Scratch, Click & Vote: E2E voting over the Internet

Mirosław Kutyłowski and Filip Zagórski*

Institute of Mathematics and Computer Science
Wrocław University of Technology
Miroslaw.Kutylowski@im.pwr.wroc.pl
Filip.Zagorski@im.pwr.wroc.pl

Abstract. We present *Scratch, Click & Vote* voting scheme, which is a modification of the Punchscan and ThreeBallot systems. The scheme is end-to-end veryfiable and allows for voting over the Internet. Security against malicious hardware and software used by a voter is due to the fact that a voter's computer does not get any knowledge about the voter's choice. Moreover, it can change successfully a voter's ballot only with a small probability.
As a side result, we present a modification of the ThreeBallot that eliminates Strauss'-like attacks on this scheme.

Keywords: Internet voting, evoting, E2E, veryfiable voting scheme
Version: last corrected 5 VII 2008 (this temporary version will expire on 31 VIII 2008)

1 Introduction

There is a lot of public interest in electronic remote voting schemes. On one hand it is necessary to provide practical voting methods for societies becoming more and more mobile. On the other hand, a voting process must guarantee the highest standards of anonymity, dependability and resistance to coercion and frauds. This is due to increasing possibilities to derive sensitive data, both through data mining and (cryptoanalytic) attacks on available election data (encrypted ballots on bulletin board, verification data, ...).

E-voting research and development projects are concentrated on two main scenarios: the first one is to use electronic machines in polling stations, the second one is to enable a voter to cast a vote over communication network using her or his own computing equipment. In the first

* contact author

scenario there might be a strong control over the equipment used, but e-voting mainly helps to process the ballots. In the second case, the voter is no longer obliged to appear in person at the polling station (sometimes he simply cannot vote for this reason). On the other hand, we are faced with a totally uncontrolled environment in which the voter casts the vote (this concerns in particular the computer used by the voter). So far, there have been a lot of efforts to implement solutions in the first scenario (e.g. in USA), remote voting was implemented in parliament elections in Estonia.

So far, practical deployment of e-voting systems has a rich history of all possible faults. Moreover, most schemes designed in the academic community do not fulfil all security demands. The most acute problem is that almost all schemes ignore the fact that electronic voting equipment should be considered as a potential adversary. On the other hand, the most dangerous in remote voting systems is the equipment on the voter's side. Voters' machines can be infected with malware that reveal the voter's preferences or even change the encrypted ballot cast by the voter.

Recently, many significantly new ideas emerged; voting schemes that allow verifiable elections in the polling stations have been proposed (see [2–4, 13, 1]). All of them present the idea of end-to-end auditable voting systems (*E2E*). These systems provide the voter receipts, which can be used to check if the voter's ballot has been included in the tally. It is also possible to verify that the results are computed correctly. On the other hand, the receipt obtained by the voter does not enable her to prove how she voted. Therefore, it is impossible to use receipts to sell or buy votes - they provide no proof that the voter behaves as required.

In this paper we concern *E2E* systems for remote voting over electronic networks. Due to lack of control over voting environment and potential coercion, constructing such systems is far more challenging than designing *E2E* voting procedures for polling stations. Moreover, there is a common belief even among IT professionals that it is impossible to provide such systems.

1.1 Related Work

Three important ideas concerning *E2E* voting systems have been presented during the last few years: Prêt à Voter, Punchscan, ThreeBallot (and related schemes). All of them are devoted for electronic, paper-

based elections at polling stations. Recently, Punchscan and Prêt à Voter have been adjusted to mail-in voting [12]. Since these methods are closely related to our scheme, we recall them briefly.

Prêt à Voter [4]. A voter, say Alice, obtains a ballot which consists of two parts. The left part contains the official list of the candidates, however after applying a circular shift by x positions, where x depends on the ballot. The right part contains boxes where Alice can put the $\times$-mark. In order to vote, she puts the $\times$-mark in the row that contains the name of her favorite candidate in the left part. On the right part, there is a kind of ballot's serial number S which is then used for decoding Alice's vote (namely, for reconstructing the shift value x). The serial number is also included in the voting receipt obtained by Alice.

Candidate	*S*
2 Jerry	
3 Edgar	
0 Ervin	
1 Donald	

(a) Prêt à Voter ballot with shift $x = 2$

Candidate	*S*
2 Jerry	
3 Edgar	
0 Ervin	×
1 Donald	

(b) A vote for candidate number 0 (Ervin)

S
×

(c) The receipt of the vote

Fig. 1. Ballot example for Prêt à Voter scheme

After making her choice, Alice separates the left and right parts. The left part goes to a shredder, while the right is scanned and entered to the system. The shift used in the ballot is committed by the value S.

Punchscan [3]. The original ballot design of Punchscan is quite different from Prêt à Voter, however it has been shown in [16] that the crucial mechanisms of Punchscan can be used together with Prêt à Voter ballot layout.

The key issue is that Punchscan offers a complete back-end to perform *E2E* verifiable elections. The values that are used in a ballot construction are committed and can be verified. The verification process is twofold and consists of a pre-election audit and a post-election audit. If

the authority responsible for preparing ballots passes both audits, then with an overwhelming probability the integrity of the elections is guaranteed.

