Tel Aviv University
Raymond and Beverly Sackler
Faculty of Exact Sciences
The Blatavnik School of Computer Sciences

An Implementation of Dual (Paper and Cryptograhic) Voting System

Submitted as a partial fulfillment of the
requirements towards the Master of Science degree by

Niko Farhi

The research work has been conducted
under the supervision of

Prof. Amnon Ta-Shma

March 2013

Abstract

This thesis reports on the design and implementation of a cryptographic voting system, named *Wombat*. The system is designed to retain the ”look and feel” of standard paper-based plurality voting, while enhancing security using methods from modern electronic voting literature. To achieve this, the system executes two voting processes in parallel: one is electronic and *end-to-end verifiable*, while the other is paper based and emulates more traditional processes (to which the voters are accustomed). Consistency between the two processes is enforced by means of a new specially-tailored paper ballot format.

In addition, this work examines the practicality of the Wombat protocol through implementation and field testing in two student council elections with over 2000 voters and party premiership elections with almost 900 voters. During these field test the usabilty, performance and voter satisfaction was examined. Overall, voters trusted the system and found it comfortable to use.

Parts of this work were presented in EVote2012.

Acknowledgments

I wish to thank my advisor, Prof. Amnon Ta-Shma for his patience with me.

I also wish to thank Mr. Ben Riva for providing aid when it was needed.

Contents

Chapter 1

Introduction

The foundations of modern cryptographic voting systems were laid in the 1990's introducing powerful techniques such as homomorphic tallying and mixing networks. Almost all early work assumes the voter has access to some trusted computational device at voting time. In 2004, Chaum [Cha04] and independently Neff [Nef04] proposed cryptographically secure voting systems in which the voter has access to no computational device at voting time, and since then most research has focused on such *bare-handed, end-to-end verifiable* voting systems.

In 2008 Benaloh [Ben08] suggested *dual voting*. In Benaloh's system the voter fills in a plaintext ballot and a scanning machine produces from it a printed plaintext ballot (that is cast into a ballot box) together with a cryptographic encryption (that is uploaded to a public web page) and an electronic receipt (that the voter takes home). The system is end-to-end verifiable using standard cut-and-choose techniques.

There are several advantages to dual voting. Cryptographic voting, in general, is more vulnerable than paper-based voting to *global* failures and attacks. This can be demonstrated with a simple global failure. Many cryptographic protocols use a k-out-of-n threshold encryption scheme. It may happen that (accidently or deliberately) too many keys are lost, in which case, the elections' results cannot be recovered. Paper-based systems are, in contrast, more resilient to global failures. Thus, dual voting systems supply the stronger guarantees of end-to-end verifiability characteristic to electronic cryptographic voting while retaining resiliency against global failures.

Another major advantage of dual voting is psychological. Dual voting systems often retain the look and feel of paper-based systems, which makes these systems more familiar and trusted to voters used to paper-based voting. Furthermore, during the experiments it became evident that people trust paper, probably because paper is something one can hold and read *on his own*. The fact that Wombat had paper backup made it easier for Meretz Party to decide in favor of using it.

In dual ballot systems, an adversary wishing to forge the elections needs to break both the paper based and the cryptographic systems. On the downside, it is enough to break one system to breach privacy.

Finally, in dual ballot systems it should be decided in advance when to count which system. Indeed, in some states (like California) the law requires to count paper ballots, while in others, only a sample is required. The following options seem reasonable:

- Use the paper-based system as backup only for disaster recovery, e.g., when private keys are lost, or when the bulletin board goes down during the election.
- Count both systems (for all polling stations or for a sample of them) and if they substantially differ hold a police investigation.

While the theory of cryptographic voting is extensive, and quite well understood, not too many cryptographic voting systems have been actually tested in practice. Helios [Adi08, APMjQ09], which is a web based voting system, had been used in several elections totaling more than 25,000 voters. Pret-a-Voter was tested at the University of Surrey Student Union elections in 2007 [BHP+09]. A recent version of Pret-a-Voter [LR08] also supports dual voting. Punchscan was used at the University of Ottawa in 2007 [EC07]. Scantegrity II was used at the Takoma Park, Maryland municipal elections in 2009, servicing over 1,700 voters [CCC+10]. Scantegrity II also supports dual-voting. Except for Helios, those systems use pre-prepared ballots.

A common criticism on cryptographic voting systems concerns the *usability* issue. It is often said that cryptographic voting systems are too complicated for the common voter. This work sets to design and implement a dual ballot system that retains the feel and look of paper based elections in Israel, trying to prove that such systems suffer from no usability issues. This work describes the implementation of a bare-handed, end-to-end verifiable,

dual (paper and electronic) system with ballots printed on-demand (as opposed to pre-prepared ballots). The design is closest to Benaloh's system [Ben08] adapted to Israel's paper-based system.

Wombat was successfully tested thrice. It was first used in an the Interdiciplinary Center at Herzlya student council election held in May 2011, again in Meretz's party leader election held in February 2012, and again in the 2012 IDC student council elections.

An essential part of this work has been published in [BNNFR+12].

1.1 Thesis Outline

The rest of this thesis is arranged as follows: Chapter 2 reviews the entities, trust model, desired properties and cryptographic tools used by the Wombat and many other end-to-end verifiable protocols. Chapter 3 describes and analyzes the Wombat protocol in detail. Chapter 4 details the implementation of the Wombat protocol, the three elections in which it was used and a security analysis performed on Wombat's implementation.

Chapter 2

Preliminaries

This chapter will sketch the desired properties of any end-to-end verifiable voting protocol, the entities that may play a roll in an election, the cryptographic tools used by Wombat and its trust model. As these are common to many cryptographic election systems, this chapter will provide only short explanations.

2.1 Desired properties

The most crucial property required from electronic voting systems is *integrity*, meaning that it is impossible to forge the election results. Another crucial property is *privacy* meaning that no one can link a voter to his vote, and even further, a voter cannot convince a coercer what his vote was, even when the voter cooperates with the coercer (such a system is *coercion-free* or *incoercible*).

A system is *voter-verifiable* if any voter can verify that his vote was correctly recorded and is included in the tally. A system is *universally-verifiable* if anyone can verify that all recorded votes are properly tallied. A system having both properties is *end-to-end verifiable*.

