to her vote, even

*Contact author: poorvi@gwu.edu. This work was partially supported by NSF Grant ITR-0325207 and NSF Grant SGRE-0505510

though encrypted, and the requirement 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 explores the use of an additional CAPTCHA-like primitive to securely (and electronically) transmit the receipt to one or more third-party *verifiers*, who check for the presence of the receipt on the public bulletin board. The primitive serves as a humanly-verifiable digital signature[1] – 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. This 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). The use of this primitive addresses the problems pointed out above with receipts, and in addition, allows on-the-spot detection of an improper electronic receipt. Note 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 on the public electronic bulletin board. Further, voters should also be given the option of not sending the receipt to any verifier at all.

While several schemes have recently contributed to simplifying the user interface (ballot format) for voter-verifiability [6, 8], we are, however, not aware of any work that attempts to transmit the receipt electronically to a verifier, or uses hard problems in AI as the basis of the security mechanism. Although the ideas presented here can be applied to a variety of voting protocols, this paper, for ease of exposition, explores the use of this primitive with two well-known protocols: Punchscan [2] and ThreeBallot [7]. The use of this primitive is not without some weaknesses. First, it requires continuous maintenance of secure connections between polling machines and verifiers. Second, a defective or malicious verifier can interfere with voting by sending back incorrect responses.

This paper is organized as follows. Our approach is described in Section 2. Section 3 contains formal statements of protocol properties, and concluding remarks are presented in Section 4.

## 2 How it Works

Before we describe our enhancements, we provide an overview of PunchScan, ThreeBallot, and CAPTCHAs. 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 CAPTCHA-based enhancements of Punchscan and ThreeBallot, which we term C-PunchScan and C-ThreeBallot 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. In C-Punchscan, a single receipt is sent to a single verifier who might be chosen by the voter or at random by the machine. In C-ThreeBallot, the three ballots are sent to three different verifiers, randomly assigned by the machine.

- *Polling machine*. To enable communication with the verifier, our approach requires a polling 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.

[1] Our thanks to an anonymous referee for suggesting that our use of the primitive was as a humanly-verifiable digital signature

## 2.1 Overview of Punchscan

We describe Punchscan for the simple case of two candidates. 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 1). 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.

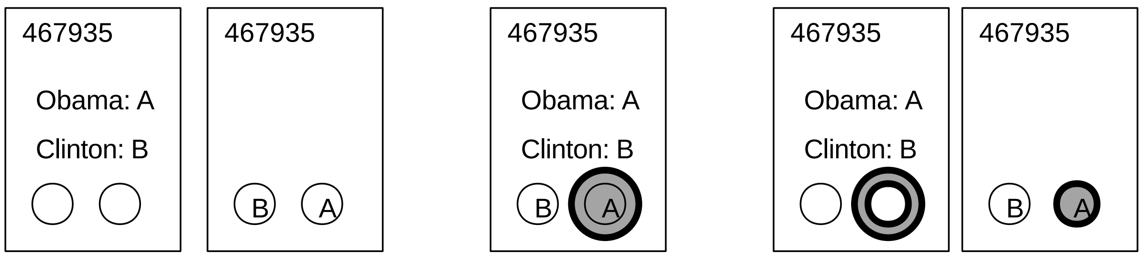


Figure 1: A Punchscan Ballot. From left to right: upper layer of unmarked ballot; lower layer of unmarked ballot; a marked ballot for Candidate Obama with layers superimposed; upper layer of marked ballot; lower layer of marked ballot

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 displayed on a public bulletin board. It is also the voter's receipt. The EA is able to decrypt the ballots as it 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.

The printed ballots are audited for correctness through the opening of the mappings (stored and committed to) for each of half of the total number of ballots. These audited ballots are treated as spoiled and are not used for the election. The decryption process is audited through the use of a process similar to randomized partial audits of mixes. These details are described in [6].

The integrity of the casting stage of Punchscan 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 stage of C-Punchscan, on the other hand, makes it possible to electronically check the presence of *all requested* receipts on the bulletin board without any voter follow-up, and sidesteps the issue of unforgeability because no receipts are given to voters. Instead, its integrity depends on the unforgeability of regular digital signatures and on the security of the humanly-verifiable digital signature primitive. The privacy of Punchscan depends on the security of the encryption and commitment schemes used. So does the privacy of C-Punchscan. That is, the use of the humanly-verifiable digital signature primitive does not affect the privacy properties of Punchscan.