ThreeBallot [13]. This scheme, presented by R. Rivest, is particularly appealing despite of certain privacy weaknesses [5]. A voter Alice obtains a sheet of paper which contains four parts. In the left part, there is a list of candidates (no shift is used). The next three columns are used to mark her choice. If she wants to vote for a candidate V, then she puts two marks $\times$ in the row containing the name of V, while she puts exactly one $\times$ mark in all remaining rows.

Candidate	*A*	*B*	*C*
0 Ervin			
1 Donald			
2 Jerry			
3 Edgar			

(a) An empty ballot

Candidate	*A*	*B*	*C*
0 Ervin	×		×
1 Donald			×
2 Jerry	×		
3 Edgar		×	

(b) A vote for Ervin

A
×
×

(c) Exemplary receipt

Fig. 2. The ThreeBallot scheme

After Alice makes her choice, all three columns (ballots) are separated and cast into a ballot box. Alice obtains as a receipt a copy of one of the columns/ballots of her choice (but the system does not know which one).

2 Design Goals of *SC&V*

In this paper we focus on possibility to cast votes over computer networks, when the voter cannot trust her or his computer. Moreover, like designers of the schemes like ThreeBallot, Punchscan and Prêt à Voter, we abandon cryptographic methods in the sense that it is no more a core element of the construction. This approach responds to declining strength of cryptographic algorithms that may lead to security threats in the future, potentially with a complete break of voters anonymity - disclosing

the voter's choices based on the remaining data records. Also, one has to be very careful about the use of malicious cryptography [7] which cannot be excluded as long as cryptographic protocol makes use of random numbers.

Let us mention that a solution based on cryptography has been already proposed in paper [11]. It offers some functionalities that are missing in the present proposal, however the overall construction and the security mechanism are quite involved and difficult to follow. Definitely, it is too complicated to answer the demands of an average voter.

We propose *Scratch, Click & Vote* (*SC&V*), a voting method for casting a vote over the Internet (or similar communication means) using untrusted computers that potentially may try to change the vote cast or to to betray voter's preferences.

We combine a few well known *E2E* schemes into one system. The backend of the scheme is based on Punchscan [3] in its version from [16]. In *SC&V* we modify the layout of the Punchscan's ballot and combine it with Prêt à Voter and Rivest's ThreeBallot layout. Such a combination borrows security mechanisms of all of these schemes; in particular it adjusts these methods to secure casting ballots over the Internet.

The main idea of the solution is that a voter uses the Internet and her computer as communication terminal and not as a trusted party. So in some sense we are focused on mail-in voting, where the voter's computer is an electronic post office. Compared to the traditional mail-in procedures, *SC&V* offers instant delivery confirmation and verification possibilities.

We assume that there are two authorities that do not collude. Under this assumption, *SC&V* offers verifiable Internet voting preserving privacy of a voter.

In *SC&V*, we use a two-stage verification mechanism. The first level relies on Punchscan techniques, and is necessary to convince a voter that her or his vote is in the virtual ballot box. The second level is based on an idea from the ThreeBallot and Punchscan; its purpose is to provide verfication of vote counting.

Main properties of *SC&V*. Let us state the most important properties of *SC&V*:

1. Malicious platform immunity: any misbehavior of voter's computer is detected with substantially high probability.
2. Ballots can be cast even from untrusted machines – a machine is unable to get any information about voter's choice, so it cannot betray it to a curious third party.
3. Vote selling is harder than in widely-acceptable mail-in voting: in order to sell a vote, a voter has to give away: her ballot (obtained in a non-electronic way), credentials needed to login to the election server and a record of the session of ballot casting. In particular, vote selling purely over the Internet is impossible.
4. Self-contained ballot auditing: ballots obtained by voters in a non-electronic way contain all the necessary information for auditing. Misbehavior of the voting authorities or voter's PC can be easily detected (with a high probability) by a voter. Moreover, in order to detect such a misbehavior a voter does not need to perform any mathematical calculations.

Voter's point of view. The scheme consist of steps that are simple and well known from the voter's experience:

Registering: Alice registers herself as for the regular mail-in voting. Then she obtains a set of ballots through traditional mail or she gets it from an election office. (Note that in particular the ballots are unknown to the voter's PC.)

Scratching: Alice chooses one of the ballots for voting and scratches the layer off to see the shift of the candidates.

Login: Alice enters the web page which is a electronic polling station. Both the voter and the web page are authenticated with strong methods while the connection is established.

Clicking: Alice makes her choice by clicking appropriate boxes on the web page according to the vote of Alice and to the ballot's layout.

Confirmation: Alice receives confirmation (a receipt) which can be printed. The confirmation may be used for a post-election audit.

3 Ideas Overview

The voting process is a combination of mail-in voting and paper based protocols. In order to vote one has to get additional ballot information

that remains hidden for the computer used for vote casting. This information might be obtained by the voter during voter's registration, mailed to her home address or sent over an independent electronic link. Vote casting is done via electronic networks. The (virtual) ballots used in our scheme have a new design that can be used as well for the Punchscan and Prêt à Voter schemes.

