Scratch, Click & Vote: E2E voting over the Internet

Mirosław Kutyłowski[⋆] and Filip Zagórski [⋆⋆]

Institute of Mathematics and Computer Science
Wrocław University of Technology
mirekk@im.pwr.wroc.pl filipz@im.pwr.wroc.pl

Abstract. We present *Scratch, Click & Vote* remote voting scheme. The scheme is end-to-end verifiable and allows for voting over the Internet. It guarantees security against malicious hardware and software used by a voter; a voter's computer does not get any knowledge about the voter's choice. Moreover, it can blindly change the voter's ballot with a small probability only.

Keywords: Internet voting, e-voting, E2E, verifiable voting scheme, ThreeBallot, Punchscan

1 Introduction

There are two main scenarios of e-voting: advanced voting procedures at polling places and remote electronic voting.

Polling station voting Recently it has became evident that badly designed e-voting machines can be extremely dangerous to a voting process [1, 9, 13]. Fortunately, a number of end-to-end auditable voting systems (*E2E*) has been presented recently. Interestingly, some recent designs implement *electronic voting* without any electronic voting machines [2, 6, 5, 4]. Moreover, for these schemes each voter gets a receipt, which may be used to check if the voter's ballot has been included in the tally. It is also possible to verify correctness of the results. On the other hand, the receipt cannot be used even by the voter to prove how she voted. So they cannot help to sell or buy votes.

Internet voting The Internet voting has not much in common with polling station scenario. At the polling station it is relatively easy to preserve voter's privacy; coercion and vote buying is hard to hide. Another source of problems is that a voter must use some electronic device. There is no convincing argument why a voter should blindly trust this device. Malware can endanger integrity of the elections as well as privacy of the voter.

⋆ Partially supported by Polish Ministry of Science and Higher Education, grant N206 2701 33

⋆⋆ Partially supported by Foundation of International Education (FEM), programme Lower Silesia Research Grants, project *Electronic Identification*

In case of remote voting one has also to deal with remote voter identification. Fortunately, it can be solved in many ways, depending on a situation. For national elections one can use advanced electronic signatures, especially if supported by personal, government-issued ID cards, or a novel technique described in [3]. For other elections logins and passwords seem to serve well their purpose.

In this paper we are concerned with an *E2E* systems for remote voting over electronic networks. We assume that the electronic devices used by a voter might be infected by malicious code, and that voter's privacy and election integrity must be guaranteed in a verifiable way.

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 dedicated to paper-based elections at polling stations. Recently, Punchscan and Prêt à Voter have been adjusted to mail-in voting [15]. Since these methods are closely related to our scheme, we recall them briefly.

Prêt à Voter [6]. A voter, say Alice, obtains a ballot which consists of two parts. The left part contains the official list of the candidates, altered by 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 on the left side. On the right side there is a kind of ballot serial number S that is 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. After making

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	$\times$
1 Donald	

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

S
$\times$

(c) The receipt of the vote

Fig. 1. Ballot example for Prêt à Voter scheme

her choice, Alice separates both parts. The left part goes to a shredder, while the right part is scanned and entered to the system.

Punchscan [2]. The original ballot design of Punchscan is quite different from Prêt à Voter, however it has been shown in [19] 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. Similar back-end is also used in Scantegrity [5] and Scantegrity II [4]. The values that are used in the 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 [16]. This scheme, presented by R. Rivest, is particularly appealing despite of certain privacy weaknesses [7]. A voter, Alice, obtains a sheet of paper consisting of four parts. The leftmost column contains the 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. After Alice makes her choice, all three columns (ballots) are separated and cast

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

(a) An empty ballot

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

(b) A vote for Ervin

A
$\times$
$\times$

(c) Exemplary receipt

Fig. 2. The ThreeBallot scheme

into a ballot box. As a receipt Alice obtains a copy of one of the columns/ballots of her choice (but the system does not know which one).

Internet voting schemes. So far, the schemes designed by the academic community do not fulfill 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 ([12, 14]). Meanwhile, potentially 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.