## 2.2 An Overview of ThreeBallot

The ThreeBallot ballot consists of the list of candidates, arranged one below the other in a fixed, pre-defined order, and three columns next to the candidates. To choose a candidate, the voter marks two of the three columns corresponding to the candidate. For all other candidates, the voter marks exactly one column. The three columns are separated out and cast separately, each as a ballot. Each ballot has an associated serial number, though the serial numbers are independent. The voter scans in all three and takes exactly one home with her.

All ballots are posted online with the corresponding serial numbers. Each voter checks if the ballot she took away is on the bulletin board. She does not know if the other two ballots are there too (and unchanged), but because the EA cannot guess which piece she took home with her, it will be caught with high probability if it changes even a few ballots. Anyone can tally the votes – the winner will be the candidate with the most marks. The number of votes obtained by each candidate is the number of marks less the total number of voters. The integrity of this scheme depends on the scanner and the EA not being able to anticipate which ballot will be kept by the voter.

We demonstrate the challenges in retaining the integrity and privacy of ThreeBallot. The integrity of the original ThreeBallot scheme depends on the voting machine not knowing the voter's choice of receipt from among the three ballots cast by each voter; however this condition cannot be satisfied when the polling machine sends the receipt to the verifier. Hence, C-ThreeBallot requires that the machine send all three ballots, each to a verifier. To preserve the involuntary privacy of voting, the voter may not choose the verifiers, as their collusion will reveal the vote. The collusion of a single verifier with the voting machine can change the vote, hence the integrity of ThreeBallot is considerably weakened, and it is not as well-suited to our approach as is Punchscan. Further, additional cryptographic checks are required to ensure that the verifiers are randomly chosen by the machine, defeating one original purpose of ThreeBallot – to obtain a voting scheme that did not use cryptography.

### 2.3 An Overview of CAPTCHAs

In the general Human Interactive Proof (HIP) problem, the goal is to distinguish between a human and machine using a simple test. By focusing on exceptional cognitive abilities such as visual perception, a HIP tries to place a high barrier to machine duplication of human ability. A *CAPTCHA* is an application of this notion to security problems: it is a security primitive whose hardness assumption is based on a problem in Artificial Intelligence [9]. The AI problems are somewhat distinct from the usual hard problems used in cryptographic schemes (problems in algorithmic number theory) because standard approaches to breaking them require the use of a large data set to learn from. The applications of CAPTCHAs have also hence been different – they have typically been used in situations where the security depends on the machine not solving the hard AI problem in real time. Thus, while a typical cryptographic scheme is required to be unbreakable into the future, it is usually enough if a CAPTCHA cannot be broken in a few minutes.

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. This problem can be made quite difficult by incorporating not simply visual recognition, but also face and theme recognition, common historical knowledge (for example, identities of presidents), or emotions (happy vs. sad) in the images. Further, one may similarly use audio-based CAPTCHAs that exploit human abilities to recognize speakers and intonations in a way that has been out of reach for machines.

We do not use our secure primitive for the purposes CAPTCHAS have typically been used for, which is why we do not refer to it as a CAPTCHA (even though its security is also based on a hard problem in AI). 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. *We also assume that the hard AI problem is not solved by a human in real time either*. A similar primitive is used in [4] for the purpose of document authentication.

In our approach, we require the verifier to generate a composite image representing the voter's receipt, using a specific format and set of images (the private key) known only to the voter and the verifier. We require that the computer providing the composite image to the voter not be able to determine, in a short time period, the format and set of images. Hence, it would not be possible for a computer to change the receipt received from the verifier – that is, it would not be possible, given the signature on one receipt, to forge it onto another.