One can roughly divide the new voting systems into two classes: voting systems where ballots are pre-prepared before election day [Cha04,RP05,FCS06,AR06,CEC+08,CCC+08], and voting systems where ballots are printed on-demand in the voting booth behind curtains [Nef04,MN06,Ben06,Ben08,SDW08]. On-demand systems often have easy interface

to the voter (often using touch screens). Regarding privacy, in print-on-demand voting the voter often has to tell his choices to the voting machine - thus losing privacy towards the voting machine, whereas pre-prepared ballots avoid this problem. On the other hand, when ballots are printed in advance it is crucial to guarantee that these ballots are kept secret (and in particular that the ballots are not, e.g., photocopied by an adversary) leading to the *chain of custody* problem. Another privacy issue in print-on-demand systems is the possibility of subliminal channels where the booth leaks information about the votes to outsiders. [FB09, AN09, GGR09] show how to mitigate these types of attacks.

2.2 Cryptographic tools

Wombat uses only standard cryptographic primitives used in other cryptographic election protocols and therefore they will be mentioned without much details.

ElGamal encryption. ElGamal [Gam85] is an asymmetric homomorphic encryption scheme which works over a multiplicative group G of prime order q.

Zero-knowledge proof of 1-out-of-ℓ re-encryption. [CDS94] presents a zero-knowledge proof that a cipher-text c is an encryption $c = Enc_h(m, r)$ of some message m from a set of elements $m_1, m_2, \ldots, m_\ell$. The proof does not reveal any other information about m (or anything else). The proof is an efficient three-move, honest-verifier zero-knowledge public-coin protocol. Combining it with the Fiat-Shamir method [FS86] results with an efficient non-interactive zero-knowledge proof that c is an encryption of a message from the specific domain.

(t, n)-treshold verifiable ElGamalkey generation. A scheme based on [Ped91, Ped92] that generates an ElGamal public key and n secret shares distributed amoung parties such that any t parties may decrypt any message but any $t - 1$ parties cannot.

Mix-net. A single trusted party can easily generate an ElGamal public key and later decrypt a list of ciphertexts and output the resulting plaintexts in random order. A mix-net [Cha81] is a protocol executed by multiple parties that provides the same functionality, but in a weaker trust model where only a certain fraction of the parties are required to remain uncorrupted.

More precisely, the parties first execute a distributed key generation protocol to generate a joint ElGamal public key such that the corresponding secret key is (t, n)-threshold verifiable secret shared among the parties. This means that at least t honest parties must cooperate to recover the secret key and no set of up to $t - 1$ parties has any knowledge about the secret key. Given a list of ciphertexts, the parties then employ a protocol to jointly decrypt the ciphertexts and at the same time randomly permute the resulting plaintexts without leaking any knowledge about the permutation used.

Wombat uses a *universally verifiable* mix-net where the parties prove that they mixed and re-encrypted correctly using honest-verifier, public-coin zero-knowledge proofs. Another possibility was to use homomorphic tallying, however mix-nets support more complex voting schemes.

Fiat-Shamir heuristic. A cryptographic scheme presented in [FS86] that allows to transform honest-verifier public coin zero-knowledge proofs to non-interactive proofs verifiable by any outside party.

2.3 Entities

Voter. Any person who is registered for voting. He does not have any access to the internet or any computational power at the voting booth (hands-free voting).

Election officials. Officials appointed by the government entrusted with running the voting process and tallying the paper votes.

Polling station committee. A group of officials appointed by the government to a specific polling station in which they are entrusted with managing the voting process, including

identifying voters before they are allowed to vote. The committee has a computer with a scanner and connection to the internet, used to identify voters and upload the ciphertext part of their ballot onto the bulletin board. The uploading may be deferred till the end of the election day.

Voting machine. A deterministic box (i.e., a machine that does not use randomness of its own), built and programmed by the government. It has a touch screen and a printer as its only input/output devices. It gets as input the voter's choice and prints a dual-ballot containing the voter's choice in the plaintext part and an ElGamal encryption of the vote in the ciphertext part, together with some authentication info (see below for details). The voting machine must be kept behind curtain to keep the voter's privacy.

Smart cards. Each voting machine is equipped with two smart cards (and each smartcard is associated with a specific voting machine). The smart cards generate the randomness used by the voting machine in a secure and verifiable manner (see 3.3 for a discussion of the role of the smart cards in the system).

Dual-ballots. Paper notes, printed by the voting machine. Ballots are divided into two detachable parts: the electronic ballot and the physical (*plaintext*) ballot. The electronic ballot has the encrypted vote along with a digital signature certifying the electronic ballot. The physical ballot has the actual vote printed on it. It can be folded into half, and then glued using some standard adhesive strip, thereby hiding the plaintext inside.

Ballot box. Identical to ballot boxes in current election processes.

Bulletin board. Secured and authenticated public database, accessible through a web site interface, which contains the public key for threshold encryption, signature and public keys of all parties participating in the elections, all the casted encrypted votes and audit information about the tallying process. The bulletin board is accessible through the web by all voters and participating parties. The bulletin board supports (1) *read request*, which can be requested by anyone, and (2) *append request*, which can be requested only by official

authenticated players. All bulletin board answers are signed with its signature key. The bulletin board's public key is known to all parties before the election starts (as that key cannot just be published on the bulletin board).

Mix-Net Mixing network is the electronic module responsible for generating public encryption key and decrypting the votes in verifiable and privacy preserving manner. It is composed of several computers executing a predefined protocol where any quorum of predefined size can successfully decrypt the votes. See 2.2 for further information.

Mix-Net parties - A set of officials, each one controlling a single mix-net server. The mix-net parties should typically be either appointed by the government or by the candidates, although this does leave room for representatives of neutral bodies and institutions serving the general public as part of their observing the election. After executing the mix-net, these parties publish their results and audit information on the bulletin board.

Auditors - There are four types of auditors:

- Auditors assuring the orderly conduct of the elections process and overseeing polling station committees.
- Auditors verifying that the electronic ballot appears on the web.
- Auditors auditing the voting machines.
- Auditors for the tallying process.

Auditors of the first type are officials appointed by the government. Auditors of the second type are the voters themselves or anyone given access to their receipt. Auditors of the third type are officials appointed by the government and the voters themselves. Anyone can be an auditor of the fourth type.

2.4 Trust model

This section will explain the assumptions required to achieve the desired properties of the protocol.

Assumptions assuring integrity. The polling station workers are semi-honest, i.e., they will not allow to upload encrypted votes or to cast plaintext votes that were not created by legitimate voters.