1.2 Our Contribution

Design goals. The main goal behind design of *SC&V* is to get an *E2E* scheme that would be acceptable for casting votes over Internet. Unlike in the previous works, we demand all key requirements to be satisfied. In particular, the scheme must be secure without assuming that the voter's PC or any of its components is trustworthy. We present the scheme which is:

human verifiable: a receipt obtained by a voter is human-readable and easy to examine by a moderately educated voter,

voter friendly: a voter needs not to perform any complicated (and hard to understand by an average voter) operations like: re-encryption, getting a blind signature, executing oblivious transfer protocol etc.

malware immune: integrity of the elections and privacy of votes do not rely on any assumption on trustworthiness of the equipment used by the voter,

efficient: computational overhead as well as communication volume are low.

Scheme design. In the next sections we provide a full description of *SC&V*. The scheme can be described as a layered design in which we combine a number of techniques/tricks:

Version 1: We start from a straightforward Internet-version of the ThreeBallot: a voter needs to fill in a voting card using an application run on her PC. The filled ballot is then sent to a voting server (*Proxy*).

Version 2: In order to balance the number of $\times$ marks in the columns (in order to make the scheme immune against Strauss'-like attacks [7, 10, 17, 18]) we add another column and another $\times$-mark ("FourBallot").

Version 3: Since the voter's PC knows exactly the voter's choice – we introduce a coding card which is prepared by a *Proxy* (see the diagram below). The coding card hides from the PC the meaning of the voter's choices (every candidate is "clicked" exactly once). Moreover coding card is constructed in a way that the possibility of modification of voter's choice by her PC is reduced.

Version 4: Since *Proxy* still knows the voter's choice, we introduce another server – Election Authority (EA), which is responsible for preparation of ballots. In this way *Proxy* does not know the choice, while Election Authority does not know who voted.

Election Authority prepares voting cards in a similar way as for the Punchscan scheme. Together with the ThreeBallot mechanisms this ensures verifiability of the voting process.

Under the assumption that *Proxy* and *Election Authority* do not collude, *SC&V* offers verifiable Internet voting with unconditional integrity and full privacy of a voter.

2 Ideas Overview

2.1 Ballots and coding cards

The voting process is a combination of paper based protocols. In order to vote one has to get additional information that remains hidden from the computer used for vote casting. This information might be obtained by the voter during voter's in-person registration, mailed to her home address or sent over an independent electronic link. In case of small scale elections or elections of limited importance one can use emails with CAPTCHA to sent information readible for a human voter, but hard to read by the voter's computer. Vote casting is done via electronic networks.

There is an *Election Authority* (*EA*) and a *Proxy*. *EA* prepares the *ballots* while *Proxy* prepares *coding cards*. There is an *Auditor* which is responsible for pre- and post-election audits.

In this section we describe the scheme from the point of view of a voter Alice. For the sake of simplicity of exposition we assume that there is a single race where the voter has to choose one out of m candidates (the pictures presented below depict the case of $m = 4$).

Ballot layout. In order to cast a vote, Alice needs a *ballot* and a *coding card*. The **ballot** is prepared by *EA*, it consists of the following values covered by a scratch surface:

- list of candidates permuted with some random permutation π. Later we shall represent $\pi = \pi' \circ \pi''$ where π', π'' are random.
- 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.

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

Fig. 3. A ballot with $\pi(i) = i + 2$

The **coding card** 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 the other choices.
- coding card serial number S_r

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

Fig. 4. A coding card

2.2 Voter's Point of View

Alice obtains both: the ballot and the coding card[1]. 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[2], so Alice can easily find one she trusts and gets coding cards from this Proxy. A complete ballot (which Alice may put on her desk) may look as follows:

complete ballot

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

PC screen

■			
		■	
■			
	■		

S_r

ballot matrix

		×	×
×			×
	×	×	×
×		×	

S_l

receipt

$C, \begin{array}{|c|}\hline \times \\ \hline \\ \hline \times \\ \hline \times \\ \hline \end{array}, c$

Alice visits an election website operated by the *Proxy*. She authenticates herself with appropriate authentication method (login/password pair, electronic signature etc). She 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 of the remaining rows, she clicks on one of the positions of `n`'s.

The *Proxy* commits to Alice's clicks (the commitment is passed to *EA*), then Alice enters coding card serial number S_r. The *Proxy* checks S_r and then transforms the choice of the voter into an internal form called *ballot matrix*: *Proxy* puts × mark for each `n` which has 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. Note that *Proxy* knows which row corresponds to the vote cast. On the other hand, due to the random permutation *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). In the next step *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. The receipt contains:

[1] Alice obtains it at registration office, by mail or by email, depending on election settings.

