# Explaining the PunchScan voting system

Stefan Popoveniuc, Ben Hosp

George Washington University - CS Dept.
Washington DC 20052
{poste,bhosp}@gwu.edu

**Abstract.** PunchScan is a precinct-read optical-scan balloting system that allows voters to take their ballot with them after scanning. This does not violate the secret ballot principle because the ballots cannot be read without secret information held by the distributed authority in charge of the election. In fact, this election authority will publish the ballots for everyone to see, allowing voters whose ballots were incorrectly omitted to complain. PunchScan vote-counting is performed in private by the election authority – who uses their secret information to decode the ballots – but is verified in public by an auditor.In this paper we describe how and why PunchScan works. We have kept most of the description at an outline level so that it may be used as a straw model of a cryptographic voting system.

## 1 Motivation

The accurate results of a democratic election are at the heart of any modern society. Democracies are built throughout the world with the commitment to have elected individuals representing the entire population of a nation. To be able to record the wish of the people accurately we need to have a voting system that is transparent, reliable and verifiable. We need to be able to prove that the elections are run correctly, that every vote counts, and that the every person going to the polls and exercising their right to vote can make a difference. At the same time, we have to respect the secret nature of any vote. Linking a voter to a vote should not be possible, with or without the complicity of the voter.

PunchScan is a novel voting system and extremely easy to use, both by the voter and by the people running the elections. It is transparent and reliable, and provides public verifiability, election integrity and enhanced voter privacy.

## 2 Key elements/Ideas

There are three key elements that make PunchScan work:

1. The ballot is made out of two separate pages. When the two pages are put together, the resulting ballot reveals the choices of the voter. When only one page is viewed, it gives no information – in the computational sense – about what candidates the voter chose. Thus, if one page of the ballot is destroyed, the voter can keep the other page, without violating ballot secrecy.

2. A mechanism allows the recovery of the candidate choices from only one page of the ballot
3. The integrity of the election is provable through pre- and post-election audits.

These ideas are common both to PunchScan and to a previous method of David Chaum's [Cha]. However, PunchScan is more practical, because it does not suffer from the perfect alignment problem of the previous method, because the cryptography used is simpler, and because the time required to find the result and obtain the integrity proof is smaller.

## 3 High-level system design

PunchScan achieves publicly verifiable integrity while maintaining a voter friendly interface using an optical scan-like ballot. It gives each voter the opportunity to take their vote home and check that it is counted in the final tally. In this section, we first describe the ballot itself, then we present all the phases of the voting process as seen by all the participants: voters, the election authority, and candidates.

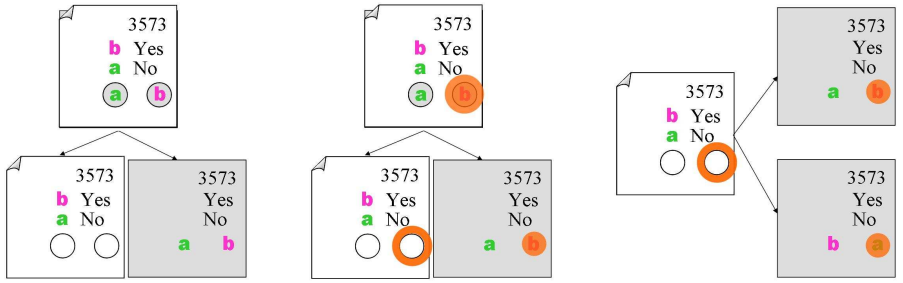
We assume that the candidates are auditing the election, since they are the ones that should care most about a correct outcome; in particular, each candidate would want to check that his rightful votes were not given to another candidate.

### 3.1 Ballot design

A ballot consists of two stacked sheets of paper. The top page of the ballot has holes in it, and the information on the bottom page can be read through the holes. Both pages also contain all the text needed on the ballot, such as contests (i.e.: ballot questions) and the candidates' names. On the top page, every answer has a symbol assigned to it and the assignment of symbols to answers varies from ballot to ballot. On the bottom page of the ballot, there is an (apparently) unordered list of symbols and their order differs from ballot to ballot. The top and the bottom ballot pages are aligned in such a way that when they are overlaid, for every question on the ballot, the symbols from the bottom page are visible through the holes made on the top page (see figure 1(a)).

In PunchScan, the voter uses a dauber to mark the selection of candidates. A dauber is a pen that leaves a disk of ink on the paper when it makes contact, just like the ones used by Bingo players to mark the numbers on their tickets. The diameter of the ink disc is greater then the diameter of the hole punched through the top page, which means the dauber leaves a mark on both the top and bottom ballot pages. Figure 1(b) contains a ballot voted for "Yes".

Because the order of the symbols on the two pages of a ballot is different (and independent), one cannot determine which mark is for which candidate by viewing only one page. We assume that the association of candidates with symbols and the order of the symbols on the bottom page are uniformly random. Figure 1(c) has the right answer selected on the top layer; depending on which possible bottom layer is this ballot's actual bottom layer, that mark could represent a vote for "Yes" or a vote for "No", both with a probability of 50%.




(a) A sample ballot. When the two pages are overlayed, the symbols on the bottom page are visible through the holes.

(b) A voted ballot. If you look at each layer individually, you cannot say that the mark is for "Yes" or for "No".

(c) Given only one layer of the ballot, the marks on that layer are equally likely to represent a vote for any candidate.