There is an *Election Authority* (*EA*) and a *Proxy*. *EA* prepares the *ballots* and *Proxy* prepares the *coding cards*. The technique used for ballot creation is based on Punchscan scheme and is described below. There is an *Auditor* which is responsible for pre- and post-election audits.

In this section we describe the scheme from Alice's (voter) point of view. We assume that there is a single race where the voter has to choose one out of m candidates (the pictures are presented for the case of $m = 4$).

Ballot layout. In order to cast a vote, Alice needs a *ballot* and a *coding card*:

A ballot is prepared by *EA*, it consists of the following values covered by a scratch surface:

- a list of the candidates shifted circularly by x positions, (later we shall represent x as $x = x' + x''$ mod m where x', x'' are random),
- the ballot serial number S_l,
- four *confirmation tokens*: A, B, C, D – one per column, they are prepared in a special way that will be described below.

So a ballot can look as follows (for this ballot we have shift value $x = 2$):

Candidate	A B C D
2 Jerry	
3 Edgar	
0 Ervin	
1 Donald	
S_l	

So, apart from the left hand side that contains A, B, C, D (and four empty columns instead of one), the ballot design is borrowed from Prêt à Voter.

A coding card: it is prepared by *Proxy* and consists of:

– four columns. In each row there is exactly one mark `Y` standing for YES, and 3 marks `n` standing for NO. The placement of `Y` in each row is random and independent from other choices.
– coding card serial number S_r

This is an example of a coding card:

	n Y n n
	n Y n n
	Y n n n
	n n n Y
	S_r

On the picture we can see that a coding card is a *right part* corresponding to the *left part* formed by a ballot. If we put them together, we obtain a complete ballot ready to cast a vote as shown on the next picture below.

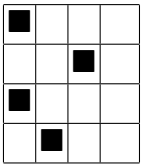
Ballot casting. After obtaining both: a ballot and a coding card, Alice lays them side by side and thus obtains a complete ballot. Let us note that Alice gets exactly one ballot, but she is allowed to have as many coding cards as she likes. Moreover, we assume that there are many Proxies in the system, so Alice can easily find one she trusts and get coding cards from this Proxy. A complete ballot (which Alice may put on her desk) may look as follows:

Candidate	*A B C D*
2 Jerry	n Y n n
3 Edgar	n Y n n
0 Ervin	Y n n n
1 Donald	n n n Y
S_l	S_r

Now Alice visits an election website operated by a *Proxy*. She enters coding card serial number S_r and then clicks on the screen in the following way:

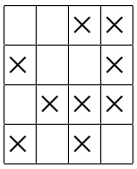
– she clicks on the position of `Y` in the row corresponding to the candidate that she votes for,
– in each remaining row, she clicks on one of the positions of `n`'s.

So in particular, if Alice votes for Ervin she may click the following positions on the screen:



S_r

Proxy checks S_r and then transforms the ballot into a *ballot matrix* in the following way: *Proxy* puts × mark for each `n` which have not been used yet (this transformation depends deterministically on the positions of `Y`'s and `n`'s and the voter's choice). So, for a row with the candidate chosen by the voter *Proxy* puts three × marks, while in each row corresponding to different candidates, there are only 2 marks. In the case of our example it looks as follows:



Note that *Proxy* knows which row corresponds to the vote cast. On the other hand, due to the random shift *Proxy* does not know which candidate is corresponding to this row.

Then the columns of the ballot matrix, called *ballot columns*, are processed separately (analogously to ThreeBallot). Then *Proxy* obtains a blind signature (BS) of *EA* under each ballot column. (A blind signature is necessary in order to prevent changing the ballot contents by *EA* at this moment). The voter enters ballot serial number (S_l), then *Proxy* unblinds the signature, and sends ballot columns with S_l to *EA*. Simultaneously, the voter requests one ballot column as a receipt - for this purpose 1 from 4 oblivious transfer (OT) protocol is executed between the voter and *EA* via *Proxy*. The receipt contains:

- $T \in \{A, B, C, D\}$ a value of a confirmation token;
- y - a ballot column,
- t such that $T = \text{sign}_{EA}(t, S_l)$, such a t is called a pre-token of T.

Below we show an example of a receipt in case when the third column has been chosen, and $C = \text{sign}_{EA}(c, S_l)$:

C ,  , c

The ballot columns are now separated and are published just like for Punchscan scheme, and then decrypted in a similar way. The number of votes for each candidate is counted like for ThreeBallot. Also, there is verification procedure checking correctness of vote counting and presence of all ballots cast. In the next sections we describe details of these procedures.

4 *Scratch, Click & Vote* Scheme

4.1 The Ballots and Audit Tables

The ballots are created by *EA*. In order to guarantee election integrity, *EA* generates audit tables P and R. The table R is similar to Punchboard used in Punchscan. Each row of table P corresponds to a single ballot matrix (which is a set of columns with the same ballot serial number and the shift). Entries of R correspond to single ballot columns. Below we describe details of their construction.

Table *P*. The table P has 2 columns, called P1 and P2. It has $2n$ rows, where n is greater or equal to the maximum number of voters.

