Introduction to Electronic Voting

Filip Zagórski

Wzrost liczby absolwentów w Politechnice Wrocławskiej na kierunkach o kluczowym znaczeniu dla gospodarki opartej na wiedzy.

DRAFT: V November 20, 2012

Projekt współfinansowany ze środków Unii Europejskiej
w ramach Europejskiego Funduszu Społecznego

Preface

This book was written to support students of the course *Selected topics from computer science*, a lecture taught at the Institute of Mathematics and Computer Science, Department of Foundamental Problems of Technology, Wroclaw University of Technology.

Contents

Chapter 1

Introduction

> Those who cast the votes decide nothing.
> Those who count the votes decide everything.
>
> Josef Stalin

The words of the dictator, are cited to focus on the main goal of this book: presenting, describing, discussing and explaining techniques that are used to design election systems where those who cast the votes decide everything.

This book focuses on techniques that may lead to schemes that satisfy all following properties:

Eligibility: only legitimate voters vote; each voter votes at most once.

Fairness: voting does not reveal any partial (early) results.

Verifiability: individual (local), universal (global).

Privacy: no information about individual votes is revealed.

Receipt-freeness: a voter cannot prove she voted in a certain way.

Coercion-resistance: a voter cannot interact with a coercer to prove that she voted in a certain way.

In fact, we concentrate on verifiability and how introducing verifiability influences on other properties (*i.e.,* privacy, receipt-freeness, coercion-resistance).

The traditional paper-based voting process is not a perfect one. It uses tons of resources, nation-wide elections require tons of paper that need to be printed and later distributed. In addition, in some jurisdictions, ballots need to be prepared in many languages. But the main problem is with manual tallying. Depending on difficulty of a voting scheme, it requires a lot of man-power to calculate election results. Man-power is one of the main cost-factors of elections but is also the main source of threats and failures. This is the reason why election officials prefer to use electronic devices, which provide better managing options than paper based solutions. Also, election results may be obtained within minutes after polls are closed, the speed-up is especially seen with preferential systems, *i.e.,* IRV (instant-runoff voting).

1.1. E-voting

Use of electronic means during polling place voting has two main flavors:

DRE machines: - in that case a voter interacts with an DRE (Direct Recording Electronic) machine and usually a ballot has only a digital form.

optical scan: - in this scenario, voters use paper ballots which are later scanned and processed electronically.

Optical scan systems are preferable from the election integrity perspective, because even if technology fails, there is still an opurtunity to perform a manual recount. Usually, DRE machines might be a great help for voters with disabilities but cannot be as trusted as paper based systems, even if they are equipped with VVPAT (voter verifiable paper audit trail).

Designing a good electronic voting system is not as easy as one may think. ATM machines for the first sight seem to be very similar to DRE-machines. There used to be one company which was manufacturing both. After a couple years of running the business in both areas, the CEO and President of Diebold Inc., confessed in the interview for the Fortune Magazine[1].

> We didn't know a whole lot about the elections business when we went into it.

[1]Source: http://money.cnn.com/magazines/fortune/fortune_archive/2006/11/13/8393084/index.htm, last visited on Oct 30, 2012

> Here we are, a bunch of banking folks thinking making voting machines would be similar to making ATMs. We've learned some pretty painful lessons.

The words were said one year before the Secretary of State of California withdrew a certification for the Diebold's machines (Bow) (and at the same time conditionally recertified them) because of the numerous security flaws.

Introduction of electronic voting to polling places may badly influence on both crucial aspects: *ballot secrecy* and *election integrity*.

1.2. Ballot secrecy

Traditional use of paper may guarantee high level of ballot secrecy. In general, in order to tell anything about a ballot that was cast by a given voter one may need to access a ballot box and to find out a corresponding ballot by inspecting fingerprints.

Use of electronic machines at polling places increases privacy threat. First of all, like in the case of using ATM machines, voting machines learn how one votes. This information (i.e. correlated with the order of voters approaching machines) may be accessible for poll-workers but it may also leak outside. But poll workers are not the only one who can gain access to individual ballots. Voting machines are stored between elections in places with weak access control – anyone who has access to them may alter software that is run by a machine and get access to private data by applying appropriate modifications.

Sometimes, electronic systems have both logical and implementation bugs that lead to revealing how people voted. This is for example the case of the Brazilian DRE machines, where due to wrong use of a pseudo-random generator, receipt reveals how a voter voted (for more details see Example 3 in Chapter 3).

Similar problems with ballot secrecy, but at smaller scale, are connected with optical scan approach, namely – scanning may be done with much higher resolution that may lead to recording fingerprints together with a image of a ballot.

1.3. Election integrity

Threat to the ballot secrecy is very severe and can influence on election outcome by vote buying or coercion. But the most devastating type of attacks are those on election integrity – when someone is able to modify election outcome (without the need of any external co-operation like vote-buying or coercing other voters).

Election integrity may be violated at different stages. One may ask following questions about what happens with a ballot in electronic voting system:

(1) Is my ballot cast as intended?

(2) Is it recorded as cast?

(3) Is it tallied as recorded?

End-to-end (E2E) verifiable voting schemes try to give a positive answer to all above questions, moreover those systems provide proofs that support the claims.

In fact more detailed conditions must be met in order to call a system an end-to-end verifiable one, but for the next couple of chapters we just want to focus on those three (until we reach Chapter 5).

The problem is that almost every electronic voting scheme that was/is used so far, does not provide E2E verifiability. Problems start at very early stage of vote-casting *i.e.,* DRE-machines do not ensure that a vote is *cast as intended* nor *recorded as cast*; optical scan solutions may have problems with correct vote-interpretation or adjusting correct treshold levels during scanning.

1.4. Absentee voting

As in the case of polling place voting, we are comparing systems to those that use paper ballots. In the case of electronic absentee schemes one should compare to the current vote by mail solutions.

Absentee voting schemes, comparing to polling place voting are facing additional problems:

- remote voter authentication,
- ballot secrecy: voter's choice may:

  - be revealed to a voter's computer (case of Norway, Estonia, Switzerland),
  - be revealed malware on voter's machine,
  - become public.
- vote buying on mass scale,
- coercion, both:
  - in-person,
  - "online" – thanks to a special software.
- availability of the system:
  - may be limited to all voters by DDOS attacks,
  - may cause abstention of targeted group of voters.

If vote casting is made, using electronic means then all threats corresponding with election integrity corresponding with DRE machines apply to that case as well.

1.5. Summary

This long list of obstacles does not mean that designing reliable electronic voting scheme is not possible. It just make things more complicated but at the same time more interesting. There is a bunch of cryptographic techniques that may be used in order to meet requirements expected from voting systems.

In order to achieve ballot secrecy (in the case when a voter interacts with a voting machine), it is common to use indirection like code voting .

Election integrity may be achieved at the same time. To ensure "cast as intended" (see Chapter 2), usually one of the two approaches are applied: *cut & choose* or *commit & prove*. Vote casting step usually provides a voter a receipt.

To verify "recorded as cast" a voter may use a receipt generated during ballot casting (see Chapter 3).

Finally, to ensure "tallied as recorded", verifiable mix-networks are used (with help of *zero knowledge proofs*) – this is the subject of the Chapter 4.

In order to perform many cryptographic protocols that are involved as building blocks of an electronic voting scheme, one needs to use other cryptographic primitives like: non-malleable encryption, bit-commitments, digital signatures, oblivious transfer that are described in the Chapter 4.

In the Chapter 5 we discuss a more preceise definition of end-to-end verifiability.

The last two, Chapter 6 and Chapter 7 describe end-to-end verifialble voting schemes that were used in the binding public elections.

ATM vs DRE

1. Describe a protocol that is performed between a customer and an ATM machine (and a bank and a payment card). Consider the following cases:
   (a) a customer asks for a receipt,
   (b) a customer does not ask for a receipt,
   (c) a customer asks for a receipt but the receipt is not printed.

   A customer wants to withdraw amount x. Assume that an ATM is altered and it requests from a bank amount $x + 50$ but pays a customer x. What steps needs a customer to follow in order to detect ATM's "mistake"? Can she prove that, is the proof dispute free?
2. Some ATM machines perform a slightly different protocol – they ask a customer if she/he wants to get a receipt before performing any operation. Explain the difference between that protocol and scenarios considered in the previous exercise.
3. Describe what is:
   (a) security policy – what is and what is not allowed,
   (b) security mechanism – methods, tools and procedures that are used to enforce security policy,
   (c) threat model,
   (d) risk assessment,
   (e) countermeasures

   in the context of accessing bank account within an ATM card.

Chapter 2

Cast as Intended

End-to-end verifiable voting schemes allow a voter to obtain a receipt generated at the stage of vote casting, as a result of a voter's interaction with a voting system, such a receipt constitutes a private proof that her vote intent was correctly "encoded" and her ballot was cast. A receipt allows a voter to verify *cast as intended*.

In this chapter we discuss ways in which a voting receipt is generated. Such a receipt is : DRE/scanner or a voter's computer in the remote scenario setting. But the role of a receipt is not only limited to detect possible malfeasance.

Dispute freeness. One less obvious property an E2E voting system should provide is dispute-freeness (KY02) (or accountability (KTV10a)). If the verification of some aspect of the election fails, implying an error or fraud, the voter should be able to demonstrate that it failed and which entity is responsible. For example, in polling place elections, it is not very important to differentiate between attacks due to a malicious DRE/scanner and insider attacks conducting by the EA: the EA is responsible for both. With online voting, the EA cannot assume accountability for the state of voters' computers. If vote verification fails, the EA must ensure that it is not incorrectly blamed for compromised voter machines. Likewise, voters want assurance that a malicious EA cannot modify ballots and blame the voters' computers if the modification are detected. It is also important that voters

or political parties cannot easily fabricate false evidence that an election has been compromised, casting doubt on the final tally.

Section outline. In this chapter we describe some of the important schemes that have been presented during the last few years: Chaum's Punchscan ((Pun), Scantegrity (CEC$^+$08) and Scantegrity II (PCC$^+$08)), Rivest's VAV (Vote Anti-vote Vote) and ThreeBallot (RS07) and Prêt à Voter (CRS05). All of them are dedicated to paper-based elections at polling stations – ballots are scanned. We also shortly describe the first voter-verifiable voting scheme that was presented in 2004 by Chaum (Cha). All mentioned schemes use at some point *cut-and-choose* technique.

We also discuss some aspects of Estonian, Norwegian and Swiss internet voting schemes (although those schemes do not provide end-to-end verifiability) and compare them with Helios voting system that is voter-verifiable. We stress our attention to *cast as intended* part of the schemes. Helios system implements Benaloh (Ben06) idea of using *commit and prove* approach in the setting where a voter interacts with an electronic voting machine (as opposed to optical-scan scenario and interacting with a paper ballot).

There are two main approaches that are used to verify correctness of the protocol at the stage of ballot casting. The first one is called: *cut-and-choose*, the second one is *commit and prove* (but sometimes it is called Benaloh challenge).

2.1. Cut-and-choose

Cut-and-choose approach is used in interactive protocols where a prover answers to the challenges generated by a verifier. After a series of such random challenges a verifier is convinced, with high probability that a prover is honestly following the protocol.

In the following examples, *cut an choose* is used as a part of voter-initiated print audit. A voter may ask for more than one of encoded ballots and verify if they are printed (encoded) correctly, at the same time a voter keeps one ballot that she uses for cast.

The power of the *cut and choose* comes from the fact that those audits are performed independently. To achieve high level of confidence *i.e.,* $P_c \geq 1 - 2^{-n}$ that ballots indeed are *cast as intended* it is sufficient that voters are limited to pick only two ballots, audit one and cast the other one. If

the probability to detect a misprinted ballot at random is $P_{mis} \geq p$ then thanks to the independence, the probability that m voters do not detect any misprinted ballot is $< (1 - P_{mis})^m$.

2.1.1. Prêt à Voter. (CRS05) A voter, obtains a ballot which consists of two parts (see Fig. 2.1). The left part contains the official list of the candidates, shifted by x positions, where x is chosen at random, independently for each ballot. On the right hand-side of each ballot, a voter can mark her choice of her preffered candidate. At the bottom of the right hand-side of a ballot there is a serial S printed.

Candidate	
2 Jerry	
3 Edgar	
0 Ervin	
1 Donald	
	324989

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

Candidate	
2 Jerry	
3 Edgar	
0 Ervin	×
1 Donald	
	324989

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

×
324989

(c) A receipt of the vote

Figure 2.1. Ballot example for Prêt à Voter scheme

A voter makes her choice and then separates both parts. The left-part is shredded and the right part is scanned, copied as a voter's receipt, placed in a ballot box.

