On the Independent Verification of a Punchscan Election

Richard T. Carback III
Center for Information
Security and Assurance,
University of Maryland,
Balitmore County.
carback1@umbc.edu

Jeremy Clark
School of Information
Technology and Engineering,
University of Ottawa.
jclar037@site.uottawa.ca

Aleks Essex
School of Information
Technology and Engineering,
University of Ottawa.
aesse083@site.uottawa.ca

Stefan Popoveniuc
Department of
Computer Science,
George Washington University.
poste@gwu.edu

Abstract— **Punchscan is a cryptographic voting system providing full transparency throughout the entire election process: a mandatory pre-election public audit, a mandatory post-election public audit, and the ability for a voter to check the correct printing and recorded marks on a paper receipt she keeps. Even though a voter can verify that her vote is counted as she cast it, the ballot receipt does not contain enough information to show someone else how she voted.**

These unique properties produce a system with a voluntary and universally available process that establishes an overwhelmingly high statistical degree of confidence in the integrity of the outcome—in other words, they allow for unparalleled independent verification of election results. These ideas are new and have the potential to radically change the way we think about and build the voting systems of the future.

I. Introduction

Punchscan is an open-source voting system, the results of which are verifiable by voters and independent observers through an election audit—a voluntary and universally available process that establishes an overwhelmingly high statistical degree of confidence in the integrity of the outcome. Engaging voters and observers more centrally in the election process through independent verification, observability and transparency is the motivation behind the development of the Punchscan voting system. At the same time this system was developed around the recognition that the secret-ballot, and in turn voter privacy, remains a fundamental requirement in modern voting systems. Engineering a voting system that offers voters the ability to "see their vote count," while at the same time protecting against improper influence was an enormous design challenge, and Punchscan represents the current state-of-the-art in the field of voting systems research.

Unlike conventional electronic voting systems, the judicious use of cryptography alleviates the concern over the strict use of trusted hardware and software components. The polling place hardware/software never encounters a voter's unencrypted vote. The entire process leaves an end-to-end audit trail that allows voters to independently verify for themselves that their ballot was correctly included in the tally, and that the tally was performed correctly. Additionally by design, audits can be performed throughout the election making Punchscan highly effective at preventing deliberate cheating or software errors that might necessitate a re-vote.

Unlike other cryptographic voting systems, you don't need a degree in abstract algebra to understand how Punchscan works. The cryptographic clockwork that protects, shuffles and retrieves the ballots is similar in technique to the "secret decoder ring" one might encounter in a cereal box. And because Punchscan does not lean on trusted software, it is easier to understand than the code running in a modern electronic voting machine—even if we were allowed to see it!

We begin with an emphasis on the importance of independent verification in voting systems and define what it entails. We then begin to describe the Punchscan voting system, beginning with a high-level design view that will give an overview of the core components and the auditing protocol. Afterward, we give a more detailed architectural view, describing the physical hardware and software components, user roles, and how they work together in the system. Finally, we demonstrate how Punchscan meets the criterion of end-to-end (E2E) cryptographic independent verification. This paper is an introduction to the longer and more comprehensive submission of Punchscan to the 2007 Vocomp[1]. Parts of this work may incorporate text or figures, with permission, from the published work done in [13], [9], [8], [7].

II. Independent Verification

The independent verification process is Punchscan's *raison d'tre*. The notion of independent verification refers to an individual or organization unassociated with the election trustees that undertakes to perform an audit of election results to verify the correctness of the final tally. Independent verification (IV) represents a new paradigm in elections bringing citizen oversight to an unreached level in current voting systems. Unlike conventional optical scan voting systems, the Punchscan audit is:

- End-to-end (E2E)—the audit is performed on all ballots not just a small percentage.
- Compulsory—the audit is performed in every race and not only under exceptional circumstances.

[1] Available from `http://punchscan.org/vocomp`

- Available and repeatable—the audit can be performed by anyone willing to take the time.
- Open specification and open source—the auditing tools can be self-created. The implementation offered by the Punchscan team is simply offered for convince, and it is open source and documented for easy scrutiny and verification.
- Software/hardware independent—the auditing is mathematical and does not depended on a specific implementation, programming language, operating system, or proprietary computers.
- Easy—while cryptography can be incredibly complex, Punchscan uses easy to understand encryption functions.

A. The Cost of Independent Verification

For all its benefits, independent verification does comes with one complexity: ballots must contain unique information and serial numbers. Because Punchscan adheres to the traditional principle of the secret ballot, the system must contain additional privacy enabling and assurance components. Therefore as we begin examining the Punchscan system, keep in mind that we are attempting to design a voting system with the following two properties:

- **Integrity** of election results through ballot "receipts" and audits.
- **Secrecy** of voters' vote through strong cryptographic design and distribution of trust.

Intuitively one might expect that integrity of election results and ballot secrecy are mutually exclusive. However as we will see, through the use of cryptography and careful design, these two properties are both simultaneously realizable.