Assumptions assuring incoercibility (and privacy). The voting booth will not collaborate with any coercer or with any of the smart cards it uses. Furthermore, the smart cards are manufactured by different companies and do not collaborate among themselves. Third, the smart cards can be initialized only once and their internal memory cannot be read or modified externally. Last, there is no dishonest subset of the mix-net parties large enough to be able to decrypt messages.

Chapter 3

The Protocol

The protocol is based on the protocols from [Ben06, Ben08]. Since the voting booth prints ballots on-demand, the protocol prevent the voting booth from using subliminal channels to compromise the voter's privacy and does so by splitting some of the booth's functionality to external smart cards. For more details see 3.3.

3.1 High Level Description

This section will describe the voting protocol, as seen from the voter's point of view.

The voter first enters the polling station and identifies himself to the polling station committee. Given permission, the voter proceeds to the voting booth and makes his selection on a touch screen. The voting machine then prints a *dual-ballot*. At this point in the process the voter can either audit the machine, or, use the ballot for casting. Is is similar to Benaloh's [Ben06] *cast-or-audit* method.

The dual-ballot is a paper note, divided into two detachable parts: the electronic ballot and the physical (*plaintext*) ballot (see Figure 3.1). The electronic ballot has the encrypted vote along with a digital signature certifying the electronic ballot. The physical ballot has the actual vote printed on it. It can be folded into half, and then glued using some standard adhesive strip, thereby hiding the plaintext inside.

If the voter intends the ballot for casting, the voting machine prints "For Casting" on the ballot (see Figure 3.1). The voter then folds and glues the physical ballot (see Figure 3.3)

and exits the voting booth. The electronic ballot is scanned by the polling station committee and the information is uploaded to the public electronic bulletin board. The committee stamps both parts of the ballot and detaches them in front of the voter. The physical ballot is cast into the ballot box and the electronic ballot is taken home by the voter as a receipt (see Figure 3.4).

If the voter intends the ballot for auditing the voting machine prints additional audit information on the ballot (see Figure 3.2). Audit ballots allow checking the consistency of the voting machine, and inconsistent audit ballots serve as a proof that a given voting machine does not function correctly. Audit ballots cannot be used for voting, and instead the voter enters again to the voting booth to actually vote.

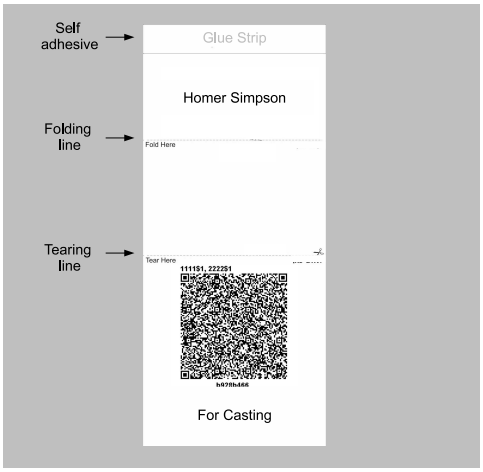

Figure 3.1: Dual-ballot before folding. Since it is for casting, there is no barcode in the lower part of the ballot.

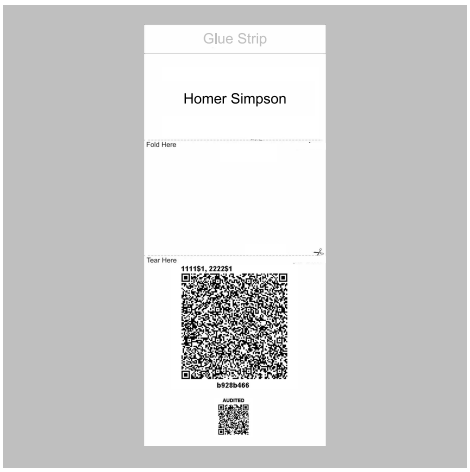

Figure 3.2: Audit ballot. The audit information is printed in the barcode in the lowest part of the ballot.

Tallying. Once the polling stations close, the electronic tallying process takes place publicly on the bulletin board. The tallying is performed by the universally verifiable mix-net (see 2.2 for details). Manual tallying of the paper ballots may be performed at the polling station after it is closed. The decision whether to count/sample the paper ballots or not is left to the discretion of the officials organizing the elections and a policy defining when paper ballots are tallied should be published prior to the elections.

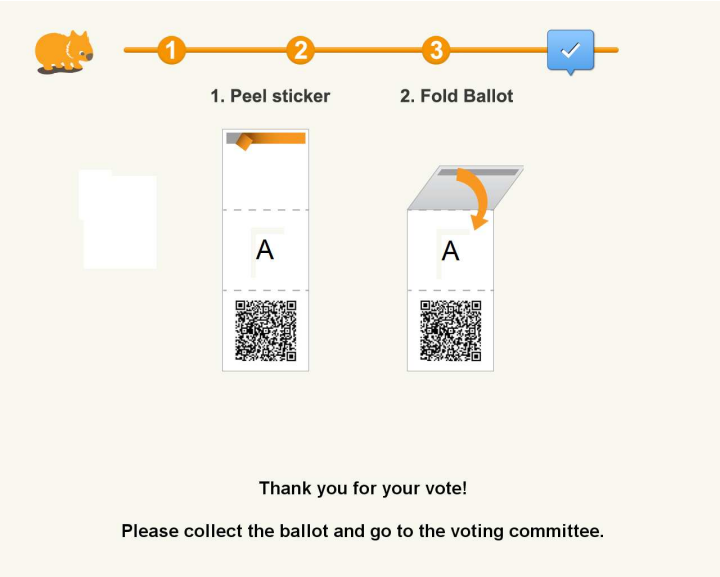

Figure 3.3: Folding a ballot (screenshot from the voting machine).

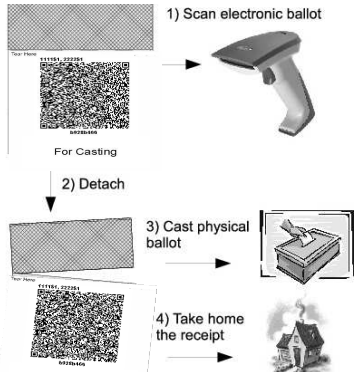

Figure 3.4: The casting process.

3.2 Detailed Description of the Protocol

This section will lay out in details the full protocol from cryptographic point of view.

3.2.1 Setting up the election

The mix-net parties jointly generate a master public-key using the distributed key generation of the threshold ElGamal cryptosystem. Let G, q, g be the public parameters and let h

be the generated threshold ElGamal public key.

The bulletin board and all polling station committee computers generate signature key-pair. The bulletin board's public key is distributed to all participants separately from the bulletin board.