2.1.2. VAV. The *VAV* (Vote Anti-vote Vote) scheme was presented by R. Rivest and W. D. Smith in (RS07) and is an example of an end-to-end verifiable voting scheme that does not use cryptography. It is a paper based voting system which uses a *cut-and-choose* concept. In contrast to the previous examples (Prêt à Voter and Scantegrity), cut and choose is not used to ensure *cast as intended* part of a voting process but rather *recorded as cast*. *Cut and choose* helps also to generate a receipt that does not reveal the vote.

VAV protocol uses three ballots of the two kinds: two ballots are labeled as a "Vote" and the third ballot is labeled as an "Anti-vote" (see Figure 3(a)). Each of the ballots has a unique serial number that is covered with a scratch-off.

To cast a vote in VAV scheme, a voter performs the following steps (compare with Figure 2.1.2:

marking: The voter:
(1) The voter chooses one of the Vote ballots, and marks a candidate of her choice,
(2) The voter marks any candidate on the other Vote ballot and marks the same candidate on the Anti-Vote ballot.

casting: The voter interacts with the system:
(1) the ballots are verified by a reader/checker (which checks if ballots are filled correctly),
(2) the ballots are separated; serial numbers are revealed (but still a voter cannot see them);
(3) the voter obtains a copy of one of the ballots as a receipt (but the system does not know which one);
(4) all ballots are inserted into a ballot box.

publishing: Election Authority publishes all ballots collected in the ballot box on a public Bulletin Board.

tallying: Election Authority:
(1) all votes from Vote parts are added up,
(2) all votes from Anti-vote parts are subtracted from the totals for a given candidate.

verification: A voter:
(1) checks if her receipt is published on the Bulletin Board,
(2) verifies if a tally is computed correctly.

In the example below, Alice votes for Cat – she chooses the rightmost ballot to make her choice, while she selects Fish on the remaining ballots (Vote and Anti-Vote).

Verifiability. After closing of the elections, all ballots from the ballot box are published. Every voter can check, if his receipt is present on the list. In order to perform this step, each ballot must be marked with a unique serial

Figure 2.2. VAV voting scheme

Vote 1920		*Anti − vote* 8239		*Vote* 7281	
1. Dog	☐	1. Dog	☐	1. Dog	☐
2. Cat	☐	2. Cat	☐	2. Cat	☐
3. Fish	☐	3. Fish	☐	3. Fish	☐
4. Parrot	☐	4. Parrot	☐	4. Parrot	☐

(a) Plain voting card (plain ballots). All serial numbers are hidden under the scratch-off *surface*.

Vote 1920		*Anti − vote* 8239		*Vote* 7281	
1. Dog	☐	1. Dog	☐	1. Dog	☐
2. Cat	☐	2. Cat	☐	2. Cat	☒
3. Fish	☒	3. Fish	☒	3. Fish	☐
4. Parrot	☐	4. Parrot	☐	4. Parrot	☐

(b) Vote for Cat

Vote 1920	
1. Dog	☐
2. Cat	☐
3. Fish	☒
4. Parrot	☐

(c) A receipt corresponding to the left ballot. Serial number of the ballot 1920 revealed to the voter.

number (which cannot be seen by a voter before he obtains a receipt – to prevent "random attacks" (KRMC10)).

Election authority does not know which copy of a ballot is taken as a receipt by a voter. If any ballot is modified by the election authority, then a certain voter can detect the manipulation with probability $\geq \frac{1}{3}$.

2.2. Cryptographic commitments

A *cryptographic commitment* is a scheme that allows a participant $\mathcal{A}$ of a protocol to commit $c := \mathsf{Comm}(x, open)$ to a value x without it to anyone during *commiting phase* and to reveal x in the *reveal phase* (by providing *open*). We are focused on schemes that are:

binding: $\mathcal{A}$ cannot change the value x that she commited to.

hiding: c does not reveal information about x.

A value c ($= \mathsf{Comm}(x, open)$) is called a commitment to x.

The concept of commitments was formalized in (BCC88) by G. Brassard, D. Chaum and C. Crapeau but existed in the literature for many years ().

Commitments cannot be at the same time both *perfectly binding* and *perfectly hiding*. Depending on properties of an application, one should one of the following combinations available:

- perfectly binding and computationally hiding, or
- computationally binding and perfectly hiding.

In the first case, a sender of a commitment is unable to find a different value that would be consistent witha commitment – even if it has unbounded computational power. While someone who has enough computational power is able to discover a commited value.

2.2.1. Examples.

Perfectly binding scheme based on the discrete logarithm problem.

Setup:

(1) chooses $p \in Primes$, $0 \leq x \leq p-1$.
(2) chooses a generator $g \in Z_p^*$.
(3) return public parameters: $\langle p, g \rangle$.

Commit: $\mathcal{A}$ in order to commit to x:

(1) publishe $c := g^x \bmod p$.

CheckReveal: to verify a commitment c to x:

(1) checks if $c = g^x \bmod p$.

The scheme presented above is *perfectly binding* since $\mathcal{A}$ cannot find any $x' \neq x$ such that $g^{x'} \bmod p = g^x \bmod p$. The scheme is not perfectly hiding because of computational infeasibility of the discrete logarighm problem states that is is not possible to compute x from c. In fact the scheme does not satisfy a formal definition of the security because with a limited message-space it is possible to find x without need of computing discrete log.

Pedersen commitment. The Pedersen (Ped92)

Setup:

(1) choose $p, q \in Primes$ such that $q|(p-1)$.

(2) choose g, a generator of G – the order-q subgroup of Z_p^*.
(3) choose a at random.
(4) compute $h := g^a \bmod p$.
(5) return: public parameters: $\langle p, q, g, h \rangle$.

Commit: To commit to a value x:
(1) choose $r \in Z_q$.
(2) compute $c := g^x h^r \bmod p$.

Open: To open a commitment c, the sender reveals x, r, the receiver:
(1) checks if $c = g^x h^r \bmod p$.

Pedersen commitment scheme is *perfectly hiding* – for a given commitment c, every value x is equally likely to be the commited value. The scheme is *computationally binding* since if a sender can solve discrete log problem, she can find different values x and x' and different opening values r and r' such that $g^x h^r = g^{x'} h^{r'}$.

2.3. Commit and prove

Commit and prove approach is one of the standard ways of veryfing correctness of a protocol. In a voting scenario it is often used at a ballot generation stage. *Commit and prove* appeared in the voting literature relatively recently – when voting protocols started to distinguish between a voter and a computer that is used to cast a vote. One of the the first protocols that use this approach is (Ben06) by Benaloh and thous sometimes *commit and prove* in the voting setting is called *Benaloh challenge*.

Here we describe not the original scheme but the Helios (Adi08) voting system – internet version of the Benaloh scheme. We refer to the important differences between polling-place Benaloh scheme and its electronic implementations in Section 6.1.

VAV and ThreeBallot

1. Assume that you have a receipt generated by the ThreeBallot scheme with k marked rows and n candidates. Find the number of Ballot pairs that form a valid tripple with your receipt. Compute how many different receipts are possible in that scheme.

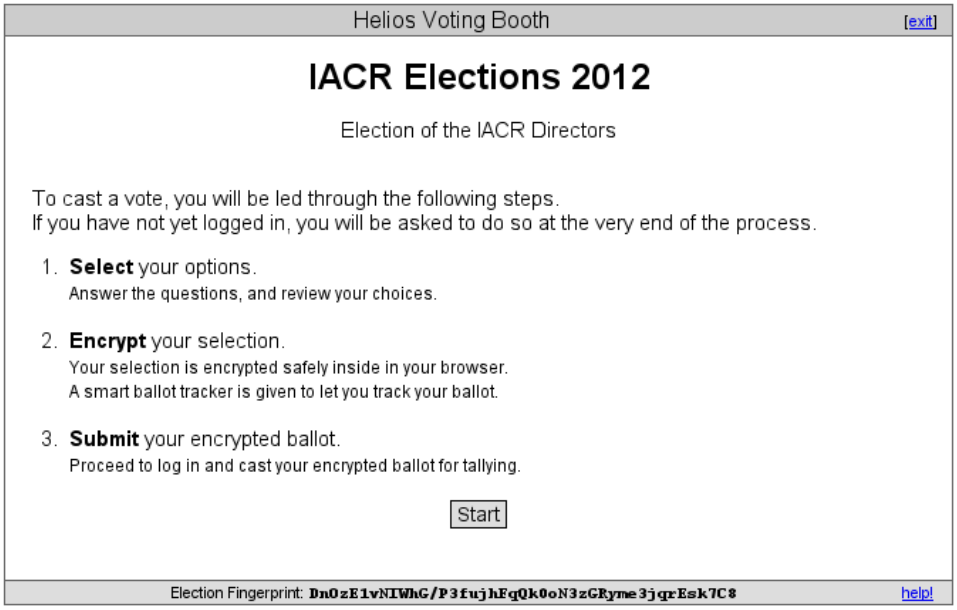


Figure 2.3. Helios welcome screen presenting voting steps.

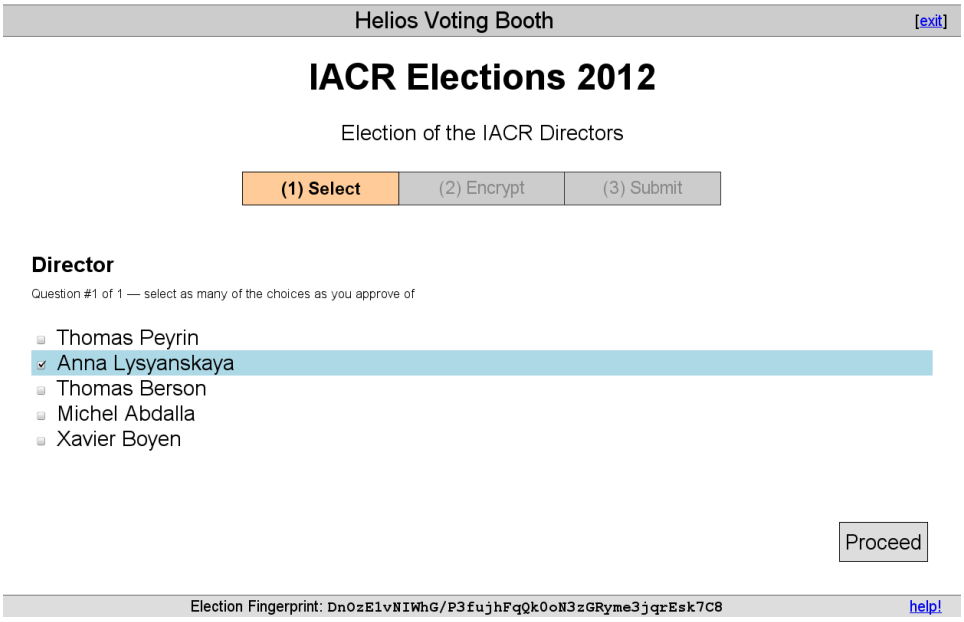


Figure 2.4. In this step, a voter marks her choice.

Figure 2.5. Confirmation screen.

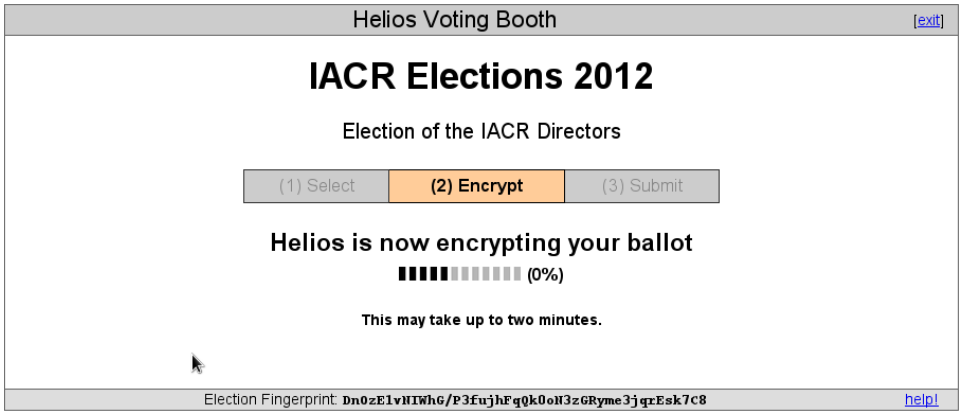


Figure 2.6. After the confirmation, ballot is encrypted and stored in a browser's memory.

2. Consider a modification of the VAV scheme, where one can vote for up to k out of n candidates. On the figure below there is an example of a valid vote for Dog. Find a number of correct receipts.

3. Assume that Election Authority knows serial numbers of all ballot tripples that were cast and tries to modify election outcome. If they modify one of your ballot-parts, what is the probability that you do not detect that while checking a receipt on a bulletin board? Assume that k voters whose ballots were modified are checking their receipts online. What is the probability of undetected malfeasance?

