Building Trust in Remote Internet Voting

Thesis for the degree Master of Science

Lars Hopland Nestås
University of Bergen
Department of Informatics
June 2010

NoWires Research Group

www.nowires.org

Preface

During the past decades, a lot of research has been done to create voting protocols and election systems that facilitate voting via the Internet. Many universities and private organizations are now using such systems for referendums, or to elect individuals for its leading positions. Several countries are also moving towards electronic voting over the Internet for legally binding elections.

Cryptographers and system designers have until now mainly focused on the mathematical and the technical aspects of electronic voting systems. How to build trust in these systems have not been considered in any depth (as far as the author has been able to establish). Whenever there is a change in election procedures, voters', candidates', and election officials' trust in the election system may be challenged. In this thesis, we focus on different aspects of how to build and preserve trust in remote electronic voting systems.

The intended audience for this thesis is first and foremost students and IT professionals interested in remote electronic voting systems. However, the first two chapters, and several parts of Chapter 3 and Chapter 4, are not very technical oriented. Readers with some basic IT knowledge, and who are interested in the topic, are encouraged to continue reading.

Acknowledgements

I am grateful to many people who helped me make this thesis possible. First of all I would like to thank my supervisors Kjell Jørgen Hole and Håvard Raddum for excellent and enthusiastic help and guidance throughout the work on this thesis. I am also especially grateful for our joint work on the Encap paper, and the two e-voting chronicles. It has been a privilege to be a member of the NoWires Research Group.

A special thanks goes to Vidar. I have really appreciated your creative ideas on how to develop new attacks targeting various computer systems. Thank you Olav Arthur, for being the very local Apple support.

I would also thank my brother Eirik and my future college Lars-Helge for giving me constructive and useful feedback on this thesis.

Finally I would like to thank my family—especially my beloved wife Ingeborg. You have not only given me your unconditional support during the work on this thesis, but also provided valuable advices in how to structure my work.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Distrust and caution are the parents of security.
Benjamin Franklin

"Free and fair" elections are often referred to as the foundation for a democracy. OSCE's *Charter of Paris for a New Europe* [1] states:

> "Democratic government is based on the will of the people, expressed regularly through free and fair elections. [E]veryone also has the right [...] to participate in free and fair elections."

Elklit and Svensson [2] clarify and differentiate between the terms "free" and "fair". *Freedom* entails the right and opportunity to choose one thing over another, and at the same time, certain choices should not have negative consequences for one's own or one's family's safety, welfare, or dignity. *Fairness* means impartiality, i.e., no individuals are given unreasonable advantages.

During the past centuries, the introduction of technical equipment has radically changed the way many elections are organized [3, Ch. 1]. Preprinted paper ballots—technical equipment in its simplest form—were introduced to improve accuracy. Similarly, the introduction of envelopes improved the voters' privacy. Later, electronic equipment have been widely used to cast and count votes in countries like the U.S., the Netherlands, Germany, France, and Austria [3, Ch. 1].

The term electronic voting (or e-voting) denotes different means of casting and counting votes electronically. A *remote* electronic voting system gives the voter the opportunity to cast his vote outside a controlled environ-

ment.[1] Several countries are now moving towards remote electronic voting over the Internet. Regardless of the technology used to organize elections, the principles of free and fair elections are the same.

1.1 Trusting the electoral process

In Norway, the Ministry of Local Government and Regional Development runs the E-vote 2011 project[2] to establish and implement routines for e-voting that ensure a correct result and build trust. According to the project directive [4], the project shall work to achieve acceptance for e-voting as a secure and reliable solution amongst both policy makers and the general population. These goals are important, but often overlooked. According to Schneier [5, p. 114], voting is as much a perception issue as it is a technological issue. He states that:

> "It is not enough for the result to be mathematically accurate; every citizen must also be confident that it is correct."

In other words, it is not enough for election observers to declare an election free and fair—the public must have some level of trust in the electoral process to give the winner legitimacy to govern. Usually, people do not protest simply because their candidate lost the election, but they do if they believe that their candidate lost unfairly. It is therefore often said that the main purpose of election fairness is to convince the loser (and his supporters) that he lost the election fair and square [6]. If trust is low, or lacking, this is a very hard task.

It is difficult to give a precise and universal definition of trust. Because of the large variety of definitions (e.g. see [7]), it seems reasonable to conclude that trust must be considered in light of its context. In this thesis we will, thus, use Marsh and Dibben's definition of trust [7]:

> "Trust concerns a positive expectation regarding the behavior of somebody or something (*the trustee*) in a situation that entails risk to the trusting party (*the truster*)."

Marsh and Dibben observe that the level of trust must be above a certain threshold, denoted the *cooperation threshold*, to make the decision to trust someone [8]. If the level of trust is below the cooperation threshold they call

[1] Note that all further usage of the term 'e-voting' in this thesis refers to remote electronic voting on the Internet, unless something else is specified.

[2] `http://www.e-valg.dep.no/`

this *untrust* or *distrust* (see Figure 1.1). Untrust is a measure of how little the trustee is actually trusted, and distrust is a measure of how much the truster believes that the trustee will actively work against him or her in a given situation.

To further clarify the differences between untrust and distrust, we introduce a second threshold, namely the *uncooperation threshold*. If the truster's level of trust is below the uncooperation threshold, the truster believes that the trustee *deliberately* acts against him. For example, if a voter has a high degree of distrust, he may strongly believe that the election officials in charge of an election, controls the outcome—the election officials are acting uncooperative. Likewise, if election officials choose to ignore strong evidence of election fraud, many voters are likely to object because they distrust the election—the voters are acting uncooperative.

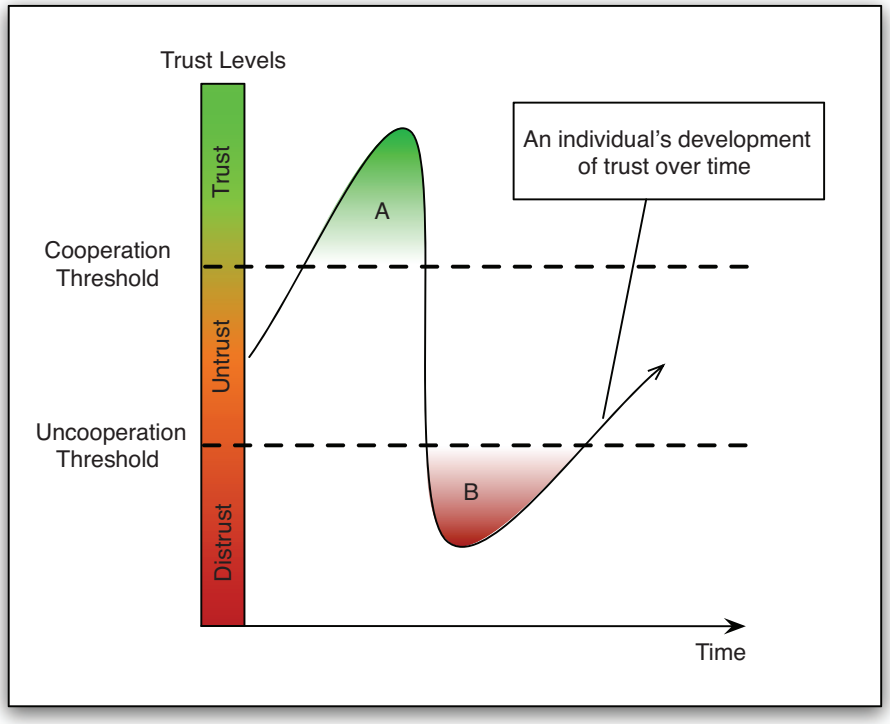

Figure 1.1: *Trust is situational, and varies over time. In area A the truster trusts the trustee enough to cooperate. In area B, the truster will act against the trustee, because the truster is convinced that the trustee will do the same.*

In between the cooperation threshold and the uncooperation threshold we find the area untrust. Marsh and Dibben give an example of how to understand the term untrust [8]:

> "—I trust you, but not enough to believe you'll be of any help in this situation if a push comes to shove."

Similarly, a voter may trust the election officials' intentions to organize a free and fair election, but the voter is uncertain about the election officials' capability to detect an attack on the election, and thereby ensure the election's integrity.

1.2 Is trust important?

The public's confidence in an electronic election system can be fragile. One relatively small incident can lead to dramatic changes in the public's confidence. The 2006 municipal election in the Netherlands exemplifies this problem. A suspicion of fraud was raised when a candidate got 181 preferential votes at one particular polling station, but only obtained a total of 11 votes elsewhere [9]. Moreover, the suspicion of fraud was further raised by the fact that the candidate was a polling worker at the polling place where he obtained the unexpected large number of votes. Because it was impossible to perform a meaningful recount, due to the design of the voting machines, voters were asked to come back to cast their vote once more. During this election the suspected candidate only got a handful number of votes. As a result of this shadow election and testimonials given by voters, the candidate was convicted of election fraud.