Last, the election officials initialize two smart cards SC_1, SC_2 for each voting booth. The initialization of smart card SC_i consists of generation of a unique identification number id_i, generation of a signature key pair (possibly the same for all booths) and setting the internal counter $rnd\_cnt_i = 1$. Also, the election public-key is stored on the card along with the list of valid candidates. All the smart cards' public keys are stored on the bulletin board.

3.2.2 Election day

Voting. The voter enters the polling station and identifies himself. Given permission of the poll workers, the voter enters the voting booth. The voter chooses his vote v using a touch screen.

Denote the smart cards by SC_1, SC_2. The booth itself is a deterministic machine that cannot generate randomness. The booth requests randomness from the smart cards (to avoid the subliminal channel problem). Each smart card $i \in \{1, 2\}$ increases by one its internal counter rnd_cnt_i, and returns amessage consisting of

$[rnd\_cnt_i, r_i, g^{r_i}, Signature_{SC_i}(id_i|rnd\_cnt_i|g^{r_i})]$ where g is the generator from the election public key and r_i is uniformly random.

The booth encrypts the vote by $c = Enc_h(v, r_1 + r_2)$. It also generates a non-interactive zero knowledge proof π_c that c is an encryption of a valid vote (using 1-out-of-ℓ zeroknowledge proof from 2.2). The booth sends $[rnd\_cnt_1, rnd\_cnt_2, c, \pi_c]$ to SC_1 (SC_1 is chosen before the election day, e.g. the smart card with lower id number). The smart card verifies that the proof π_c is valid for c, and that its internal counter sig_cnt_1 is smaller than rnd_cnt_1. If all verifications pass, the smart card sets its internal counter $sig\_cnt_1 = rnd\_cnt_1$ and returns $[Signature_{SC_1}(id_1|rnd\_cnt_1|rnd\_cnt_2|c)]$. Otherwise it returns an error message. (The 1-out-of-ℓ zero-knowledge proof is required to prevent the voting machine from leaking previous votes in the encrypted message, thereby violating voter privacy.)

The booth prints the first and second parts of the ballot (see Figure 1). More specifically, in the physical ballot part it prints v and in the electronic ballot it prints:

$$
\begin{aligned}
&id_1, id_2\\
&rnd\_cnt_1, rnd\_cnt_2\\
&g^{r_1}, Signature_{SC_1}(id_1|rnd\_cnt_1|g^{r_1})\\
&g^{r_2}, Signature_{SC_2}(id_2|rnd\_cnt_2|g^{r_2})\\
&c = Enc_h(v, r_1 + r_2), Signature_{SC_1}(id_1|rnd\_cnt_1|rnd\_cnt_2|c)
\end{aligned}
$$

The counters are used to prevent chain voting and reuse of randomness.

The voting machines have shielded printer outputs such that the voter can see that a ballot is printed but cannot extract it before he chooses whether to audit the ballot or not.

Using the information printed in the electronic ballot, anyone can verify that the encryption was computed with randomness that was produced by the smart cards. That can be checked simply by verifying all signatures, and, computing $g^{r_1}g^{r_2}$ and comparing it with the first element of $Enc_h(v, r_1 + r_2)$.

Now, the voter can (but does not have to) audit the voting machine and verify that this ballot was properly produced. If the voter wishes do check it, he presses "Audit the machine" on the touch screen. Otherwise, the voter presses "Cast".

Auditing the machine. The booth prints "Audit information: r_1, r_2" in the bottom of the ballot. After the voter exits the booth the poll-workers verify that all signatures are valid and that the randomness counters are equal and increased by one over the counters of previously casted ballots. The poll workers also verify using the randomness printed as audit information that the ciphertext printed on the electronic part of the ballot really encrypts the plaintext printed on the other part. If so, they stamp the ballot and the voter can return to the booth to continue his voting. The voter may also verify those properties at home.

Casting. If the voter presses ”Cast” the booth prints ”For Casting” in the bottom of the ballot. The voter folds the first part of the ballot. Next, the voter leaves the voting booth and presents his folded ballot to the poll workers. The poll workers verify that his ballot is not yet detached. They scan the electronic ballot, verify its signatures and randomness counters, stamp both parts of the ballot and detach the physical ballot from the electronic one. All of this is done in front of the voter. The physical ballot is publicly casted to the ballot box and the stamped electronic part is uploaded to the bulletin board and returned to the voter as receipt.

The voter then leaves the polling station with the electronic ballot.

3.2.3 Tallying

After the election is over, for each polling station the mix-net takes all the encrypted votes $c_1, c_2, ..., c_N$ and passes it through a (re-encryption) mix-net. The mix-net is made of n mixes, each one belongs to a different party. After the last mix outputs a list of ciphertexts, $c'_1, c'_2, ..., c'_N$, verifiable threshold decryption is executed by t parties. The result of this decryption is the tally result for this specific polling station.

In parallel, the physical ballots may be counted according to the policy of the officials organizing the elections.

3.2.4 Auditing

Auditability of casting. The voter can check whether his casted electronic vote is posted correctly on the bulletin board. Also, he can choose to audit the voting machine and receive an audit ballot that he can check at his home, using his own computer. Because the machine has to commit to the ballot by printing it before it knows whether it is audited or not, the machine has to decide whether to ”cheat” or not before knowing whether the ballot will be audited.

Auditability of tallying. Universal verifiability of the tallying is achieved using the standard primitives of verifiable shuffles and verifiable threshold decryption. Anyone can

download a program to check those proofs using his own computer. Anyone with sufficient knowledge can write a program to verify those proofs himself.

Cross checking. At the end of the election the organizers will have two parallel systems that can validate each other. The decision whether to count the paper based system or not should be determined before the elections.

3.3 Informal Security Analysis of the Protocol

This section will analyze the Wombat protocol in order to show it meets the reuired properties from 2.1.

3.3.1 Integrity

In order to manipulate the results of the election, one must affect votes by either:

- Producing invalid ballots, or,
- Tampering with the contents of the ballot box, or,
- Changing votes to the bulletin board, or,
- Forging the tallying computation.

To prevent the first,

- The validity of all the electronic ballots is verified before the mix-net operation.
- A voter or an auditor can verify that the machine encrypts a correct vote by choosing to audit the machine after entering a vote.

The second issue is solved by the assumption that the polling station committee is semi-honest and the ballot box is placed in a widely-visible spot in the polling station, so the polling station officials will not allow the ballot box to be tampered by anyone.