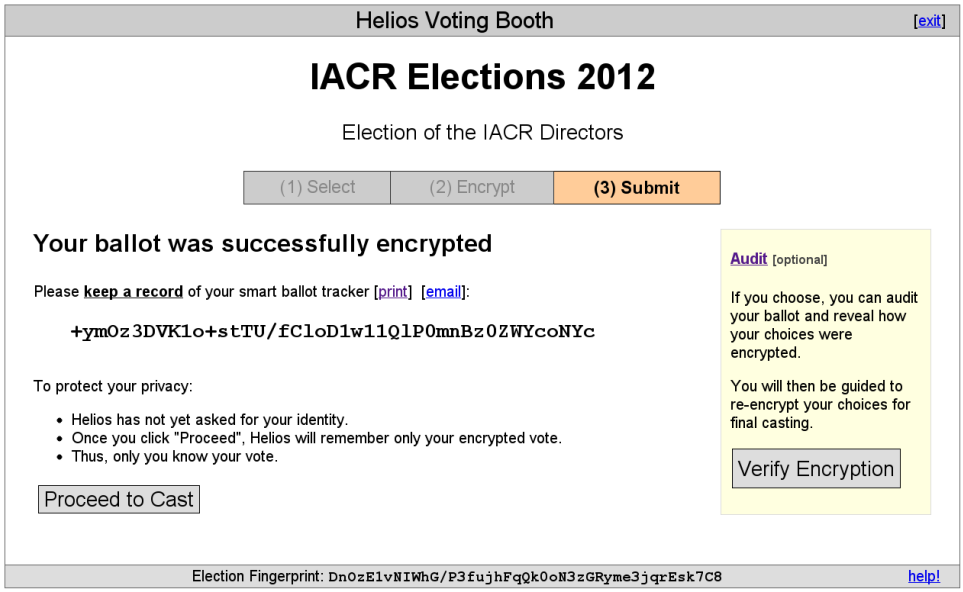


Figure 2.7. A commitment to the ballot's encryption is displayed to the voter. At this point a voter can *cast* and thus go to the screen presented on Figure 2.10 or *audit* the ballot encryption (Figure 2.8 and Figure 2.9) and after the auditing go back to the step presented on the Figure 2.5.

4. Assume that N votes were cast ($3N$ ballots) and Election Authority modified k ballots (each one from a different tripple), what is the probability that the malfeasance is detected when a random sample of m voters verified their receipts online? Compute that probability for different parameters $N = 10, 100, 1000, 10\ 000$, $k = cN, c = 0.01, 0.1$ and $m = pN, p = 0.01, 0.05, 0.1, 0.2$.

5. Election Authority may modify election outcome by changing only those ballots that were cast by themselves – then, there are no voters who can complain about a receipt. Analyze what can be achieved with that approach in three candidate race. Consider both VAV and ThreeBallot schemes. Can this situation be detected by just observing bulletin board? Give necessary and sufficient conditions for detection (it there are any).

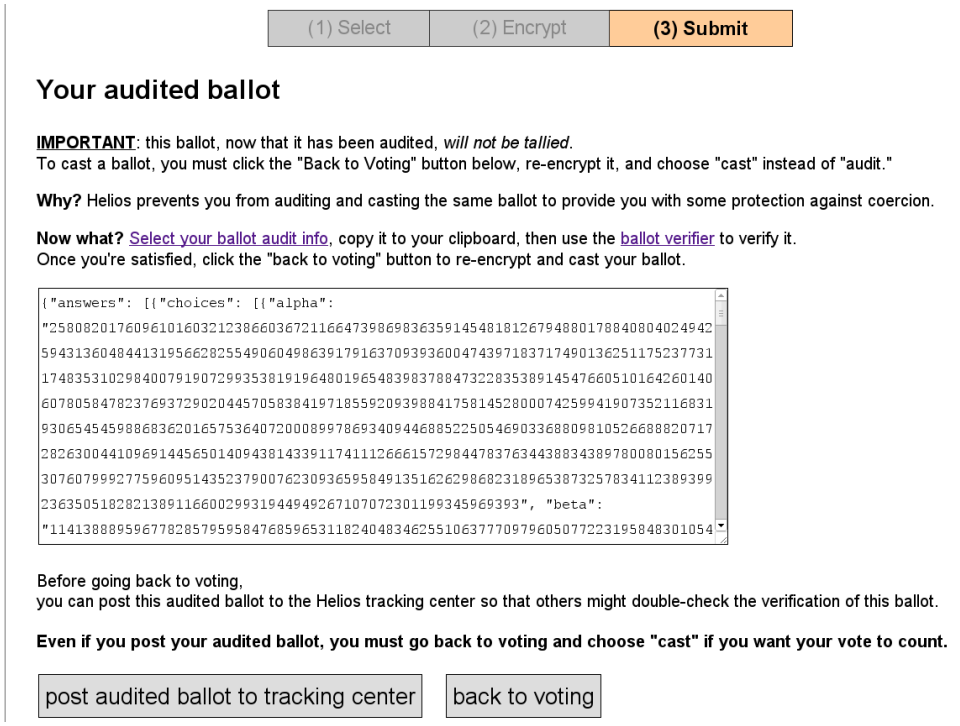


Figure 2.8. Audit screen: randomness used to generate the encryption is revealed to the voter so she can verify if the ballot was encrypted as intended. Audited ballot cannot be cast.

References

[1] Adida, Ben and Neff, C. Andrew, **Efficient Receipt-Free Ballot Casting Resistant to Covert Channels**, Proceedings of the 2009 conference on Electronic voting technology/workshop on trustworthy elections, http://static.usenix.org/event/evtwote09/tech/full_papers/adida-casting.pdf

Figure 2.9. Single-ballot verified: online service that allows to verify if a browser correctly encrypted voter's choice.

Figure 2.10. Voter is authenticated only if she is ready to cast a ballot.

Vote	*Anti − vote*	*Vote*
1. Dog ☒	1. Dog ☐	1. Dog ☐
2. Cat ☐	2. Cat ☒	2. Cat ☒
3. Fish ☒	3. Fish ☒	3. Fish ☐
4. Parrot ☐	4. Parrot ☐	4. Parrot ☒

Chapter 3

Recorded as cast

End-to-end verifiable voting schemes allow a voter to obtain a receipt that is used to verify that her ballot was *cast as intended*. A receipt plays also another role – it helps to check if a vote is included in a set of ballots that are going to be tallied – so it is also used in the verification of *recorded as cast* step (for more details see Chapter 5).

Receipts allow voters to verify election outcome but badly designed receipts may be a threat to ballot secrecy. So designing an end-to-end verifiable voting system is mainly about designing receipts:

- they need to allow a voter to publicly prove (potential) malfeasance,
- they need to hide a voter's choice.

We saw an example of two very similar systems: ThreeBallot and VAV, each of them have similar way of generating a receipt which is a copy of one of the ballots that a voter casts. In the case of VAV, with high probability a receipt does not tell how someone votes (assuming that the number of candidates is smaller than the number of voters and that voters mark their vote and antivote parts at random). The case of ThreeBallot is different, one can show that even in the case of not too long list of candidates – slightly above 6, it is possible to effectively reconstruct votes of single voters (Str06; CKW08).

On the other side there are systems like Prêt à Voter and Scantegrity II, in that systems a receipt that is taken home by a voter does not reveal the

vote. Moreover for the Scantegrity II system it was proven (KTV10b) that the system is coercion-resistant.

Example The Brasilian system is badly designed. Among other problems, it is possible to recover votes without having physical access to a voting machine. This is possible because shuffling is made basing on a pseudo-random number generator whose seed is a part of a public record. Lines responsible for generating a shuffle are similar to those below:

```
srand(time(NULL));
fprintf(public document, ‘‘%dnn’’, time(NULL));
```

System manufacturer made at least three mistakes:

- using weak PRNG,
- using seed that was not random enough,
- making seed public.

Anyone who had access to votes and knew how they are shuffled could influence on election outcome by buying votes or by coercing voters. In such a case, a voter had to deliver a receipt as a proof of vote (see Figure 3.1) – time of ballot (Hora) casting could be used as a pointer to the actual ballot stored by the system.

3.0.1. TWIN. TWIN is not in fact pure "end-to-end" scheme – it does not allow a voter to check that her or his vote has been counted.

In the case of TWIN (RS07), each voter marks a ballot and puts it in the ballot box, receiving a copy of a randomly chosen previously-cast ballot. This copy is a voter's take-home receipt.

Obviously, such a receipt does not reveal any information about voter's choice. At the same time, election authority cannot modify ballots, because they do not know which ballots have been taken as receipts.

To allow verification, all ballots have serial numbers that are hidden under a scratch-off layer. This layer prevents a voter to know the serial number of her or his ballot. A checking machine check that the marking of a ballot is valid, removes scratch-off and places it into the ballot box.

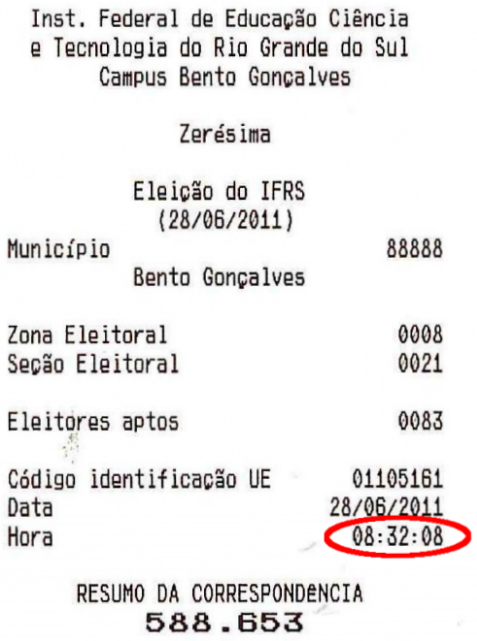


Figure 3.1. Data printed on a receipt are used as a seed of a random generator which output is used to permute ballots.

References

[2] **Software Vulnerabilities in the Brazilian Voting Machine**, Diego F. Aranha, Marcelo M. Karam, Andre de Miranda, Felipe B. Scare, 2012 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections.

Chapter 4

Tallied as recorded. Cryptographic building blocks

This chapter describes basic cryptographic primitives that are used as building blocks in voting schemes. We start our description from *blind signatures* and thus motivate need of use *mixing networks* (mixing nets, mix-nets) that allow voters to preserve ballot secrecy. Next we describe the idea of *homomorphic encryption* that motivates use of *zero-knowledge proofs* (ZKP) and *non-interactive zero-knowledge proofs* (NIZKP). We also describe ideas of *commitments* and *oblivious transfer*.

4.1. Blind signatures

A *blind signature*, introduced in (Cha82) by David Chaum, is a form of *digital signature* in which a signer does not see a message that is signed.

> A digital *blind signature* has the following real-world interpretation: an author of a message writes it down on a slip of paper that is put inside a carbon paper lined envelope. A signer writes a signature on the outside of the envelope which results leaving a carbon copy of the signature on the paper inside the envelope.
> A signer does not learn what it signed. A message author can obtain a signer's signature under any message.

There is a few versions () of blind signatures, we just briefly describe *RSA*-based version.

4.1.1. RSA.

KeyGen: Key generation procedure:

(1) choose p, q at random, two primes of equal length.
(2) compute $N := pq$.
(3) compute $\varphi(N) = (p-1)(q-1)$.
(4) choose $1 < e < \varphi(N)$ (usualy $e = 65\ 537$).
(5) compute $d = e^{-1} \bmod \varphi(N)$ (so $ed = 1 \bmod \varphi(N)$).
(6) return:

public key: $K_{pub} \leftarrow \langle e, N \rangle$
private key: $K_{priv} \leftarrow \langle d, N \rangle$

Enc: To send a message M to a user with a public key $K_{pub} = \langle e, N \rangle$:

(1) compute a padding: $m \leftarrow padding(M)$ that transforms message M into a number $0 \leq m < N$ using a predefined *padding* function *i.e.,* OAEP (Optimal Asymetric Encryption Padding).
(2) compute a ciphertext: $c \leftarrow m^e \bmod N$.
(3) return: c.

Dec: To decrypt a ciphertext c with a private key $K_{priv} = \langle d, N \rangle$:

(1) compute $m \leftarrow c^d \bmod N = (m^e)^d \bmod N = m^{ed} \bmod N = m^{ed \bmod \varphi(N)} \bmod N = m$.
(2) return: m.

Sign-simple: To sign[1] a message M with a private key K_{priv}:

(1) compute $mHash \leftarrow \mathcal{H}(M)$.
(2) compute $S := mHash^d \bmod N$.
(3) return: $Sign := \langle M, S \rangle$.

Verify-simple: To verify a signature $sign = \langle M, S \rangle$ with a public key $K_{pub} = \langle e, N \rangle$:

(1) compute $mHash = \mathcal{H}(M)$.
(2) return:

true: if the value $S^e \bmod N$ is equal to the hash value $mHash$.
false: otherwise.