## 2.4 The Enhanced Protocol: A Sketch

Our protocol, described in general for both C-Punchscan and C-ThreeBallot, proceeds as follows (note that, in the protocol description, *digital signature* and its derivatives refer to the classical digital signature technique):

**Step 1: Prior to election**.

- The EA posts information about candidates and verifiers, polling sites and the election schedule.
- The voting 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 (large) random verification numbers and a set $\mathcal{F}$ of internally-generated formats and image sets that the trustee 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. For Punchscan, the tickets for each verifier are placed in separate boxes. For ThreeBallot, tickets of all verifiers are placed in a single well-shuffled box. 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* for Punchscan, and three tickets from the single box for ThreeBallot.
- The voter is given a paper ballot in much the same way as with the original PunchScan or ThreeBallot protocols, and directed to a voting booth where she will cast her vote.
- The voter makes her selections and scans in her ballot.
- The machine presents a summary ballot containing the two layers (for Punchscan) or the three ballots (for ThreeBallot). Also presented is a textfield where the voter can enter her ticket number(s) $v$. A function (not necessarily one-way) of the ticket number(s) identifies the verifier(s) to the polling machine.

- The voter enters the ticket number(s) present on her ticket(s). For Punchscan, she also chooses a layer.
- The machine then sends the digitally signed chosen layer (for PunchScan) or all three ballots (for ThreeBallot) to the associated verifier(s) using the secure connection(s).
- The verifier server(s) 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(s) to the voter, along with an option to "confirm" or finalize the vote.
- The voter sees her summary layer in the image returned (for Punchscan), or the three ballots in the three images returned (for ThreeBallot), *in the corresponding format(s)*, $g(v)$, and confirms the vote. Note that, for ThreeBallot, she needs check only one of the three pieces at random, as the polling machine would not know which one she would check.

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. 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[2] of $v$.

**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.

We note that, as with the original protocols, a voter may waive the option to verify her vote, in which case she would choose not to pick up a ticket. A voter may also choose to take a paper receipt from the original protocol, and not send it to a verifier – that is, a voter may choose to stay with Punchscan and not participate in C-Punchscan, for example. As with the original versions of Punchscan or ThreeBallot, if even a small number of concerned voters engage in using verifiers, the probability of a cheating polling machine being caught is very high.

**The CAPTCHAs Used**

In this section we show some sample CAPTCHAs for C-Punchscan and C-ThreeBallot. We request the reader to reserve judgement on the breakability of these particular CAPTCHAs – they are merely for illustration. Far harder CAPTCHAs can be designed using the power of human vision and cognition.

We first show how C-Punchscan can mimic the use of Punchscan. Consider the ticket in Figure 2. It depicts a ticket number shown in a particular CAPTCHA-style font. A voter with this ticket can assume that only she and the verifier know that font, and that the font is difficult to reverse-engineer.

If, in the manner of Punchscan, the voter chooses the top-layer, the top-layer is sent in plain text to the verifier by the machine. The verifier then returns the image displayed in Figure 3. 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, the image displayed in Figure 4 is returned by the verifier, showing the ticket number and the selection made.


Figure 2: C-Punchscan Ticket

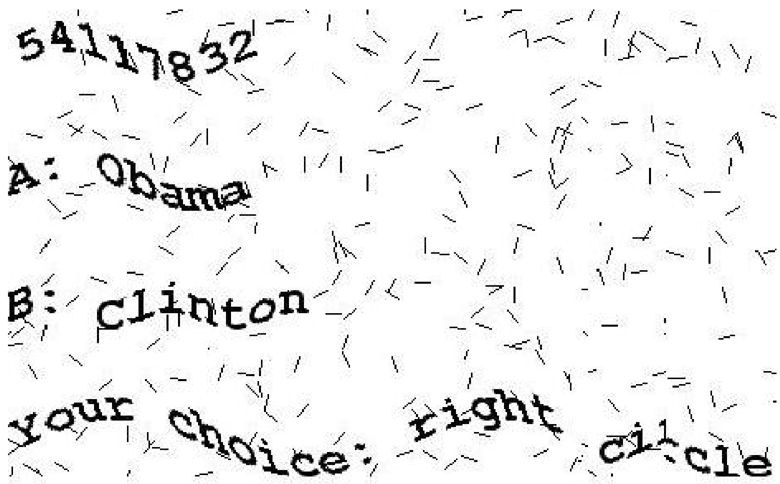


Figure 3: C-Punchscan Composite Image Returned: Top Layer

The particular CAPTCHA technique above uses distorted fonts, and is the most widely used CAPTCHA for login applications. For illustration, our example for C-ThreeBallot shows how pictures can be used. Figure 5 shows one of the three tickets on the left. The ticket contains a description of a visual theme, in this case an "outdoors" picture of a candidate implies a mark in the column corresponding to that candidate. On the right, are two pictures forming the composite image returned by the verifier. It shows that the ballot the verifier received was a mark for Obama (the other image is not outdoors). *Note that the serial number of the receipt may be displayed using distorted fonts as well.*

# 3 Formal Statements

In this section, we state more formally our assumptions and properties of C-Punchscan and C-ThreeBallot that we believe to be true. For some properties, we provide proof sketches.