**Fig. 1.** PunchScan's ballot

### 3.2 Chronological description

There are three phases of the voting process:

- the preelection phase (labeled B for *Before*)
- the election phase (labeled E for *Election*)
- the postelection phase (labeled A for *After*)

**The preelection phase** The preelection phase is preparatory and allows the setup of the election and integrity proofs. During the preelection phase, the ballots are generated, printed and audited. Also, the information that allows recovering the choice from one page of the ballot is generated and checked. The chronological order is the following:

B.1 The election authority generates ballots and commits to them.
B.2 The election authority generates and commits to the information necessary for decrypting one page of the ballot when the other one is destroyed.
B.3 The candidates challenge the election authority and ask to see some of the ballots (say half), along with the information from B.2.
B.4 The election authority provides the requested ballots, and opens the commitments associated with them, thus spoiling them.
B.5 The candidates check to ensure that the commitments are consistent with the opened ballots.

**Election day** On election day, the voters go to their assigned polling places, authenticate themselves as legitimate voters, and get a ballot from the election officials.

E.1 The voter is given a sealed ballot.
E.2 Without seeing the order of the symbols on either page, the voter commits to the page that will be kept (e.g by making a special mark on the other page).
E.3 The voter marks the hole that has the symbol associated with their favorite candidates on the ballot.
E.4 The voter separates the two pages, destroys the unchosen one and keeps the one chosen in [E.2].
E.5 The surviving page is scanned, and the positions of the marks are recorded and made public. Henceforth, all references to "ballot" will refer to this surviving page.

In an earlier version, the voter chose which page to keep after seeing and marking their ballot. The early choice of the page to become a receipt is necessary to prevent an attack described by John Kelsey.

**The postelection phase** After all the polls close, the election is audited and proofs carried out to ensure the integrity of the election. The chronological order of the events following an election is as follows:

A.1 Any voter can go to the election authority web site, enter a serial number for her ballot, check that the ballot is there, and that it accurately resembles the page she possesses.
A.2 The election authority processes all ballots to produce decrypted versions, along with a partially decrypted form of all the ballots.
A.3 The candidates ask to see some of the transformations from the original ballots to the partially-decrypted forms, and some of the transformation from the partially-decrypted form to the clear form.
A.4 The election authority replies to the challenges made by the candidates in [A.3].
A.5 The candidates check to see if the reply of the election authority is consistent with the commitments made in the preelection phase [B.2] and with the information made public in [A.2].

## 4 Description by roles

### 4.1 The voter

On Election Day, a voter comes to the assigned polling place and authenticates herself as a legitimate voter. She gets a dauber and a ballot, and before seeing it, commits to the page that she will keep. She enters a private voting booth. She chooses her favorite candidates by making a mark with the dauber on the hole that has the symbol associated with her favorite candidate. She then shreds the unchosen page and keeps the other one. Then, she scans the kept page. She may walk out of the polling place with this page, which serves as her (encrypted) receipt. Later, she can go to a web site, type in the serial number of her ballot, and check that the ballot is there. No other checks are required from the voter.

### 4.2 The election authority

In the preelection phase, the election authority decides the format of a canonical ballot. This is the one from which all the other ballot variants will be generated. Also, the canonical ballot is used to recover the choices of the voters, after one page of the ballot has been destroyed.

The election authority generates at least twice the number of ballots needed in the election, and commits to them (making the commitment public; the ballots themselves remain secret). It also generates and commits to information necessary to recover the intent of a voter from one page of the ballot.

In response to the preelection challenge [B.3], the election authority discloses all the information about half (or a significant fraction) of the ballots (thus spoiling them). This allows the candidates to check the commitments and ensures (with high probability) that all the ballots have been correctly generated.

After the election, the election authority posts partially decrypted ballots and cleartext ballots. To prove that both decryptions (partial and final) were done correctly, for each vote the election authority will reveal either how it transformed the voted ballot into a partially decrypted one, or how it transformed a partial decrypted ballot into a cleartext one, but not both for the same ballot. The auditors choose which part will be revealed, and the chances of a cheating election authority being detected grow exponentially with the number of votes cheated on.

### 4.3 The candidates

We assume that the candidates are competing in an election. Because of this, we can safely allow the candidates also to play the role of auditors. As auditors, the candidates challenge the election authority during preelection and postelection and check that the replies are consistent with the commitments.

## 5 An Example

We describe a minimal example: the election consists of a single binary contest; the voters vote "Yes" or "No". The election authority decides that, in the canonical ballot, the symbol "a" is associated with "Yes" and the symbol "b" with "No" on the top page. The election authority also decides that the order is "a" "b" on the bottom page. The canonical ballot is presented in figure 2(a). The election authority defines what is a shift of one from the canonical form on top and bottom pages. The canonical ballot corresponds to a shift of 0 (call it a non-flipped ballot) and the non-canonical ballot corresponds to a shift of one (call it a flipped ballot). Figure 3(a) contains all the possible top and bottom pages. Any top page can be combined with any bottom page to produce a ballot as seen in Figure 3(b). The four types of ballots are equally likely.

A non-flipped top page combined with a flipped bottom page results in a flipped ballot. All the possibilities are in table 1. Note that we are only interested in knowing if the entire ballot is flipped or not, not individual pages.

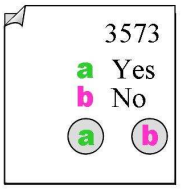
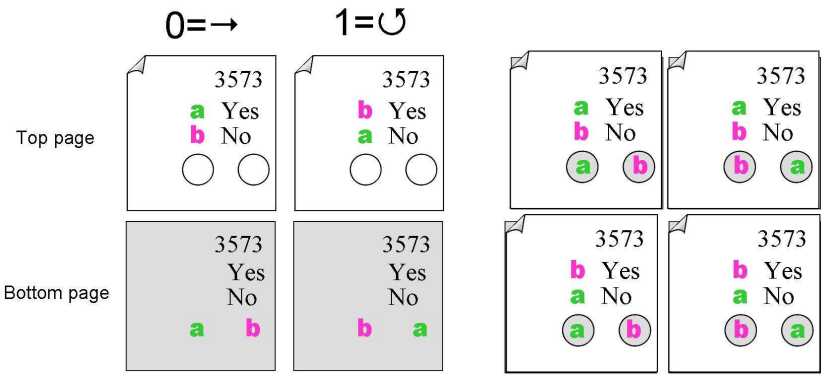