[1] This is a simplified, insecure version of RSA signature scheme, the real-world version is described below

Sign: To sign a message M with a private key K_{priv} (*RSA-PSS* scheme presented below):

(1) compute $mHash \leftarrow \mathcal{H}(M)$, where $\mathcal{H}()$ is *e.g.,* SHA-512, Keccack.

(2) compute EM according to the scheme presented on the Figure 4.1:

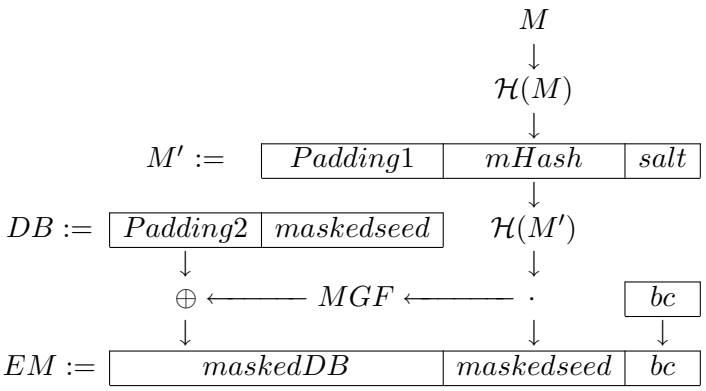


Figure 4.1. RSA-PSS encoding operation, transforming a message M into an encoded message EM.

(3) compute $S = EM^d \bmod N$

(4) return: $sign = \langle M, S \rangle$

Verify: To verify a signature $sign = \langle M, S \rangle$ with a public key $K_{pub} = \langle e, N \rangle$:

(1) compute $mHash = \mathcal{H}(M)$

(2) recover EM from S: $EM = S^e \bmod N = EM^{de} \bmod N$

(3) return:

true: if the encoded message EM is a valid transform of the hash value $mHash$.

false: otherwise.

4.1.2. Blind RSA signatures. For the blind signatures, the key generation (*KeyGen*) procedure looks exactly the same as for the *RSA*. There is a difference in the signature generation, which is performed by the two parties:

Blind-sign-simple: A participant $\mathcal{A}$ to obtain a blind signature of $\mathcal{B}$ under a message M performs the following steps:

	$\mathcal{A}$		$\mathcal{B}$
	$\mathcal{B}$'s public key $\langle e, N\rangle$		private key $K_{priv} = \langle d, N\rangle$
	message M		
(1)	computes $mHash := \mathcal{H}(M)$		
(2)	generates random r		
(3)	computes $x = r^e \cdot mHash \bmod N$		
(4)	sends x to $\mathcal{B}$	$\xrightarrow{x}$	
(5)			computes $y = x^d \bmod N$
(6)		$\xleftarrow{y}$	sends y to $\mathcal{A}$
(7)	removes a blinding factor: computes $S := y/r \bmod N$		
(8)	return: $sign = \langle M, S\rangle$		

Figure 4.2. RSA blind signature scheme.

4.1.3. Blind signature based voting. Here we present a simple electronic voting scheme that uses blind signatures. This is rather a toy example that we use as a motivation for use of verifiable mixing networks (see Section 4.3).

The settings for the scheme are the following:

- Election Authority $\mathcal{EA}$ has an RSA private key K_{priv} and public key K_{pub}.
- Each voter V_i wants to submit a ballot b_i, each V_i has a token t_i that can be used to authenticate to Election Authority.
- A public *bulletin board* $\mathcal{BB}$ accepts (in append-only mode) entries that are signed with the Election Authority's private key.

The Figure 4.3 presents data flow of the scheme. We assume that $\mathcal{EA}$ has a set $T_V = \{t_1, \ldots, t_n\}$ of all tokens. A bulletin board can be interpreted as a list $BB = BB(\pi, k) = BB_k = \langle \langle b_{\pi_1}, S_{\pi_1}\rangle, \ldots, \langle b_{\pi_k}, S_{\pi_k}\rangle\rangle$, where a permutation π depends on the order in which voters cast their ballots and k is the length of the list (k is the number of votes that were cast so far). We also assume that for the bulletin board only two operations are allowed: $\mathsf{Read}(BB)$ which returns all elements on the BB list and $\mathsf{Append}(BB_k, \langle b_j, S_j\rangle) = BB_{k+1}$, where $\pi_{k+1} = j$.

The blind singnature base voting scheme presented on the Figure 4.3 has a nice property that if the Election Authority do not cooperate with the Bulletin Board operator then votes remain secret (bulletin board can

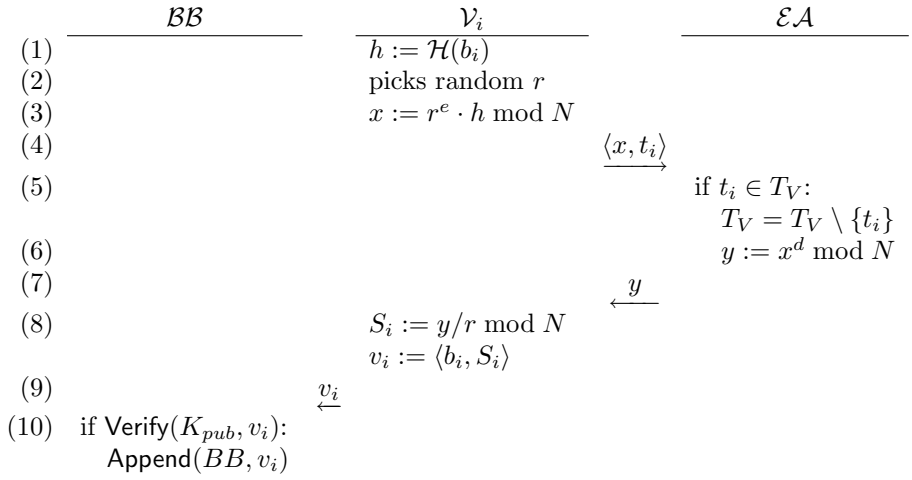


Figure 4.3. Blind signature based voting scheme.

be implemented in distributed manner, with support of P2P networks and anonymizing services like TOR to achieve this property).

But there are the following problems with the scheme:

- it reveals partial results because all votes are posted in cleartext $\xrightarrow{\text{Solution}}$ post encrypted votes (and we already know how to encrypt votes in verifiable manner - Chapter 2). But in that case, there should be a verifiable way of decrypting encrypted votes from the Bulletin Board (see Section 4.3).

- Election Authority can add votes to the Bulletin Board since only a signature is verified. There are a few possible solutions to that problem:
  - EA can generate *commitments* to each admisible token before election and only votes with corresponding tokens are counted

- If Election Authority generates tokens (if tokens are one-time passwords) and passes them to voters then Election Authority may use them to cast the votes $\xrightarrow{\text{Solution}}$ use *oblivious transfer* during token issuing – oblivious transfer-like physical *i.e.,* with a use scratch-off cards – compare with Section 7.1.

4.2. Homomorphic encryption

In this section we show how to use homomorphic encryption schemes to design voting schemes. At the same time we present reasons of using: non-interactive zero knowledge proofs, non-mallable encryption and other techniques as building blocks of voting protocols.

We say that an encryption scheme $\mathsf{Enc}()$ is homomorphic if the following equality holds:

$$\mathsf{Enc}_k(m_1) \cdot \mathsf{Enc}_k(m_2) = \mathsf{Enc}_k(m_1 + m_2).$$

If we interpret m_1 and m_2 as numbers, then homomorphic encryption allows for computing the ciphertext of $m_1 + m_2$ from the ciphertexts of m_1 and m_2 (but without knowledge of m_1 and m_2 themself).

For a race with m candidates/options, an electronic ballot sent by i^{th}-voter may have the following form:

$$b_i = \langle \mathsf{Enc}_k(x_{i,1}), \mathsf{Enc}_k(x_{i,2}), \ldots, \mathsf{Enc}_k(x_{i,m}) \rangle ,$$

where[2] $x_1 + \ldots + x_m = 1$ and $x_1, \ldots, x_m \geq 0$. Then we can easily compute the encryption of a sum of the votes cast for each candidate, see Figure 4.2.

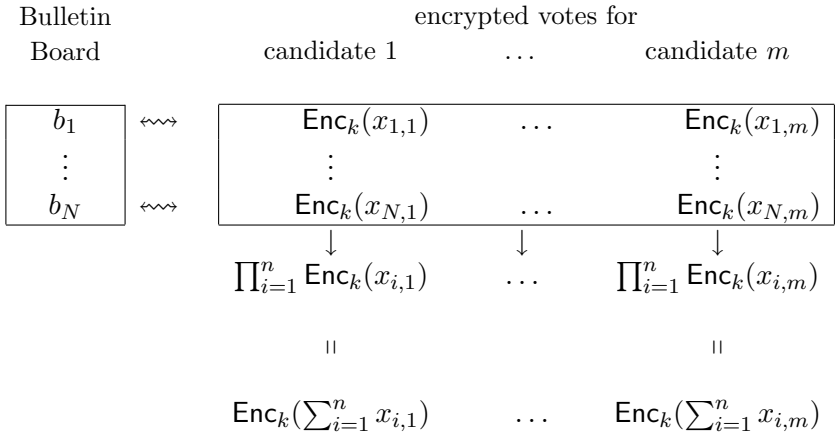


Figure 4.4. Scheme applies homomorphic encryption as a voting scheme.

Homomorphic encryption is a very apealing approach, but unfortunatelly it cannot be applied right away. We need to take care of few things:

[2]For a race when only one candidate can be selected, and $x_1 + \ldots + x_m \leq l$ if up to l choice may be selected.

(1) A voter needs to be convinced that her vote is encrypted correctly – but we already know how to approach that problem – by using *commit and prove* or *cut and choose* approaches.

(2) An election authority should not be able to link a decrypted vote to a voter – we need to use a *verifiable mix network* to decrypt votes.

(3) All encrypted ballots should have proof of correctness – use Zero Knowledge Proofs – Section 4.4.

4.2.1. Homomorphic encryption of counters. Here we present an additively homomorphic variant of ElGamal to encrypt ballots (this variant is *e.g.,* used in VoteBox (SDW08) and RemoteBox (SW08) schemes).

For group generators f, g, private key a, public key $h = g^a$ and plaintext (counter) c, the encryption is defined as follows:

KeyGen: To generate a key:

(1) choose p, q at random, so $p = 2q + 1$.
(2) choose f, g generators of a subgroup of order q in Z_p^*.
(3) generate a at random ($a < q$).
(4) compute $h := g^a \bmod p$.
(5) return:

public key:: $K_{pub} \leftarrow \langle f, g, h, p\rangle$.
private key:: $K_{priv} \leftarrow \langle a, f, g, p\rangle$.

Enc: To encrypt a counter i:

(1) generate r at random.
(2) return:

$$\mathsf{Enc}(i, r, h) := \langle x, y\rangle = \left\langle g^r, h^r f^i \right\rangle$$

Dec: To decrypt $\langle x, y\rangle$ knowing a private key K_{priv}:

(1) return:

$$\mathsf{Dec}(\langle x, y\rangle, a) = \mathsf{Dec}(\left\langle g^r, g^{ar} f^i \right\rangle, a) := \frac{g^{ar} f^i}{(g^r)^a} \bmod p,$$

Dec: To decrypt $\langle x, y\rangle$ knowing a nonce r and public key K_{pub}:

(1) return:

$$\mathsf{Dec}(\langle x, y\rangle, a) = \mathsf{Dec}(\left\langle g^r, g^{ar} f^i \right\rangle, a) := \frac{g^{ar} f^i}{(g^a)^r}.$$

SumUp: To compute an encryption of two encrypted counters i, j from: $\langle x_i, y_i\rangle$ and $\langle x_j, y_j\rangle$:
(1) return:

$$\langle x_i, y_i\rangle \odot \langle x_j, y_j\rangle = \mathsf{Enc}(i, r_i) \odot \mathsf{Enc}(j, r_j) = \left\langle g^{r_i}, g^{ar_i} f^i \right\rangle \odot \left\langle g^{r_j}, g^{ar_j} f^j \right\rangle$$
$$= \left\langle g^{r_i+r_j}, g^{a(r_i+r_j)} f^{i+j} \right\rangle = \mathsf{Enc}(i + j, r_i + r_j).$$

ReEnc: To re-encrypt a given ciphertext $\mathsf{Enc}(i, r) = \langle x, y\rangle$ *i.e.,* to transform it into a ciphertext with a different nonce, one needs only to know the public key K_{pub}:
(1) generate s at random.
(2) return:

$$\mathsf{Enc}(i, r + s) := \langle g^s x, h^s y\rangle .$$

Computing a tally (decryption) corresponds to computation of a discrete logarithm of f^m, but since this value is limited by a number of voters, it is feasible.