P1	P2
⋮	⋮
$S_l(i)$	$BC(i_A, i_B, i_C, i_D)$
⋮	⋮

Example: Audit table P

The column P1 records the ballot serial numbers. The second column, called P2, contains commitments to 4 pointers to the rows of table R. Say, if a serial number S is in P1, then in the same row P2 contains a commitment

$$BC(i_A(S),\ i_B(S),\ i_C(S),\ i_D(S))$$

to numbers

$$i_A(S),\ i_B(S),\ i_C(S),\ i_D(S)\ .$$

where $i_X(S)$ is the row number such that row $i_X(S)$ of table R contains an entry for the column X of the ballot with the serial number S.

Table *R*. The table R contains three parts: the *starting part*, the *middle part* and the *final part*. Each part consists of a set of consecutive columns.

Each row in the starting part is devoted to a single ballot column of some ballot. Let $W(i)$ denote the column corresponding to this ballot column for row i of R. For constructing R two random and secret permutations are used, we call them π_1 and π_2. Then for each i, data concerning $W(i)$ are:

- in row i in the starting part,
- in row $\pi_1(i)$ in the middle part,
- in row $\pi_2(\pi_1(i))$ in the final part.

Moreover:

- the starting part of row i will contain the ballot column $W(i)$ as filled by the voter,
- the middle part at row $\pi_1(i)$ will contain $W(i)$ shifted circularly by $-x' = -x'(i)$ positions,
- the final part at row $\pi_2(\pi_1(i))$ will contain $W(i)$ shifted circularly by $-x(i) = -x'(i) - x''(i)$ positions. Hence the marks of $W(i)$ will be shifted back $x(i)$ positions so that they correspond to the standard ordering of candidates.

Below we describe a row i of R. Let $j = \pi_1^{-1}(i)$ and $k = \pi_1^{-1}(\pi_2^{-1}(i))$.

starting part (for $W(i)$)						middle part (for $W(j)$)				final part (for $W(k)$)
i	$\widehat{x(i)}$	$H(t(i))$	$\widehat{t(i)}$	$y(i)$	$\widehat{\pi_1(i)}$	$\widehat{x'(j)}$	$y(j) \ll x'(j)$	$\widehat{x''(j)}$	$\widehat{\pi_2(i)}$	v

Organization of a row of table R

The starting part contains the following entries in row i (see the diagram above):

- i - the row index ($i \in [1, 8n]$)
- $\widehat{x(i)}$ - a bit commitment to the shift $x(i)$ used in the ballot containing $W(i)$,

– $H(t(i))$ - hash of a *confirmation pre-token* $t(i)$, which satisfies the condition

$$T(i) = \mathrm{sign}_{EA}(t(i), S_l(i)) \,,$$

where $T(i)$ is the confirmation token used in conjunction with $W(i)$, and $S_l(i)$ is the serial number of ballot containing $W(i)$.
– $\widehat{t(i)} = BC(T(i), S_i)$ - a bit commitment to:
  - the ballot serial number $S_l(i)$ of the ballot containing $W(i)$, and
  - the confirmation token $T(i)$,
– $y(i) = [y_0(i), y_1(i), \ldots, y_{m-1}(i)]$ - a vector holding 1 on each position l such that $W(i)$ contains the $\times$ mark in row l. Initially, during creation of table R, the vector $y(i)$ is empty. It becomes filled after casting a vote using $W(i)$.
– $\widehat{\pi_1}(i)$ - a commitment to the value $\pi_1(i)$.

The middle part of R in row i contains the following entries:

– $\widehat{x'(j)}$ - a commitment to $x'(j)$, where $x(j) = x'(j) + x''(j) \bmod m$,
– $y(j) \ll x'(j)$ - the vector $y(j)$ shifted circularly by $-x'(j)$ positions,
– $\widehat{x''(j)}$ - a commitment to $x''(j)$,
– $\widehat{\pi_2(i)}$ - a commitment to $\pi_2(i)$.

The final part of R contains $y(k) \ll x(k)$.

4.2 Preparation of Ballots and Audit Tables

The ballots and the audit tables P and R are created by EA in the following way:

1. EA determines the election parameters: the number of candidates m, the official list of candidates (with their official ordering), and an upper bound n on the total number of voters.
2. EA chooses at random $2n$ serial numbers; for each serial number S:
   – EA chooses at random a shift $x(S) < m$,
   – EA chooses at random four confirmation pre-tokens $t_A(S)$, $t_B(S)$, $t_C(S)$, $t_D(S)$ and computes confirmation tokens $T_A(S)$, $T_B(S)$, $T_C(S)$, $T_D(S)$:

$$T_X(S) := \mathrm{sign}_{EA}(S, t_X(S)) \,.$$

3. *EA* creates audit table P: For this purpose, *EA* chooses at random a permutation ρ of $1, \ldots, 8n$. Then $\rho(4j-3), \ldots, \rho(4j)$ are assigned to the jth serial number $S_l(j)$. These numbers serve as pointers to the rows of the audit table R - and are called $i_A(S_l(j)), i_B(S_l(j)), i_C(S_l(j)), i_D(S_l(j))$.
   Then for each serial number S_l, a commitment to the values $i_A(S_l(j)), i_B(S_l(j)), i_C(S_l(j)), i_D(S_l(j))$ is created and inserted in the row containing $S_l(j)$.
4. *EA* prepares the audit table R: For this purpose *EA* chooses random permutations π_1, π_2 of $1, \ldots, 8n$. For each $S_l(j)$, the shift value $x(S_l(j))$ is assigned to the rows $i_A(S_l(j)), i_B(S_l(j)), i_C(S_l(j)), i_D(S_l(j))$ of the starting part of R. Separately for each row of R, *EA* chooses at random values x' and x'' that sum up to the shift value x of this row: $x = x' + x'' \bmod m$.
5. Then the entries of R are filled according to the description from the previous subsection.