Let $\mathcal{R}$ represent the set of all possible receipts (sent to the verifier), and $r \in \mathcal{R}$ a single receipt. Let $\mathfrak{R}$ be the set of all composite images (returned by the verifier). Let $\rho(r, g(v)) \in \mathfrak{R}$ represent the composite image corresponding to receipt $r$ in format $g(v)$.

We now define the security primitive. First, the primitive must be checkable by a human. That is, given a ticket numbered $v$, describing the format $g(v)$, a human must be able to recognize composite image $\rho(r, g(v))$, as being $r$ in format $g(v)$ for all possible values of $r$.

**Assumption 1**[HUMAN CHECKABILITY] The mapping $g : \mathcal{V} \rightarrow \mathcal{F}$ is *humanly checkable*. That is, $\forall v \in \mathcal{V}$, $\forall r \in \mathcal{R}$, $\rho(r, g(v))$ is read as being the receipt $r$, printed in format $g(v)$, by a human with ticket numbered $v$.

[2]Thanks to an anonymous referee for describing this problem.


Figure 4: C-Punchscan Composite Image Returned: Bottom Layer

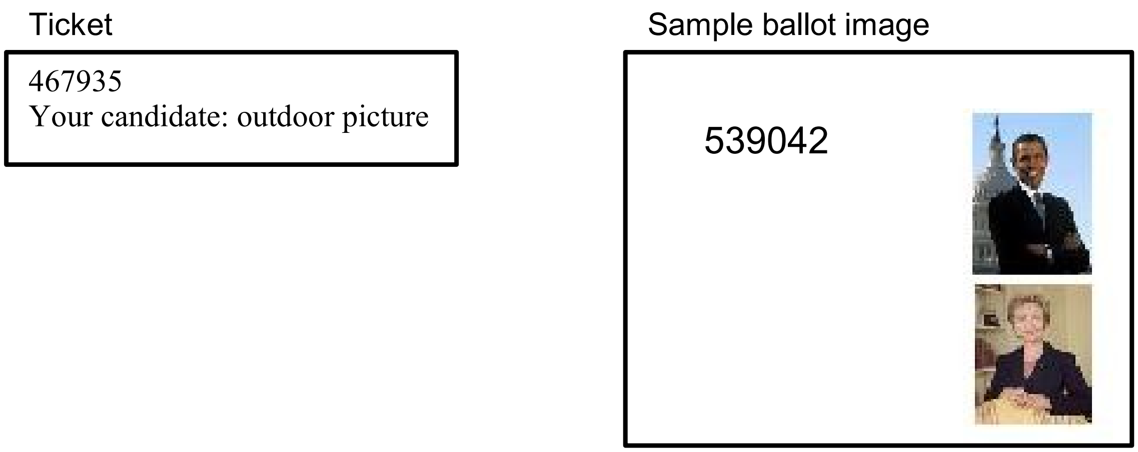


Figure 5: C-ThreeBallot Receipt Showing a Single Mark For Candidate Obama

The security requirement for the primitive is that, given the value of $\rho(r, g(v))$ for several values of $(r, v)$, the computer not be able to produce an image that is accepted by a human for any other value of $(r, v)$.

**Definition 1**[SECURITY BREAK] A program breaks the security of mapping $g : \mathcal{V} \rightarrow \mathcal{F}$ if, given some $r_1, r_2, ...r_n \in \mathcal{R}$, some $v_1, v_2, ...v_n \in \mathcal{V}$, and $\rho(r_1, g(v_1)), \rho(r_2, g(v_2)), ...\rho(r_n, g(v_n))$, for some value $n << |\mathcal{V} \times \mathcal{R}|$, it can produce a composite image that is read as being the receipt $r$, printed in format $g(v)$, by a human with ticket numbered $v$, when $(r, v) \neq (r_i, v_i) \forall i$.