4.3. Mix nets

An ElGamal mixing server takes N inputs – ciphertexs $\{c_i\}_{i\in[N]}$, permutes them according to a random permutation π and re-encrypts each of them using random factors $\{s_i\}_{i\in[N]s}$. Output of a mixing server is $\{d_i\}_{i\in[N]}$, where $d_i = \mathsf{ReEnc}(c_{\pi(i)}, s_i)$.

4.3.1. Shadow-Mix. The following protocol that verifies correctness of the mixing process was proposed by Sako and Kilian in (SK95).

- *The mix server* creates a shuffle defined by $M = \langle \pi, \{s_1, \ldots, s_N\}\rangle$ and a *shadow mix* with $SM = \langle \pi', \{t_1, \ldots, t_N\}\rangle$ (see Figure 4.5) with the same inputs but different permutation and different exponents used for re-encryption.
- *The Verifier* chooses a bit $b = 0, 1$.
- *The mix server* if
  b=0: reveals the shadow mix SM (both the permutation and exponents).
  b=1: reveals the difference between the two mixes, *i.e.,* the permutation that transforms outputs of the two mixes.

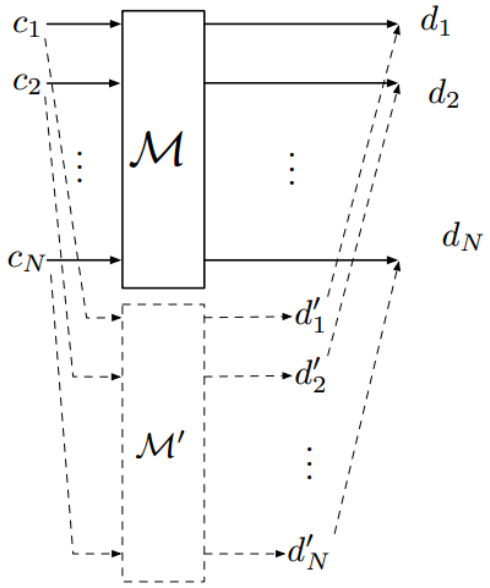


Figure 4.5. *Shadow-Mix* proof of shuffle.

An hones mix server is always able to correctly answer to either questions while a server that did not follow the protocol can correctly answer at most to one of them (can cheat undetected with probability $\leq 1/2$). To increase the assurance of integrity, a mix server needs to prepare t such a shadow mixes, and the procedure is run for each of them, resulting with probability $1 - 2^{-t}$ that the primary mix is correct.

4.4. Zero-knowledge proofs

A *zero-knowledge proof of knowledge* is a protocol that is run between a *Prover* and a *Verifier* where a prover convinces a verifier about a statement, such that the following properties hold:

Completeness: an honest prover always convinces an honest verifier of the validity of the statement.

Soundness: a dishonest prover can cheat only with small probability.

Zero knowledge: no other information is revealed.

Proof of knowledge: can extract witness from a successful prover.

4.4.1. Zero-knowledge proof of decryption. A decryption of an ElGamal ciphertext can be proven using Chaum-Pedersen protocol (CP92) for proving plaintex equality.

To prove that a given ciphertext $c = \langle x, y\rangle$ encrypts a plaintext m, the prover shows that $\log_g(y) = \log_x(y/m)$:

Prover: (1) selects $w \in Z_q$.
(2) sends $\langle A, B\rangle = \langle g^w, x^w\rangle$ to the verifier.

Verifier: chooses a random challenge $c \in Z_q$.

Proover: answers with: $t = w + ac$

Verifier: checks if:

- $g^t = Ah^c$
- $x^t = B(y/m)^c$

4.5. Non-interactive zero-knowledge proofs

Zero-knowledge proofs have one drawback which becomes an issue in the case of auditing elections – they are interactive and their transcript does not constitute a proof to someone who does not take a part in the protocol. To overcome that problem there are two approaches: the first one relies on using a public source of verifiable randomness like stock exchange data (CEA07; CH10); the other approach is to transform a proof-scheme into an *non-interactive zero knowledge proof*. The transformation is done using the Fiat-Shamir heuristic (FS86) the idea of simulating a challenger is that challenge bits are computed as the hash of all data that prover is commiting to.

The proofs of the correctness of mixing (both Shadow-Mix and RPC) and proof of decryption can be transformed into non-interactive form.

Crypto-blocks

1. What is the verification procedure for the (simple) RSA blind signature scheme? Write down the corresponding equations. Write down the procedure for the RSA blind singature and verification which use RSA-PSS padding.
2. Consider the following commitment scheme.

Commit: The Sender generates a key k and sends $\mathsf{Enc}_k(M)$ to the Receiver,

Reveal: The Sender sends k to the Receiver, the Receiver can decrypt the message.

What is wrong with the above scheme (as a commitment scheme)?

---

s

Chapter 5

End to end verifiability

In the previous sections we were analysing various voting schemes, we mainly focused our attention on checking if a system assures that the three steps can be verified:

- cast as intended,
- recorded as cast,
- tallied as recorded.

This simple test of the verifiability of the voting system is very usefull but at the same time is not sufficient. We already saw problems that may not be detected by this simple approach *i.e.,* ThreeBallot, straightforward implementation of homomorphic encryption.

In this chapter we discuss a more complete definition of what an end-to-end verifiable voting system is. We are presenting approach introduced in (3).

In order to check if a given voting protocol is end-to-end verifiable, one needs to show that the protocol passes all the checks:

(1) **Presented ballots are well-formed** – this check ensures that the representation of the voter's choices on the ballot are consistent with the representation of the ballot by the rest of the voting system, *i.e.,*:
   (a) If the ballot to be cast by the Voter is not well-formed then, the Voter, at any time after the election is able to detect if the vote

she is about to cast does not represent a vote for the candidate(s) she intended.

(b) The probability that the Voter does not detect her incorrectly formed ballot is strictly less than one.

(c) The Voter has a publicly acceptable, irrefutable proof of malfeasance if she detects that her ballot is not well formed.

The probability of detection by two different voters should be independent. Moreover, a voter should be able to check that a ballot she uses has not been issued to another voter.

(2) **Cast ballots are well-formed** – this checks ensures that cast ballots do not contain over-votes or negative votes, *i.e.,*:

(a) For any cast ballot B that is not well-formed, and is to be included in the tally, anyone, at any time after the election, is able to detect that B is incorrectly formed.

(b) The probability that no one detects that the cast ballot B is incorrectly formed is at least 2^{-20} (see (3) for discussion).

(c) A proof that a ballot is incorrectly formed is publicly acceptable[1] and irrefutable[2].

(3) **Recorded as cast** – the ballot that was cast by the voter is the one that was recorded by the system, *i.e.,*:

(a) If the Voter's cast ballot is not correctly recorded then, the Voter, at any time after election, is able to detect with the probability strictly greater than zero that her cast ballot is incorrectly recorded.

(b) If the Voter detects that her cast ballot is incorrectly recorded then she has a publicly acceptable proof of malfeasance.

(c) Detection probability is strictly greater than zero, the probability that the central claim is true is not negligible (but it does neet to be irrefutable).

(4) **Tallied as recorded** – the recorded votes are counted correctly, *i.e.,*:

(a) If n recorded ballots are incorrectly tallied, then at any time after the tally has been made public, anyone is able to detect that the declared tally does not represent the tally of all the recorded votes.

[1] A *publicly acceptable* proof is a proof that can be verified independently by anyone.

[2] An *irrefutable proof* is a proof in which the probability that the central claim is false is smaller than 2^{-112} (exact probability comes from NIST SP 800-57).

(b) The probability that one does not find an error in the declared tally is at most $c(n) < 1$ (a function specified by the system design).

(c) The proof of malfeasance is irrefutable.

(5) **Consistency** – verifies if the set of ballots subject to the *recorded as cast* check is the same as the set of ballots subject to the *tallied as recorded* check, *i.e.,*:

(a) If sets of tallied ballots is different than the set of ballots that voters are able to check in *recorded as cast*, then at any time after the tally has been made public by the election officials, anyone is able to detect that the two sets are different.

(b) The probability that one cannot detect that the two sets are different is smalle than ϵ[3].

(c) If the sets are different then the proof of malfeasance is publicly acceptable and irrefutable.

This checks verifies the chain of custody.

(6) **Each recorded ballot is subject to the "recorded as cast" check** – this ensures that no ballots that could not have been checked by at least one voter are included in the final tally, *i.e.,*:

(a) If a cast ballot B does not have a unique voter that is able to check it during the *recorded as cast* phase, then at any time after the election, anyone is able to detect that the ballot B does not have a unique corresponding voter.

(b) The probability that no one can detect that for n cast ballots, there are no corresponding voters should be lower than p_n.

(c) If one detects that some ballots do not have unique corresponding voters then the proof is publicly acceptable and irrefutable.

(7) **Irrefutable proof of following procedures** – for every integrity-preserving procedure there must be a check which can detect when the voting system does not follow the protocol. Moreover this check should provide a public proof.

---

[3]The suggested by (3) value of $\epsilon = 2^{-30}$.

E2E Definition

1. Explain why traditional paper ballot systems, optical scan systems and DREs do not satisfy the *Consistency* check and thous are not end-to-end verifiable.

References

[3] Popoveniuc, Stefan and Kelsey, John and Regenscheid, Andrew and Vora, Poorvi, **Performance requirements for end-to-end verifiable elections**, Proceedings of the 2010 international conference on Electronic voting technology/workshop on trustworthy elections, http://static.usenix.org/events/evtwote10/tech/full_papers/Popoveniuc.pdf

Chapter 6

Polling place

As mentioned in the Chapter 1, there are two main scenarios of use of electronic voting systems at polling places. The first one relies on electronic machines. The second assumes use of paper ballots which are later scanned.

DREs: a ballot has only an electronic form. Some DREs are equipped with VVPAT (voter verifiable paper audit trail) – a voter can see, behind a transparent shield that a receipt of her vote printed. Such a receipt is a part of *verified paper record* and thous can be used to audit the machine (not by a voter but by election auditors).

marking devices: a voter interacts with a machine which prints a marked ballot, a voter can check if a ballot is marked as intended and cast a ballot (through a ballot-box or a scanner).

There are also hybrid solutions which assume that a voter uses a computational assistant to cast a ballot (CFN$^+$12) or uses a dedicated machine to check if a ballot is marked correctly before it is cast (SFCC11).

There is a series of schemes that implement *commit and prove* approach of Benaloh (Ben06) scheme: VoteBox (SDW08), Wombat Voting. In the next section we show importance of the commitments – we present an attack on a paperless implementation of Benaloh scheme (VoteBox). The problem lies in the fact that a voter is unable to verify a commitment in a voting booth.

6.1. Paper is needed(?)

We limit the description of the VoteBox system to the fragile part (Election day: casting votes), we leave original numbering of the protocol steps, according to (SDW08).

(1) The poll worker authorizes from the supervisor console one of the booth machines. Supervisor console broadcasts an authorization message (and ballot definition if needed) directing the selected machine to interact with a voter.
(2) The booth presents the ballot to the voter.
(3) The voter makes selection on a screen of a VoteBox machine.
(4) The booth shows a review screen, listing the voter's choices.
(5) If the voter needs to make changes goes back to Step 2, if not goes to the next step.
(6) The booth:
  (a) encrypts and publishes cast ballot.
  (b) presents "challenge or cast" screen.
(7) The voter chooses between "to cast" or "to challenge".
(8) The booth:
  to cast: logs the voter's choice (to cast).
  to challenge: publishes the value r.

Figure 6.1. VoteBox – ballot casting protocol

The original Benaloh's scheme (Ben06) uses a paper as a "commitment medium". Two Benaloh's scheme implementations: Helios and VoteBox use a "computer screen" as a commitment medium. For the first sight both Helios and VoteBox offer exactly the same – an electronic version of the Benaloh's challenge. The difference is that in the case of Helios, a voter may use a different machine (*e.g.,* smart-phone) to verify a commitment while in the case of the VoteBox, a voter does not have access to any computational device since it is forbidden at polling places. Because of that fact, the following attack is possible.

Attack. An attacker needs to modify a code run by voting machine on one of the booths. The modification is as follows: the booth always encrypts a voter's choice exactly as a voter wanted. So if a voter performs a challenge then the result of every audit is correct. But when a voter presses the cast button then the machine publishes r (making the ballot cast invalid and pretending that a voter initiated an audit). Just after

that, the attacker (being supervisor console's operator) authorizes the booth to cast another vote. The booth prepares another encrypted ballot (with appropriate choices) and cast that.

6.2. Scantegrity

The Scantegrity II (CCC+09; CCC+10b) scheme is an enhancement for traditional paper ballots that are used in optical scan systems. The system offers much higher election integrity thanks to use of confirmation codes that are printed in invisible ink[1] (the system can also use scratch-offs instead of invisible ink). A voter marks her candidate just as in traditional paper systems but with a special pen, marking reveals a code (see Figure 6.2) that a voter can write down on a receipt.