Finally, the ballots are printed so that their contents (the shift on the list of names, confirmation tokens and serial numbers) is hidden under a scratch layer.

4.3 The Pre-election Audit

As for Punchscan, the following steps are executed in order to check that the audit tables have been created honestly:

1. The Auditors pick at random a set *AS* of n ballots. The remaining ballots create so called election set *ES* (and are not checked).
2. The contents of all ballots from *AS* is revealed, so in particular their serial numbers. Based on the serial numbers we can indicate the rows of P corresponding to the ballots from *AS*.
3. *EA* opens all bit commitments from table P corresponding to the ballots from *AS* as well as all bit commitments from table R corresponding to the ballot columns of the ballots from *AS*.
4. The Auditors check that the ballots and the entries in the audit tables were created honestly.
5. All ballots from the audit set *AS* are discarded; the ballots with serial numbers in *ES* are used for election.

In practice, the Auditors may confine themselves to controlling only a limited number of ballots from *AS*, and check more ballots on demand.

4.4 Preparing Coding Cards

The coding cards are prepared in electronic form and published (as commitments) on a webpage by *Proxy*. Their correctness is checked in a standard way:

1. *Proxy* creates an audit table X in which it commits coding card serial numbers S_r and positions of `Y`-marks on each coding card.
2. The Auditors select at random some number of coding cards to an audit set (these coding cards are not used for elections).
3. *Proxy* opens all bit commitments from the cards of the audit set.
4. The Auditors check, if coding cards revealed have been created honestly.

4.5 Elections

The following steps are executed by a voter:

1. A voter receives a ballot (e.g. by visiting certain authorities, from a special courier delivering the ballots at residence area, by certified mail services, ...). At the same time, identity of the voter is verified in a non-electronic way and the ballot is given to her or his own hands. The ballots are not transmitted electronically, so in particular they do not go through any computer used for casting votes. Distribution of ballots is organized so that nobody knows who gets which ballot. Since the ballot information is covered with a scratch surface, this is not particularly hard to achieve.
2. The voter obtains coding cards from one or more Proxies. Here, a secure electronic link (anonymous and encrypted, e.g. like in [10]) may be used. Link protection has to prevent *EA* from seeing the coding cards obtained by a voter in the plaintext.
3. The voter visits an election webpage run by a *Proxy*:
   (a) she peels-off the scratch-layer from the ballot,
   (b) she chooses one of the coding cards at hand and lays it next to the ballot,
   (c) she enters S_r from the coding card chosen, clicks on the screen on radio buttons corresponding to her choice –that is, according to the shift used for the ballot and alignment of `n`'s and `Y` marks on the coding card.

A *Proxy* executes the following steps:

1. it transforms the voter's choice into ballot columns,
2. it obtains a blind signature from *EA* under each of the ballot columns (these signatures are then stored by *Proxy* for a post-election audit),
3. it asks the voter for S_l,
4. it passes S_l and the ballot columns to *EA*.

The *EA* executes the following steps:

1. it enters obtained ballot columns into appropriate rows of the starting part of the table R (but publishes them when the election are closed),
2. it executes the oblivious transfer protocol in order to send a receipt of a ballot column chosen by the voter. All messages between the voter and *EA* are processed via *Proxy* and anonimized.

4.6 Tallying

1. When the voting time is over, *EA* publishes voter's choices by inserting them in $y(i)$ in the starting part of the table R. Then it computes the entries for the middle part of R: $y(j) \ll x'(j)$, and for the final part: $v = y(k) \ll x(k)$.
2. From the entries v in the final part *EA* calculates the tally: If the number of ballot columns is $4k$ (meaning that k votes have been cast) and there are together M marks $\times$ in row j of all ballot columns in the final part of R, then the number of votes cast for the candidate j is $M - 2k$.

4.7 Post-Election Audit

First, each voter can check if her or his ballot column corresponding to the receipt appears in the table R. This is possible, since due to knowledge of the verification pre-token t , one can locate the right row containing $H(t)$. If it is missing or the contents of the ballot column disagrees with the receipt, then a fraud is detected.

Checking integrity of table R and the election results is performed in public by the auditors. For this purpose the standard Randomized Partial Checking [9] procedure is executed for R :

1. the auditors choose $2n$ rows of R at random and ask *EA* to open commitments $\widehat{\pi_1(i)}$ in these rows. Then we get $2n$ out of $4n$ values of the permutation π_1 (for the sake of simplicity assume that n voters participated in the elections). Then for each row $\pi_1(i)$ in the middle part, for which $\pi_1(i)$ has been revealed, the commitment $\widehat{x'}$ is opened and it is checked that the ballot column from the starting part shifted by $-x'$ yields the ballot column in the middle part.
2. For each row j in the middle part, where no commitment has been opened during the first step, *EA* has to open the commitments $\widehat{\pi_2}$ and $\widehat{x''}$. Then the ballot columns are checked for consistency with x''.