B. Two Important Cryptographic Concepts

Two cryptographic concepts are at the heart of the independent verification provided by Punchscan. The first is a commitment scheme and the second is a cut and choose (CNC) protocol [5]. In a commitment scheme, Alice has a piece of information that she wishes to keep secret for a period of time but wants to prove to Bob that she knows the secret now. A commitment scheme allows Alice to compute a piece of data from her secret, called the *commitment*, that can only be computed from her secret. She gives the commitment to Bob and Bob should not be able to work backwards and recover the secret from the commitment even if he knows the exact algorithm Alice used for computing the commitment. When Alice reveals her secret to Bob and claims it is the same secret she committed to, Bob can take the secret he is given and compute the commitment for it. If this matches the commitment he was originally given, he knows that it is the same secret Alice committed to. For example, Alice may claim to have made a great scientific discovery but is waiting to settle some business plans before revealing it. However she is concerned it may be independently discovered in the meantime and wants to ensure her recognition of making the original discovery. She can published a commitment of her discovery and then later when it is revealed, it can be independently verified that her discovery matches the commitment.

When the election trustees generate the unique information that will be contained in each ballot, they commit to it using a cryptographic one-way function. They also generate the data necessary to tally the election and this committed to as well. Though these commitments and the commitment function used to generate them are made public, the actual information contained on the ballot remains sealed. This is because the function is one-way—it is computationally infeasible to determine the information on the sealed ballot given only its publicly posted commitment. The reason for committing to this data is to ensure that it is not changed at any point during or after the election. In lieu of committing to this data, the election trustees could, for example, change it after the election to engineer a desired result.

In a cut and choose protocol, Alice makes a claim about a piece of data and Bob must verify that claim without seeing the data itself. To verify Alice's claim, Bob asks Alice to create many substitutable pieces of data matching her claim and to individually encrypt them. Bob then *chooses* one piece of data to be set aside, and requests the decryption keys for the remaining *cut* pieces of data. He checks each one to verify that they all meet Alice's claim, and if they do he authorizes the use of the remaining chosen piece of data. For example, Alice may claim that a sealed envelope contains a hundred dollar bill but she does not want Bob to see the serial number on it. Bob can ask Alice to produce n sealed envelopes, each with a hundred dollar bill in them. Bob opens all but one of the envelopes and if they all satisfy Alice's claim, he is reasonably sure that the chosen one does as well without opening it. In order to successfully cheat without being caught, Alice creates 1 bad piece of data (such as an empty envelope), and hopes with probability $1/n$ that Bob will choose that piece of data to remain unopened. If n is large enough, there is an overwhelming probability that Alice will get caught if she tries to cheat.

When we use CNC in Punchscan, Bob is an auditor of election data and could be a voter or any independent verifier. The verifier cannot see all of the election data to protect voter privacy. However the verifier must be assured that this data has not been changed since it was committed to and that all the tally operations were performed correctly. To facilitate this goal, while preserving privacy, half of the data is selected to be opened for auditing. Furthermore, many functionally equivalent sets of election data are created and the verifier can select different halves to be opened in each. This is a notable difference from the types of CNC protocols described above. In this construction, a corrupt election trustee will have a 50% chance to get away with cheating in one set of data (as he must hide the cheating in one half or the other) but since many sets are created, he must cheat in the exact same way in each set for them all to be functionally equivalent and produce the same final tally. So instead of succeeding with probability $1/n$, the corrupt trustee will only succeed with probability $1/2^n$, where n is the number of sets. As n grows, this quickly results in an even greater probability of being caught. For example, if only ten independent sets are created and the corrupt trustee cheats only once, his probability of being caught is over 99.9%.

III. Core Components of Punchscan

At its core, Punchscan is a derivative of an earlier unnamed E2E system proposed by Chaum that used visual cryptography [4]. Like that system, Punchscan also uses a two-sheet ballot and an audited mixing of ballots, but both of these components are substantially less complex in Punchscan.

The key advantage of Punchscan over that system is that the voter marks a pre-printed ballot instead of trusting a machine to generate one. The ballot is still split by the voter but it does not use visual cryptography. We believe the new ballot is an improvement because the voter verifies that her positions and letters are correct and need not make an exact comparison of the position of black pixels on a screen. In a sense, the Punchscan ballot receipt is human readable even though it does not reveal information about a voter's vote.

Instead of a series of tellers performing mix operations, Punchscan relies on a simple auditable series of ballot/group operations. These operations use a cryptographically secure pseudo random number generator (CSPRNG) [12]. As previously stated, to provide independent verification of these operations, Punchscan additionally requires an unconditionally secure bit commitment scheme (USBCS) [11] and a proper formation of a cut and choose protocol [5]. There are some trade-offs in losing the tellers, but our tallying function permits us to achieve the security goals of *unconditional integrity* and *computational privacy*.

We now discuss the Punchscan ballot, the form of the tallying data (called the Punchboard), and the independent verification of the correctness of the inputs to the tallying function and the tallying itself.

A. Ballot

The Punchscan ballot, as illustrated in Figure 1 is created by combining a top and bottom sheet of paper. The top sheet has letters or symbols next to candidate names and holes in it to show letters that are printed on the bottom sheet. The letters on both sheets are ordered randomly.

To vote, each voter uses a bingo-esque ink dauber to mark the letter seen on the bottom sheet that is next to the candidate of her choice on the top sheet. This action creates a mark on both sheets, because the ink dauber is larger than the hole through which the letter is viewed. Afterward, either the top or the bottom sheet is destroyed (as randomly determined by a poll worker before the voter is issued her ballot), and the surviving sheet is scanned, publicly posted, and kept by the voter as a receipt[2]. As shown in Figure 1, neither half of the ballot can reveal the original vote by itself. Only the election trustees can determine the original intent, and it can only do so using the publicly committed to election data. The position marked by the voter is known as the mark position, and in subsequent diagrams is either 0 for the left mark or 1 for the right mark. For races with more than two candidates, we would indicate the choice as 0, 1, ..., or n, with numbering starting at the leftmost position.