A voter can choose to audit ballot printing – to verify if codes are printed correctly (to ensure *cast as intended*). Voters are able to verify if their votes are correctly included (*recorded as cast*) in the tally without revealing how they voted (the code itself does not reveal a vote). The scheme allows anyone to verify that the tally is computed correctly (*tallied as recorded*).

Figure 6.2. Scantegrity II uses invisible ink. To reveal a code a voter uses a special pen that develops a code corresponding to her choice. The code is used as a receipt.

[1] If codes are printed in regular ink – like in the Scantegrity I then the dispute resolution procedure is less efficient.

	#32490
Jerry	$VT8$
Edgar	$J4P$
Ervin	$LA4$
Donald	$DJ9$
	#32490

(a) A ballot seen by a voter – codes corresponding to candidates are hidden thanks to invisible ink (or scratch-off surface).

	#32490
Jerry	$VT8$
Edgar	$J4P$
Ervin	$LA4$
Donald	$DJ9$
	#32490

(b) A vote for *Ervin* reveals a code *LA4*.

	#32490
Jerry	$VT8$
Edgar	$J4P$
Ervin	$LA4$
Donald	$DJ9$

LA4	#32490

(c) A voter copies a code to a receipt slip which can be taken home for verification.

Figure 6.3.

References

[4] David Chaum and Aleks Essex and Richard Carback and Jeremy Clark and Stefan Popoveniuc and Alan T. Sherman and Poorvi Vora, **Scantegrity: End-to-End Voter Verifiable Optical-Scan Voting**, IEEE Security and Privacy, 2008.

[5] J. Benaloh and M. Byrne and P. Kortum N. McBurnett and O. Pereira and P. B. Stark and D. S. Wallach, **STAR-Vote: A Secure, Transparent, Auditable, and Reliable Voting System.**, 2012 arXiv preprint arXiv:1211.1904.

Chapter 7

Remote voting

Chaum proposed the first approach to vote encryption where the voter does not have access to trusted computational power (Cha04) while voting. This resulted in an *independently-verifiable* voting protocol whose verifiability properties did not rely on the voting system or election officials to correctly perform their functions. Many protocols with similar properties followed (Nef04; CRS05; Ben06; PH06; RS06; AR06; MN07; RTS07; CEC+08; CCC+08). Some of the corresponding voting systems are very practical for verifiable in-person voting and have been deployed in real elections (Pun; CCC+10a). These voting systems do not, however, serve the needs of absentee voters.

A large number of voters in the US have no choice but to vote absentee because they are located outside the country. For example, more than 333,000 votes were cast under the Uniformed and Overseas Citizens Absentee Voting Act (UOCAVA) in the 2006 US election (Com, page 4). Additionally, many voters choose to vote absentee. Mail-in ballots are widely used in the US: ballots in the state of Oregon are exclusively cast by mail, while about 80% of those in the state of Washington are, and about half of US states allow ballots to be cast absentee without voters having to provide a justification for the choice (Com). One of the most significant problems with absentee voting is that marked ballots are not always received in time — 19% of absentee ballots cast in the US Election of 2008 were not received in time to be counted. For this reason, a system that enables voters to check

that their ballots have been delivered in time and correctly would be very useful.

Some countries have introduced proposals for internet voting, or even run elections with such systems. Jefferson *et al.* provide a security analysis of SERVE, a proposal for the U.S. Department of Defence, and discover significant vulnerabilities (JRSW04). More recently Clarkson *et al.* examine a system developed by Scytl for the Okaloosa Distance Balloting Pilot and also discovered significant vulnerabilities (CHI+08). Internet voting systems have also been used in Estonia, The Netherlands, and Switzerland. The Netherlands system, REIS, purported to be verifiable however flaws were discovered that undermined both its verifiability and ballot secrecy (Gro04; HJS+08; GHH+09). A recent proposal has been made for a (partially) verifiable remote voting system for Norway (Gjo10).

Aside from the governmental use of REIS, verifiable remote systems have been used in other binding elections: *Helios* (Adi08) for the Recteur election at the Universite Catholique de Louvain (UCL), Belgium (AdPQ09) and student elections at Princeton University; and *SCV* (KZ10) for student elections at Collegium Civitas, Poland and works councils elections at Sanofi-Aventis and Zentiva. Helios makes it possible to tell if a vote has been modified, but it is impossible to tell whether the modification was caused by election authorities or by malware on the voter's machine. Those issues are at least partially addressed by *Scratch-Click-And-Vote (SVC)* (KZ10) and its predecessor (KZ07), however, both solutions are too cumbersome for use in a real public election.

In the research literature, one approach to mitigating the issues of voting from an untrusted platform is to use code voting. This approach originated with Chaum's *SureVote* (Cha01) and there are many proposals for taking it in various directions (HS07; JR07; HSS08; OSL08; JRF09; RT09; HLL10; Pop10). Remotegrity belongs in this class of systems.

The literature also addresses the tangential problem of coercion resistance in remote voting scenarios where vote casting is unsupervised. This line of research was originated by Juels *et al.* (JCJ05) and implemented as *Civitas* (CCM08). Recent improvements include more efficient tallying (AFT07) the use of panic passwords (CH11). These systems assume the

voter votes on a trusted machine, which is not an assumption made by Remotegrity.

7.1. Remotegrity

The Remotegrity part of a voting system is a distributed protocol with a Dealer (EA) who enables N participants (voters) to efficiently and securely send their messages (encoded ballots in the case of voting) to a public Bulletin Board.

The system uses a combination of physical and electronic media/channels, and is hence not a fully-electronic system, but a hybrid system. The hybrid nature of the system enables several of the security properties not present in fully-electronic systems. Credentials, issued by the Dealer to participants, are printed on paper forms (covered by scratch-off layers) and sent to participants by postal mail. Participants are allowed to send their messages over the Internet. If participants discover problems with the Internet channel, they may mail their messages back with their credentials, using the postal system or physical delivery.

We assume that each participant x_i of the protocol wants to post $V(x_i)$ on the Bulletin Board. In the case where Remotegrity is used with a voting system, $V(x_i)$ will denote voter x_i's coded vote. We denote by $S(i)$ the bit representing success, at protocol end, for the i^{th} voter: $V(x_i) \in BB \Leftrightarrow S(i) = 1$. The protocol ends with success if $S(i) = 1 \; \forall \; i$.

We do not require the Dealer to be honest. Moreover, our security arguments assume a powerful Adversary (such as malware) which has the same knowledge as the Dealer and colludes with the Dealer. We do require, however, that, of all computers used by the participant (voter) to check the bulletin board, at least one is honest (though the participant may not know, *a priori*, which one is honest). The goal of the Adversary is to change a few messages and publish them incorrectly—the Adversary wishes to achieve, at protocol end, $S(i) = 0 \; \forall \; i \; \in \; C$; where C is some subset of participants. That is, for all participants $x_i \; \in \; C$, the Bulletin Board publishes a message $m \neq V(x_i)$. It is not sufficient for the Adversary to prevent the message from being published. This is because, given the presence of at least one honest computer with which to view the Bulletin Board, the voter can determine that her message is not published and deliver it by postal mail. We describe

how she can do so using her Authentication Card. Thus, at the end of the protocol, for all participants x_i, either $S(i) = 0$ (the message is published incorrectly) or $S(i) = 1$ (the message is published correctly). (We provide more rigorous support for these statements later in the paper; now, it suffices to say that a voter will notice if his or her message is not posted and will be able to ensure it is posted, hence the message will be posted, either correctly or incorrectly.)

Our goals for the ith voter at the end of the (hybrid) protocol are:

- $S(i) = 1$, which means $V(x_i) \in BB$, or
- $S(i) = 0$ and, with non-negligible probability, x_i has verifiable proof that the Dealer cheated.

7.1.1. Setup. We use the following parameters:

N: - number of voters,

M: - security parameter (number of Authentication Cards opened during the print audit),

K: - a "usability" parameter (number of attempts a voter may make to return his coded vote over the Internet, using different computers if necessary). If the voter is not able to post his message correctly after K attempts, he would need to use the physical channel. If he mails it in, he can check that it is correctly posted, and, if not, as a last resort, he can vote at the polling place and avail of the security properties of the polling-place end-to-end paper-ballot-based voting system.

There is also another security parameter L – the length of numbers used for authentication. This will be described later.

Setup(N, M, K). A Dealer generates tuples

$$t_i = (serial_i, LockIn_iA, LockIn_iB, LockIn_iS, OTP_i(1), OTP_i(2), \ldots, OTP_i(K))$$

and publishes commitments to $T = \{t_1, \ldots, t_{2(N+M)}\}$ of the form $com(t_i) = (serial_i, com(LockIn_iA), com(LockIn_iB), com(LockIn_iS), com(OTP_i(1)), \ldots, com(OTP_i(K)))$.

The parameter $OTP_i(j)$ is called a *one-time-password* (of length L), and all the one-time-passwords are distinct. That is, $OTP_i(j) \neq OTP_{i'}(j')$ for $(i, j) \neq (i', j')$.

$LockIn_iA$ and $LockIn_iB$ (of length L) are used by the voter to "lock-in" his vote (to confirm that what is posted on the bulletin board agrees with their intensions) and are similarly unique. $LockIn_iX \neq LockIn_{i'}Y$ for $i \neq i'$ and $X, Y \in \{A, B, S\}$ and similarly for $LockIn_iB$. Also $LockIn_iA \neq LockIn_{i'}B$ for all i, i'.

$LockIn_iS$ is a *system lock-in* and is used by election officials when a given ballot is mailed back (with authentication card).

7.1.2. Pre-election audit. A (pseudo)random process determines a subset $PreA(T) \subset T$ of size $N + M$ that is audited. The Dealer opens the commitments to tuples $t \in PreA(T)$, and auditors check the corresponding commitments published in Setup.

7.1.3. Printing. After a successful audit, the Dealer prints $N + M$ Authentication Cards, each corresponding to a tuple $t_i \in Printed(T) = T \setminus PreA(T)$. The following values:

(1) $LockIn_iA, LockIn_iB$,

(2) $LockIn_iS$,

(3) $OTP_i(1), OTP_i(2), \ldots, OTP_i(K)$.

are printed and are individually covered with scratch-off layers. Additionally, $serial_i$ is printed without scratch-off and is visible.

7.1.4. Print Audit. Auditors choose, uniformly at random, a subset $PrintA(T) \subset Printed(T)$ of size M. (The subset may also be chosen pseudo-randomly). The correctness of the printing process is checked by auditors who scratch off the layers and check the revealed values against the corresponding commitments opened by the Dealer. The Mailing Authority obtains the remaining set of N Authentication Cards $AC(T) = Printed(T) \setminus PrintA(T)$ from the Dealer and a set of N Ballots from the Ballot Maker. Each participant obtains one Authentication Card and one Ballot in a package with instructions on how to vote. It is also possible to print a larger number of ballots and authentication cards so that voters may obtain two of each, auditing one and using the other to vote.

7.1.5. Posting On BB. The Bulletin Board is run by the Dealer. It accepts either

(a): a message of the form (OTP, m), publishing a corresponding entry of the form:

$$(id, OTP, m, serial, hash, sign)$$

if (OTP, m) is a valid pair, or

(b): a message of the form $(LockIn)$, publishing:

$$(id, LockIn, serial, hash, sign).$$

if $LockIn$ is a valid LockIn

The message m for the voting problem is the serial number of the ballot and coded votes for various contests. id, $hash$ and $sign$ are the Bulletin Board item ID, a (chained) hash of the information published and the (chained) signature of the Bulletin Board.

Loosely speaking, (OTP, m) is a valid pair if and only if

(1) OTP is not yet published

(2) $\exists\ i, j$ such that OTP is a valid j^{th} one-time password of the i^{th} tuple $t_i \in AC(T)$

(3) If any other pair (OTP', m') has been published on the Bulletin Board such that OTP' is a valid j'^{th} one-time password of the same i^{th} tuple t_i, the ballot serial numbers for m and m' are identical and

(4) Neither $LockIn_iA$ nor $LockIn_iB$ is published on the Bulletin Board.

This ensures that, loosely speaking,

(1) An OTP is used only once (so that an adversary may not use a voter's already-posted OTP to change his or her vote or to cast another vote).

(2) OTP is a valid authentication code on some authentication card (so that only those in possession of valid authentication cards can post messages).

(3) A voter updates only the vote corresponding to her ballot and does not attempt to change someone else's ballot by using their already-posted ballot serial number with her OTP and

(4) An adversary does not attempt to change a locked-in vote (because by locking-in her vote the voter communicates that she will not further change it).

The Bulletin Board may also provide some kind of error-detection for the message itself, if this is incorporated into the lock-ins, OTPs, ballot serial numbers and coded votes.