An action group named "we don't trust voting computers" was founded after this election.[3] The group managed to get access to voting machines from one of the major vendors and found several serious vulnerabilities [10]. The Dutch government was, thus, forced to initiate further investigations of the voting machines [11]. On October 1st, 2007, the District Court of Amsterdam decertified all voting machines in use. Finally, the Dutch returned to paper based voting for the 2009 European Parliament election and all work on remote Internet voting was put on hold.

The level of public confidence in Norwegian elections is deemed to be very high [12, 13]. This does not mean that the Norwegian election system is—and has always been—perfect. For example, election observers during the 2005 parliamentary election were rather critical to the tabulation process. They pointed out that there were no official specifications on how to use electronic equipment to process votes. In particular, detailed counting procedures were lacking. The election observers concluded [13]:

[3] `http://www.wijvertrouwenstemcomputersniet.nl/English`

> "Again, such a system, which would appear rather vulnerable and open to abuse in other circumstances, does not appear to create any problems in Norway, where there is a strong tradition of accountability and the level of public confidence in the honesty and integrity of the election administration is very high."

Trust is a well-established topic in the areas of psychology and the social sciences. Trust has also gotten more attention in the areas of information science and technology with the growth of the Internet and, in particular, e-commerce. Trust is essential for e-commerce: with trust people will buy things in an online store, without they will not [14]. If the level of trust is low, a customer has several opportunities. Either he will purchase the goods in a competitor's online store, which he is likely to trust more, or he will buy the goods in a traditional store. Trust regarding e-voting systems is not quite as simple. A voter may choose not to trust an e-voting system for many reasons. If privacy concerns are the main reason for the voter's untrust or distrust, he can cast a paper ballot if this is an option. In a situation where no alternative exists, the voter is left with the option not to participate in the election. Moreover, if the integrity of the election result is the voter's main concern, it will not help to use a different "channel" to cast the vote. A voter that does not trust the integrity of one part of the tally, will obviously not trust the complete tally. The voter is again left with the option of not voting. If the voter decides not to participate in the election, we can say that his level of trust is below the cooperation threshold.

1.3 E-voting affects the voters' trust

In this thesis we will argue that there are some general properties of electronic voting that can affect the voters' level of trust in a negative direction towards distrust. For example, introduction of a new electronic election system may change known and trusted procedures radically. The voters have to develop new internal trust models. It is also clear that election observers are struggling to find a reasonable way to observe remote electronic elections [15], and likewise, auditors find it difficult to verify the integrity of the election result. These problems are related to the fact that a remote e-voting system reduces the number of laymen involved in the election process in favor of a smaller group of IT professionals [16, Ch. 9]. Finally, there is a growing concern about the ongoing "privatization of the democracy" [17]. Commercial entities are in greater extent delivering critical components to the election process, components that may have an impact on the election outcome. Examples are electronic voting machines, optical ballot scanners,

and electronic electoral rolls. Information about these systems is very often not available to the public, or even to the election management board. Such lack of transparency may decrease the overall trust level.

1.4 Scope

The main goal of this thesis is to explore how different design properties of remote Internet e-voting systems influence the voters' level of trust. We will also discuss how behaviors of election officials have an impact on voters' trust in an election result.

Cryptography is often used to ensure election integrity and voter privacy. In this thesis we will present some cryptographic properties for building trust in e-voting systems.

Legal issues, like whether a remote e-voting systems violates the United Nations' *Universal Declaration of Human Rights*, will not be considered in this thesis.[4] Our focus is how to build trust in the technical and organizational parts of an e-voting system, after a decision about implementing such a system has been made.

1.5 Structure of thesis

Chapter 2

Chapter 2 presents commonly accepted requirements for e-voting systems, and discusses the importance of specifying requirements in order to build trust. The chapter also outlines an analysis technique for e-voting systems based on the requirements.

Chapter 3

With an analysis technique in place, the third chapter presents an analysis of the e-voting system utilized during the 2009 rectorial election at the University of Bergen (UiB). The chapter concludes with some considerations about performing security tests on a live system.

Chapter 4

Election fraud, or even suspicion of election fraud, can undermine public confidence in the outcome of an election. This chapter discusses which in-

[4] `http://www.un.org/en/documents/udhr/`

formation election officials must gather in order to support later audits, recounts, and forensic analysis. This chapter also explores some techniques for detecting election fraud and other irregularities in e-voting systems.

Chapter 5

Verifying the correctness of an election is important to avoid untrust and distrust. Different ways of verifying election results will be explored in the chapter. In particular, the chapter discusses how an enhanced level of voter interaction throughout the voting procedure can help to increase the publics' level of trust in the electoral process.

Chapter 6

Chapter 6 presents a short summary, and concludes this thesis. Finally, some suggestions for further work within the field trust and remote electronic election systems is given.

Chapter 2

Requirements and Analysis Technique for E-voting Systems

Defining problems accurately lays the foundations for solving them.
Trust Matters, S. Bibb & J. Kourdi

Different aspects of the electoral process must be considered to declare an election "free and fair". In this chapter we define a set of security and privacy requirements especially relevant to e-voting systems. Furthermore, we discuss how to utilize these requirements in order to build trust among the public. Finally, the chapter outlines a simple analysis technique for e-voting systems based on the requirements. The reader interested in other election criteria/requirements, not directly related to the e-voting system itself, is referred to the *Declaration on criteria for free and fair elections* [18], and Elklit and Svensson [2, Table 1].

2.1 Requirements for a free and fair e-voting system

It is common to use the general terms *confidentiality*, *integrity*, and *availability* when considering security and privacy requirements in information systems. Confidentiality is the protection against unauthorized disclosure of data, integrity refers to protection against unauthorized modification of data, and availability is the ability to use a resource as desired [19].

For complex systems, like an e-voting system, it may be better to use a set of requirements especially tailored to the domain of the system. A list of

eleven high-level, non-functional requirements for e-voting systems follows. This list is based on commonly accepted requirements found in the research literature [6, 15, 20, 21].

2.1.1 Requirements

Eligibility All eligible voters are accurately identified and registered, and an authentication procedure ensures that only registered voters can vote.

Uniqueness Only one vote per voter. If a voter is allowed to vote multiple times, then only the last vote counts.

Accuracy All valid votes are counted correctly. It is not possible for anyone to alter, delete, invalidate, or copy any vote.

Soundness Invalid votes, for example votes with invalid format or content, are not counted.

Privacy All cast votes are kept secret during the election and it is never possible to link any vote to the voter.

Fairness No early results, in particular no partial tally, can be obtained by anyone during the election period.

Transparency Information about the e-voting system itself and the information it publishes must be available to all stakeholders.

Robustness No reasonably sized coalition of stakeholders is able to disrupt or influence the voting process during the voting period, or manipulate the final tally in any way.

Uncoercibility A coercer cannot be sure he vas able to force a voter to cast a particular vote later tallied by the system.

Receipt-freeness To prevent vote buying and selling, no voter can obtain or construct a receipt to prove the content of a vote to a third party during the election or after the election.

Verifiability Each voter can check that his vote was correctly received by the central infrastructure (voter verifiability), and anyone can count the votes (universal verifiability).

2.1.2 Selecting requirements

Understanding the context in which an e-voting system will be deployed is important when selecting its requirements. The collection of requirements defining an e-voting system for private elections at universities will not be equal to the collection of requirements associated with a system for governmental elections. Deciding upon a set of requirements for an e-voting system must be seen in the light of which threats the system is likely to face.

It is important to notice that we distinguish between the terms *vulnerability* and *threat*. A vulnerability in an e-voting system is a (security relevant) design flaw or an implementation bug, while a threat is a person or a group that has the needed capabilities to exploit a vulnerability either intentionally or accidentally [22, Ch. 4]. Malicious software developers, hackers, criminal organizations, protest groups, malicious election officials, and rouge system operators are examples of threats deliberately exploiting vulnerabilities to misuse e-voting systems. Legitimate voters and benevolent operators are examples of threats inadvertently exploiting vulnerabilities due to carelessness or bad usability.

2.1.3 Further requirement engineering

The selected list of high-level, non-functional security and privacy requirements is vital in the further process of requirements engineering. Every new low-level functional requirement suggested for the system should be held up against the list of non-functional security and privacy requirements. If it is not compliant, it should be discarded.

2.2 Requirements as a means for building trust

The process of developing (non-functional) security and privacy requirements can be used to increase the level of public trust in a forthcoming e-voting system. Presenting well-formulated requirements may show insight, and if done right, the requirements can be seen as an expression of will to consider important security and privacy aspects of e-voting in the further development process. Choosing *not* to announce security and privacy requirements, may cause suspicions and untrust, and in worst case distrust. If it is not possible

to achieve public acceptance for the non-functional requirements early in the development process, it is more likely that the process will fail later.