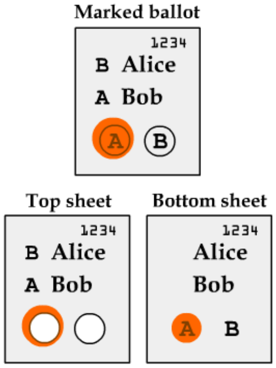


Fig. 1. A simple Punchscan ballot showing candidate list, serial numbers and letters appearing on each respective sheet. The letters on each sheet are independently randomly ordered. This diagram denotes a vote for “Bob.”

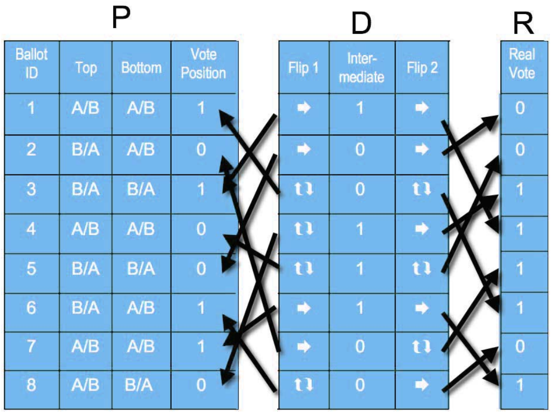
B. Punchboard

In order to determine voter intent, the trustees must know the letter ordering on the destroyed half of the ballot. This information is contained in a special data structure which we refer to as the “Punchboard,” and example of which is shown in Figure 2. To represent votes or marks in the Punchboard by candidate order. Thus a 0 in one of the cells of the results table represents a vote for the first candidate listed on the ballot, a 1 represents the second candidate and so on. We refer to this as the “canonical ordering;” the ordering of candidate names as it appears on the ballot.

The Punchboard is used to provide voter privacy and election integrity. If we post it as shown in the figure, there is no privacy in the system, but if it remains secret, we provide no publicly verifiable integrity to the counting process. In order to achieve both of these properties, Punchscan uses its own USBCS to commit to certain data before ballots are printed for the election, and CNC is used to reveal parts as the election progresses. This method enforces integrity by making public certain values as we progress through the election, allowing anyone interested to check to make sure the public values, or revealed data, match what election authorities committed to before the election. The data not made public protects the privacy of voters.

The first type of commitment, the printing commitment, is a commitment of each cell for the top and bottom sheets in the P table. The printing commitment data is revealed when a ballot is spoiled, or after results are posted when the EA knows which sheet each voter took as a receipt. Thus, depending on the sheet chosen to be destroyed, the receipt not only verifies the positions chosen by the voter but also permits voters to check on the printing process.

[2] Alternatively, the top sheet of the ballot could consist of a random ordering of the candidates, and the voter would mark next to the candidate. This change removes the need for a choice of sheet (forcing the bottom sheet to be the choice). We do not do this for two reasons: we wish to preserve an ordered candidate list as may be required by state laws, and the random choice serves as a cut and choose protocol to check the integrity of the printing process.



Ballot ID	Top	Bottom	Vote Position
1	A/B	A/B	1
2	B/A	A/B	0
3	B/A	B/A	1
4	A/B	A/B	0
5	B/A	B/A	0
6	B/A	A/B	1
7	A/B	A/B	1
8	A/B	B/A	0

Flip 1	Inter-mediate	Flip 2
➡	1	➡
➡	0	➡
⇅	0	⇅
⇅	1	➡
⇅	1	⇅
➡	1	➡
➡	0	⇅
⇅	0	➡

Real Vote
0
0
1
1
1
1
0
1

Fig. 2. The Punchboard demonstrating 8 ballots in the simplest election: a single contest, two-candidate plurality (i.e. "first past the post") vote. "Encrypted" votes are displayed on the left side in the P table and "unencrypted" votes are shown in the R table. The Flip columns contain either a straight arrow, which leaves the vote position mark alone, or a circular arrow which flips a 0 to 1 and a 1 to 0. The top and bottom sheet columns considered together should match the Flip 1 and Flip 2 columns considered together such that 0 corresponds to the first candidate in the R table and 1 the second.

The second type of commitment is a D-row commitment, which encompasses both flips in the D table and the corresponding permutations. This data is released only when a ballot is spoiled. Revealing this data can check on both the printing process and serve as an integrity check to make sure the flip columns correspond to the top and bottom sheet columns in the table.

The last kind of commitment, the mix commitment, consists of each of the entire flip columns (i.e. Flip 1 and the permutation to the P table or Flip 2 and the permutation to the R table). After results are posted the auditor chooses which of the two commitments for each D table to reveal. This is an explicit CNC operation, and allows us to verify that the table was filled out correctly by the EA, but does not permit us to determine what rows or votes in the P table corresponded to what rows or votes in the R table.

It is important to note that all of the data in the Punchboard is deterministically generated by a secret master key. No one trustee has access to the shared key, however they each own a share of it. If a predetermine threshold of trustees enter there shares (in the form of passphrases), the master key can be generated and used to recreate the Punchboard exactly as it was the first time. For this reason, the Punchboard does not need to be created and stored in memory somewhere. It is recreated by the trustees each time the data it contains is needed for some procedure within the election and is never stored in non-volatile memory.