5 Security Concerns

We focus now on security of *SC&V* scheme. In the next three subsections we discuss what happens if one of the parties of the protocol does not behave honestly.

5.1 Alice's PC misbehavior

Here we assume that the Alice's PC is dishonest, while *EA* and *Proxy* behave correctly. This corresponds to the case when Alice's PC is infected by malware.

In order to manipulate voter's choice (change from Alice's choice to a random candidate), the PC has to switch Alice's choice from `Y` into `n` in the row corresponding to the candidate chosen by Alice and at the same time, change `n` into `Y` in one of the remaining rows (for an example see Fig. 3). In order to do that, PC has to guess which row corresponds to the chosen candidate and succeeds with probability $\frac{1}{k}$. Then, the PC has to choose one of the remaining rows and guess which one of the unchosen three columns corresponds to the mark `Y` – this succeeds with probability $\frac{1}{3}$. So the total success probability of correct switching Alice's choice is only $\frac{1}{3k}$.

Even if the PC succeeds, Alice can detect a fraud by discovering that her receipt does not fit her choice. Let us notice that at least one of the ballot columns is modified during such a change (and sometimes it is just one column - see Fig. 3).

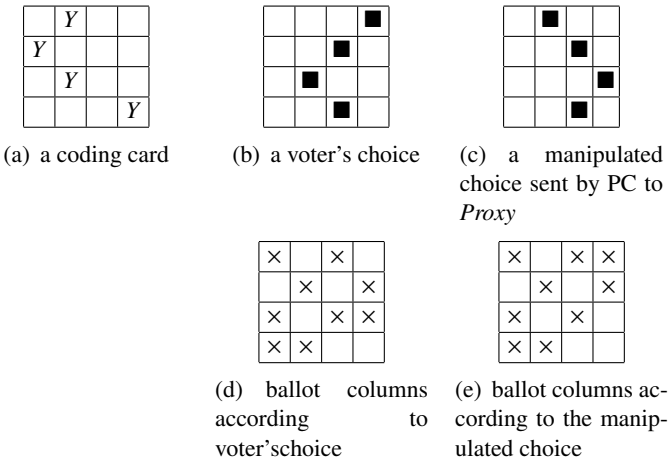


Fig. 3. PC can modify Alice's choice so that only one column of the ballot columns differ from the original choice.

5.2 *EA* misbehavior

Now assume that *EA* is dishonest, but the PC of Alice and *Proxy* are honest. The possibility of *EA*'s misbehavior in ballots' preparation and counting is limited by the pre- and post-election audits just like in Punchscan.

Replacing ballot columns when inserting them to table R is risky, since finally a voter gets a receipt of one of her four the ballot columns. If the receipt disagrees with the contents of table R, then one can catch *EA* since the same ballot column has been signed blindly by *EA*. Note that due to the hash value of pre-token, the voter can prove which entry in the starting part of R corresponds to her ballot column from the receipt.

5.3 *Proxy*'s misbehavior

Now we assume that *Proxy* is dishonest, while *EA* and the PC of Alice behave correctly. The assignments of `Y`'s, `n`'s are known to *Proxy*, so *Proxy* knows the row corresponding to the voter's choice. However, *Proxy* does not know the shift used in the ballot.

Proxy can change the voter's choice and create valid ballot columns. However, at least one column must be changed. So with probability at least $\frac{1}{4}$ the column chosen by *EA* as a receipt will reveal the fraud (prob-

ability of undetected modification of 10 votes is $< 5.7\%$, 20 votes – $< 0.032\%$).

If *Proxy* changes S_l in order to change the shift then Alice obtains a different confirmation token than it is stated on her ballot. Thus it will be detected immediately by Alice.

5.4 Attacking ballot's secrecy

***EA*'s point of view.** The situation is like for Punchscan: If *EA* knows which ballot was used by Alice, then *EA* knows the vote of Alice. So it is crucial to apply appropriate procedures of ballot distribution. Keeping the sensitive information under scratch surface helps a lot - the ballots can be mixed before distributing and become indistinguishable. Also, it is crucial that a voter never sends ballot information directly to *EA* - all communication goes through *Proxy*.

***Proxy*'s point of view.** In Sect. 6 we propose a simple modification of the basic that leads to full ballot secrecy even if *Proxy* communicates directly with a voter and knows voter's identity. Here, we describe threat to the voter's privacy when the scheme is not implemented with modification from Sect. 6.

During a voting process, *Proxy* knows how each voter filled the ballots - *Proxy* knows assignments of $\times$-marks in each column, but still does not know the shift. *Proxy* might try to link voter's ballot columns, call them A, B, C, D, appearing on the *starting part* of the R audit table with ballot columns on the *final part* of R. The advantage of *Proxy* is that it knows that the ballot columns A, B, C, D belong to a ballot of the same voter (even if it is unknown which positions they take in R).