The issue of changing the bulletin board is addressed by two requirements. First, the polling station must sign any vote it uploads to the bulletin board and keep a confirmation,

signed by the bulletin board, that the votes were uploaded. That way, if some legitimate votes are missing from the bulletin board, the polling station could prove to auditors and/or the government that it indeed uploaded the votes. Second, at the end of the elections day, the voting station committee should verify that the number of votes in the bulletin board matches the number of voters that voted in that polling station.

The issue of *Ballot Stuffing*, i.e., adding legal votes by unauthorized party, is addressed by the counters mechanism. For the polling station committee to do ballot stuffing it would need to produce legal ballots with increasing counters and legal signatures. In order to produce them they will have to have access to the voting machine and station computer. As was assumed, the polling station committee is semi-honest and the contradictory interests of the members of the committee and external oversight will be enough to prevent ballot stuffing.

The fourth concern is handled by the verifiable mix-net and verifiable decryption process. Each mixnet party has to output a non-interactive zero-knowledge proof that it mixed and re-encrypted its input ciphertexts into its output ciphertexts correctly. Each party participating in the decryption process produces a similar proof. Together they prove with probability exponentially close to 1 that the plaintexts are valid decryptions of the input ciphertexts. Using t-threshold decryption system assures that as long as there are at least t honest decryption parties, the decryption process cannot be obstructed.

As explained before, an important feature of the system is the cross-check of the electronic tally against the paper tally, and the way this cross-check can address concerns regarding the assumptions underlying the electronic system.

3.3.2 Privacy

Assumptions. Obviously, the voter has to disclose his choice to the voting machine. In order to lower the risk of data leakage from the voting machine, it is assumed that only one among the two smartcards and the voting machine is malicious, in which case privacy would still be preserved. As for the mix-net, in which in theory the link between a voter and his vote may be disclosed, the protocol assumes that there is at least one honest party (a maximum of $t-1$ malicious parties) participating in the mix-net protocol, in which case

privacy would still be preserved.

Subliminal channels. If the voting machine is capable of picking its own randomness, it can pick randomness r such that the last two digits of g^r which are printed on the ciphertext part of the ballot and uploaded to the bulletin board, are identical to the candidate's index. To prevent this, the voting machine's output is made deterministic and uses the smart cards to generate the needed randomness.

In practice, the voting booth knows the votes and the smartcards know only the randomness, so they cannot generate randomness that discloses any information on the actual votes. This makes the booth's public output fully determined by the smart cards' randomness, so the booth cannot leak any information. As stated in the system's trust model (see 2.4), only one of the three components, the booth and the two smart cards, may be malicious. If any two of them are malicious then it is possible to leak information (e.g., the booth sends the vote to one smart card which uses this information to choose randomness that depends on that vote).

Using the smartcards does not help if the booth may request new randomness from the two smartcards until that randomness serves its malicious purpose. The prevent this, the smartcards increase their internal counters for every randomness requested. These counters are printed on the ballot and provide a way to verify that the booth does not *skip* randomness requests.

Gaps in the printed counters may occur as a result of three cases:

- A smart card is skipping legitimate counter values.
- The voting machine is repeatedly querying a smart card for randomness as described.
- A voter has printed several ballots which he did not hand in to the polling station committee for stamping.

The third case may be distinguished from the rest by logging the generation of randomness in the voting machine or even with real time monitoring, i.e., using a counter display outside the voting booth that enables the polling station committee to follow the voting booth's counters throughout election day. If the polling station committee finds regular

gaps in the counters of the smart cards, then it should decommission the voting machine and send it to forensic analysis by election officials and/or auditors.

Finally, a voting machine may leak information by printing a string that encodes current and past votes, instead of printing a valid ballot. This would normally be detected only at the end of election day when all ballots are decrypted, so it does not pose an integrity threat but it does pose a privacy threat. The 1-out-of-ℓ encryption zero-knowledge proof is used to prevent this attack. In order for the ballot to be processed, the ciphertext needs to be signed by one of the smartcards. For the smart card to sign a ciphertext, the voting machine has to produce a zero-knowledge proof that the ciphertext is an encryption of a valid vote. As the voting machine and the smart cards are not collaborating (see 2.4) and therefore the machine cannot obtain a signature on an invalid ciphertext.

Is should be noted that in systems where the booth prints paper ballots, there is always the possibility of ”local” subliminal channel. Consider a booth with a printer that prints the barcode slightly (say 1mm) shifted to the right or to the left, depending on the last vote. Such channel and similar ones are hard to discover. However, this attack is only local since the channel receiver has to see the actual receipt.

3.3.3 Incoercibility

Regular coercibility. The cryptographic assumptions imply that any two distributions of ElGamal encryptions for different parties are computationally indistinguishable. Therefore the coercer cannot distinguish between a vote for his preferred party and a vote for the voter’s preferred party.

Chain voting. Chain voting is a scenario in which the coercer is able to obtain a single legal ballot for his chosen candidate and take it outside the polling station. Outside, he can record the ciphertext part of the ballot. Next, he gives that ballot to a voter, demanding that the voter vote with the coercer’s ballot, present the stamped receipt as proof that he did, and provide the coercer with another legal ballot for the same candidate. The counters mechanism prevents this scenario. A ballot held by the coercer becomes invalid when another person uses the same voting machine, increasing the counters to a value above the

counter printed on the coercer's ballot.

Coercion to vote randomly. In this scenario a coercer attempts to force a voter to vote randomly by demanding a receipt with specific statistical properties, which is easy to generate if the voter agrees to pick a candidate according to the required property but hard to generate if the voter insists on voting according to his preference.

Because the voter makes his choice before the random string is shown to his, thus making the randomness independent of the voter's choice, the voter cannot be coerced according to this scenario.

Chapter 4

Implementation and Experiments

4.1 Implementation

4.1.1 Objectives

The ultimate goal of Wombat was to adhere as much as possible to the current system, and make Wombat simple enough for voters, election officials and auditors so that it can be run by parties with minimal skills. And to do this without compromising on system secrity.

Wombat was implemented in the Java programming language, using virtually only open source components. The interoperability between the different modules was to be implemented using Web Services. Where secure communication was needed, Wombat used Web Services over the HTTPS protocol.

4.1.2 The voting booth

The voting booth machine is a metal box containing a PC, touch screen and an industrial printer capable of printing both text and 2D-barcodes. In order to prevent the user from extracting the ballot before due time, the printer output slot was protected by a transparent plastic slot, so that the voter could see the partially printed ballot but not extract it.