IV. Independent Verification in Punchscan

Let us reiterate that independent verification is the ultimate purpose of Punchscan and E2E systems in general. Punchscan realizes independent verification through four verification mechanisms:

1) Audit Challenge Geneartion
2) Pre-Election Audit
3) Receipt Checking
4) Post-Election Audit

As we will discuss, the election trustees will "publish" (i.e. make publicaly available) selected pieces of information about the Punchboard that will be used for the purposes of independent verification. This published information does not allow anyone to link voters to votes, but does offer a high statistical assurance that the election results are legitimate.

A. Audit Challenge

We begin by discussing the notion of an audit challenge. To preserve ballot secrecy, we cannot open up the entire Punchboard to be audited. However the Punchboard has been modularly designed such that pieces of it can be revealed without compromising ballot secrecy. Which subset of the Punchboard is to be revealed is decided by the audit challenge generator. This could simply be candidates each requesting ballots to be audited, however we have implemented our audited challenge generator to be such that no one will know *a priori* which selections will be made. The audit challenge generator performs the following:

- **Pre-Election Challenge**: Given the total number of ballots in the election, randomly select serial numbers of half the ballots to be published and audited.
- **Post-Electon Challenge**: Given the serial numbers of the ballot receipts, for a given ballot select either the left or right half of the decryption table to be published and audited.

If the election trustees do not know *a priori* which ballots will be checked, they faces a high probability of getting caught if they publishes false commitments. The ballots to be selected for the audit should not be guessable with any advantage over a random guess at the time the commitments are published. Additionally the challenges themselves must be verifiable after the challenges are made. On first blush it would seem we would require some form of a true random process, like the rolling of dice, to perform these challenges. However unless you trust the dice and are personally available to witness them being rolled, you cannot be fully satisfied process was not manipulated to ensure a particular outcome. Instead an audit challenge generator can be created using high-entropy stock market data to produce fair, observable, and independently verifiable challenges. Provided that a sufficiently large stock portfolio is used with sufficiently volatile stocks, and a sufficient period of time is allowed to elapse between the publishing of the commitments and the generation of the selection, this method has been found secure [7].

B. Pre-Election Audit

The pre-election audit ensures proper construction of the Punchboard. During the pre-election audit, *half* the ballots generated by the trustees are selected to be examined. These ballots are unsealed (and spoiled) and checked to ensure they are properly formed and that they match their commitments.

Given a fair (i.e. randomly chosen) selection, we can be satisfied that the remaining sealed ballots are properly formed.

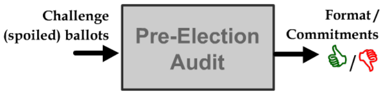


Fig. 3. Pre-Election Audit

C. Post-Election Audit

During the post-election audit, for each ballot cast in the election, one half of its decryption transformation (either left or right) is unsealed, published and checked to ensure it correctly performs its half of the decryption. The left partial-decryption step corresponds to the decryption of the ballot receipt to the partially decrypted result. Conversely the right partial-decryption step corresponds to decryption of the partially decrypted result to the result (i.e. vote). Again, given a fair (i.e. randomly chosen) selection, we can be satisfied that the remaining sealed half of the decryption is valid.

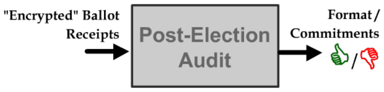


Fig. 4. Post-Election Audit

D. Receipt Check

The receipt check is the process by which the voter verifies that the information contained on their paper ballot receipt matches the information that was used in decryption/tally. The receipt check needs to be easy and available to voters to promote their participation in the process. It is however not mandatory for the voter to perform this step, and even a high degree of confidence in the election results can still be obtained with a relatively small number of checks. Additionally since the receipt does not contain information about how you voted, you can share this information with others, and allow them to perform this verification step on your behalf.

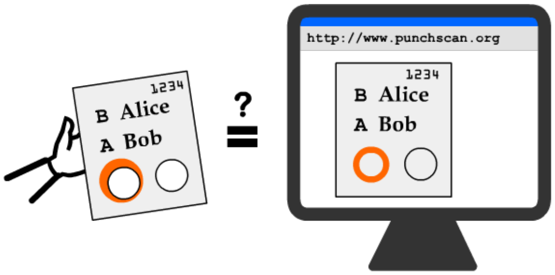
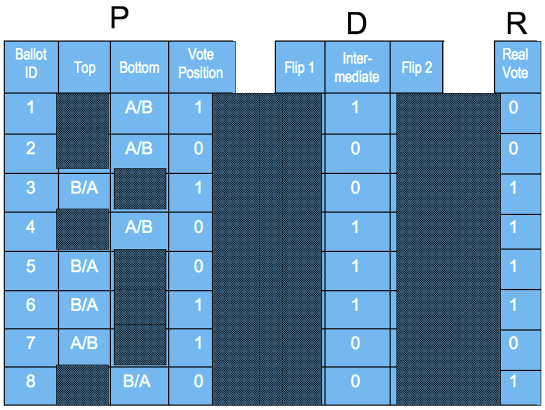