Let $S(X)$ be a set of columns which can be obtained by applying circular shifts to ballot column X. Note that $S(X)$ is an equivalence class which is called a k-bead necklaces with 2 colors [6]. The number of k-bead necklaces with 2 colors equals (see [6]):

$$\mathcal{S}(k) = \frac{1}{k} \sum_{d|k} \varphi(d) 2^{\frac{k}{d}} ,$$

where $\varphi(k)$ is the Euler quotient function, i.e. the number of integers $i \in \{1, \ldots, k\}$ that are relatively prime to k.

The problem arises when for a voter's ballot column X, no other ballot column from $S(X)$ appears in the starting part of R. Then the final part of R contains only one element of $S(X)$, namely a shifted version of X. It reveals the shift used for X (or a few possible values, if X is periodic) and thus the voter's choice (note that all four ballot columns of the voter are known by *Proxy*).

In order to at least confuse the malicious *Proxy* we need at least one other ballot column in $S(X)$. Since we do not know X of Alice in advance, we need at least to make sure that each equivalence class contains some ballot columns in the set of ballot columns cast. This might be problematic: If we assume that assignments of $\times$-marks are independent at random, then by the coupon collector problem, the minimum number of voters needed is equal to:

$$V(k) = \mathcal{S}(k)\ln(\mathcal{S}(k)) \approx \frac{2^k}{k}(\ln(2^k) - \ln(k)).$$

The values from the above formula can be summarized as follows:

k	2	3	4	5	6	7	8	9	10	11	12	13	14	15	20
$V(k)$	3	6	11	17	37	60	129	246	506	984	2 064	4 076	8 363	16 862	570 457

The meaning of the above results is that in practice we are severely confined in using this method as long as proxy has all described information.

The simplest way to avoid this situation is the use of anonymous communication between a voter and *Proxy*, like proposed in [10]. Then, even if *Proxy* can link ballot columns, it does not know voter's identity.

External observer's point of view. Here, we assume that the observer is not physically present during casting a vote and does not control the PC used by the voter. We assume that a voter casts a vote and then passes to the observer (e.g. by mail or fax): the ballot, the coding card, the receipt and informs which fields have been clicked. Of course, the observer has access to the bulletin board.

The key point is that voter could use a different coding card that she has shown the observer. So, the situation of the observer is much different from the situation of a *Proxy*: she obtains only one ballot column (receipt) and cannot be sure if the coding card obtained was really used.

If the observer somehow guesses the shifts, then we consider anonymity for a design that can be called *FourBallot* and which is an extension of

ThreeBallot. Anonymity issues for ThreeBallot have been investigated in [5, 8, 14, 15]. There are the following key techniques in attacking voters' privacy in ThreeBallot:

Reconstruction: the idea of the attack is that if two ballots have two rows with two $\times$ marks in each, then they certainly do not come from the same vote. This bounds the number of admissible combinations and, for a sufficient number of candidates, reveals unique admissible combinations of ballots. FourBallot is more immune for such an attack: any two ballot columns may come from the same vote.

Skewed probabilities: the conditional probability that a given ballot column occurs in a vote for candidate X depends on X. Compared to ThreeBallot, these probabilities are more uniform (shift of the candidates is used, moreover probability that in given row there is a $\times$-mark is equal to $\frac{2k+1}{4k} \approx \frac{1}{2}$, while in ThreeBallot this probability is equal to $\frac{2k+1}{3k} \approx \frac{2}{3}$), therefore the stochastic advantage of the observer is lower.

Requesting characteristic ballots: on request, a voter uses special patterns of $\times$ marks that can be later identified on the bulletin board. However, in case of *SC&V* it is harder to demand the patters - the voter is confined by the coding cards. The voter can also examine many coding cards and use one of them, in order to vote freely and fulfill the requirements. Also two voters may collude to provide together the ballot columns requested - but coming from two votes instead of one.

5.5 Vote selling

In case of mail-in voting, a vote buyer has to be present during inserting a ballot into an envelope. A voter can also make a copy of a ballot, but the buyer cannot be sure that the copy has not been manipulated. Alternatively, the buyer should get the whole paper ballot.

In case of *SC&V* a voter is identified electronically by a *Proxy*. The identification protocol should guarantee that the voter would not risk to transmit her electronic identity to the buyer. (In this way electronic voting becomes superior over mail-in procedures, for which transferring a ballot to a buyer is inevitable.) This holds for instance, if the voter is using an electronic ID card or ID codes that are used also for other purposes (like submitting a tax declaration).

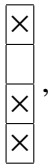
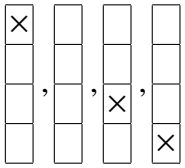
Even if the voter casts the vote herself, she can record the whole voting session and present it to the buyer together with the ballot and the coding card used. The ballots and coding cards have a non-electronic, paper form, so they can be presented to the remote buyer as electronic copies. However, the scan of the ballot can be digitally manipulated and the coding card presented need not to be the one used.

The things become more complicated for the buyer if the authentication protocol is based on a zero-knowledge protocol – then the buyer cannot be even sure that the voter is casting a vote unless he is controlling directly the PC of the voter.

The only thing that the buyer can be convinced about is the receipt and the matching entries in the bulletin board. However, at this moment we fall back to the case of the external observer considered above.