The software was designed to run on a Live Linux OS, although as the PC came off the shelf set up with Windows XP which was eventually used for the elections. The software configuration consisted of the booth's private signature key, all possible candidates

with their full names and numeric representation, mapping between track and available candidates for that track and the public threshold key and smart card keys.

In the IDC student council election, when a voter entered the voting booth, he was asked for his track number. Accordingly, the voter was asked to vote for the corresponding races.

After several tests examining how volunteer subjects operate the booth with little to no instructions beforehand, it was decided to hide the ballot auditing feature. Instead of asking the voter explicitly whether he would like to audit the ballot, the voter could audit his ballot only by pressing a hidden button while the ballot was printed. This way the voting process for the common voter became simpler while still allowing ballots to be audited by auditors and sophisticated voters. The instructions for auditing the voting booths were available at the polling station on election day and also on the election website before the elections.

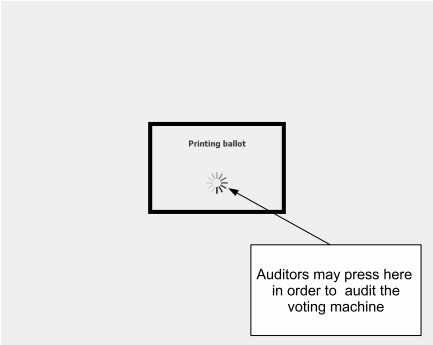

Figure 4.1: Screenshot of the Printing window with the (transparent) audit button.

4.1.3 The polling station committee

The polling station manager computer had three tasks: to access the database of eligible voters assigned to that polling station and their track number, to keep track of voters that voted, and to periodically upload scanned votes to the bulletin board. It was made of a PC running Windows XP and attached to a barcode scanner, compatible with 1D barcodes and 2D barcodes. The computer was pre-configured with the public signature keys of the voting machines in the polling stations and of the smart cards in each voting booth. It run a Java GUI software that used the barcode-scanner to scan voter identification cards and

mark voters in the database. The same program also scanned ballots, verified the signatures on them and accumulated votes in the database. In addition, the polling station committee had a paper stamp to stamp voted ballots as described by the protocol.

The polling station also had a dedicated computer for accumulating the scanned votes from polling station manager computers (over a LAN), as well as for managing the voter registry and for provisioning and monitoring the entire system. The PC ran a MySQL database server and, in addition, a program uploading the signed accumulated votes file to the bulletin board every few minutes.

4.1.4 The website (www.wombat-voting.com)

The website was installed on a virtual private server (VPS) on the internet. The standard software installed on the server included Linux operating system, Apache Web Server, MySQL database and Oracle Weblogic Application Server, The dedicated software on that server was divided between a module for uploading the votes to the database and the website which displays information about the election process and which allows searching the database for a specific vote (or viewing all votes together). The uploading server was implemented as a Java EE application that runs on the Weblogic application server. It exposed a Web Service interface for uploading scanned votes.

The website itself was written in PHP using the Wordpress infrastructure. It contained explanations about the voting, auditing and tallying processes, all public keys, the mix-net proofs of correctness, the uploaded votes file and signature and election results. The site further allowed voters to find their votes inside the vote file thereby making sure that their vote participated in the tallying process.

Although the internet was used for communication network between the different modules of the system, this is not mandatory. In case one worries about *Denial of Service* attacks, the system can be setup to communicate over a private network, and use the website only as a front-end.

4.1.5 Android ballot verifier

In order to allow voters who used the audit feature to audit their vote more easily, an open source Android application was developed for the project. The application allows voters to take a picture of the ciphertext part of the ballot and the audit part of the ballot (if it exists) using the smartphone's camera. The Smartphone would then verify that the signatures on the ballot are correct. If the ballot is an audit ballot then it would verify the ciphertext was generated using the randomness specified in the audit part. If it is "For Casting" ballot, then it would verify that the ciphertext information is posted correctly on the website. It should be noted that the application was developed independently of Wombat by Tel-Aviv University students, thus demonstrating the feasibility of having external observers of Wombat - may they be voters, candidates or neutral parties - develop their own auditing tool.

4.1.6 Mix-Net

The mix-net was implemented using Verificatum [Ver11], which is a free and open source implementation of an ElGamal based mix-net. The goal is to be faithful to cryptographic theory, yet general and fast enough for practical applications. Most of the code is written in Java, but native arithmetic code is also available for improved speed. The complete protocol is a combination of several subprotocols, most importantly threshold key generation, mix-net and verifiable threshold decryption.

Due to lack of computers on which to run the mix-net, the Wombat system has several Linux LiveUSBs, with a full distribution of Linux and verificatum installed. In this way, the mox-net could be run on any available computer.

4.1.7 Unimplemented functionality

Due to time constraints and limited resources not all planned features were implemented in time for the first IDC elections. These features were replaced as follows:

Smartcards. Due to lack of testing time, smartcards loaded with the smartcard application were not deployed to the voting machines. Instead, the machines were set with internal smartcard simulator.

Multiple polling stations. Wombat currently supports only one polling station with multiple booths. The only addition required in order to support more is key managment feature in the web service.

Merkle hash trees. In Wombat's original design the polling stations would upload only the newer votes to the website. In order to make sure that the website does not remove chunks of votes from the list, the posted votes were to be protected by Merkle Hash Tree [Mer87]. Due to time restrictions, it was decided that the polling station should upload all the votes to the website every time, signed by the polling station as a single file.

Secure communication. The Wombat protocol specifies that the communication between the polling station committee and the bullettin board should be 'secure'. Wombat's original design required an SSL connection. Due to resource restrictions and the fact the all the information that passes between the bullettin board and the polling station committee is public, we have decided to replace SSL with a simple challenge response protocol.

4.2 Elections with Wombat

The Interdisciplinary Center's student council elections took place for three consecutive days, May 17th to 19th, 2011. There were several simultaneous runs: 78 candidates competed over 56 available seats in the student council, in addition to runs for the student council president and vice president, and runs for representatives of 27 special tracks. About 2097 voted out of about 6000 registered voters (about 33it took a voter 1-2 minutes to vote, comprising of about 30 seconds of interacting with the polling station worker before voting, one minute interacting with the voting machine and 30 seconds of interaction with polling station workers after voting. After the closing of the polling stations, the mix-net was run on a single machine. The whole process took slightly less than 20 minutes and the election results were announced 45 minutes after the closing of the polling station on the last day of the elections. No contentions were filed.