**Fig. 2.** The canonical ballot for a Yes/No contest




(a) All possible pages of a Yes/No ballot

(b) All possible ballots containing Yes/No contest

**Fig. 3.** PunchScan's ballot

To decrypt one page of the ballot, it is necessary to know if it came from a flipped or non-flipped ballot, to know if it should be flipped or not to get the canonical ballot. In PunchScan, this information is split into two flip/non-flip operations (flip1 and flip2) for each ballot. When combined, these operations transform the ballot page to the canonical ballot. The information is split so that one half can be made public for auditing purposes. The relation that has to hold between the pages of the ballot and the information used for recovering is: top $\oplus$ bottom = flip1 $\oplus$ flip2.

| $\oplus$ | Non Flipped | Flipped |
|---|---|---|
| Non Flipped | Non Flipped | Flipped |
| Flipped | Flipped | Non Flipped |

**Table 1.** Flipped / Non Flipped logic

The election authority makes public commitments to the ballots and to flip1 and flip2. The candidates choose half the ballots at random and the election authority makes public the requested ballots along with the flip1 and flip2 for each ballot. Anyone can check that the equation top $\oplus$ bottom = flip1 $\oplus$ flip2 holds. Only the ballots that were not made public in this phase (pre election) will be further used in the election.

During the election phase, the election authority publishes all the marked pages (half ballots) as voted on by voters. After the election, it publishes the intermediary state of the ballots (ballots $\oplus$ flip1) and the decrypted ballots (ballots $\oplus$ flip1 $\oplus$ flip2). These are commitments to the values of flip1 and flip2 used in the decryption of the voted half ballots.

During the postelection phase, the election authority is asked to open either flip1 or flip2 but not both, since opening both would allow the linking of a voted ballot to the corresponding decrypted one. Also, it is necessary that the partially-decrypted ballots and the decrypted ones be in a random order (distinct from each other and from the order of the voted ballots).

The election authority defines the following tables:

- $P$ (for **Print**)
- $D$ (for **Decrypt**)
- $R$ (for **Results**)

The $P$ table is indexed by ballot serial number and contains the top page ($P_1$), bottom page ($P_2$), and space for the filled-in vote (to be entered after the election). It also contains commitments to $P_1$ and $P_2$.

The $D$ table contains the first ($D_2$) and second ($D_4$) mark permutations (flips), the partially-decrypted vote ($D_3$) to be filled in during decryption, and information to connect it with the $P$ table ($D_1$) and the $R$ table ($D_5$). It also contains a commitment for each row of $D$, as well as a commitment for columns $D_1$ and $D_2$, and another commitment for columns $D_4$ and $D_5$.

The $R$ table contains the cleartext votes (after postelection decryption).

For example, consider an election with six votes. The clear data in all the tables is in Table 2. (No single person will ever see all of this information.) Before the election, but after the election authority has made the commitments, the tables look as they do in Table 3.

The candidates challenge the election authority to open a random half of the ballots, say the ones numbered 2, 4 and 5. The election authority reveals the requested information, and the tables look as they do in Table 4. Ballots 2, 4, and 5 now cannot be used in the election and are excluded from any further representation of the tables (see Table 5).

Assume that the voters mark their ballots as follows: on ballot 1, the left mark is marked, and the top page is chosen; on ballot 3, the right mark and the bottom page are chosen; on ballot 6, the left mark and the top page are chosen. Because the canonical ballot is “ab”,“ab” (that is, “ab” on both pages), left is associated with “a”, and right with “b”. The voters’ choices eventually end up in $P_3$, and when they do, each row describes what can be learned through knowledge of the ballot page chosen by the voter.

The election authority performs the first flip to ballots 1,3 and 6 to obtain the partially decrypted ballots as in $D_3$, and the totally decrypted ballots as in $R_1$ (see Table 6). The ballots in both $D$ and $R$ are shuffled independently, so it is not possible to link rows among tables $P$, $R$ and $D$. During the postelection phase, the auditor asks the election authority to open either the left or the right side of $D$ (but not both). If the election authority cheats, the auditor will catch it with probability 0.5 (for a higher probability see section 6.4). In our example, suppose the auditor chooses the right side. The election authority then reveals $D_4$ and $D_5$. The auditor can now check that $D_3 \oplus D_4 = R_1$, and that the commitment $CD_{4,5}$ to the columns $D_4$ and $D_5$ is valid.

| Ballot ID | $P_1$ | $P_2$ | $P_3$ | $CP_1$ | $CP_2$ |
|---|---|---|---|---|---|
| 1 | ab | ab | | $C_{1,1}$ | $C_{1,2}$ |
| 2 | ab | ba | | $C_{2,1}$ | $C_{2,2}$ |
| 3 | ba | ab | | $C_{3,1}$ | $C_{3,2}$ |
| 4 | ba | ba | | $C_{4,1}$ | $C_{4,2}$ |
| 5 | ab | ba | | $C_{5,1}$ | $C_{5,2}$ |
| 6 | ba | ab | | $C_{6,1}$ | $C_{6,2}$ |

| $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $DC$ |
|---|---|---|---|---|---|
| 6 | → | | ↻ | 5 | $C_A$ |
| 5 | ↻ | | → | 4 | $C_B$ |
| 2 | ↻ | | → | 1 | $C_C$ |
| 1 | ↻ | | ↻ | 3 | $C_D$ |
| 4 | → | | → | 2 | $C_E$ |
| 3 | → | | ↻ | 6 | $C_F$ |
| $CD_{1,2}$ | | | $CD_{4,5}$ | | |

| $R_{id}$ | $R_1$ |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |

**Table 2.** $PDR$ tables as the election authority sees them, with all the information available. The tables are properly formed, because, for all the ballots, $D_2 \oplus D_4$ correctly represents whether $P_2$ is a flipped version of $P_1$ or not. For example, for ballot number 3, on the top page, "a" is associated with "Yes", and b with "No". On the bottom page, the order is "ba", thus $P_2$ is a flipped version of $P_1$. In the $D$ table, in the row corresponding to 3, we have $\rightarrow \oplus \circlearrowleft$ = flip. For ballot 1, $C_{1,1}$ is a commitment to $P_1$, $C_{1,2}$ is a commitment to $P_2$ and so on.