6 Advanced Version of Unlinkability

Here we propose the final solution solving the problems that have been mentioned in Sect. 5.4. The idea is that in R table a single ballot column is not represented by a single entry in the starting part, but is split into k different *masked ballot columns* in different rows of R. See an example of a ballot column and its masked versions:

a ballot column:  , masked ballot columns: 

Simply, the ith masked ballot column contains no $\times$ mark except for the row j, provided that the original ballot column contains a $\times$ mark in the row j. Let X be a ballot column and t be its pre-token. Then the jth masked ballot column for X in the starting part of R is marked by the value $H(t, j)$ (instead of $H(t)$, as it was for the first design). This enables the voter to check the entries of the bulletin board as before - however the voter has to look for k different hashes and k rows instead of one.

Checking integrity of R table is performed just as before, as well as vote counting: the number of $\times$ marks does not change, we have only to note that the number of votes is now the number of rows of R divided by $4k$.

How do the masked ballot columns help to preserve anonymity? The key observation is that

> the number of masked ballot columns of each kind is determined **uniquely** by the election result.

Therefore R table provides **no** additional information. So the Strauss' attack and any other attack based on the particular choice of ballot columns fails.

7 Final Remarks

We have presented an Internet voting scheme based on the ideas of Punchscan, ThreeBallot and Prêt à Voter schemes. The scheme allows for secure vote casting over the Internet. A voter cannot prove how she voted unless vote-casting is physically supervised by an adversary.

On the other hand, the scheme presented allows voter to check if her ballot is indeed included in a tally – which is impossible with most mail-in voting schemes. The scheme does not support revocation possibility.

We also provide a patch to ThreeBallot design that makes it resistant to anonymity threats pointed to in many papers.

References

1. Ben Adida and Ronald L. Rivest. Scratch & vote: Self-contained paper-based cryptographic voting. In *WPES '06: Proceedings of the 5th ACM workshop on Privacy in electronic society*, pages 29–40, New York, NY, USA, 2006. ACM.
2. David Chaum. Secret-ballot receipts and transparent integrity. better and less-costly electronic voting and polling places. In *Security & Privacy*. IEEE Computer Press, 2004.
3. David Chaum. Punchscan, 2005. `http://www.punchscan.org`.
4. David Chaum, Peter Y. A. Ryan, and Steve Schneider. A practical voter-verifiable election scheme. In *ESORICS*, volume 3679 of *Lecture Notes in Computer Science*, pages 118–139. Springer Verlag, 2005.
5. Jacek Cichoń, Mirosław Kutyłowski, and Bogdan Węglorz. Short ballot assumption and threeballot voting protocol. In *SOFSEM 2008: Theory and Practice of Computer Science*, volume 4910 of *Lecture Notes in Computer Science*, pages 585–598. Springer Verlag, 2008.
6. Philippe Flajolet and Robert Sedgewick. Analytic combinatorics, 2008.
7. Marcin Gogolewski, Marek Klonowski, Mirosław Kutyłowski, Przemyslaw Kubiak, Anna Lauks, and Filip Zagórski. Kleptographic attacks on e-voting schemes. In *Emerging Trends in Information and Communication Security*, volume 3995 of *Lecture Notes in Computer Science*, pages 494–508. Springer Verlag, 2006.
8. Kevin Henry, Douglas R. Stinson, and Jiayuan Sui. The effectiveness of receipt-based attacks on threeballot. Cryptology ePrint Archive, Report 2007/287, 2007. `http://eprint.iacr.org/`.

9. Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *USENIX Security Symposium*, pages 339–353, 2002.
10. Marek Klonowski, Mirosław Kutyłowski, and Anna Lauks. Repelling detour attack against onions with re-encryption. In *Applied Cryptography and Network Security (ACNS)*, Lecture Notes in Computer Science. Springer Verlag, 2008.
11. Mirosław Kutyłowski and Filip Zagórski. Verifiable internet voting solving secure platform problem. In *Advances in Information and Computer Security*, volume 4752 of *Lecture Notes in Computer Science*, pages 199–213. Springer Verlag, 2007.
12. Stefan Popoveniuc and David Lundin. A simple technique for safely using punchscan and pret a voter in mail-in elections. In *VOTE-ID 2007*, volume 4896 of *Lecture Notes in Computer Science*, pages 150–155. Springer Verlag, 2007.
13. Ronald L. Rivest and Warren D. Smith. Three voting protocols: Threeballot, vav, and twin. In *EVT'07: Proceedings of the USENIX/Accurate Electronic Voting Technology on USENIX/Accurate Electronic Voting Technology Workshop*, pages 16–16, Berkeley, CA, USA, 2007. USENIX Association.
14. Ch. Strauss. A critical review of the triple ballot voting system. part 2: Cracking the triple ballot encryption. 2006. `http://www.cs.princeton.edu/~appel/voting/Strauss-ThreeBallotCritique2v1.5.pdf`.
15. Ch. Strauss. The trouble with triples: Acritical review of the triple ballot (3ballot) scheme, part 1. 2006. `http://www.cs.princeton.edu/~appel/voting/Strauss-TroubleWithTriples.pdf`.
16. Jeroen van de Graaf. Merging pret-a-voter and punchscan. Cryptology ePrint Archive, Report 2007/269, 2007. `http://eprint.iacr.org/`.