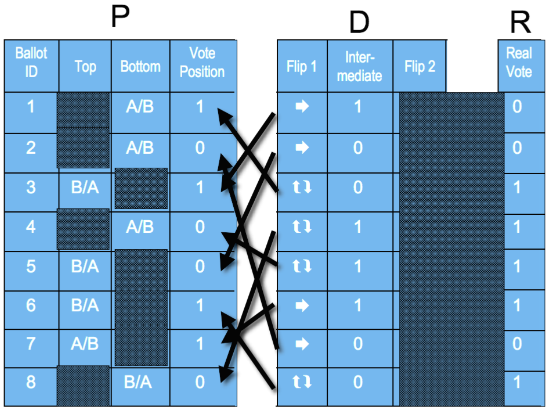
Fig. 5. Online Receipt Check

Therefore receipt information can be published freely, for example in a newspaper. We have chosen instead to implement the receipt check as an online solution. The voter visits the election website and types in the serial number appearing on their ballot receipt. A diagram of their ballot is displayed allowing the voter to verify that what they hold in their hand matches what was used in the decryption/tally.



Ballot ID	Top	Bottom	Vote Position	Flip 1	Inter-mediate	Flip 2	Real Vote
1	▨	A/B	1	▨	1	▨	0
2	▨	A/B	0	▨	0	▨	0
3	B/A	▨	1	▨	0	▨	1
4	▨	A/B	0	▨	1	▨	1
5	B/A	▨	0	▨	1	▨	1
6	B/A	▨	1	▨	1	▨	1
7	A/B	▨	1	▨	0	▨	0
8	▨	B/A	0	▨	0	▨	1

Fig. 6. The publicly available information about the Punchboard after the election and before the post-election audit. When voters check their receipts online, they are essentially ensuring that the revealed values in the P table are correct.

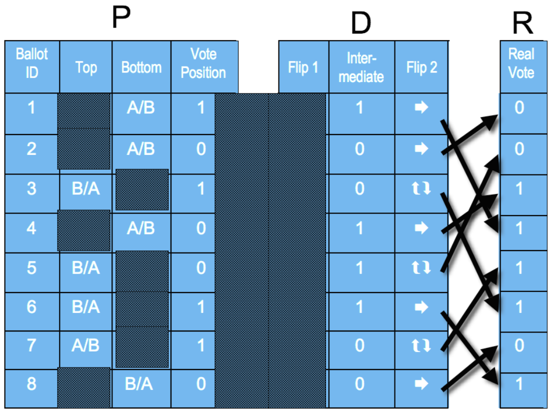


Ballot ID	Top	Bottom	Vote Position	Flip 1	Inter-mediate	Flip 2	Real Vote
1	▨	A/B	1	➡	1	▨	0
2	▨	A/B	0	➡	0	▨	0
3	B/A	▨	1	⇅	0	▨	1
4	▨	A/B	0	⇅	1	▨	1
5	B/A	▨	0	⇅	1	▨	1
6	B/A	▨	1	➡	1	▨	1
7	A/B	▨	1	➡	0	▨	0
8	▨	B/A	0	⇅	0	▨	1

Fig. 7. A final version of the Punchboard after the post election audit showing the left side of each row being selected for auditing.

Posting Results. When the election results are posted, election trustees enter their passphrases to regenerate the Punchboard data. They reveal the data on the Punchboard that corresponds to the information on the sheet that each voter took home as a receipt. Each voter is able to verify that her ballot was included with the correct marks for the final tally, that her receipt matches the revealed data, and that it was well-formed. Any independent verifier is able to verify that revealed data matches what was committed to before the election. An illustration of the information made public at this stage of the election is shown in Figure 6.

Post-Election Audit. The post-election audit ensures, among other things, that the counting process was executed properly while maintaining voter privacy. For each published D tables, either the two columns to the left or to the right of the intermediate column is revealed. In the original version of Punchscan, this was performed for the entire column, as illustrated in Figure 7 and Figure 8. However, the newest version of Punchscan allows the right or left half to be



P					D			R
Ballot ID	Top	Bottom	Vote Position	Flip 1	Inter-mediate	Flip 2		Real Vote
1	▨	A/B	1	▨	1	→		0
2	▨	A/B	0	▨	0	→		0
3	B/A	▨	1	▨	0	⇅		1
4	▨	A/B	0	▨	1	→		1
5	B/A	▨	0	▨	1	⇅		1
6	B/A	▨	1	▨	1	→		1
7	A/B	▨	1	▨	0	⇅		0
8	▨	B/A	0	▨	0	→		1

Fig. 8. A final version of the Punchboard after the post election audit showing the right side of each row being selected for auditing.

independently opened for each individual row in the D table. Either way, this enables independent verification that the marked positions match the intermediary values, or that the intermediary values match the final results. Because the election trustees do not know which half of each row in the *D* table will be selected before they produce the election results (recall this selection is made through random stock data), improperly publishing a result in either column would result in an overwhelming probability of being caught through this cut and choose protocol.

V. System Components

This section will cover the components of the system that are not people and are necessary at an architectural level. Specifically, this does not include independent or unconnected things like the ballot clipboard that is discussed later.

A. Web Server

A web server or group of Web Servers serves as the communications hub for all election parties. It is used to post receipts and Punchboard data. Independent verifiers can also use the web server to submit challenge and audit requests. The election trustees may respond to these requests by updating the copy of the partial Punchboard stored on the server(s). While this server contains important election data, its corruption (via hardware failure or malicious attack) does not imply voter privacy or election integrity has been violated. All data can be regenerated by the election trustees, and the election protocol can continue when the web server is reestablished in the case of a denial-of-service attack. However, since news that the web server has been compromised might adversely affect voter confidence, it is still important to keep the server properly secured and maintained.