| Ballot ID | $P_1$ | $P_2$ | $P_3$ | $CP_1$ | $CP_2$ |
|---|---|---|---|---|---|
| 1 | | | | $C_{1,1}$ | $C_{1,2}$ |
| 2 | | | | $C_{2,1}$ | $C_{2,2}$ |
| 3 | | | | $C_{3,1}$ | $C_{3,2}$ |
| 4 | | | | $C_{4,1}$ | $C_{4,2}$ |
| 5 | | | | $C_{5,1}$ | $C_{5,2}$ |
| 6 | | | | $C_{6,1}$ | $C_{6,2}$ |

| $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $DC$ |
|---|---|---|---|---|---|
| | | | | | $C_A$ |
| | | | | | $C_B$ |
| | | | | | $C_C$ |
| | | | | | $C_D$ |
| | | | | | $C_E$ |
| | | | | | $C_F$ |
| $CD_{1,2}$ | | | $CD_{4,5}$ | | |

**Table 3.** $PD$ tables in the preelection phase, as the public sees them.

| Ballot ID | $P_1$ | $P_2$ | $P_3$ | $CP_1$ | $CP_2$ |
|---|---|---|---|---|---|
| 1 | | | | $C_{1,1}$ | $C_{1,2}$ |
| 2 | ab | ba | | $C_{2,1}$ | $C_{2,2}$ |
| 3 | | | | $C_{3,1}$ | $C_{3,2}$ |
| 4 | ba | ba | | $C_{4,1}$ | $C_{4,2}$ |
| 5 | ab | ba | | $C_{5,1}$ | $C_{5,2}$ |
| 6 | | | | $C_{6,1}$ | $C_{6,2}$ |

| $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $DC$ |
|---|---|---|---|---|---|
| | | | | | $C_A$ |
| 5 | ↺ | | → | 4 | $C_B$ |
| 2 | ↺ | | → | 1 | $C_C$ |
| | | | | | $C_D$ |
| 4 | → | | → | 2 | $C_E$ |
| | | | | | $C_F$ |
| $CD_{1,2}$ | | | $CD_{4,5}$ | | |

**Table 4.** $PD$ tables after the election authority has replied to the request to open ballots 2, 4, and 5.

| Ballot ID | $P_1$ | $P_2$ | $P_3$ |
|---|---|---|---|
| 1 | | | |
| 3 | | | |
| 6 | | | |

| $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| $CD_{1,2}$ | | | $CD_{4,5}$ | |

**Table 5.** Ballots that can be used by voters on election day. The other ballots were spoiled during the preelection phase. The row commitments are not shown anymore because they won't be checked, since no other complete row will ever be opened.

| Ballot ID | $P_1$ | $P_2$ | $P_3$ |
|---|---|---|---|
| 1 | ab | | a |
| 3 | | ab | b |
| 6 | ba | | a |

| $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ |
|---|---|---|---|---|
| | | a | | |
| | | b | | |
| | | b | | |
| $CD_{1,2}$ | | | $CD_{4,5}$ | |

| $R_{id}$ | $R_1$ |
|---|---|
| 3 | a |
| 5 | b |
| 6 | a |

**Table 6.** $PDR$ snapshot after the polls close. One cannot say what row in the $D$ table corresponds to what row in the $P$ or $R$ table, because the rows are shuffled. Thus, the secret ballot principle is satisfied.

| Ballot ID | $P_1$ | $P_2$ | $P_3$ |
|---|---|---|---|
| 1 | ab | | a |
| 3 | | ab | b |
| 6 | ba | | a |

| $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ |
|---|---|---|---|---|
| | | a | ↺ | 5 |
| | | b | ↺ | 3 |
| | | b | ↺ | 6 |
| $CD_{1,2}$ | | | $CD_{4,5}$ | |

| $R_{id}$ | $R_1$ |
|---|---|
| 3 | a |
| 5 | b |
| 6 | a |

**Table 7.** $PDR$ snapshot after the postelection audit. The election authority was asked to open the right side of the $D$ table. Anyone can check that the partially decrypted result transformed by $D_4$ gives the result in $R$ ($D_3 \oplus D_4 = R$), so the election authority did not cheat. Also $CD_{4,5}$, the commitment to $D_4$ and $D_5$, is checked. Note that there is still no link between $P$ and $R$, so privacy is maintained.

## 6 A more technical description

This section provides a more technical description of PunchScan.

### 6.1 The ballot

Let $S$ be a set of symbols. The symbols in $S$ will appear on both the top and bottom page. We assume that $S$ is sorted and the order is fixed. We denote by "canonical ballot" a ballot that will have $S$ printed in order on both the top and bottom page. Let $T_p$ (top permutation), $B_p$ (bottom permutation), and $D_2$ be three random, independent permutations of $S$ (in an implementation, the permutation would be pseudorandomly generated as described in section A).

Compute $D_4$ such that $B_p o T_p^{-1} = D_2 o D_4$. Therefore, $D_4 = D_2^{-1} o B_p o T_p^{-1}$.