[2] Moreover, a "decoy service" can be introduced – then Alice may obtain many different but fake coding cards with the same serial number – in order to cheat a coercer or a vote-buyer.

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

The ballot columns are now separated and 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.

3 *Scratch, Click & Vote* Scheme

The Ballots and Audit Tables The ballots are created by EA. In order to guarantee election integrity, EA generates audit tables P and R (see below for details). 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 permutation of the candidates). Entries of R correspond to single ballot columns. R is based on the same idea as Punchboard used in Punchscan.

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

Example: Audit table P

The column P1 records the ballot serial numbers. The column 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, column P2 contains commitments

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

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 consists of three parts: the *starting part*, the *middle part* and the *final part*. Each part consists of a set of consecutive columns. R has $8n$ rows; this corresponds to $2n$ ballots and $4 \cdot 2n$ ballot columns. There are two types of permutations used for constructing table R:

– ρ_1, ρ_2: permutations of rows of the R (i.e. permutations over $\{1, \ldots 8n\}$),

- permutations π'_i, π''_i for $i = 1, \ldots, 8n$ over $\{1, \ldots, m\}$, where m is the number of candidates (i.e. 2 permutations per each of $8n$ ballot columns). These permutations have to be applied to ballot columns.

Each row i in the starting part of R is devoted to a single ballot column of some ballot, (and for each ballot column from some ballot there is exactly one such a row of R). Let $W(i)$ denote the ballot column corresponding to the ith row of R. Then for each i, data concerning $W(i)$ are placed in:

- row i of the starting part,
- row $\rho_1(i)$ of the middle part,
- row $\rho_2(\rho_1(i))$ of the final part.

Moreover:

- the starting part of row i will contain the ballot column $W(i)$ as filled by the voter (the order of the candidates is determined by $\pi_i = \pi'_i \circ \pi''_i$, i.e. the entry for a candidate j is given in row $\pi_i(j)$,
- the middle part at row $\rho_1(i)$ will contain $W(i)$ permuted by $(\pi'_i)^{-1}$),
- the final part at row $\rho_2(\rho_1(i))$ will contain $W(i)$ permuted by $(\pi''_i)^{-1} \circ (\pi'_i)^{-1}$. Hence the marks of $W(i)$ will be permuted according to the standard ordering of candidates: $(\pi''_i)^{-1} \circ (\pi'_i)^{-1} \circ \pi_i = (\pi''_i)^{-1} \circ (\pi'_i)^{-1} \circ \pi'_i \circ \pi''_i = id$.

Below we describe the ith row of R. Let $i = \rho_1(j)$ and $i = \rho_2(\rho_1(k))$.

starting part (for $W(i)$)						middle part (for $W(j)$)				final part (for $W(k)$)
i	$\widehat{\pi_i}$	$H(t(i))$	$\widehat{t(i)}$	$y(i)$	$\widehat{\rho_1(i)}$	$\widehat{\pi'_j}$	$y(j) \ll (\pi'_j)^{-1}$	$\widehat{\pi''_j}$	$\widehat{\rho_2(i)}$	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 ($i \in [1, 8n]$),
- $\widehat{\pi_i}$ – a bit commitment to permutation of candidates π_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 the 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 to the confirmation token $T(i)$,
- $y(i) = [y_0(i), y_1(i), \ldots, y_{m-1}(i)]$ – a vector holding mark $\times$ on those positions 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.

– $\widehat{\rho_1(i)}$ – a commitment to the value $\rho_1(i)$.

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

– $\widehat{\pi'_j}$ – a commitment to permutation of candidates π'_j, where $\pi_j = \pi'_j \circ \pi''_j$,
– $y(j) \ll (\pi'_j)^{-1}$ the vector $y(j)$ permuted by $(\pi'_j)^{-1}$,
– $\widehat{\pi''_j}$ – a commitment to permutation π''_j,
– $\widehat{\rho_2(i)}$ – a commitment to $\rho_2(i)$.