As the central communications hub for election participants, the web server performs many important functions. When voters enter the Ballot ID from their receipt, the server's Web Application Software accesses mark and ballot configuration data from the public Punchboard to render a virtual copy of the receipt. Voters can inspect this virtual copy to ensure it is identical to their paper receipt. Independent verifiers can download all public election data, including the Punchboard, from the server in an open data format for automated processing or manual inspection. At the appropriate times, the server will accept challenges and audit requests as generated by the random stock data. In response, the election trustees must be able to log onto the web server to securely upload updated election data. Only election trustees require authenticated access to the server; all other users may remain anonymous. All data and software on the web server are public, therefore there is no risk a malicious user could obtain sensitive data.

B. Trusted/Diskless Workstation

While the web server is a public and marginally expendable computer, election trustees require a special, high-security Trusted Workstation with which they can process important election data with verified software. The workstation has no need of a hard drive and therefore should contain no information or programs when it is not in use. The Workstation also has no network interface or modem. Election trustees supply an operating system, programs and election data on removable media that is posted online for anyone to check before or after an election, and program output is stored on recordable media before the workstation is powered down. The Punchboard ensures the integrity of election, so the reason for the high-security of the workstation is to protect voter privacy.

A simple USB key may serve as a removable and recordable storage medium. Any such device can adequately supply and store data, as long as it features a write-protect switch to optionally prohibit the deletion or alteration of data or programs. Implementations may employ CD or DVD media and a combination of read-only and recordable disc drives to accomplish the same task.

Since the Diskless Workstation is the only computer to process election data in unencrypted form, a high threshold is set on its security and integrity. Its hardware configuration limits its ability to store or transmit sensitive information, and its Verified Trusted Software should faithfully process all data according to the descriptions earlier in the chapter.

All source code for the Workstation's operating system and user applications are open and published on the Web Server along with any derivatives, including compiled binaries and optical disk images. All published code and binary data are accompanied by their public hash value and the steps necessary to reconstruct any derivative from the original source code. This allows anyone to use publicly available tools to examine, build, test and verify the software to be run on the Diskless Workstation.

C. Printer

Since Punchscan is a hybrid paper/electronic voting system, separate hardware is necessary to manage and process paper ballots and receipts. Paper ballots can be printed with an ordinary inkjet Printer, although for large elections this task may be delegated to an industrial printing firm. The Printer must be trusted to print each ballot as directed by the Punchboard's print table. Printing distribution strategies can be used to minimize the impact of a violation of that trust.

D. *Scanner*

Within the polling place, voters mark their ballot and separate its pages. One page is destroyed by a cross-cut paper Shredder with a battery backup. Shredded ballot pages are properly disposed of using standard procedures for handling sensitive documents. The remaining page is scanned using an optical scanner with battery backup attached to a computer workstation. The workstation includes software to detect marks made by the voter and a screen to allow for corrections and final confirmation. Once verified, the vote is encoded in an XML file as a list of marks on a specified ballot page. The file is transmitted to the web server or stored on removable storage for later hand delivery. The scanner must be properly calibrated to recognize all possible valid marks on each ballot. This can be done using software algorithms or by calibrating the scanner with a sample ballot with all positions marked.

E. *Ballot Authoring*

One final software program is needed to specify key ballot parameters. The election trustees uses any program to author the ballots, with special graphical elements that are recognized by an automated application. The program outputs a standard file containing this information, which is transmitted to the web server for public examination. Once all errors have been detected and corrected, the file is locked to prevent further editing.

VI. User Roles

There are a few defined roles that users play in the system, but anyone, if they choose to do so, can play the role of an idependant verifier by looking at publicly provided data and verifying its correctness. We now present and describe the four roles of election trustees, poll worker, voter, and indepedent verifier.

A. *Election Trustees*

The election trustees are responsible for administrating and running the election. As a group they are trusted to handle all election data, including the Punchboard, in both its encrypted and unencrypted forms. It is essential that the election machinery distribute access to sensitive data across a super-majority of trustees (typically with competing interests or political affiliations), such that no minority of colluding trustees could ever view the unencrypted secret election data.

B. *Poll Worker*

Poll workers are the volunteers and other election officials that are responsible for the proper operation of each polling place. We think of the election trustees as actually a small group (no more than 5 or 10 people). Poll workers themselves have various roles from manning tables, to passing out ballots, to directing what polling places printers send their ballots.