### 6.2 The tables

We describe the $PDR$ tables using notation from relational algebra, a system of notation heavily used in databases. It has the notions of relations (tables), projections ($\pi$ - SQL SELECT), selection ($\sigma$ - SQL WHERE) and join ($\bowtie$). A relation $R(A, B), A \rightarrow B$ means that $A$ implies $B$ (given $A$, $B$ is uniquely identified). $A$ is called a key of relation $R$.

Let $P$ (print) be the following relation:

$$P(B_{id}, P_1, P_2, P_3, CP_1, CP_2), B_{id} \rightarrow (P_1, P_2, P_3, CP_1, CP_2)$$

where $B_{id}$ is the ballot id (the serial number of the ballot), $P_1$ is $T_p$, $P_2$ is $B_p$, $P_3$ is a projection of $B_p o T_p^{-1}$ (voter choices), $CP_1$ is a commitment to $P_1$, and $CP_2$ is a commitment to $P_2$. The commitments are cryptographic commitments (see Section B.2 for details). $P$ contains $2n$ records.

Let $D$ (decrypt) be the following relation:

$$D(D_1, D_2, D_3, D_4, D_5, DC), D_1 \rightarrow (D_2, D_3, D_4, D_5, DC)$$

where $D_1$ is a foreign key pointing to the $B_{id}$ attribute of $P$, $D_5$ is a foreign key pointing to the $R_{id}$ attribute of $R$ (see below),$D_2$ and $D_4$ are permutations of $S$ described above, $D_3$ is $P_3 o D_2$, and $DC$ is a commitment to the tuple $(D_1, D_2, D_4, D_5)$. $D$ contains $2n$ records.

Let $CD$ (commitments to the columns of $D$) be the following relation:

$$CD(CD_{1,2}, CD_{3,4})$$

This relation has only one record. $CD_{1,2}$ is a commitment to $D_1$ and $D_2$; $CD_{4,5}$ is a commitments to $D_4$ and $D_5$.

Let $R$ (results) be the following relation:

$$R(R_{id}, R_1), R_{id} \rightarrow (R_1)$$

where $R_{id}$ is a unique identifier and $R_1$ is $P_3 o D_2 o D_4$. $R$ contains $2n$ records.

To select all the information for a ballot, we write:

$$(P \bowtie_{B_{id}=D_1} D) \bowtie_{D_5=R_{id}} R$$

### 6.3 The timeline

Before the election the election authority computes $P(B_{id}, P_1, P_2, CP_1, CP_2)$, $D(D_1, D_2, D_4, D_5, DC)$, $CD(CD_{1,2}, CD_{4,5})$ and makes public $P(B_{id}, CP_1, CP_2)$, $D(DC)$ and $CD(CD_{1,2}, CD_{4,5})$.

In the preelection audit, the auditor randomly selects half of the records in $P$. The election authority reveals $P \bowtie_{B_{id}=D_1} D$ for all the requested records. The auditor can check that $B_p o T_p^{-1} = D_2 o D_4$. and that the commitments $CP_1$, $CP_2$, and $DC$ are valid.

During the election, the voters fill in $P_3$.

After the election, the election authority computes $D_3 = P_3 o D_2$ and $R_1 = D_3 o D_4$ and makes $D_3$ and $R_1$ public.

In the postelection audit, the auditor asks the election authority to either reveal $(D_1, D_2)$ or $(D_4, D_5)$, but not both. The election authority reveals the requested information. The auditor can either check that $P_3 o D_2 = D_3$ (using $P \bowtie_{B_{id}=D_1} D$) or $D_3 o D_4 = R_1$ (using $D \bowtie_{D_5=R_{id}} R$). The chance of the election authority cheating and not being caught is 50% (see section 6.4). $CD_{1,2}$ and $CD_{4,5}$ are also checked.

### 6.4 Multiple instances of *D*

Because the election authority can cheat with 50% probability of success (i.e., nondetection), we introduce multiple instances of $D$. In other words, we modify the relation $D$ as follows: Let $D$ (decrypt) be the following relation:

$$D(i, D_1, D_2, D_3, D_4, D_5, DC), (i, D_1) \rightarrow (D_2, D_3, D_4, D_5, DC)$$

where $i$ is the instance number and the rest is as described in Section 6.2

Let $CD$ (commitments to the columns of $D$) be the following relation:

$$CD(i, CD_{1,2}, CD_{3,4}), i \rightarrow (CD_{1_2}, CD_{3,4})$$

where $i$ is a foreign key pointing to the $i$ attribute of $D$.

In the postelection audit, we can now make $k$ challenges, where $k$ is the number of $D$ instances. The auditor will ask to open either $(D_1, D_2)$ or $(D_4, D_5)$ for each instance of $D$. The chance that the election authority cheats successfully is one out of $2^k$. We can make this probability arbitrarily small by increasing $k$.

### 6.5 Multiple-question Ballots

We have been implicitly assuming that there is only one question per ballot. The situation becomes slightly more complicated if this is not the case. PunchScan works just fine for multiple-question ballots but the decrypted ballots will preserve the "cross-question" relationships: for example, if 90% of the people who voted for Alice for Governor also voted for Bob for President, the results will reflect this. However, PunchScan can be extended to hide these correlations if desired.

Trivially, of course, if PunchScan works for one-question elections then we can conduct an $n$-question election by giving each voter $n$ one-question ballots. If we want to preserve the cross-question relationship among two or more questions (perhaps if someone voted "No" for a recall election then they are not allowed to vote for a replacement candidate) then we could group those questions on the same ballot. This would work but seems to us to be not as good (from a ballot design, system overhead, and printing cost point of view) as the case when we are using one ballot and running one election.