Stakeholders often present the expectations they have to a system in a natural language. Without a common basis for communication, statements are likely to be misinterpreted. Defining a list of easy to understand, and well-defined requirements makes it easier to have an enlightened debate whether or not an e-voting system is suitable for a particular type of election.

2.2.1 System requirements for the Norwegian E-vote 2011 project

The Norwegian E-vote 2011 project chose a different approach then ours when selecting requirements for their system. Requirements for the e-voting system were first and foremost a result of their acquisition form: *competitive dialogue*. In a competitive dialogue, several tenderers are invited to compete for the delivery of a final system. During the competition phase, the tenderers submit several partial deliveries. The contractor forms specifications for the next deliveries based on the partial deliveries. In this way the tenderers are participating in the process of developing the final requirements for the system.

In the Norwegian E-vote 2011 project, a set of system requirements was finalized two and a half weeks before the tenderers submitted their final proposal for an e-voting system [23]. At the same time the requirements were made publicly available. However, regular voters were not among the intended audience for the requirement documents. It may seem that the project management did not encourage a public debate on the system's non-functional requirements since the requirements were inaccessible almost until a full system solution was described.

One of the greatest challenges when introducing a remote Internet e-voting system is to overcome the bad reputation of e-voting in general. The Norwegian E-vote 2011 project has already experienced this. The Ministry of Local Government and Regional Development invited fifty-six municipalities to apply for participation in a limited e-election in 2011. Only limited information about the project was available at the time the politicians had to make a decision about participation. Twenty-one municipalities decided not to apply to the Government. Some of the municipalities declined mainly because they were concerned about the voters' privacy [24, 25].[1]

[1]Other municipalities declined mainly because of the unclear costs associated with the participation in the e-election trial.

The E-vote 2011 project's privacy requirements are actually very stringent [23, pp. 32, 33]. So why did the politicians have doubts about the privacy? Were they not convinced that the Ministry would be able to provide an e-voting system that would meet the project's security and privacy requirements? There is no easy answer to these questions. But clearly, some of the politicians did not have enough (or any) knowledge about the system's privacy requirements and based their decision on their own perceptions about the system.

2.3 Analysis technique for e-voting systems

Inspired by Hubbard's critical evaluation of risk analysis methods [26] and other methods for discovering security weaknesses [27, 28], this section outlines a technique for deciding if an e-voting system is in compliance with a set of requirements. The technique's main focus is to discover technology oriented attacks. It is useful for individuals with knowledge of information security, but not necessarily much experience with security evaluation of e-voting systems. Our analysis technique is not an alternative to formal methods for evaluating the security of a system design, but rather a supplement that is especially useful when evaluating prototypes and fully implemented systems.

It is possible to rate a system's compliance with a requirement using, e.g., the three levels *low*, *medium*, and *high*. One motivation for introducing levels is to make it easier for a team of executives to address the most critical problems first [29, Ch. 6]. Unfortunately, the introduction of levels of compliance makes it more likely that different groups of analysts will come to different conclusions [26, Ch. 7].

We will rather try to determine requirements *not* satisfied by an e-voting system. A requirement is said to be broken, or *violated*, if we can find a practical attack, else the requirement is assumed to be satisfied. Note that we define an attack as the combination of a vulnerability and a threat. An attack is said to be *practical* if it can be carried out by one or more threats. The practicality of an attack is best demonstrated by developing a proof of concept-attack.

The main advantage of this approach is that once the requirements have been selected and accepted by the stakeholders, it is hard to argue that an e-voting system is suitable for real elections when practical attacks violating one, or several of the requirements, are discovered.

2.3.1 Three step analysis technique

Step 1 – Develop system models

To analyze an e-voting system it must first be modeled. A model is an abstraction of the system that focuses on interesting aspects an ignores irrelevant details [30]. The purpose of modeling is to deal with complexity. It may be useful to develop several models, each focusing on different aspects and parts of the system.

First, it is necessary to get a general overview of the electoral processes that the e-voting system is a part of. A generic model of the electoral process, presented in Figure 2.1, can be used as a starting point. The figure is based on [6, Ch. 1], where supplementary descriptions of the different phases in the process can be found.

An e-voting system's most valuable assets are the values of the cast votes. To outline the entire electoral process it may be useful to describe:

- The information that is available at each phase of the election.
- In what form the information is stored.
- The actors (people and components) with access to the information.
- How the actors interact with the information.
- How the information flows within each phase.
- How the information flows between each phase.

Even if a system description is available from the system provider, the analysts should (as closely as possible) try to verify that the description is correct.

Step 2 – Determine practical attacks violating the requirements

In this step the analysts should look for threats exploiting vulnerabilities in each phase of the electoral process. The rich literature describing attacks on web based e-voting systems and web applications in general, (see e.g. [28, 31, 32, 33]) may be of great help during this step. Most of the practical attacks described in Chapter 3 were discovered using the methodology presented in [28, Ch. 20].

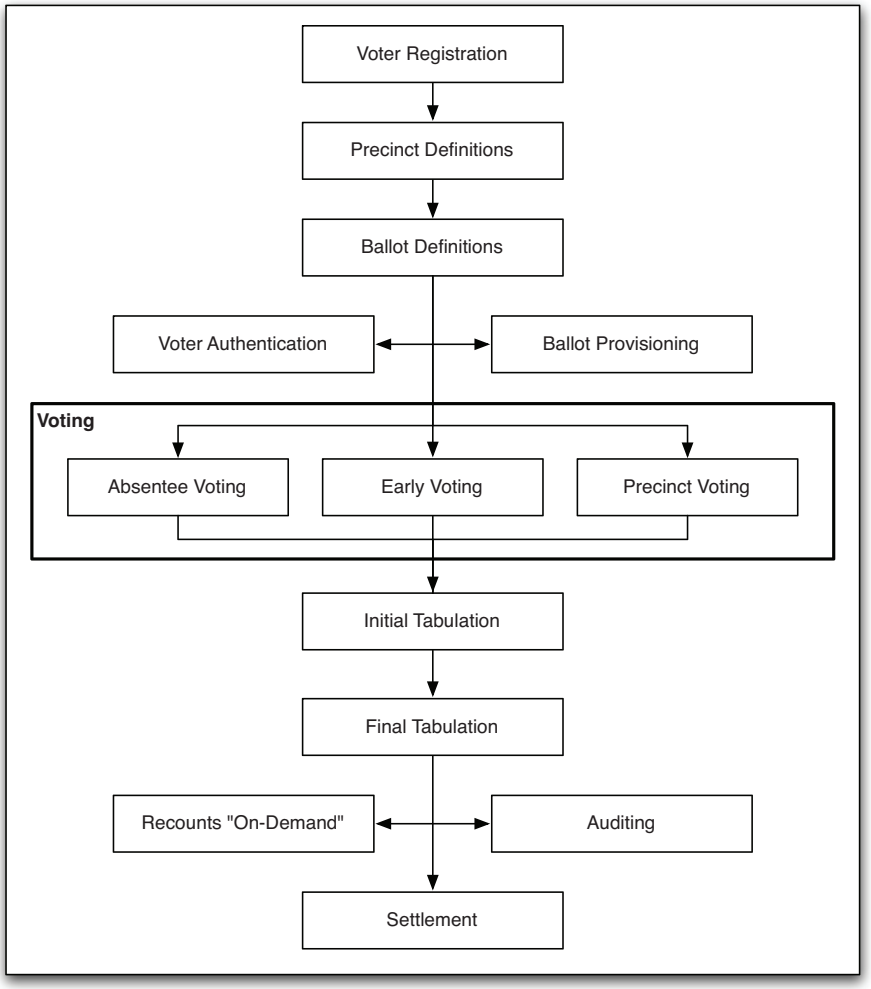

Figure 2.1: *Generic model of the electoral process.*

Step 3 – Document attacks and identify violated requirements

It is important do document all practical attacks for several reasons. First, claiming that a requirement is violated may be met with objections from different stakeholders. Providing detailed documentation makes it easier for others to verify the results of a completed analysis. Second, well-formed documentation makes it easier for developers (if possible) to deploy counter-measures. Finally, it is important to document all attacks to simplify the next round of analysis.

2.3.2 Recommendations

The outlined analysis technique can never prove that a requirement holds for a real system. It is therefore preferable to use multiple groups of analysts with diverse skill sets, looking at the systems from different angles, to minimize the likelihood of overlooking serious vulnerabilities.

Analysts should begin to scrutinize the system early in the development process. Since new vulnerabilities and threats may appear over time, the e-voting system should be repeatedly analyzed during its lifetime, and not only when system updates are deployed.