When an honest computer checks information on the Bulletin Board, it checks the correctness of the hash and the signature, that all items are ordered by id, and that items of lower id are present. It also checks consistency of the bulletin board with previous versions of the Bulletin Board if it has viewed the Bulletin Board previously. Similarly, $LockIn$ is a valid LockIn if and only if $LockIn$ is not yet published and is either $LockIn_iA$ or $LockIn_iB$ for the tuple $t_i \in AC(T)$.

Note that the above criteria will be checked on the published (signed) bulletin board. We anticipate a significant time delay (three hours) between the the receipt of a message and its formal publishing with digital signatures. Thus a voter should provide a few hours between the sending of two messages to the Bulletin Board. (Note that we have not incorporated some of the criteria above in the pseudo-code yet.)

Let l denotes the current number of entries on the BB:

Post(OTP, m, l**):** procedure

(1) finds i, j such that $OTP = OTP_i(j)$
(2) if $OTP_i(j)$ is not yet published then {
(3) $serial := serial_i$
(4) $hash := hash(BB[l], OTP_i(j), m, serial)$
(5) $sign := sign(sign[l], OTP_i(j), m)$
(6) $(l + 1, OTP, m, serial, hash, sign) \rightarrow BB$
(7) $BB[l + 1] := [(OTP_i(j), m, hash, sign]$
}

Post($LockIn, l$**):** procedure

(1) finds i, α such that $LockIn = LockIn_i\alpha$
(2) if $LockIn_i\alpha$ is not yet published then {
(3) $serial := serial_i$
(4) $hash := hash(BB[l], LockIn_i\alpha)$
(5) $sign := sign(sign[l], LockIn_i\alpha)$
(6) $(l + 1, LockIn_i\alpha, serial, hash, sign) \rightarrow BB$

(7) $BB[l+1] := [LockIn_i\alpha, hash, sign]$
}

7.1.6. Sending messages. Each participant (a voter) performs the following protocols which consist of three procedures. In the first procedure, the participant scratches off any one-time password of her choice. She then sends the pair—the message and the recently-scratched-off one-time-password—to the Dealer. In the voting problem, the message consists of the Ballot serial number and the coded vote(s).

In the next procedure, the participant checks the Bulletin Board, preferably from one or more different computers, to see if the message is published as described above (i.e. participant's computer checks the hashes, signatures, ordering, etc.) If she finds that the message is not posted or is posted incorrectly, she repeats the first procedure with another OTP, and then repeats the second procedure. This continues till she runs out of OTPs, at which time she goes to vote in person.

If she does not run out of OTPs and is satisfied that her vote was posted correctly, she performs the third procedure, where she scratches of one of the lock-ins of her choice and sends it in.

She does not scratch off values she does not need.

Send(i, attemptNo): procedure

(1) $success := false$
(2) while($!success$ && $attemptNo < K$) {
(3) $(V(x_i), OTP_i(attemptNo)) \rightarrow BB$
(4) $success := [(V(x_i), OTP_i(attemptNo)) \in BB]$
}

Check(i, attemptNo): procedure – should be performed from a different location, on a different machine than the one used for the **Send** procedure.

(1) if $(V(x_i), OTP_i(attemptNo)) \in BB$
then: go to **LockIn** procedure
else: if $attemptNo < k$ go to **Send** procedure
else: go to polling site

LockIn(): procedure

(1) pick at random $\alpha \in \{A, B\}$
(2) $LockIn_i\alpha \rightarrow BB$
(3) $success := [LockIn_i\alpha \in BB]$

7.1.7. Post audit. A Dealer opens and publishes all tuples.

7.1.8. Using the Physical Channel. When a voter appears to vote in person, she provides a valid ID. Polling officials have noted down the correspondence between $serial_i$ and the voter, and cancel all previous votes cast using authentication codes from tuple t_i. The voter then casts a vote using a new ballot.

7.1.9. Voter's perspective. The authentication card (see Figure 7.1) contains two sets of numbers under scratch-offs: *one-use or one-time passwords* (i.e., $9764 - 5930 - 4195 - 1472$) and *audit codes or lock-in codes* (i.e., $3509 - 4903 - 8255 - 8937$) on the bottom. It also contains a visible *serial number* (i.e., $EB3C15$) and a system password (i.e., 9768603372754008) – a number that is posted online if a corresponding ballot is received physically by the Election Authority.

Voters are able to:

vote by mail: A voter marks her choices on the Ballot, puts it into a reply-envelope and mails it back (exactly as in traditional vote-by-mail).

vote online:

The voter choosing to vote online performs the following steps.

- She uses the paper ballot to determine the confirmation number corresponding to her vote. With the Scantegrity ballot shown in Figure **??**, for example, the receipt value corresponding to candidate *Candidate 1* as first choice is Scantegrity confirmation number 6055.
- To cast her ballot, the voter goes to the Absentee Voting Website and enters the Scantegrity confirmation number of the candidate she selects and a one-use password (OTP) scratched-off at random from the authentication card (see Figure 7.1.9).

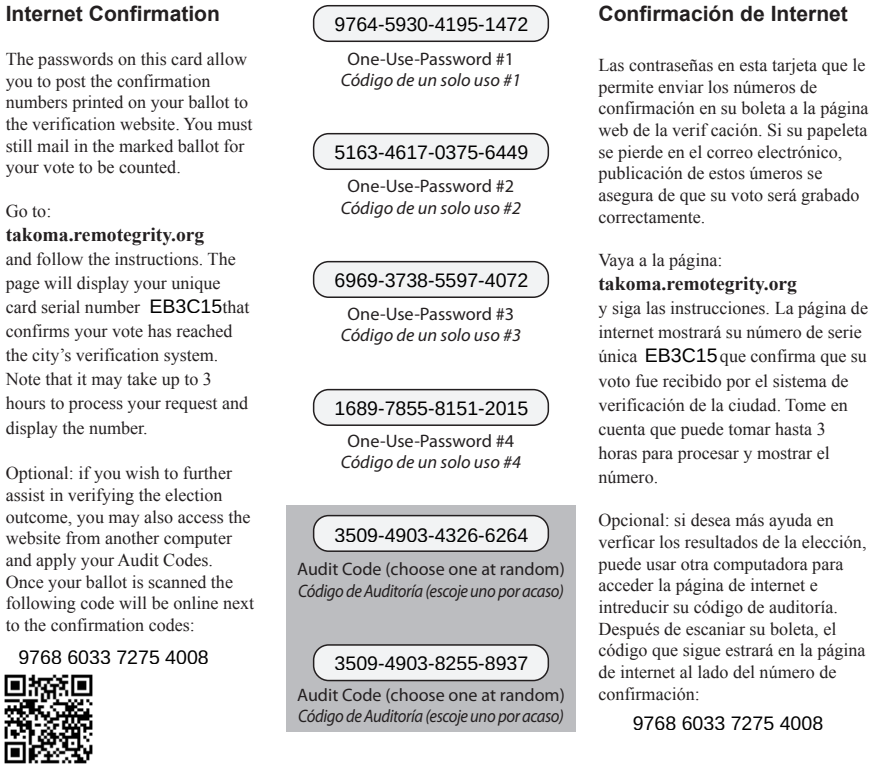
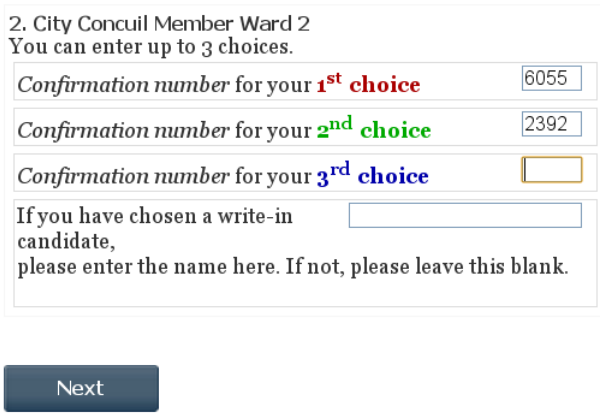


Figure 7.1. Remotegrity *Authentication Card*. All *one-use-passwords* and *audit codes* are under scratch-off.

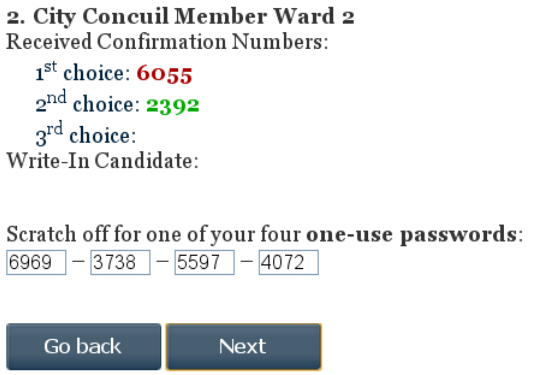
Note that: An adversary that intercepts her submission will not know which candidate the voter is attempting to vote for. Further, the adversary cannot reliably change the vote to a value corresponding to another candidate on the voter's ballot. The adversary could be malicious software on the computer she is using to cast her ballot or any party along the route.

- Once the values are submitted, the voter conducts an initial verification that the website correctly posts these values. The verification is preferably performed from a different computer. Some voters check the digital signatures on the ABB data.

- After waiting $T + 1$ hours, a voter may choose to go back to the Absentee Bulletin Board (ABB), preferably but not necessarily from yet another computer or device. She checks that her confirmation number, authentication card serial number and one-use password are correctly listed.



(a) Voter enters confirmation codes corresponding to her choice



(b) Voter checks her choice and enters one time password

Figure 7.2. The Online Vote-Casting Procedure

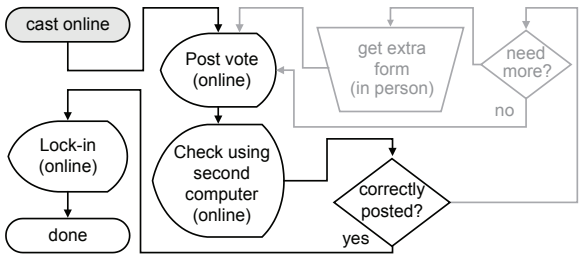


Figure 7.3. The example vote casting flow chart.

Note that: The voter did not submit the authentication card serial number when casting her vote, and the display of the proper number weakly authenticates the update.

- If the information is correct, she locks-in her vote by submitting a randomly scratched-off lock-in code, which is then displayed on the ABB.

Note that: This completes the voting process, however she is encouraged to monitor the ABB, either by just looking at it and checking that her numbers are correctly posted, or by using a program that checks digital signatures and previously-posted data for consistency.

- If the ABB does not display her confirmation number, the voter may retry by sending in her confirmation number from another computer, with the same one-use password.

  Note that: If it displays an incorrect number, she may retry, but should use a different one-use password.

  Note that: If the voter revisits the ABB after locking-in and the ABB displays another one-use password or lock-in code against her authentication card serial number, she may have proof of cheating (with non-negligible probability) because the numbers may not be the ones she scratched-off. Additionally, if she checks digital signatures immediately after the initial verification or after locking-in, she has proof if the ABB later attempts to change any data. Voters can choose the physical channel at any time during the protocol, or even later. Paper ballots over-rule electronically-communicated confirmation numbers, and a vote cast in person on election day over-rules all other votes.

Further, note the following: commitments to the printed ballots and cards are opened as necessary after the election, and absentee packages containing ballots and authentication cards are audited before the election. The ABB can detect errors in single digits in the confirmation numbers, as well as the transposition of two digits. It can also detect more errors in the codes and ballot serial numbers.

The offline server can determine whether a confirmation number is valid (but not what candidate it represents). It hence does not sign an invalid confirmation number. In such cases, the ABB will post, instead, a note denoting that the received confirmation number for a particular ballot number was invalid. The offline server knows all correspondences between one-use passwords, lock-in codes and authentication card serials, all of which are unique.

It is hence able to determine the authentication serial number corresponding to a one-use password, and is able to link all confirmation numbers sent using the same authentication card.

To summarize the voter's experience if voting online (also see Figure 7.3):

(1) visits the election website and:
   (a) enters confirmation codes corresponding to her choices (from the Scantegrity ballot),
   (b) learns one of the OneTimePasswords by scratching off the layer from the Authentication Card, and enters that password.

(2) visits the election website (possibly) from another computer and checks if the data sent by her in the previous step was correctly posted:
   if correctly posted: she goes on to the next step,
   if incorrectly posted: she repeats the previous step with a fresh OneTimePassword.

(3) visits the election website and enters the value of one of the LockIns (learns the value of the LockIn by scratching-off one at random).

Chapter 8

Summary

Currently used electronic voting systems are very dangerous to the democratic process. This script presented techniques that can be used to construct electronic election schemes that are free of the threats of the current systems – we call such schemes *end-to-end verifiable voting schemes*. E2E schemes protect election integrity but may introduce new threats that are not existing in the current systems.

Everlasting privacy. Current paper based systems provide no verifiability but at the same time they protect ballot secrecy quite well. Electronic systems in order to be verifiable post some data online to provide voter-verifiability. Data are protected by use of encryption/commitments but what happens if an underlying encryption scheme becomes broken? One can determin how people voted!