However, we can readily modify this scheme to fix this problem. Suppose we are running $n$ one-question elections. That is, each voter receives one ballot for each of $n$ elections and votes, then the votes are counted separately for each election. In this situation, there is one $P$-table and one set of $D$-tables (and associated $R$-table) for each of the $n$ elections. Let us note that there is no information contained in the $D$-tables for election $A$ that can be used to decrypt the ballots for election $B$. Since the shuffles for each election are also independent, we do not need to obscure the link between voter $x$'s encrypted ballots in election $A$ and $B$, because when they are decrypted the shuffling will obscure the cross-question relationship for us. In other words, we can print these ballots together, on the same piece of paper, with the same serial number (and the same $P$-table row), just as in the original scheme that reveals the correlations. Because the ballots are decrypted separately, this does not provide any more information regarding the cross-question relationships.

## 7 Proofs

This section contains proofs of some security properties of PunchScan.

### 7.1 Privacy

In this context, the maintenance of privacy requires that an observer's probability distribution of the contents of a given ballot $i$ (i.e.: the value of voter $i$'s vote) be unchanged by observation of the cryptographically-hidden data. In other words,

$$p(b_i|PDR) = p(b_i|R),$$

where $b_i$ is the value of ballot $i$, $PDR$ is the entire publicly-observable ballot data matrix, and $R$ is the results column of that matrix.

**Attacks on $P$** The most straightforward way for an attacker to use the secret parts of $PDR$ to reveal the vote of voter $i$ would be to simply decrypt $P_{1,i}$ and $P_{2,i}$ and use those to decode $P_{3,i}$. If the attacker is unable to break this cryptography, then learning $P$ would not affect his probability distribution on $b_i$. This cryptography can be made arbitrarily strong in order to protect privacy at any desired level of computational security.

**Attacks on $D$** Another method would involve an attack on the shuffle; that is, decrypting the unrevealed link between $P$ and $D$ ($D_1$) or between $D$ and $R$ ($D_5$). However, the same cryptography is used to secure those columns of $D$, so again, an attacker unable to break the cryptography could not learn anything useful from $D$.

### 7.2 Integrity

There are four elements of the PunchScan process that are vulnerable to some extent to manipulation of the vote tally by the election authority.

- The ballots may be improperly formed.
- The ballots may be improperly printed.
- The ballot markings may be improperly recorded.
- The marked ballots may be improperly decrypted.

Each of these vulnerabilities is addressed by an audit procedure.

### 7.3 First Audit

The first audit procedure ensures that the ballots are well-formed, meaning that for each ballot, $P_1 \oplus P_2 = D_2 \oplus D_4$ for the row in each $D$-matrix associated with that ballot. This involves spoiling some fraction of the ballots by unlocking this secret data.

In general, suppose there are $n$ ballots, the election authority has cheated by malforming $k$ of them, and $f$ ballots are chosen at random to be examined. The probability that the election authority gets away with this attack is the number of possibilities where the auditor chooses only valid vote divided by the number of all possible choices.

The number of all the possible choices is $\binom{f}{n}$ ($n$ choose $f$). The number of ways to choose $f$ valid ballots from a total of $n$ ballots where $k$ of then are invalid, is $\binom{f}{n-k}$ (choose $f$ votes out of $n-k$ that are valid). So the election authority cheats successfully with the following probability:

$$p = \frac{\binom{f}{n-k}}{\binom{f}{n}} = \frac{\frac{(n-k)!}{f!(n-k-f)!}}{\frac{n!}{f!(n-f)!}} = \frac{\frac{(n-k)!}{(n-k-f)!}}{\frac{n!}{(n-f)!}}$$

Note that $f + k < n$, so that $n - k - f > 0$ and $(n - k - f)!$ exists and is not the special case 0!. If $f + k > n$ then the probability is 0.

In the interest of simplicity, from here we may compute two upper bounds on the chance that this attack will not be detected:

$$\begin{aligned}
\frac{(n-k)!}{n!} \times \frac{(n-f)!}{(n-k-f)!} &= \frac{(n-f) \times (n-f-1) \times ...(n-f-k+1)}{n \times (n-1) \times ... \times (n-k+1)} \\
&= \frac{n-f}{n} \times \frac{n-f-1}{n-1} \times ... \times \frac{n-f-k+1}{n-k+1} \\
&= (1-\frac{f}{n}) \times (1-\frac{f}{n-1}) \times ... \times (1-\frac{f}{n-k+1}) \\
&< (1-\frac{f}{n})^k \\
\frac{(n-k)!}{(n-k-f)!} \times \frac{(n-f)!}{n!} &= \frac{(n-k) \times (n-k-1) \times ... \times (n-k-f+1)}{n \times (n-1) \times ... \times (n-f+1)} \\
&= \frac{n-k}{n} \times \frac{n-k-1}{n-1} \times ... \times \frac{n-k-f+1}{n-f+1} \\
&= (1-\frac{k}{n}) \times (1-\frac{k}{n-1}) \times ... \times (1-\frac{k}{n-f+1}) \\
&< (1-\frac{k}{n})^f
\end{aligned}$$

Thus, our upper bound on the probability that the election authority gets away with malforming $k$ out of $n$ ballots when $f$ of those ballots are audited is $min[(1-\frac{f}{n})^k, (1-\frac{k}{n})^f]$.

### 7.4 Second Audit

In order to check that a given ballot receipt was properly printed, one can reencrypt it (that is, recompute the commitments) and compare it with the $P$-matrix. Suppose $n$ ballots remain unspoiled after the first audit, $f$ are actually used by voters who later check the commitments, and $k$ of them are improperly printed. Once again, the upper bound on the probability that none of the misprinted ballots are detected is $min[(1-\frac{f}{n})^k, (1-\frac{k}{n})^f]$.