One should remember that discovering vulnerabilities, and presenting practical attacks might influence the public's level of trust. Voters, electoral candidates, and politicians are likely to lose trust in e-voting systems yielding to attacks, even though the attacks may be relatively limited in nature. Hence, it is of major importance that system owners are prepared to deal with discoveries of system vulnerabilities. A system owner who decides not to scrutinize his system properly, or ignores reported vulnerabilities, runs a greater risk. Regarding trust, nothing seems worse than reports of successful attacks during, or after an election.

The final question system owners must answer, is whether known vulnerabilities shall be publicly disclosed. This is not a new problem, and it was already discussed in *Locks and Safes: The Construction of Locks*, published in 1853 [34]:

> "Rogues knew a good deal about lock-picking long before locksmiths discussed it among themselves, as they have lately done. If a lock, let it have been made in whatever country, or by whatever maker, is not so inviolable as it has hitherto been deemed to be, surely it is to the interest of honest persons to know this fact, because the dishonest are tolerably certain to apply the knowledge practically; and the spread of the knowledge is necessary to give fair play to those who might suffer by ignorance."

In [32], Jones discusses this question in the context of e-voting. He argues that compared to the short-term benefits of not discussing vulnerabilities in e-voting systems openly, suppressing such information may lead to an uninformed electorate making uniformed decisions.

Chapter 3

Analysis of the Rectorial Election at UiB in 2009

As long as I count the votes, what are you going to do about it?
—William Marcy "Boss" Tweed, 1871

The University of Bergen (UiB) has long elected individuals for its leading positions through democratic elections. In 2005, there was a major change in the election procedure—the university introduced remote electronic voting over the Internet for most of its elections. Despite this major change in the procedure, UiB's electronic election system did not get much attention until the rectorial election in 2009. In that election, 9 irregular votes were rejected [35] and several serious vulnerabilities in the system were discovered [36].

This chapter gives an overview of the election system used, and the vulnerabilities discovered in the system.

3.1 System overview

My Space ("Mi side" in Norwegian) is UiB's intranet for students.[1] The system is an implementation of .LRN (Learn, Research, Network),[2] which is an open source e-learning system originally developed at Massachusetts Institute of Technology (MIT). In My Space, students can join different community and course groups. Groups of students at the department level are examples of community groups. The course groups often provide information about classes, such as schedules, lecture notes, discussion boards, and assignments.

[1] `http://myspace.uib.no/`
[2] `http://dotlrn.org`

The .LRN framework consists of several modules. The focus in this chapter is on the survey module. Group administrators can create polls, usually for evaluating courses or administer tests, using the survey module. UiB utilized the survey module to arrange elections through My Space. Figure 3.1 shows an overview of the important voting and tabulation phases in UiB's election system. A more detailed description of the system, based on all phases of the 2009 rectorial election, follows.

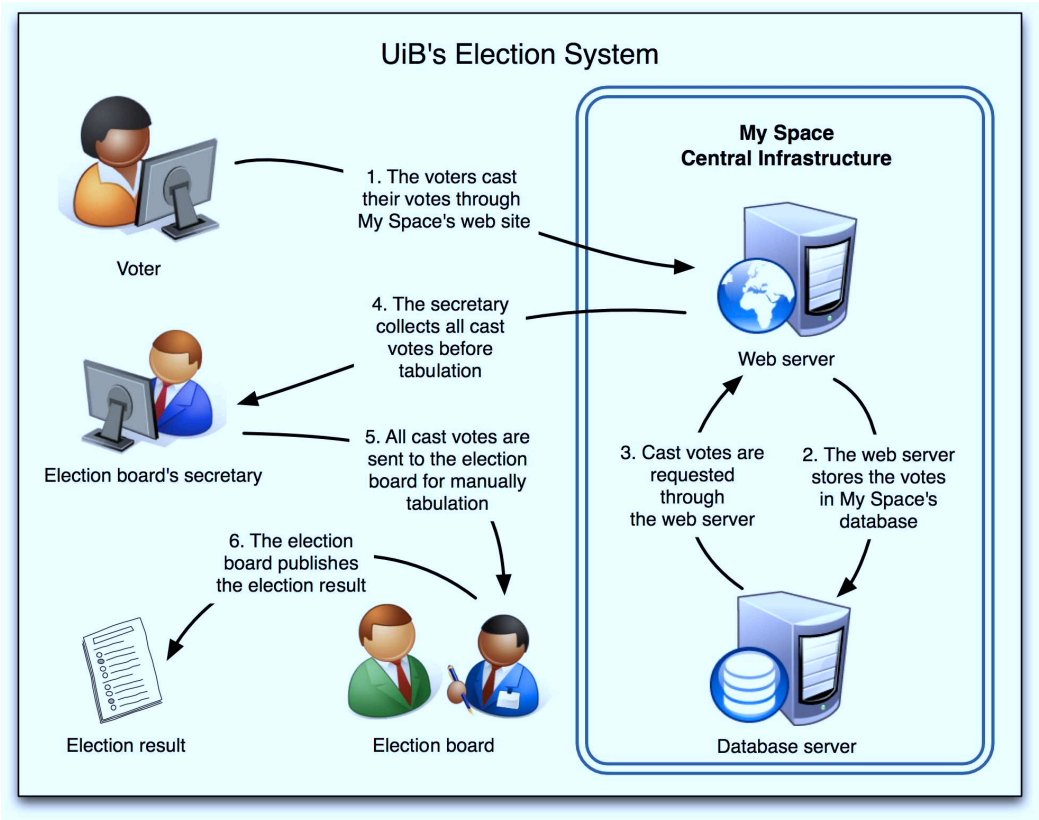

Figure 3.1: *Overview of the voting and tabulation phases in UiB's election system.*

3.1.1 Voter registration

The official electoral roll at UiB was maintained outside My Space. For the rectorial election two separate web pages, one for staff[3] and one for students,[4]

[3] `http://www.uib.no/ansatt/manntall/`
[4] `http://www.uib.no/info/student/manntall/`

were set up for voters to check if they were correctly registered and allowed to vote.

Voters in the rectorial election were divided into four groups. The votes from the different groups were weighted (weight in parenthesis). Group A: permanent academic staff (29,5 %); Group B: temporary academic staff (29,5 %); Group C: technical and administrative staff (16 %); and Group D: students (25 %). Before the election started, the voters within the official electoral roll were divided into different subgroups in the UiB community group at My Space.

3.1.2 Authentication and authorization

Access to the voting page at My Space is protected by a voter's username and password. Usernames and passwords are managed by the central user administration system at UiB called SEBRA. Password requirements are given in UiB's password policy [37]. User rights are managed within My Space.

3.1.3 Ballot definition

Two pairs of candidates, *Team Grønmo and Rokne* and *Team Reed and Nordtveit*, ran for election as rector and prorector in the rectorial election. The election administrator was responsible for setting up the voting page. The ballot (Figure 3.2) was presented as an HTML form with one visible field (containing the parameters *Response_to_question* and *Question_id*) and three hidden fields (*Survey_id*, *Section_id* and *New_response_id*). All parameter values in the HTML form were numeric values. These values, together with *user_id* and *creation_id*, formed an electronic vote.

Table 3.1 gives an example of values for each parameter in the 2009 rectorial election at UiB. In the following, we give a description of each parameter.

Survey_id

Responses to the same survey, or votes in the same election, always have the same value for *Survey_id*.

Section_id

In a survey with multiple questions, it is possible to group the questions into sections. Every section has its own *Section_id*. The rectorial election only had one question, and therefore only one *Section_id*.

Figure 3.2: *Ballot in the 2009 rectorial election at UiB. The button "Send svar" submits the form.*

New_response_id

New_response_id is a unique value for each response to a survey, or each ballot in an election. This value is the primary key when a response is stored in My Space's database.[5]

Question_id

Each question in a survey gets its own value. In the rectorial election there was only one question.

Response_to_question

A voter had the opportunity to either cast a blank vote, or to vote for one of the candidate pairs. Radio buttons were used in the HTML form to ensure that a voter only chose one out of the three options. Each option had its own value.

User_id

A voter's *User_id* was obtained from a cookie and attached to the vote.

[5]The value of a primary key is unique, and can therefore be used to identify each tuple in a relation (or a vote in this context) [38].

Example ballot

Parameter	Value
Survey_id	55492801
Section_id	55492802
Question_id	55492803
Response_to_question	176758
New_response_id	55566305
User_id	18880680
Creation_id	129.177.123.4

Table 3.1: *A vote cast by the author in the 2009 rectorial election at UiB. Response_to_question has three possible values: 176758 is a blank vote, 176756 is a vote for Team Grønmo and Rokne, and 176757 is a vote for Team Reed and Nordtveit.*

Creation_id

The voter's IP address (*Creation_id*) were also stored together with the votes.