Meretz is a small political party in Israel, with about 3The party council consists of about 950 members and elects the party's leader. There was a high turnoff at the elections

with about 830 voters (about 88of the voters were over 50 years of age. The Meretz party leadership elections were a simple plurality vote.

The second IDC student council elections were held between May 22th and May 24th 2012. The number of candidates, eligible voters and actual voters was almost the same as in the first elections.

In order to educate potential voters about the system, in both elections the voting process was explained in the website in advance. Furthermore, at the entrance to the polling station one of the developers stood and explained the process to waiting voters and what they had to do once inside the polling station. There were also large posters explaining the process posted outside the polling station.

4.2.1 Lessons Learned

Without guidance, many voters did not fold their ballot at all or folded it incorrectly (in both elections). This was partly due to bad design of the ballot that made it possible to fold the ballot in two different ways. When one of the system developers demonstrated the correct folding before entering the voting booth the error rate virtually dropped to zero.

The developers of Wombat explained to interested voters the dangers of DRE voting (i.e., where a computer simply stores the votes internally). Voters quickly understood the issue and many of them reported that they feel better knowing they can actually *see* their vote in plaintext. Also, many voters (especially the younger ones) enjoyed voting with the new technology, and as a result, were more open-minded to learn about the system.

4.2.2 The Questionnaire

In the first election, voters were asked to fill in an on-line questionnaire or participate in an exit survey. The online questionnaire was composed of 10 questions, 2 of which administrative, 6 about the voter's understanding of the voting process and his/her satisfaction and two about the perceived privacy and integrity of the system. In total, 481 voters participated in the survey, 403 of them answering the on-line survey and 78 the exit survey. The survey response rate was just under 23%. About 37% of those who answered were female and 62% were male; with 4 voters declining to state their gender. In general, survey

participants were well distributed among seven fields of study. The majority (about 73%) of survey participants verified their ballot. It should be noted that the high participation rate is due to a lottery of two campus parking lots (a desirable bonus) among those who participated.

Information on a voter's satisfaction with the voting process was captured via the survey question, "Thinking about your overall experience at the polls today, how satisfied are you with your voting experience?". Over 85% of respondents reported being satisfied.

Voters were also asked their opinion over the simplicity of the voting process. The majority of survey respondents believed the voting process was clear and simple. Across all survey participants, 60% of respondents strongly agreed that the voting process was clear and simple; with just over 1% of respondents strongly disagreeing. About three-quarters of survey respondents agreed with the statement that they understood why the ballot was separated.

Furhter discussion of the questionnaire and its implications can be found at [BNNFR+12].

4.3 Security Analysis of the Implementation

Guy Lando and Bar Perach have performed a security analysis of the Wombat implementation and have published a detailed report of their findings [LP12]. The analysis confirms the security the Wombat protocol under the stated assumptions and presents implementation exploits only. It lists 50 different findings, each representing a bug, missing feature or other weakness of the implementation. These findings are then combined into 7 attack scenarios of which 6 were implemented according to the report. The report concludes by stating that most of the findings are easy to fix and if all findings are fixed the implementation can be considered secure. If these finding are not fixed, however, using them or simply publishing them may damage the system's credibility.

This work cannot list and answer all findings. However the findings can be grouped into several categories with common properties.

Many of the findings exist due to insufficient development time before the first elections the used Wombat. Findings 4, 5, 10, 12, 14, 24, 25, 27, 34, 35, 40, 42 and 50 could be found and averted had there been more time for development and testing. Other findings exist due

to lack of experience in secure programming and security in general in the development team. Findings 5, 13, 15, 16, 20, 21, 25, 26, 29, 30, 31 and 32 could have been averted if the the development team of Wombat had the needed experience in programming secure systems.

Another part of the findings (8, 9, 17, 33 and 39) refer to the storage of and access control to different components of the wombat system before, during and after elections. These issues are chain of custody issues and appropriate procedures should be defined by the election authority to restrict the access to system componets.

A third part of the findings refers to the wombat website and fails to make a distinction between the site's functions as a bulletin board (receiving, validating and publishing the signed file containing all the votes) and its functions as a verifier, enabling a voter to verify that his vote is in the votes file. Findings 2, 3 and attack scenarios 5.1, 5.2, 5.3 attack only the verifier functionality of the website. If Wombat is used in future elections of more significance, third parties will write independent verifiers to make sure that a given vote is in the votes file and thus preventing these findings of damaging the system.

A final set of findings (43, 44, 45, 47, 48 and 49) are merely informative. They do not create any weakness in the system but allow an adversary to gain more information about different components of Wombat.

Several of the findings deserve a more detailed explanation:

- Finding 1 shows how an adversary could crash the voting booth application by pressing two buttons simultaneously and then gain control over the booth. This attack is used to create a ballot with a very high serial number. The report doesn't mention that the voting booth will continue to create ballots with high numbers, and the voting process will continue. Also, this attack is easy to recognize.

- Finding 5 describes a man-in-the-middle scenario where the adversary can gain a valid polling station signature on malicious votes file by intercepting the communication between the polling station and the bulletin board. The finding is caused by faulty implementation of the challenge-response protocol that was used instead of SSL due lack of resources. See 4.1.7 for further details.

- Finding 7 refers to a feature added to the system for the Meretz elections. The party

required the ability to vote with pre-printed ballots for persons with special needs. Therefore a bypass for the serial numbers verification was added to the system. The risks were explained in detail to Meretz's personnel and they were accepted.

- Finding 11 refers to system's source code, potential voters list and other information accessed with encryption. This information is public and therefore there is no need to protect it.

- Finding 19 states that the bulletin board erases the old votes file whenever a new votes file is successfully uploaded. The bulletin board also saves a copy of the old file as backup, making all denial-of-service attacks that use this finding recoverable.

- Finding 23 discusses the bulletin boards administrative console being accessible through the internet but shows no actual attack that uses console.

- Finding 37 describes a bug in the polling station committee software that allows to scan a ballot after faulty identification scan. To turn this bug into an actual attack the person manning the computer is required to collude with the adversary by scanning a ballot even though the computer shows an error in the identification scan.

- Finding 41 states that it is possible to make the same encryption exist twice in the votes file. If that happens, the mixnet code will find the duplicate and remove it.

- Finding 46 shows that the random number generator used to create the ElGammal encryptions in the voting booth is weak. That finding can endanger voter privacy because an adversary can guess the randomness used in the ElGammal encryption and decrypt the voter's receipt. The finding was caused due to the unimplemented smartcard functionality, which would have made the voting booth deterministic.