### 7.5 Third Audit

In addition to checking that the ballot is correctly printed, one can also verify that the recorded ballot mark matches the mark on the receipt. In effect, this verifies the correctness of $P_3$. Again, if $n$ ballots remain unspoiled after the first audit, $f$ are actually used by voters who verify that their ballot marks are correctly recorded online, and $k$ ballots are incorrectly recorded, then the upper bound on the probability that none of the incorrectly-recorded marks are detected is $min[(1-\frac{f}{n})^k, (1-\frac{k}{n})^f]$.

### 7.6 Fourth Audit

The election authority may influence the vote tally by incorrectly decrypting the ballots. There are two methods we may use for auditing the election authority to ensure that this does not occur.

**Ballot-wise Auditing** Suppose the auditor goes through a $D$-matrix ballot-by-ballot (that is, row-by-row) and randomly chooses whether to inspect (open) the "left" or "right" commitment for each ballot. This situation is different from the first three audits because all ballots are inspected, but each inspection has only a $\frac{1}{2}$ chance of catching a modification. This makes the situation simpler; the chance of $k$ modified ballots all escaping detection is $2^{-k}$.

**Table-wise Auditing** On the other hand, the auditor may choose to open all the "left" or "right" commitments for a given $D$-matrix. Assuming that the election authority intends to cheat during the decryption and is aware of this, he will put all his cheating in a given $D$-matrix in either the "left" or "right" commitment, so that he has a $\frac{1}{2}$ chance of escaping detection when that $D$-matrix is inspected. If there are $n$ $D$-matrices, then the chance of escaping detection if any ballots are incorrectly decrypted is $2^{-n}$.

**Comparison** Both of these methods have desirable properties. The ballot-wise method has the feature that the probability of detecting cheating is a function of the number of ballots cheated on, and increases exponentially with a linear increase in number of cheated ballots. The table-wise method has the feature that the audit does not reduce the size of the anonymity set created by the shuffle.

## 8 Related Work

Verifiable electronic voting has been introduced by David Chaum in 1981 [Cha81]. The first voter verifiable version used visual cryptography [CvdGRV07]. Peter Ryan introduced the candidate permutation idea and developed an improved ballot, much more usable and implementable in a voting system called Pret-A-Voter [CRS05]. An early stage of PunchScan was analyzed by John Kelsey [JK07] who came up with an attack based on the fact that the voter can see the ballot and then decide which page to keep (see Section 3.2).

## 9 Appendix

## A Permutations

PunchScan requires two types of permutations to be generated:

- row permutations
- mark permutations

Row permutations refer to the permutations of the rows of the $D$ table and mark permutation refer to the order in which the positions are associated with marks on the ballot and to $D_2$ and $D_4$.

### A.1 Row Permutations Generation

Consider an "unshuffled" $D$-matrix where $D_1 = [1, 2, \ldots 2n]$, so row $x$ of $PDR$ represents ballot $x$ across the entire row, and $D_5$ is blank. The election authority should generate this matrix as the first step; call it $\delta$. Generating the row permutations will therefore take the form of the generation of $D^1 \ldots D^{n_D}$, where $D^i$ denotes the $i^{th}$ shuffled $D$-matrix.

The $D$-matrices will be generated from $\delta$ as follows:

1. Randomly shuffle the rows of $\delta$; call this $D^1$.
2. Let $D^1_5$ equal a random shuffling of $\{1, 2, \ldots, 2n\}$.
3. For each $i$ from $2, 3, \ldots, 2n$, let $D^i$ equal a random shuffling of the rows of $D^1$.

This involves $n_D + 1$ permutations of $\{1, 2, \ldots, 2n\}$. It should be clear that if $(y, D^i_1) = x$ and $(y, D^i_5) = z$, then for all $j$, $(y, D^j_1 = x)$ implies that $(y, D^j_5) = z$; in other words, since each row of $D^1$ contains a pointer to a (unique) row (ballot) of $P$ in $D_1$ and a (unique) pointer to $R$ in $D_5$, reordering its rows does not change the destination (in $R$) of any ballot in $P$.

### A.2 Implementation

**Permutation Algorithm** We use the following permutation algorithm to permute the unshuffled matrix. This algorithm generates a permutation $\pi$ of $1, 2, \ldots m$, given as input $m$, some encryption function $E$, and some key $K$.

First, create a table with $m$ rows and 2 columns. Populate column 1 of the table with $1, 2, \ldots m$ and column 2 of the table with $E_K(1), E_K(2), \ldots E_K(m)$; in other words, $(i, 2) = E_K((i, 1))$ for every row $i$. Next, sort the table according to column 2. Let $\pi(i) = (i, 1)$; column 1 is now a permutation of $1, 2, \ldots m$.

If the key $K$ were generated randomly, and the function $E$ is a good encryption algorithm, then the permutation output by the algorithm will be random. (That is, it will preserve any randomness in $K$.)

**Application of the Algorithm** The election authority can use this algorithm to implement the $D$-matrix generation algorithm above as follows:

1. Generate a permutation $\pi_{D^1}$ of $1 \ldots 2n$. Let $D^1_1 = \delta$, sorted by $\pi_{D^1}$; that is, row $x$ of $\delta$ becomes row $\pi_{D^1}(x)$ of $D^1$.
2. Generate a permutation $\pi_R$ of $1 \ldots 2n$. Let $D^1_5 = \pi_R$.
3. For each $i$ from 2 to $n_D$, create $D^i$ by generating a permutation $\pi_{D^i}$ of $1 \ldots 2n$. Let row $y$ of $D^1$ become row $\pi_{D^i}(y)$ of $D^i$.

### A.3 Mark Permutations

The mark permutations, in contrast, are much simpler to generate. In order to produce all possible associations of candidate names with ballot symbols, it is not necessary to randomly permute both lists; it is only necessary to cyclically shift

both lists a (different) random amount. So to generate the mark permutations for ballot $x$, where the ballot has $c$ candidate names on the top page and $c$ mark symbols on the bottom page, the election authority only needs to generate two random numbers between 1 and $c$, and record these numbers as $P_1$ and $P_2$ to indicate the shift distance for the pages of ballot $x$.