3.1.4 Voting

A voter could choose between two different procedures to access the voting page in the rectorial election. Either he could log in to My Space and choose the hyperlink "Rektorvalg /Rectorial election 2009" on the front page, or he could use a hyperlink received in an e-mail from the election board secretary. The hyperlink pointed directly to the voting page in My Space. If the voter was not logged in to My Space when clicking on the link, My Space would first direct the voter to the login page, and then redirect the voter to the voting page after a successful login.

3.1.5 Data storage

My Space stores its data in an Oracle database [39]. Election data, such as the electoral roll, election settings, and votes, are stored together with other data from My Space. The votes are stored *unencrypted* [40].

3.1.6 Tabulation

Results from a survey can either be read directly from the survey's administration web page, or be exported to a file as comma separated values (CSV

file). This could be done at any time during the rectorial election period [41].

To simplify the tally, the rectorial election was organized as four separate surveys within different community groups at My Space. My Space's survey module does not contain any functionality for merging results from different surveys or functionality for handling weighted votes. As a consequence, the election board did the calculation of the final results in the rectorial election manually.

3.2 Vulnerabilities in the election module

We evaluated UiB's election system twice. First, we performed some limited tests during the rectorial election. Later, a second and more systematic evaluation was carried out on behalf of the IT Department at UiB. We completed our evaluation on May 20th 2009, and reported our findings to the IT Department the same day. The IT Department has since deployed countermeasures for some of the vulnerabilities. We have not evaluated the new countermeasures.

This section presents practical attacks exploiting the vulnerabilities found during the two evaluations, and explains how the election system violated the security and privacy requirements listed in Chapter 2.

3.2.1 Voter registration

Tampering with the electoral roll

It was possible for external threats to trick the election administrator into changing the electoral roll in My Space during the voting period. Changes could be initiated by sending HTTP requests (POST or GET) via the survey administration page. If the election administrator visited a malicious web page containing the code from Example 1, the code would first delete a voter with user ID 18880680, and then add a voter with user ID 13881433 to the electoral roll. This is a classical example of a Cross-Site Request Forgery (CSRF) attack. Note that the administrator had to be logged into My Space while he visited the malicious web page.

Using the survey module as an election system violated the *eligibility requirement* because it was possible for threats to deny eligible voters the right to cast a vote by deleting them from the electoral roll. We can also conclude that the system violated the *uniqueness requirement* for the rectorial election because a threat could cast four votes in the election by adding himself as a voter in the four different groups (A to D).

Example 1 Code deleting and adding voters to the electoral roll

```
<html>
<body>
<img src="https://studentportal.uib.no/[...]/deregister?user_id=18880680&[...]" />
<img src="https://studentportal.uib.no/[...]/add-member?target_user_id=13881433&[...]" />
</body>
</html>
```

3.2.2 Authentication and authorization

Privilege escalation

A modified version of the previous CSRF attack could give any user of My Space administrator rights to a survey. In Example 2, the argument *role_type* is added to the *add-member* request. An execution of this request would give the targeted user administrator rights to the survey.

Example 2 Code adding an administrator to a survey

```
<html>
<body>
<img src="https://studentportal.uib.no/[...]/add-member?target_user_id=13881433&
            role_type=dotlrn_cadmin_rel[...]" />
</body>
</html>
```

The opportunity for a threat to become administrator of an election violates several of the basic requirements. First, it violates of the *eligibility* and *uniqueness requirements*, because an adversary with administrator rights can add several unqualified (or fictitious) voters to the electoral roll. Second, administrators are able to read early election results, which violates the *fairness requirement*.

3.2.3 Ballot definition

Voting for a fictitious candidate

During the initial evaluation of the rectorial election, Vidar Drageide and the author submitted a vote for two fictitious candidates. To cast a vote for one of the official candidates, the voter submitted the *Response_to_question* value assigned to that particular candidate. Casting a vote for a fictive candidate needed some preparation. First of all, a new survey in My Space had to be created. The new survey was a so-called "deactivated survey", created within any group in My Space. A deactivated survey is only visible to the administrators of the My Space group the survey belongs to. In this survey,

the name of the fictive candidate was set as one of the predefined answers to a question. As a result, an identifier (*Response_to_question*) for the fictive candidate was stored in the My Space database. Next, when submitting a fictitious vote to the server, the value for *Response_to_question* was replaced with the fictitious candidate's value. When the central server performed the tally, the votes for the fictive candidates were presented together with the votes for the real candidates.

We might argue that the system violated the *soundness requirement* because it accepted votes for fictitious candidates, even though the votes were manually removed during the tabulation in the rectorial election.

3.2.4 Voting

Casting multiple votes

A voter in the rectorial election should only be able to cast a single vote. To cast multiple votes, a voter could submit one vote using the regular procedure and at the same time use a proxy based tool to capture the request. However, submitting exactly the same request once more failed because a vote with the same *New_response_id* was already stored in the database. Instead, if the voter manually increased the value of *New_response_id* (or at random picked a value not used before), then the server accepted this second vote. By repeating this last step, and picking new unused values for *New_response_id*, a voter could cast an unlimited number of votes.

A less technical method, *using only a web browser*, could also be used to cast multiple votes. The first step was to open multiple copies of the voting page in one web browser, before submitting any vote. Each copy of the voting page got its own unique value for *New_response_id*. Going through each page in succession, and casting one vote per page, generated multiple votes accepted by the survey module.[6] Once again we have showed that the system violated the *uniqueness requirement*.

The major flaw in the design of My Space's survey module, which made these attacks possible, was the lack of input validation. When delivering a vote, the server did not check if the voter already had cast a vote, it only controlled the voter's permission against the electoral roll.

[6]Two students cast multiple votes in the rectorial election, probably unintentionally using this method.

Casting votes on behalf of other voters

A CSRF based attack could be used to cast votes on behalf of other voters. This attack succeeded even if the voter already had voted. Multiple votes could be cast. The first step of the attack was to set up a malicious web page that automatically generated a new value for *New_response_id* and then submitted the vote to the election server. The second step was to trick voters, logged in to My Space, to visit the malicious web page.

Phishing

One of the most common techniques used to trick people into visiting malicious web pages is phishing. There are three reasons why the rectorial election was particularly vulnerable to phishing attacks. First, several e-mails with voting instructions were sent to all voters. A threat that wanted to carry out a phishing attack could simply reuse one of the official e-mails and spoof the sender address. Second, the hyperlinks in the official e-mails were created using UiB's service for making short aliases for redirections of long URLs.[7] A voter would not be able to control if the shortened URL pointed to a malicious server or the legitimate voting page in My Space. Third, My space has a built-in redirection functionality. If a user asks for a subpage of My Space before he has logged in, My Space first redirects the user to the login page. After a successful authentication, My Space directs the user back to the requested subpage. In a phishing attack, this subpage would be the page for delivering a pre-completed ballot made by the threat (using CSRF based techniques).

In summary, the phishing attack violates the *uniqueness requirement* and *eligibility requirement*. It also violates the *soundness requirement* because votes submitted from other web page's than My Space (using CSRF based techniques) are counted.

Coercion

Two features of My Space violate the *uncoercibility requirement*. First, a coercer may stand behind the voter's back and watch the voting. The coerced voter is not able to recast his vote. Second, a receipt of the cast vote is available on My Space throughout the election period.

[7] `http://link.uib.no/`

3.2.5 Data storage

My Space's survey module states that the rectorial election at UiB in 2009 is anonymous (see Figure 3.2, where the survey type is set to anonymous). Except for the first section in §7 of UiB's election regulations [42], there is no other references to the voters' anonymity. This section states that the election board must ensure that the electoral system takes care of anonymity and security in a paper based election. It is natural to put a question mark behind the anonymity statement in My Space, especially since every voter could read the content of his cast vote on My Space throughout the voting period. This ability indicates that there is a strong link between the voter and his vote when it is stored in the central database.

Access to anonymized data

A respondent can edit his answers in retrospect if a survey is set to be "editable". To edit a submitted response, the user enters a web page in My Space. This web page uses the parameter *response_id* (which is set in the URL) to look up the requested response. The server does not validate that the user is the legitimate owner of the previous submitted response. It only checks that the user is a member of the group the survey belongs to. If a threat (also member of the same My Space group) iterates through a sequence of *response_id*s, all answers to a given survey are revealed. (We remark that My Space did not disclose any information about the user IDs connected to the various responses.)

A threat can abuse this functionality for two purposes. First, he can produce a partial tally. Second, when looking at a response, it was also possible to edit the given answer and resubmit it.