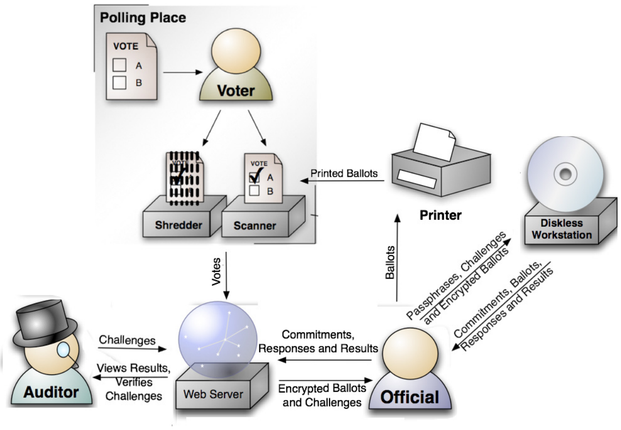

Polling Place
VOTE
A
B
Voter
VOTE
Shredder
VOTE
A
B
Scanner
Printed Ballots
Printer
Diskless Workstation
Votes
Ballots
Passphrases, Challenges and Encrypted Ballots
Commitments, Ballots, Responses and Results
Challenges
Views Results, Verifies Challenges
Auditor
Web Server
Commitments, Responses and Results
Encrypted Ballots and Challenges
Official

Fig. 9. An architecture diagram of the Punchscan System. The web server acts as a central, transparent repository for all election data. Not shown on this diagram is "everyone"; anyone can check receipts or download election data from the webserver. Note that in the version of Punchscan described here, the "auditor" is an algorithm that uses random stock data to generate the selections and not a person.

C. *Voter*

Voters contribute to any election system by casting ballots. Because Punchscan is an independently verifiable system, voters may also play an additional role as independent verifiers. They are encouraged to use a website to verify that the receipt they hold in their hand matches what was counted in the tally.

D. *Independent Verifiers*

Independent verifiers perform audit checks on the revealed election data. Any interested observer can examine this data to verify the election proceeded without irregularities or tampering.

VII. System Architecture

An illustration of our system architecture is shown in Figure 9[3]. As can be seen, the web server acts as a central repository for all election data, with the election officials responsible for populating the vast majority of that data (except for votes and audit challenges). The auditing algorithm provides the challenges (choosing random numbers, effectively using stock data), but independent verifiers must check the proper generation of the challenges and the correct disclosure of the data. Not shown is that Punchscan relies on a small number of voters to verify their receipts with the web server. This is not indicated on the diagram because anyone can check ballots (through voters giving copies of their receipts to organizations or election observers), and the web server should not make a distinction between an independent verifier and a voter. We also see the printer and diskless workstation, the two components that are trusted with privacy and interacted with exclusively by the election officials.

[3] This diagram was created from graphical elements that are found in [9]. They were created by Kevin Fisher.

VIII. E2E: End-to-End Cryptographic Independent Verification

In 2005, the American Election Assistance Commission (EAC) released a set of voluntary voting system guidelines [1] that includes a description of what they refer to as "End to End Cryptographic Independent Verification" (E2E) systems. According to the EAC, typical distinguishing features of an E2E voting system are as follows:

- A paper receipt is issued to the voter that contains information that permits the voter to verify that her choices were recorded correctly. The information does not permit the voter to reveal her selections to a third party.
- The voter has the option to check that her ballot selections were included in the election count, e.g., by checking a web site of values that should match the information on the voter's paper receipt.
- Such a system may provide an assurance not only that her ballot choices were correctly recorded (cast-as-intended), but that those selections were included in the election count (counted-as-cast).

A. Punchscan in the scope of E2E

Punchscan has been designed to be an E2E voting system. Most of the system components you have read about in the preceding sections have been developed to realize the E2E criteria. The EAC found that the range of proposed E2E systems have points of commonality, and they attempted to summarize these in a list of properties. Here we present some of them and briefly explain how Punchscan does or does not exhibit their properties. Note that this is not a list of requirements for a system to be classified as E2E but rather a preliminary sketch of the typical properties of these systems.

Property 1: Voters' ballot selections are encrypted for later counting by designated trustees.

A Punchscan ballot consists of two pages. The top page contains a list of contests and candidates with set of randomly ordered symbols beside the candidate names. There are holes in the top sheet that display a corresponding (but independently and randomly ordered) set of symbols. To vote on a Punchscan ballot, the voter observes the symbol appearing beside their chosen candidate's name, and locates the matching symbol in the holes. The voter then marks that hole with an implement such as a bingo-style dauber. The implement is sized slightly larger than the hole such that the ink mark will be made on both sheets. One of these sheets is destroyed in a cross-cut paper shredder. The remaining sheet represents the voter's receipt and is now "encrypted." [4] Only the threshold number of trustees (aka the election authority) have the ability to reconstruct the information contained on the destroyed sheet.

Property 2: Voting will produce a receipt that would enable the voter to verify that their ballot selections were recorded correctly and counted in the election.

Punchscan uses a robust audit procedure, including a process by which a voter can visit the election website and look up their ballot using the serial number contained on their receipt and verify what they hold in their hand matches what was recorded by election authority.

Property 3: The receipt preserves voter privacy by not containing any information that can be used to show the voter's selections.

Because one of the sheets is shredded, and assuming that the ordering of symbols contained on that page were uniformly random and independent from the page that was retained (aka the receipt), then no information about the destroyed sheet is contained on the retained sheet, and therefore the vote cannot be guessed with any advantage.

Property 4: No one designated trustee is able to decrypt the records; decryption of the records is performed by a process that involves multiple designated trustees.

Punchscan employs a threshold based password scheme whereby a pre-designated number of trustees must correctly enter their passwords before the records can be reconstructed.

Property 5: End to end systems store backup records of voter ballot selections that can be used in contingencies such as damage or loss of its counted records.

Punchscan in its original form relies on voter receipts to reconstruct an election should the counted records be destroyed. However an implementation of Punchscan used in a case study [8] expanded the originally proposed system to include a paper-based backup of the ballot receipts.