The final part of R in row i contains vector v equal to $y(k)$ permuted by $(\pi'_k)^{-1}$ and then by $(\pi''_k)^{-1}$ (i.e., listed according to the standard ordering of the candidates).

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 random permutation π,
   – EA chooses at random confirmation pre-tokens $t_A(S)$, $t_B(S)$, $t_C(S)$, $t_D(S)$ and computes confirmation tokens $T_A(S)$, $T_B(S)$, $T_C(S)$, $T_D(S)$
   $$T_X(S) := \mathrm{sign}_{EA}(S, t_X(S)) \text{ for } X = A, B, C, D .$$
3. EA creates audit table P: For this purpose, EA chooses at random a permutation σ of $1, \ldots, 8n$. Then $\sigma(4j-3), \ldots, \sigma(4j)$ 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 $S_l(j)$, commitments to the values $i_A(S_l(j))$, $i_B(S_l(j))$, $i_C(S_l(j))$, $i_D(S_l(j)))$ are created and inserted in the row containing $S_l(j)$.
4. EA prepares the audit table R: For this purpose EA chooses random permutations (on R-table rows) ρ_1 and ρ_2 of $1, \ldots, 8n$. For the jth serial number $S_l(j)$, its permutation π (on ballot columns) 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. (i.e., $\pi_{i_A(S_l(j))}$, $\pi_{i_B(S_l(j))}$, $\pi_{i_C(S_l(j))}$, and $\pi_{i_D(S_l(j))}$) take the value π). Separately for each row i of R, EA chooses at random permutations π'_i and π''_i such that $\pi_i = \pi'_i \circ \pi''_i$.
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 permutation of the list of candidates names, confirmation tokens and serial numbers) is hidden under a scratch layer.

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 a 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 it is possible to 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 whether the ballots and the entries in the audit tables were created correctly.
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.

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

1. *Proxy* creates an audit table X in which it commits to 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 form 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 the revealed coding cards have been created correctly.

Elections The following steps are during vote casting:

Step 1: the voter obtains a ballot (e.g. by visiting certain authorities, from a special courier delivering the ballots at residence area, by certified mail services etc.). At the same time, identity of the voter is verified and the ballot is given to her own hands. 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 easy.

Step 2: the voter fetches (still unused) coding cards from one or more Proxies (for convenience the coding cards can be printed).

Step 3: the voter peels-off the scratch-layer from the ballot.

Step 4: the voter logs in an election webpage run by a *Proxy* and authenticates herself.
Step 5: *Proxy* verifies voter's credentials.
Step 6: the voter chooses one of the coding cards and lays it next to the ballot.
Step 7: the voter clicks on the PC screen on radio buttons corresponding to her choice – that is, according to the permutation used for the ballot and alignment of `n` and `Y`'s marks on the coding card.
Step 8: *Proxy* commits to voter's clicks, sends the commitment to *EA* and to the voter (so the voter can print it)[3].
Step 9: the voter enters S_r from the coding card used.
Step 10: *Proxy* transforms the voter's choice into ballot columns.
Step 11: *Proxy* obtains a blind signature from *EA* under each of the ballot columns (these signatures are then stored by *Proxy* for a post-election audit).
Step 12: the voter enters S_l.
Step 13: *Proxy* passes S_l and the ballot columns to *EA*.
Step 14: *EA* enters the obtained ballot columns into appropriate rows of the starting part of table R (but *EA* publishes them when the election are closed), *EA* publishes commitments to the ballot columns obtained from *Proxy*.
Step 15: the voter chooses a receipt (one of the four columns).

Tallying

1. When the voting time is over, *EA* publishes voter's choices inserted into vectors $y(i)$ in the starting part of the table R. Then it computes the entries for the middle part of R: $y(j) \ll (\pi'_j)^{-1}$, and for the final part: $v = (y(k) \ll (\pi'_k)^{-1}) \ll (\pi''_k)^{-1}$.
2. From the entries v in the final part *EA* calculates the tally: If the number of ballot columns is $4N$ (meaning that N 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 jth candidate is $M - 2N$.

Post-Election Audit First, each voter can check if her 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 [11] procedure is executed for R (for the sake of simplicity of description we assume that n voters participated in the elections):

[3] This commitment can be later used during investigation in the case if a fraud was detected.