The rectorial election was set to "non-editable", but it was possible to make a survey editable through a CSRF attack. Hence, a threat could produce a partial tally and change votes, thus, breaking the *fairness requirement* and *accuracy requirement*, respectively.

Access to non-anonymized election data

My Space has an e-mail notification service. Typically, a user can ask for e-mail notification whenever a new message occurs on a bulletin board or in a forum.

Administrators can ask for e-mail notifications for public surveys (opposite of anonymous surveys). Whenever a user responds to the survey, the administrator receives an e-mail containing the respondents name and all responses to the questions in the survey. An administrator should not be

able to ask for e-mail notifications for anonymous surveys like the rectorial election. However, there is a flaw in the system that can be exploited when a public survey is converted into an anonymous survey. A malicious administrator could initially set up a public survey, activate the e-mail notification service for that particular survey, and finally make the survey anonymous. The administrator would still get e-mail notifications for every response to the survey.

Furthermore, our tests revealed that any user of My Space could sign-up for e-mail notifications for surveys. Signing up for e-mail notifications is a post-request to a web page.[8] This service is typically used by a user who wants to receive an e-mail notification when a new posting occurs on a bulletin board or in a discussion forum. The two parameters in the notification request, *type_id* and *object_id*, are essential. *Type_id* refers to different modules in My Space. The survey modules *type_id* is 672. *Object_id* corresponds to the earlier described *Survey_id*. This attack could be executed by submitting a regular notification request for a bulletin board, and changing the values for *type_id* and *object_id* into the surveys values.

We did not test this e-mail notification attack on the rectorial election. Our e-mail notification attack did only target surveys created on My Space for testing purposes. When we found this vulnerability the IT Department was notified at once. They corrected the problem and asked for a new test. It is therefore likely that this attack also would have succeeded in the rectorial election.

The e-mail notification functionality obviously violates the *privacy requirement*.

Deleting responses

A malicious administrator can delete all responses to a survey. Deleting a question, and then creating the same question once more, will remove all previous responses to that particular question. Hence, the *accuracy requirement* was violated.

Deleting a survey

It is possible to delete an entire survey through the survey's administration web page. Moreover, deleting a survey can be done via a CSRF attack (see Example 3). According to warnings given on the web page, deleting a survey includes deleting all registered responses to the survey. This violates the

[8] `https://studentportal.uib.no/notifications/request-new`

accuracy requirement. The attack can also be seen as a Denial-of-Service (DoS) attack, and a violation of the *robustness requirement*.

Example 3 Code deleting a survey.

```
<html>
<body>
<img src="https://studentportal.uib.no/[...]/survey-delete?survey_id=55492801&
            confirmation=t[...]" />
</body>
</html>
```

3.2.6 Tabulation

Partial tally

The election administrator at UiB accessed preliminary election results for the rectorial election every day during the election [41]. A system that facilitates such activity clearly violates the *fairness requirement*.

Swapping votes between candidates

Candidates to an election are defined in one single text field in the administration tool in My Space (see Figure 3.3). Each candidate is separated by a line break. Recall that in a survey each candidate is represented by a numerical value assigned to the parameter *Response_to_question*. A candidate's value for *Response_to_question* depends on his position in the list of candidates. In practice, a reordering of the list of candidates will affect the candidates' value for *Response_to_question*. A user with administrator rights can edit the list of candidates at any time during the voting phase. Thus, interchanging two candidates' values for the parameter *Response_to_question* during the voting phase, will interchange all previous recorded votes for the two candidates. This is a violation of the *accuracy requirement*.

3.3 Aftermath of the rectorial election

Scrutinizing an online system in use, without asking for permission or notifying the system owners, is not unproblematic. In the aftermath of the election, the election board's chairman described our analysis of the system as sabotage and a violation of the university democracy [43]. Linda H. Lamone, administrator of elections in the State of Maryland, had a similar reaction when Professor Aviel D. Rubin and his research group at Johns Hopkins

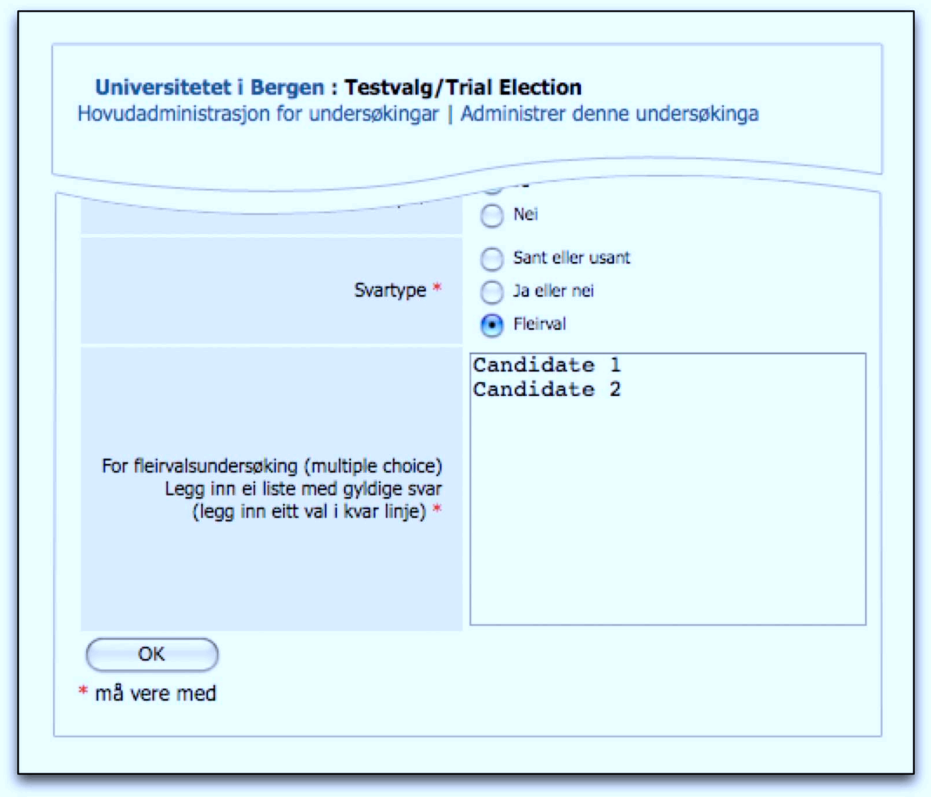

Figure 3.3: *Administrative tool in My Space for defining candidates. A reordering of candidates during the voting phase will interchange all previous cast ballots between candidates.*

University reported that some of the voting machines in several states in the U.S. had serious vulnerabilities. With a clear reference to Rubin's group she stated:

> "Computer Scientists who question the security of electronic voting machines are undermining our democracy [44, p. 216]."

A clarification about what we did *during* the voting period, and why we did it follows.

3.3.1 Motivation for analyzing UiB's election system

We wanted to collect public available information about the election system, and learn about how the system behaved during the voting period. Early in our information gathering, it became clear that the voting procedure was vulnerable to CSRF attacks. To test and document a possible CSRF attack,

a web page that automatically submitted a blank vote was created on a local web server. Initially, the plan was to only cast one vote (on behalf of one voter) using this method—and thereby not violate the election regulations. For this reason, neither the IT Department nor the election committee were notified ahead of the tests. It came as a *complete surprise* to us that by simply reloading the web page implementing the CSRF "attack", we could cast an unlimited number of votes. Using this method, the author cast a total of four blank votes, and Vidar Drageide cast one blank vote. It is important to notice that we made no attempts to penetrate any of the election servers. Nor did we try to get unauthorized access to any information, or try to manipulate the election result.[9]

During the voting period the author and Vidar Drageide also cast one vote for two fictitious candidates. In retrospect, we see that it may be naive to argue that voting for fictitious candidates in an electronic election is similar to writing the name of a fictitious candidate on a paper ballot. All other vulnerabilities presented in this chapter were discovered during a second evaluation, carried out on behalf of the IT Department.

We decided not to notify the system owners about our findings before the election had come to an end. It is reasonable to ask why. First, based on the principle that information about the votes and the election result should be unknown until the voting period has ended (cf. the privacy and fairness requirements in Chapter 2), we strongly believed that we had some more time to analyze our findings. Second, we concluded that there was a low risk that a threat would exploit the discovered vulnerabilities. We felt that it would be irresponsible to inform the system owners without taking the time to analyze our findings. Taking hasty decisions based on too little information can be damaging; and we would not force the system owners to do so. Thus, we thought it would be best to not disrupt the ongoing voting, but rather try to initiate a responsible disclosure process after the election.

3.3.2 Reactions to our analysis

Our assumptions turned out to be wrong. As mentioned earlier in this chapter, much to our surprise the election administrator at UiB looked at preliminary election results every day. Consequently, the vote for the fictitious candidates was detected, and an investigation was initiated. During the investigation, two employees at the IT Department were able to link the four blank votes and the vote for the fictitious candidates back to the author.