Property 6: The backup records contain unique identifiers that correspond to unique identifiers in its counted records, and the backup records are digitally signed so that they can be verified for their authenticity and integrity in audits.

The backup ballot receipts contained a serial number which matched the serial number of the ballot. While the ballot receipts themselves were digitally signed, the paper backups were not as it was agreed that the backup records were very unlikely to be needed. Should they have been used, they would have been published and thus their integrity would be ensured through voter verification. In future elections, consideration will be given to digitally signing the backups as well as the receipts.

Property 7: The documentation includes extensive discussion of how cryptographic keys are to be generated, distributed, managed, used, certified, and destroyed.

The source code for all the software used by Punchscan is open source and can be examined by anyone. Furthermore, the Punchscan team has attempted to document the underlying

[4] Since both sheets contain random but independent orderings of the symbols, possessing only one of the sheets does not give you information about the corresponding symbol on the other sheet.

cryptography of the system through papers, presentations, and other documentation available from the Punchscan website [5].

Property 8: Vote capture stations used in end to end systems must meet all the security, usability, and accessibility requirements.

The security of the vote capture station in a Punchscan election is similar to that of a paper ballot voting station. Two additional security measures are taken: one is to lock the ballot to a clipboard and the second is to ensure a high-integrity paper shredder.

Property 9: Reliability, usability, and accessibility requirements for printers in other voting systems apply as well to receipt printers used in end to end systems.

At pennies a ballot, Punchscan is low-cost, open source, and can be run on commodity hardware. Punchscan can be easily implemented with inexpensive off-the-shelf equipment. However Punchscan is hardware independent and could be adapted to use proprietary voting-dedicated equipment if it proved more reliable.

Property 10: Systems for verifying that voter ballot selections were recorded properly and counted are implemented in a robust secure manner.

Punchscan allows the voter to verify the proper scanning of their ballot at the polling station before it is cast, in addition to their ability to check the receipt online. The security of the Punchscan tallying process is dependent on well-studied cryptographic primitives [13] and no implementation vulnerabilities have been discovered to date.

IX. Concluding Remarks

Facilitating the end-to-end independent verification of election results is the primary goal of the Punchscan voting system. To this end, it is open standard and open-source. Since it draws its integrity from mathematics and not computer security, the polling stations computers can be off-the-shelf and based on closed-source operating systems without compromising the election integrity. This unconditional integrity is enforced through pre and post-election auditing of election data, and by allowing voters to keep a receipt they can check against the published election data. The privacy of the voters is preserved as the receipts do not contain any information about how they casted their vote [6]. The ultimate goal of this effort is a voting system with the potential to enrich the quality of democracy in this and other countries around the world.

References

[1] Voting system performance guidelines. *2005 Voluntary Voting System Guidelines*, Volume 1, United States Election Assistance Commission, Version 1.0, 2005.

[2] Brennan Center Task Force on Voting System Security (Lawrence Norden, Chair). The Machinery of Democracy: Protecting Elections in an Electronic World. *Brennan Center For Justice*, 2006.

[3] J. Buechler, T. Earnet, and B. Smith. Voting System Usability: Optical Scan, Zoomable, Punchscan. *UMBC CMSC 691/491V – Electronic Voting by Alan T. Sherman*, May 2007.

[4] D. Chaum. Secret-Ballot Receipts: True Voter-Verifiable Elections. *IEEE Security and Privacy*, IEEE Computer Society, 02.1, Los Alamitos, CA, USA, 2004, 38–47.

[5] D. Chaum, C. Crepeau, and I. Damgard. Multiparty unconditionally secure protocols. Proceedings of the twentieth annual *ACM Symposium on Theory of Computing*, 1988.

[6] J. Clark, A. Essex, and C. Adams. On the security of ballot receipts in E2E voting systems. Proceedings of *Workshop on Trustworthy Elections 2007*.

[7] J. Clark, A. Essex, and C. Adams. Secure and observable auditing of electronic voting systems using stock indices. *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) 2007*.

[8] A. Essex, J. Clark, R. Carback, and S. Popoveniuc. Punchscan in practice: an E2E election case study. Proceedings of *Workshop on Trustworthy Elections 2007*.

[9] K. Fisher, R. Carback and A.T. Sherman. Punchscan: introduction and system definition of a high-integrity election system. Proceedings of *Workshop on Trustworthy Elections 2006*.

[10] D.W. Jones. Chain voting. *Workshop on Developing an Analysis of Threats to Voting Systems*, National Institute of Standards and Technology, 2005.

[11] A. Kent. Unconditionally secure bit commitment. *Physical review letters*, American Physical Society, 83.7, 1999.

[12] A. Menezes, P. van Oorschot, S. Vanstone. Handbook of applied cryptography. CRC Press, Boca Raton, 1997.

[13] S. Popoveniuc and B. Hosp. An introduction to Punchscan. Proceedings of *Workshop on Trustworthy Elections 2006*.

[14] S Popoveniuc and J. Stanton. Undervote and Pattern Voting: vulnerability and a mitigation technique. Proceedings of *Workshop on Trustworthy Elections 2007*.

[15] S. Reiss. The Wired 40. *Wired*, 14.07, July 2006.

[16] J. Saltzer and M. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63:9, 1975.

[17] RFR-002: VoComp Evaluation Criteria. Voting System Performance Rating Organization. July, 2007. Available online: http://www.vspr.org/rfr-docs/vocompe.pdf

[5] http://punchscan.org