This is the reason why some researchers propose the following approach: instead of having *unconditional integrity* and *computational privacy* they propose to use encoding that quarantees *unconditional privacy* while having *computational integrity*.

Audits. Electronic nature of a voting scheme may badly influence on auditing. One may ask at least the following question: what happens in the case when an audit fails? And then

- is it possible to recover the results?
- or do we need to rerun whole election?

In the case of paper-audit trail, some systems offer possibility of a manual recount. If this is not a case and a problem is identified in some polling stations then some electronic voting schemes allow for rerunning elections only in that stations.

The situation is more complicated if a system does not support auditability (or dispute resolution) – auditability allows, in the case when audit fails, to blame a party that is responsible for the failure. Most internet voting schemes do not have any mechanisms that would allow to tell if quilty is: an election authority, a browser, a virus or a voter.

List of Figures

References

[Adi08] Ben Adida, *Helios: Web-based Open-Audit Voting*, Proceedings of the Seventeenth Usenix Security Symposium (USENIX Security 2008), June 2008.

[AdPQ09] Ben Adida, Olivier deMarneffe, Olivier Pereira, and Jean-Jacques Quisquater, *Electing a university President using open-audit voting: Analysis of real-world use of Helios*, EVT/WOTE 2009, Proceedings of the Electronic Voting Technology Workshop / Workshop on Trustworthey Elections, August 2009.

[AFT07] Roberto Araujo, Sebastien Foulle, and Jacques Traoré, *A practical and secure coercion-resistant scheme for remote elections*, Frontiers of Electronic Voting, 2007.

[AR06] Ben Adida and Ronald L. Rivest, *Scratch & Vote: self-contained paper-based cryptographic voting*, WPES '06: Proceedings of the 5th ACM Workshop on Privacy in the Electronic Society (New York, NY, USA), ACM Press, 2006, pp. 29–40.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau, *Minimum disclosure proofs of knowledge*, J. Comput. Syst. Sci. **37** (1988), no. 2, 156–189.

[Ben06] Josh Benaloh, *Simple verifiable elections*, EVT'06: Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop (Berkeley, CA, USA), USENIX Association, 2006.

[Bow] Debra Bowen, *California secretary of state, voting systems review: Top-to-bottom review*, Website, http://www.sos.ca.gov/elections/elections_vsr.htm.

[CCC+08] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman, *Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes*, EVT'07: Proceedings of the USENIX/Accurate Electronic Voting Technology on USENIX/Accurate Electronic Voting Technology Workshop, USENIX Association, 2008.

[CCC+09] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, Alan T. Sherman, and Poorvi L. Vora, *Scantegrity ii: end-to-end verifiability by voters of optical scan elections through confirmation codes*, IEEE Transactions on Information Forensics and Security **4** (2009), no. 4, 611–627.

[CCC+10a] Richard Carback, David Chaum, Jeremy Clark, Aleksander Essex, Travis Mayberry, Stefan Popoveniuc, Ronald L. Rivest, Emily Shen, Alan T. Sherman, and Poorvi L. Vora, *Scantegrity II Municipal Election at Takoma Park: The First E2E Binding Governmental Election with Ballot Privacy*, Proceedings of the Ninteenth Usenix Security Symposium (USENIX Security 2010), August 2010.

[CCC+10b] Richard T Carback, David Chaum, Jeremy Clark, John Conway, Aleksander Essex, Paul S. Hernson, Travis Mayberry, Stefan Popoveniuc, Ronald L. Rivest, Emily Shen, Alan T Sherman, and Poorvi L. Vora, *Scantegrity II election at Takoma Park*, USENIX Security Symposium, 2010.

[CCM08] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers, *Civitas: A secure voting system*, In IEEE Symposium on Security and Privacy, 2008.

[CEA07] Jeremy Clark, Aleksander Essex, and Carlisle Adams, *Secure and observable auditing of electronic voting systems using stock indices*, IEEE Canadian Conference on Electrical and Computer

Engineering, 2007.

[CEC+08] David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan T. Sherman, and Poorvi Vora, *Scantegrity: End-to-end voter verifiable optical-scan voting*, IEEE Security and Privacy (2008).

[CFN+12] David Chaum, Alex Florescu, Mridul Nandi, Stefan Popoveniuc, Jan Rubio, PoorviL. Vora, and Filip Zagorski, *Paperless independently-verifiable voting*, E-Voting and Identity (Aggelos Kiayias and Helger Lipmaa, eds.), Lecture Notes in Computer Science, vol. 7187, Springer Berlin Heidelberg, 2012, pp. 140–157.

[CH10] Jeremy Clark and Urs Hengartner, *On the use of financial data as a random beacon*, EVT/WOTE, 2010.

[CH11] _____, *Selections: Internet voting with over-the-shoulder coercion-resistance*, FC, 2011.

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

[Cha82] _____, *Blind signatures for untraceable payments*, Advances in Cryptology: Proceedings of Crypto'82, 1982.

[Cha01] _____, *Surevote: Technical overview*, WOTE, 2001.

[Cha04] _____, *Secret-ballot receipts: True voter-verifiable elections*, IEEE Security and Privacy (2004), 38–47.

[CHI+08] Michael Clarkson, Brian Hay, Meador Inge, abhi shelat, David Wagner, and Alec Yasinsac, *Software review and security analysis of scytl remote voting software*, Tech. report, Report commissioned by the Florida Division of Elections, 2008.

[CKW08] Jacek Cichon, Miroslaw Kutylowski, and Bogdan Weglorz, *Short ballot assumption and threeballot voting protocol*, SOFSEM (Viliam Geffert, Juhani Karhumäki, Alberto Bertoni, Bart Preneel, Pavol Návrat, and Mária Bieliková, eds.), Lecture Notes in Computer Science, vol. 4910, Springer, 2008, pp. 585–598.

[Com] Technical Guidelines Development Committee, *VVSG recommendations to the EAC*, Delivered to the EAC on September 4, 2007 for consideration and public comment., (VVSG means "Voluntary Voting System Guidelines".).

[CP92] David Chaum and Torben Pryds Pedersen, *Wallet databases with observers*, CRYPTO, 1992.

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

[FS86] Amos Fiat and Adi Shamir, *How to prove yourself: practical solutions to identification and signature problems*, CRYPTO, 1986, pp. 186–194.

[GHH+09] Rop Gonggrijp, Willem-Jan Hengeveld, Eelco Hotting, Sebastian Schmidt, and Frederik Weidemann, *RIES—Rijnland internet election system: A cursory study of published source code*, VOTE-ID, 2009.

[Gjo10] Kristian Gjosteen, *Analysis of an internet voting protocol*, Tech. report, IACR Eprint Report 2010/380, 2010.

[Gro04] Jens Groth, *Review of RIES*, Tech. report, Cryptomathic, 2004.

[HJS+08] Engelbert Hubbers, Bart Jacobs, Berry Schoenmakers, Henk van Tilborg, and Benne de Weger, *Description and analysis of the RIES internet voting system*, Tech. report, Eindhoven Institute for the Protection of Systems and Information (EiPSI), 2008.

[HLL10] Sven Heiberg, Helger Lipmaa, and Filip van Laenen, *On e-vote integrity in the case of malicious voter computers*, ESORICS, 2010.

[HS07] Jorg Helbach and Jorg Schwenk, *Secure internet voting with code sheets*, VOTE-ID, 2007.

[HSS08] Jorg Helbach, Jorg Schwenk, and Sven Schage, *Code voting with linkable group signatures*, EVOTE, 2008.

[JCJ05] Ari Juels, Dario Catalano, and Markus Jacobsson, *Coercion-resistant electronic elections*, ACM WPES, 2005.

[JR07] Rui Joaquim and Carlos Ribeiro, *Codevoting: protection against automatic vote manipulation in an uncontrolled environment*, VOTE-ID, 2007.

[JRF09] Rui Joaquim, Carlos Ribeiro, and Paulo Ferreira, *Verytvote: A voter verifiable code voting system*, VOTE-ID, 2009.

[JRSW04] David Jefferson, Aviel D. Rubin, Barbara Simons, and David Wagner, *A security analysis of the secure electronic registration and voting experiment (SERVE)*, Tech. report, Report to the Department of Defense (DoD), 2004.

[KRMC10] John Kelsey, Andrew Regenscheid, Tal Moran, and David Chaum, *Attacking paper-based E2E voting systems*, Towards Trustworthy Elections, LNCS, vol. 6000, Springer, 2010, pp. 370–387.

[KTV10a] Ralf Kusters, Tomasz Truderung, and Andreas Vogt, *Accountability: Definition and relationship to verifiability*, ACM CCS, 2010.

[KTV10b] Ralf Küsters, Tomasz Truderung, and Andreas Vogt, *Proving coercion-resistance of scantegrity ii*, ICICS (Miguel Soriano, Sihan Qing, and Javier López, eds.), Lecture Notes in Computer Science, vol. 6476, Springer, 2010, pp. 281–295.

[KY02] Aggelos Kiayias and Moti Yung, *Self-tallying elections and perfect ballot secrecy*, PKC, 2002, pp. 141–158.

[KZ07] Mirosław Kutyłowski and Filip Zagórski, *Verifiable internet voting solving secure platform problem*, Advances in Information and Computer Security (Atsuko Miyaji, Hiroaki Kikuchi, and Kai Rannenberg, eds.), Lecture Notes in Computer Science, vol. 4752, Springer Verlag, 2007, pp. 199–213.

[KZ10] ———, *Scratch, click & vote: E2e voting over the internet*, Towards Trustworthy Elections, State-of-the-Art Survey Series, vol. 6000, Springer Verlag, 2010, pp. 343–356.

[MN07] Tal Moran and Moni Naor, *Split-ballot voting: Everlasting privacy with distributed trust*, ACM Computer and Communications Security, 2007.

[Nef04] C. A. Neff, *Practical high certainty intent verification for encrypted votes*, 2004.

[OSL08] Rolf Oppliger, Jorg Schwenk, and Christoph Lohr, *Captcha-based code voting*, EVOTE, 2008.

[PCC+08] Stefan Popoveniuc, Jeremy Clark, Richard Carback, Aleks Essex, and David Chaum., *Securing optical-scan voting*, Selected proceedings of Workshop on Trustworthy Elections, 2008.

[Ped92] Torben P. Pedersen, *Non-interactive and information-theoretic secure verifiable secret sharing*, CRYPTO, 1992, pp. 129–140.

[PH06] Stefan Popoveniuc and Ben Hosp, *An introduction to PunchScan*, IAVoSS Workshop On Trustworthy Elections (WOTE 2006) (Robinson College, Cambridge UK), June 2006.

[Pop10] Stefan Popoveniuc, *Speakup: remote unsupervised voting*, ACNS, 2010.

[Pun] *Punchscan*, http://www.punchscan.org.

[RS06] P. Y. A. Ryan and S. A. Schneider, *Prêt à voter with re-encryption mixes*, In D. Gollmann, D., J. Meier, and A. Sabelfeld, editors, ESORICS, volume 4189 of Lecture Notes in Computer Science, Springer-Verlag, 2006, pp. 313–326.

[RS07] Ronald L. Rivest and Warren D. Smith, *Three voting protocols: Threeballot, vav, and twin*, EVT'07: Proceedings of the USENIX/Accurate Electronic Voting Technology on USENIX/Accurate Electronic Voting Technology Workshop (Berkeley, CA, USA), USENIX Association, 2007, pp. 16–16.

[RT09] Peter Y A Ryan and Vanessa Teague, *Pretty good democracy*, Workshop on Security Protocols, 2009.

[RTS07] Ben Riva and Amnon Ta-Shma, *Bare-handed electronic voting with pre-processing*, EVT'07: Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop, USENIX Association, 2007.

[SDW08] Daniel R. Sandler, Kyle Derr, and Dan S. Wallach, *VoteBox: a tamper-evident, verifiable electronic voting system*, USENIX Security Symposium, 2008.

[SFCC11] Alan T. Sherman, Russell A. Fink, Richard Carback, and David Chaum, *Scantegrity iii: Automatic trustworthy receipts, highlighting over/under votes, and full voter verifiability*, EVT, 2011.

[SK95] Kazue Sako and Joe Kilian, *Receipt-free mix-type voting scheme: a practical solution to the implementation of a voting booth*, Advances in Cryptology - EUROCRYPT'95, 1995.

[Str06] Charlie Strauss, *A critical review of the triple ballot voting system. part 2: Cracking the triple ballot encryption*, http://www.cs.princeton.edu/~appel/voting/Strauss-ThreeBallotCritique2v1.5.pdf, October 2006.

[SW08] Daniel Sandler and Dan S. Wallach, *The case for networked remote voting precincts*, EVT (David L. Dill and Tadayoshi Kohno, eds.), USENIX Association, 2008.

Index