Each $D$-matrix instance will require its own set of decrypting mark permutations (columns $D_2$ and $D_4$). (It is for this reason that at least the decrypting mark permutations must be performed after the row permutations.) For each row of each $D^i$, the election authority generates a random number between 1 and $c$, and records this number in $D_2^i$. $D_4^i$ is set such that the modular sum of the ballot's entries in $P_1$ and $P_2$ equals the sum of its entries in $D_2$ and $D_4$.

**Random Number Generation** The permutation algorithm described above can also be used for the random number generation. The election authority can compute a permutation $\pi$ of $1, 2, \ldots, c$ and use $\pi(1)$ as the random number.

# B Commitments

This section describes how the commitments in PunchScan are computed. We use the comma (",") to represent the concatenation operation. There are two secret AES 128-bit keys, $MK_1$ and $MK_2$, and a public 128-bit constant, $C$.

## B.1 Computing AES keys

This section requires the use of two 128-bit AES keys. Given message $M$, let $M_{128}$ be the first 128 bits of $M$ (if $M$ is shorter then 128 bits, $M$ will be padded with trailing zeros); a random key $SK_m$ is generated as follows:

$$SK_m = D_{MK_1}(C \oplus E_{MK_2}(C \oplus E_{MK_1}(M_{128})))$$

where $\oplus$ is the XOR operation and $E$ and $D$ are AES Encrypt and Decrypt EBC NoPadding operations.

## B.2 Commitment Algorithm

Given a message $M$, the commitment to $M$ is computed as follows:

1. Generate a 128-bit AES key $K_m$ as described in Section B.1.
2. Encrypt the public constant $C$ with $K_m$, using AES 128-bit ECB NoPadding. Let the result be $SK_m = AES_{K_m}(C)$. Note that $SK_m$ has 128 bits.
3. Concatenate $M$ with $SK_m$ and hash everything using SHA256, resulting in $h_1$. So, $h_1 = SHA256(M, SK_m)$.
4. Let $h_2 = SHA256(M, AES_{SK_m}(h_1))$, where the AES encryption is AES 128bit ECB PKCS#5Padding.
5. The commitment is $h_1, h_2$ ($h_1$ concatenated with $h_2$).

We now describe the computation of $M$ for all the commitments needed in PunchScan.

***M* for $P_1$** $M$ is obtained by concatenating the serial number of the ballot to a constant particular to $P_1$ and with the text on the top page of the ballot.
$M = i$,"P1", $P_1$ where $i$ is a string representing the serial number of the ballot, "P1" is a constant string (capital "P" concatenated with digit "1") and $P_1$ is the string in $P_1$ (the string representation of the top page).

***M* for $P_2$** $M$ is obtained by concatenating the serial number of the ballot to a constant particular for $P_2$ and with the text on the bottom page of the ballot.
$M = i$,"P2", $P_2$ where $i$ is a string representing the serial number of the ballot, "P2" is a constant string (capital "P" concatenated with digit "2") and $P_2$ is the string in $P_2$ (the string representation of the bottom page).

***M* for rows in *D*** $M$ is obtained by concatenating all the known values in a row in $D$. The known values are: the pointer to the $P$ table ($D_1$), the first mark permutation ($D_2$), the second mark permutation ($D_4$) and the link to the $R$ table ($D_5$).
$M = D_1, D_2, D_4, D_5$, each $D_i$ being the string representation of a field in $D$.

***M* for columns in *D*** $M$ is obtained by concatenating all the values in the first column and then concatenating all the values in the second column.
For the leftmost columns:
$M = D_{1,1}, D_{2,1}, D_{3,1}, \ldots, D_{n,1}, D_{1,2}, D_{2,2}, D_{2,3} \ldots D_{n,2}$
For the right most columns:
$M = D_{1,4}, D_{2,4}, D_{3,4}, \ldots, D_{n,4}, D_{1,5}, D_{2,5}, D_{2,5} \ldots D_{n,5}$

We only need to protect two 128-bit AES keys, $MK_1$ and $MK_2$, in order to preserve the privacy of the system. The keys can be distributed and recreated as needed, only when a certain threshold of the participants is present.

Note that the public cannot verify that the AES keys have been generated in this way, or rather in some other way. Therefore, this system unfortunately introduces a potential covert channel via the AES keys.

## Acknowledgments

We would like to thank David Chaum, Poorvi Vora, Rick Carback, Jeremy Robin and Ben Adida, for the vibrant discussions and insightful comments.

## References

[Cha] David Chaum. Secret ballot receipts and transparent integrity - better and less-costly electronic voting at polling places. `http://www.vreceipt.com/article.pdf`.

[Cha81] David L. Chaum. Untraceable electronic mail, return address, and digital pseudonym. *Communication of ACM*, February 1981.

[CRS05] David Chaum, Peter Y. A. Ryan, and Steve Schneider. A practical voter-verifiable election scheme. In *In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, ESORICS, volume 3679 of Lecture Notes in Computer Science*, pages 118–139. Springer, 2005.

[CvdGRV07] David Chaum, Jeroen van de Graaf, Peter Y. A. Ryan, and Poorvi L. Vora. Secret ballot elections with unconditional integrity. Technical report, IACR Eprint, 2007. `http://eprint.iacr.org/` or `http://www.seas.gwu.edu/~poorvi/cgrv2007.pdf`.

[JK07] Tal Moran David Chaum John Kelsey, Andrew Regenscheid. Hacking paper some random attacks on paper based e2e systems. http://kathrin.dagstuhl.de/files/Materials/07/07311/07311.KelseyJohn.Slides.pdf, 2007.