[9] In theory, it is possible that four blank votes (weighted 25 %) can affect the election outcome in the rectorial election at UiB. If no candidate pair gets majority, a second election round is held.

It was also during this investigation that the two students that had voted multiple times by accident were discovered.

The election board's chairman and the election board's secretary seemed not to be interested in our suggested responsible disclosure process. Shortly after we confirmed the factual circumstances regarding the five blank votes in a meeting with the election board's chairman, the election board's secretary, and the IT director, the chairman and the secretary publicly announced that nine irregular votes had been rejected from the final tally [41]. At this point we had not revealed any technical details about the vulnerabilities we had found, but agreed to have a new meeting with the IT Department to discuss the details of our findings.

In [41], the election board's chairman and secretary ensure the voters that the election system works as intended, despite the fact that nine "irregular" votes were rejected in the rectorial election. Moreover, they argue that in order to verify the election's integrity, it is necessary to have the opportunity to link votes to the voters. But they ensure that only two "super users" at the IT Department are able to perform this kind of surveillance and trace votes to voters.

The candidates who lost the election did find the reports on "irregular" votes and election surveillance somewhat disquieting, and asked for an external investigation of the rectorial election. Ernst & Young was hired by the University Director to perform the investigation.[10]

Even though the rectorial election was subject to an external investigation, UiB employees made several statements about the integrity of the rectorial election and My Space's suitability as election system. For example, one week before Ernst & Young was supposed to deliver their report, the election board's secretary stated that the rectorial election was absolutely anonymous, and that the election result was correct [45]. Similarly, the election board's chairman points out that the election board, after consulting with the IT Department and the Division of Academic Affairs at UiB, did not find any system errors or threats indicating that UiB will obtain invalid or fraudulent election results by continued usage of My Space's survey module [46, 47].[11] As the rectorial election got more publicity [48, 49], the election board's chairman finally conceded that My Space's survey module does not provide adequate security for election purposes [43].

[10] Ernst & Young's investigation and reports will be discussed in Chapter 4.

[11] The Division of Academic Affairs is the system owner of My Space which the survey module is a part of.

3.3.3 Is My Space suitable as election system?

It is difficult to draw general conclusions about how the discovered vulnerabilities and irregularities in the 2009 rectorial election affected the voters' level of trust. However, the fact that the candidates who lost the election filed a complaint and asked for an external investigation expresses some degree of untrusts.

Evidently, My Space's survey module violates at least eight of the requirements for e-voting systems listed in Chapter 2. The survey module was not originally designed for election purposes, and consequently important security and privacy requirements for elections were not considered during the development phase. This example demonstrates how important it is to analyze the security and privacy of an e-voting system *before* it is deployed.

Finally, UiB's IT director announced that My Space's survey module probably will be replaced with a new election system [50]. During the 2009 student parliament election, UiB used the University of Oslo's e-voting system to organize the election.[12]

[12] `https://valg.uio.no/`

Chapter 4

Election Forensics—Rebuilding Lost Trust

Smart election security not only tries to prevent vote hacking—it prepares for recovery after an election has been hacked.
—Bruce Schneier

In the aftermath of an election, the election result can be put into doubt. Reasons can be suspicion of misfeasance that may have affected the election outcome (for example reports of lost ballots), or suspicion of outright election fraud. Merly suspicion of an improper election result may be enough to undermine the public's trust.

Election forensics is important in order to rebuild lost trust. One aspect of election forensics is to gather evidence proving that the election result is correct—trust is rebuilt due to proof of correctness. Another aspect of election forensics is to detect evidence of misfeasance, misuse, and attacks on the election system, as well as technical problems that may have affected the election outcome [51]. If such evidence is found, making the necessary decisions—like announcing a new election round—can rebuild trust.

Election forensics is not only important for verifying the integrity of an election outcome; it can also be used to correct the election result. For example, if a technical problem during the voting phase corrupted some of the cast votes, an investigation may recover the lost votes [51]. Finally, election forensics can help us to understand *why* problems occur, and how to proceed in order to prevent similar problems in the future.

The first part of this chapter describes the investigation of the rectorial election at UiB in 2009. Based on experiences from that investigation, the

next sections focus on the information that must be gathered during an election in order to support later investigations, and how to detect indications of irregularities. Finally, different techniques for protecting the integrity of gathered information are presented.

4.1 Investigation of the rectorial election at UiB in 2009

As a consequence of the reported abnormalities in the rectorial election at UiB in 2009 (rejection of votes, suspicion of election surveillance, and partial tallies during the voting period), the candidates who lost the election filed a complaint and asked for an independent investigation of the election [52]. The University selected Ernst & Young to do an investigation, focusing on the following questions:

- Who have had the opportunity to access election data during the election?
- Who have actually been reading election data during the election?
- What security mechanisms were added to the system to ensure that no unauthorized individual can access the system?
- Is it possible to trace user activity in the election system?

4.1.1 Findings

Ernst & Young documented the results of their investigation in two different reports—one which UiB made publically available [40], and one that UiB chose not to disclose to the public [53].[1] The reports are based on interviews and an investigation of the election servers' electronic logs. The most important findings from the two reports follow.

Access to election data during the election

In the reports, Ernst & Young distinguishes between *anonymous election data* and *non-anonymous election data*. For data that are not anonymous, it is possible to make a link between a voter and his vote, whereas anonymous election data only give information about the content of each vote.

[1] UiB has given the author access to a copy of the private report, in which named individuals have been anonymized.

Recall that the election board's chairman and secretary stated that only two "super users" at UiB were able to read non-anonymous election data (see Section 3.3.2). Ernst & Young found that a total of twenty individuals, all employees at the IT Department and the Division of Academic Affairs, were able to read non-anonymous election data. Furthermore, eight individuals could act on behalf of the voters in My Space, and consequently read the content of their cast votes. These eight individuals could also cast votes on behalf of the voters.

Eleven individuals at the IT Department had access to My Space's database and the possibility to read, alter, and delete non-anonymous election data during the election period.[2] Two individuals had access to non-anonymous election data via backups. One person, the election board's secretary, had access to anonymous election data.

Finally, Ernst & Young found that one individual, without any relationship to the IT Department, or the Division of Academic Affairs, was registered as "administrator" in one of the voting groups in My Space. A test revealed that this user did not gain access to any kind of election data as a result of his user role. However, Ernst & Yong notes that this user's further permissions and rights are unknown.

Reading election data during the election

When the election board's secretary discovered the vote for the fictitious candidates, UiB initiated their own investigation. This investigation was started during the election. Based on interviews and electronic logs, Ernst & Young found that two individuals (including the secretary) had read anonymous election data during UiB's investigation. Furthermore, two individuals at the IT-Department had read non-anonymous election data during this investigation, in order to identify irregular votes.

Ernst & Young did not find any other individuals admitting that they had misused their rights in the system. Ernst & Young was unable to confirm many of these statements for two reasons. First, among the twenty individuals with access to non-anonymous election data, several individuals shared user credentials. Second, user activities in the election system were only partly written to the log. Especially, most queries to the database containing the election data were not logged during the election.

The only thing Ernst & Young could confirm based on the log files (assuming that the integrity of the logs are correct), was that the eight individuals with the opportunity to act on behalf of other voters, had not done so.

[2] One individual among the eight who could act on behalf of other voters is also among the eleven individuals at the IT Department with access to My Space's database.

Security mechanisms to prohibit unauthorized access

According to the reports from Ernst & Young, UiB has neither developed any formal list of requirements for their election system, nor formally specified the individuals with legitimate access to election data. Moreover, votes stored in My Space's database are not encrypted, and there exists no mechanism ensuring the integrity of stored votes. Finally, the reports state that UiB has not performed any kind of formal risk assessment of My Space's survey module, or My Space as a system.

Other observations

During the election, the election board's secretary made an overview presenting voter turnout for different periods of the voting phase. In one period there is a decrease in overall turnout, compared to the previous period. The secretary explains that this decrease was caused by typing errors when making the overview. Ernst & Young was unable to verify this claim.

In order to make the survey module in My Space more "suitable" for election purposes, the IT Department has deployed several changes to the survey module's source code. My Space's system owner, the Division of Academic Affairs, has not always been involved in this process. Ernst & Young recommends that deployment of changes should be made more formally, and involving the system owner.

4.1.2 Conclusions about election integrity

It was difficult to find evidence either proving or disproving the election result's integrity because user activities in the election system were only partly logged, and the lack of security mechanisms ensuring integrity of election data. Hence, the reports from Ernst & Young do not draw any conclusions about the correctness of the election result.

Because no evidence existed to prove that the election result was incorrect, the election board concluded that the election result was correct [52].