1. The auditors choose $2n$ rows of R at random and request *EA* to open commitments $\widehat{\rho_1(i)}$ from these rows. Then for each row $\rho_1(i)$ in the middle part, for which $\rho_1(i)$ has been revealed, the commitment $\widehat{\pi_i'}$ is opened and it is checked that the ballot column from the starting part permuted by $(\pi_i')^{-1}$ yields the ballot column in the middle part.
2. For each row j in the middle part, not pointed to by any revealed commitment $\rho_1(i)$, *EA* has to open the commitments to $\rho_2(j)$ and $\widehat{\pi''}$. Then the ballot columns in the middle part of row j permuted with π''^{-1} and the ballot column in the final part of row $\rho_2(j)$ should be equal.

4 Security Concerns

Voter's PC misbehaviour. 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.

Integrity In order to manipulate voter's choice (change from Alice's choice to any other candidate, even a random one) 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. In order to do that, the 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 probability of correct switching Alice's choice is only $\frac{1}{3k}$. But even then, Alice can still detect a fraud by discovering that her receipt does not fit her choice. At least one of the ballot columns is modified during such change (and sometimes it is just one column). So the total probability of a successful and undetected vote change is $\frac{1}{4k}$.

Privacy Even if Alice's PC sends all information it is aware of to an attacker he is unable to determine the choice of Alice. Indeed, the attacker neither knows the configuration of `Y`'s on the coding card nor the permutation used on the ballot.

***EA* misbehaviour.** Now assume that *EA* is dishonest, but the PC of Alice and the *Proxy* are honest.

Integrity Misbehaviour in ballots' preparation and counting is limited by the pre- and post-election audits just like in the case of Punchscan.

Replacing ballot columns when inserting them to table R is risky, since the voter gets a receipt, which is one of her four ballot columns signed by *EA*. If the receipt disagrees with the contents of table *R*, then one can catch *EA*. Recall that the ballot columns are signed blindly by *EA* before *EA* knows ballot's serial number (and thus the permutation) and that *EA* does not know which of the ballot columns is chosen for the receipt. Note that since the hash value of the pre-token is posted in *R*, the voter can prove which entry in the starting part of *R* corresponds to the ballot column from her receipt.

Privacy The situation is like for Punchscan: If *EA* knows which ballot was used by Alice, then it knows the vote cast by Alice. So it is crucial to apply appropriate procedures of ballot distribution. Keeping the sensitive information under scratch surface is a good solution - the ballots can be mixed before distribution, becoming thus indistinguishable. Also, it is crucial that voters never send ballot information directly to *EA* - all communication must go through *Proxy*.

***Proxy*'s misbehaviour.** Now we assume that *Proxy* is dishonest, while *EA* and the PC of Alice behave correctly.

Integrity Proxy commits to the voter's clicks before it knows S_r, so *Proxy* cannot change voter's choice.

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

Privacy The assignments of `Y`'s and `n`'s are known to *Proxy*, so *Proxy* knows the row corresponding to the voter's choice. However, *Proxy* does not know the permutation used in the ballot, so it cannot link the vote with any particular candidate.

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

The key point is that Alice could use a coding card different from the one she shows to the observer – receipt does not contain S_r. So, the situation of Charlie is much different from the situation of a *Proxy*: Charlie obtains only one ballot column (receipt) and cannot be sure if the coding card obtained was really used.

Vote selling. In case of *SC&V* voter is identified electronically by the *Proxy*. The identification protocol should guarantee that the voter would not risk transmitting her electronic identity to the buyer. (In this way *SC&V* becomes superior over postal procedures, for which transferring a ballot to a buyer cannot be prevented.) 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 Alice 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 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 manipulated and the coding card presented needs not to be the one actually used.

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 voter's PC.

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.

Decoy service Decoy service can be optionally launched in order to make vote-buying and coercion harder. A voter can visit a webpage of a decoy service and download fake coding card with any given serial number and arbitrarily chosen arrangement of `Y`'s and `n`'s. Then, the voter may present such a coding card to a coercer or a vote buyer together with a receipt and ballot to prove that she voted in a certain way.

Usability issues The scheme might be uneasy to use at least for two reasons. First, a voter needs to click next to every candidate. Second, a voter needs to find her candidate on a permuted list of candidates. So use of shifted lists of candidates (as for Prêt à Voter) can improve usability. Unfortunately, shifts have bad impact of privacy.