The security analysis report then discusses how the findings can be used to undermine Wombat's properties by presenting several scenarios.

- Scenario 2.1 requires a chain of custody breach to acquire the polling station's committee private key in order to perform a ballot stuffing attack.

- Scenario 2.2 changes the votes file but cannot the polling station's signature on the votes file. Such an attack can be discovered by independent verifiers in real elections.
- Scenario 2.3 describes a legitimate ballot stuffing scenario, however it requires the adversary to create ballots with valid smartcard or smartcard simulation signature.
- Scenarios 2.4 and 2.5 only change the behavior of the website without modifying the votes file stored in the bulletin board and therefore are attacks against the verifier functionality of the website. In real elections, there will be additional verifiers to mitigate a weakness in the website.
- Scenarios 3.1 and 3.4 will indeed disable the system but will do so in a recoverable and noticeable way.
- Scenario 3.2 states that the old votes file is erased whenever a new file is uploaded. However, as mentioned a backup is saved on the bulletin board. Therefore this scenario cannot cause loss of casted votes.
- Scenario 3.4 doesn't describe any actual attack because the findings it uses are potential only.
- In scenario 3.5, a ballot with very high serial number is created and used to disable further voting. However, as stated, this attack will not disable voting because any newer ballots will be created with even higher serial number.
- Scenario 4.1 requires that plaintext paper ballots be examined for fingerprints in order to compromise voter privacy. That can be done with any paper based system.
- Scenarios 4.2, 4.4 and 4.5 require a chain of custody violation in order to gain access to Wombat's components.
- Scenario 4.3 allows violation of voter privacy by publishing private information about him, but it doesn't allow to discover the voter's vote.
- Scenario 4.6 demands that the voting booth print a small signal in the voter recipt that reveals his vote. This attack is possible in all voting systems where the printer or voting booth that print the voter's receipt also know his vote.

- Scenarios 5.1 to 5.9 deal with undermining confidence in Wombat by changing the content of the website or publishing some of the finding of the security analysis.
- Scenario 6.2 describes what is in essence a chain voting attack. The coercer creates a ballot with high serial number and forces a voter to use. In order for the voter to be able to use it, he needs to do so before any other voters can create and use a ballot with even higher serial number.

In conclusion, the security analysis exposes many weaknesses in Wombat's implementation, though some can be mitigated. It is important to resolve these findings before using Wombat again.

Bibliography

[Adi08] Ben Adida. Helios: web-based open-audit voting. In *USENIX Security Symposium*, 2008.

[AN09] Ben Adida and C. Andrew Neff. Efficient receipt-free ballot casting resistant to covert channels. In *EVT*, 2009.

[APMjQ09] Ben Adida, Olivier Pereira, Olivier De Marneffe, and Jean jacques Quisquater. Electing a university president using open-audit voting: Analysis of real-world use of helios. In *EVT/WOTE*, 2009.

[AR06] Ben Adida and Ronald L. Rivest. Scratch and vote: Self-contained paper-based cryptographic voting. In *WPES*, 2006.

[Ben06] Josh Benaloh. Simple verifiable elections. In *EVT*, 2006.

[Ben08] Josh Benaloh. Administrative and public verifiability: Can we have both? In *EVT*, 2008.

[BHP+09] David Bismark, James Heather, Roger M. A. Peel, Steve Schneider, Zhe Xia, and Peter Y. A. Ryan. Experiences gained from the first pret a voter implementation. RE-VOTE, 2009.

[BNNFR+12] Jonathan Ben-Nun, Morgan Llewellyn Niko Farhi, Ben Riva, Alon Rosen, Amnon Ta-Shma, and Douglas Wikstrom. A new implementation of a dual (paper and cryptographic) voting system. In *5th International Conference on Electronic Voting*, pages 315–329. Gesellschaft fur Informatik, 2012.

[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. In *EVT*, 2008.

[CCC+10] Richard Carback, David Chaum, Jeremy Clark, John Conway, Aleksander Essex, Paul S. Herrnson, 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. In *USENIX conference on Security*, 2010.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, 1994.

[CEC+08] David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan Sherman, and Poorvi Vora. Scantegrity: End-to-end voter-verifiable optical- scan voting. *IEEE Security and Privacy*, 6:40–46, 2008.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.

[Cha04] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy*, 2(1):38–47, 2004.

[EC07] Aleks Essex and Jeremy Clark. Punchscan in practice: an e2e election case study. In *WOTE*, 2007.

[FB09] Ariel J. Feldman and Josh Benaloh. On subliminal channels in encrypt-on-cast voting systems. In *EVT*, 2009.

[FCS06] Kevin Fisher, Richard Carback, and Alan T. Sherman. Punchscan: Introduction and system definition of a high-integrity election system. In *WOTE*, 2006.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, 1986.

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, 1985.

[GGR09] Ryan W. Gardner, Sujata Garera, and Aviel D. Rubin. Coercion resistant end-to-end voting. In *FC*, 2009.

[LP12] Guy Lando and Bar Perach. Vulnerability analysis of the wombat voting system, 2012. `http://course.cs.tau.ac.il/secws12/?q=projects/wombat-analys`

[LR08] David Lundin and Peter Y. A. Ryan. Human readable paper verification of pret a voter. In *ESORICS*, 2008.

[Mer87] Ralph Merkle. A digital signature based on a conventional encryption function. In *CRYPTO*. 1987.

[MN06] Tal Moran and Moni Naor. Receipt-free universally-verifiable voting with everlasting privacy. In Cynthia Dwork (Editor), editor, *CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 373–392. Springer-Verlag, August 2006.

[Nef04] Andrew Neff. Practical high certainty intent verification for encrypted votes. 2004.

[Ped91] Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In *EUROCRYPT*, 1991.

[Ped92] Torben P. Pedersen. *Distributed Provers and Verifiable Secret Sharing Based on the Discrete Logarithm Problem*. PhD thesis, 1992.

[RP05] Peter Ryan and Thea Peacock. Pret a voter: a system perspective. Technical Report 929, University of Newcastle upon Tyne, School of Computing Science, Apr 2005.

[SDW08] Daniel Sandler, Kyle Derr, and Dan S. Wallach. Votebox: a tamper-evident, verifiable electronic voting system. In *Proceedings of the 17th conference on Security symposium*, pages 349–364, Berkeley, CA, USA, 2008. USENIX Association.

[Ver11] Verificatum project, 2011. `http://www.verificatum.org`.