**Assumption 2** [SECURITY] In the absence of a real-time solution to an unsolved AI problem, a human and a computer together cannot break the security of $g$ in real time.

**Assumption 3** [ONE USE TICKETS] Each ticket number $v$ is used at most once.

**Property 1** [SECURE DELIVERY] If Assumptions 2 and 3 hold, and $\mathcal{V}$ is large enough, a voter with ticket $v$ is assured that her receipt $r$ has reached the verifier if she views a composite image that she reads to be receipt $r$ in format $g(v)$, unless a real-time solution to the hard AI problem is obtained.

*Proof Sketch:* If not, then an entity not possessing $g(v)$ produces a composite image that is read by the voter to be the receipt $r$ printed in format $g(v)$. If assumption 3 holds, and as $g$ is injective, the entity does not have access to $\rho(r, g(v))$. From assumption 2, this entity then breaks the security of $g$, and provides a real-time solution to the hard AI problem.

**Property 2** [NONREPUDIATION] If the classical digital signature scheme used is secure, the verifier cannot later deny that it sent a composite image that it did send.

*Proof Sketch:* Follows from the properties of the classical digital signature schemes.

**Property 3** [INTEGRITY, C-PUNCHSCAN] C-Punchscan provides at least as much integrity as Punchscan if assumption 2 holds and verifiers are honest.

*Proof Sketch:* Integrity is reduced by either (a) an incorrect receipt being sent to the verifier or (b) the verifier checking the receipt incorrectly or (c) a valid claim of receipt manipulation being made by a verifier is considered invalid. (a) is addressed by Property 1, (b) by honest verifiers, and (c) by Property 2.

**Property 4** [PRIVACY, C-PUNCHSCAN]
If Punchscan receipts reveal no information about the vote, Steps 2 and 3.1 of C-Punchscan do not reveal information connecting a voter to a vote, unless it is revealed through the physical voting process or the voting machine.

*Proof Sketch:* The only possible extra information revealed in C-Punchscan is the association between voter and receipt. Because the receipt reveals no information about the vote, C-Punchscan does not reveal information on the vote of a specific voter.

**Property 5** [INTEGRITY, C-THREEBALLOT] If assumption 2 is true, and all verifiers are assumed honest (that is, no verifier colludes with the EA to change the vote), C-ThreeBallot provides at least as much integrity as ThreeBallot.

*Proof Sketch:* This proof is similar to that for Property 3.

**Property 6** [VULNERABILITY, C-THREEBALLOT] A single verifier can collude with the EA to change any number of its ballots in C-ThreeBallot (upto the maximum allowed by the other properties of ThreeBallot).

*Proof Sketch:* The verifier provides a correct receipt to the EA, and the voter believes her ballot is correctly recorded. However, the verifier does not point out the discrepancy between the ballot provided by the polling machine to the bulletin board and the ballot provided by the voter. The Polling Machine can hence change that single ballot to any other, as long as it does not violate the other properties of ThreeBallot ballots (such as the total number of marks is $Nc + N$ where $N$ is the number of voters and $c$ the number of candidates).

## 4 Conclusions and Future Work

The use of CAPTCHAs in voting is promising because CAPTCHAs 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 CAPTCHAs, such as audio-based CAPTCHAs, for ease of use for those with disabilities. Such a CAPTCHA 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 voices can be used, the audio snippet cannot be spliced out of previous votes.

**Acknowledgments**
Our simple implementation depicted in the C-Punchscan example adapted captcha-creating code from the open-source SimpleCaptcha project on Sourceforge [1]. One particular referee provided several useful suggestions.

# References

[1] Simplecaptcha. http://simplecaptcha.sourceforge.net/.

[2] Punchscan, 2006. http://www.punchscan.org/.

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

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

[5] A. Neff. A verifiable secret shuffle and its application to e-voting. In *8th ACM Conference on Computer and Communications Security*, 2001.

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

[7] Ronald L. Rivest. The threeballot voting system. Unpublished draft, 2006.

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

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