A receipt obtained by a voter is an k-row column. If permutations are used for reordering lists of candidates, then there are $k + 1$ types of of receipts (any two receipts with the same number of $\times$ marks are equivalent).

The situation is different, if shifts are used. Then, the number of different types of receipts is equal to the number of k-bead necklaces with 2 colors which equals to (see [8]): $\mathcal{S}(k) = \frac{1}{k}\sum_{d|k}\varphi(d)2^{\frac{k}{d}}$, where $\varphi(k)$ is the Euler quotient function, i.e. for k-prime $\mathcal{S}(k) = 2^k/k$. In this case, in order to achieve privacy of votes, the number of voters has to be much higher than $\mathcal{S}(k)$. The other solution to this problem is to use so called *masked ballot columns*.

The idea is that table R stores in its rows instead of *ballot columns* – k corresponding *masked ballot colums*. Simply, the jth *masked ballot column* of a given *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. See an example of a ballot column and its masked versions:

$$\text{a ballot column: } \begin{array}{|c|}\hline \times \\ \hline \\ \hline \times \\ \hline \times \\ \hline \end{array}\,, \quad \text{masked ballot columns: } \begin{array}{|c|}\hline \times \\ \hline \\ \hline \\ \hline \\ \hline \end{array}\,,\begin{array}{|c|}\hline \\ \hline \\ \hline \\ \hline \\ \hline \end{array}\,,\begin{array}{|c|}\hline \\ \hline \\ \hline \times \\ \hline \\ \hline \end{array}\,,\begin{array}{|c|}\hline \\ \hline \\ \hline \\ \hline \times \\ \hline \end{array}$$

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, the number of votes is now the number of rows of table 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.

The same technique may be applied to ThreeBallot scheme.

5 Final Remarks

SC&V allows for secure and verifiable vote casting over the Internet with unconditional integrity. Privacy is preserved with the assumption that both authorities do not collude.

A voter cannot prove how she voted unless vote-casting is physically supervised by an adversary (it is not the case in Internet version of Punchscan [15]).

Online vote-selling is almost impossible. In order to buy a vote, a buyer needs to obtain:

- the record of a voting session from the voter's computer (the serial numbers of ballot and coding card, and the voter's choices),
- ballot and coding card used.

Even if the voter's PC is infected by viruses, her choice remains secret. Moreover, any attempt of modification of voter's choice is detected with high probability.

References

1. Top-to-bottom review. top-to-bottom report conducted by Secretary of State Debra Bowen of many of the voting systems certified for use in California, 2007. Available from: `http://www.sos.ca.gov/elections/elections_vsr.htm`.
2. David Chaum. Punchscan, 2005. `http://www.punchscan.org`.
3. David Chaum, Michael R. Clarkson, Stuart Haber, Markus Jakobsson, Stefan Popoveniuc, and Filip Zagórski. Internet voting as secure as polling-place voting. preprint.
4. David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan Sherman. Scantegrity ii: End-to-end voter-verifiable optical scan election systems using invisible ink confirmation codes. *USENIX/ACCURATE EVT 2008*, 2008.
5. David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan Sherman, and Poorvi Vora. Scantegrity: End-to-end voter-verifiable optical- scan voting. *IEEE Security and Privacy*, 6(3):40–46, 2008.

6. 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.
7. 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.
8. Philippe Flajolet and Robert Sedgewick. Analytic combinatorics, 2008.
9. 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.
10. 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/`.
11. 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.
12. Ari Juels, D. Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *WPES*, Lecture Notes in Computer Science, pages 61–70. Springer Verlag, 2005.
13. Chris Karlof, Naveen Sastry, and David Wagner. Cryptographic voting protocols: A systems perspective. In *USENIX Security Symposium*, pages 33–50, 2005.
14. Tal Moran and Moni Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *Advances in Cryptography CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 373–392. Springer Verlag, 2006.
15. 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.
16. 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.
17. 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`.
18. 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`.
19. Jeroen van de Graaf. Merging pret-a-voter and punchscan. Cryptology ePrint Archive, Report 2007/269, 2007. `http://eprint.iacr.org/`.