4.2 Preparing for election forensics

Usually, it is the election board and the election officials that have the "burden of proof" when an election outcome is put into doubt. As exemplified in the previous section describing Ernst & Young's investigation of the rectorial election at UiB, gathering necessary documentation and information for use

in a possibly forthcoming investigation is an important part of the election process. Oppliger and Rytz [54] state:

> "[E]vidence gathering focuses on collecting evidence before an event occurs, whereas forensics focuses on collecting evidence after an event. Nevertheless, evidence gathering and forensics should go hand in hand and complement each other to provide the most significant digital evidence possible."

When it comes to e-voting systems, it is not always easy to determine which information to collect. Too much information gathered may violate the voters' privacy. In [6, pp. 76–78], a list of data to collect is presented. This list is especially tailored to elections using e-voting machines in controlled environments. Based on Ernst & Young's investigation of the rectorial election, and the overviews in [6] and [51], Appendix A presents some examples of data to collect that are more relevant for remote e-voting systems.

4.3 When shall an investigation be initiated?

Bishop et al. [51] suggest that election forensics is initiated when suspicion exists that one or several basic election requirements are violated. Basic election requirements may for example be the non-functional requirements presented in Chapter 2. Information gathered during the election (see Appendix A) can be used to evaluate indicators, established ahead of the election, to determine possible violations of requirements. In e-voting systems, some indicators can be realized by the election system's software itself.

4.3.1 Establishing indicators showing possible violations of election requirements

Bishop et al. [51] lists several indicators, especially tailored to voting machines used in controlled environments. Some of the indicators hold for remote e-voting systems as well. Based on [51], this section presents examples of indicators relevant for remote e-voting systems.

Abnormal system behavior

Abnormal system behavior, like repeated system failures, or undocumented error messages from software or hardware components, may indicate the need for an investigation. In a remote e-voting system, abnormal system behavior does not necessarily occur on the server side. Voters may experience

abnormal behavior on their computers. Election officials depend upon reports form the voters to detect abnormal behavior on the voting clients. Because voters' technical skills are likely to vary, it may be wise to announce some information about what voters can expect as normal system behavior, as well as abnormal system behavior on the voting clients. All feedback from voters must be systematically categorized.

Abnormal voter turnout

An expected number of votes in each precinct can be estimated before the election. If the total number of votes, or the number of votes in a precinct, is obviously too small (or negative), or obvious too large, it is natural to raise a question about accuracy. For example, in the presidential election in the U.S. in 2000 Al Gore got a negative count of −16022 votes in Volusia County in Florida [5, Ch. 6]!

Abnormal voter behavior

Empirical data from previous, or similar elections, can be used to predict the voters' behavior. Abnormal—or unexpected—voter behavior may indicate the need for further investigation. Figure 4.1 shows the distribution of received e-votes over time in two different elections in Estonia [55]. By comparing the two elections we see several similarities in voter behavior. During the voting phase, almost every day has a local peak of received votes around 9 AM and 21 PM. Moreover, in both elections the number of cast votes peaks during the last hour of the voting phase. An unexpected large number of votes cast within a limited time frame at an unexpected point of time, may indicate irregularities.

To mitigate coercion and vote selling, the voters in Estonia are able to vote multiple times via the Internet. Only the last vote cast will be included in the final tally. Furthermore, voters who have voted via the Internet can also cast a paper vote (p-vote) on election day. A p-vote will cancel all previous e-votes cast by the voter. The Norwegian E-vote 2011 project will use a similar arrangement for the 2011 municipality election.[3]

Table 4.1 shows the number of voters who cast multiple e-votes, and the number of voters who canceled their e-votes by casting a paper vote, in four recent Estonian elections. A high number of cancelled votes may indicate that coercion is a serious problem, but there may exist other explanations as well. For example, a political event during the voting phase may lead voters to change their mind and cast a new vote. There is also the possibility that

[3] `http://www.e-valg.dep.no/`

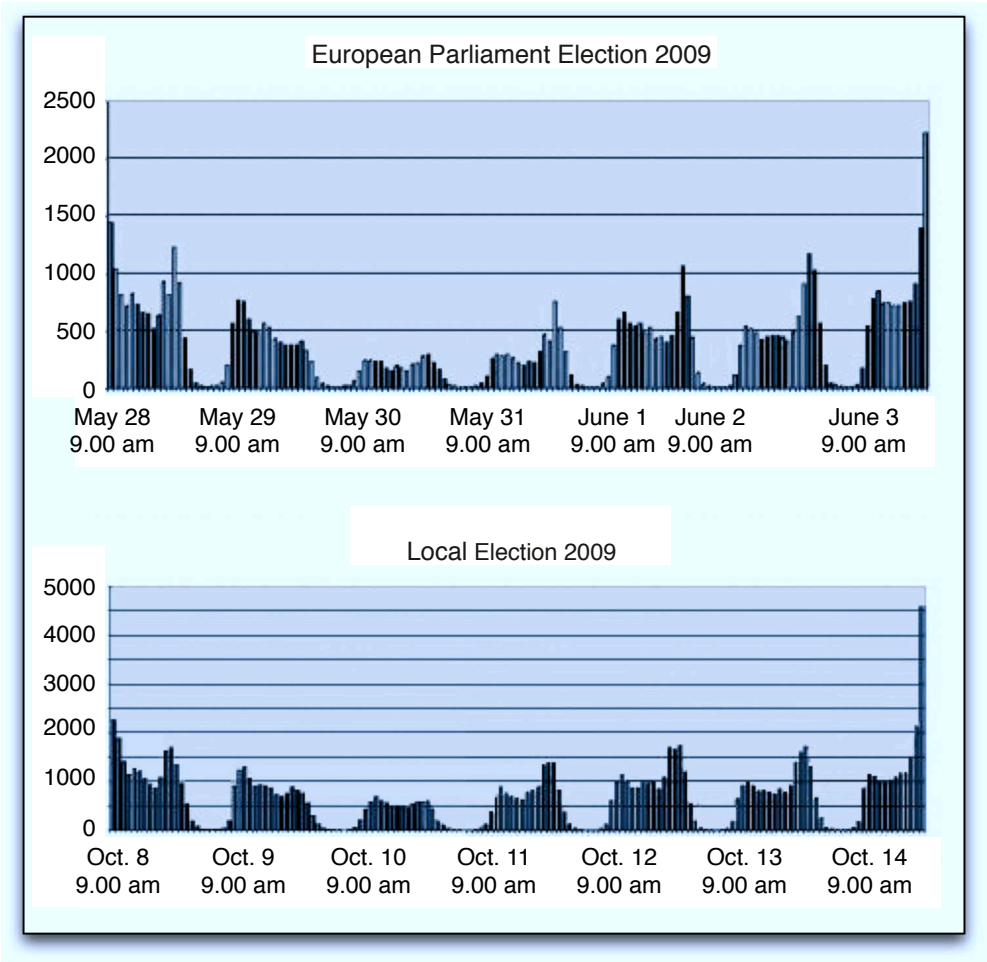

Figure 4.1: *Number of e-votes received each hour during the e-voting phase of two different elections in Estonia in 2009.*

there exists some malware, installed on the voters' computers, resubmitting votes on behalf of the voters. (Note that only a few cancelled votes do not necessary imply that the problem of coercion is non-existing. It may be the case that the countermeasures are ineffective.)

Abnormal ballot corrections

In many elections, the voters are allowed to make corrections on the ballots, for example, adding write-inn candidates to the ballot, or make a reordering of the listed candidates. An unexpected high (or low) number of corrected ballots should be investigated. It is conceivable that a special political situation, or usability issues on the voting client can affect the number of corrected

Statistics about Internet Voting in Estonia

	Internet Voters	E-votes Cancelled by P-votes	Multiple Internet Votes
Local Election 2005	9 317	30	364
Parliamentary Election 2007	30 275	32	789
European Parliament Election 2009	58 669	55	910
Local Election 2009	104 413	100	2 373

Table 4.1: *The table shows total number of votes cast via the Internet, the number of voters who canceled their e-votes by casting a p-vote, and the number of voters who cast multiple votes via the Internet, in four recent Estonian elections.*

ballots in an election. However, it may also be the case that a threat is manipulating the voters' ballots, for example using some sort of malware installed on voters' computers.

Votes cast from the same computer

A large number of votes cast from the same computer should be investigated. For example, the majority of Estonian voters only cast one e-vote (see Table 4.1). Voters within the same household may use the same computer to cast their votes, thus, one should at most expect a handful of votes cast from each computer. The exception here is computers placed in public places like libraries, schools, and Internet cafés.

Prediction of election results

Social scientists have shown that many elections are highly predictable, especially presidential elections in the USA. Alvarez and Katz [56] suggest that prediction of election results can be used to indicate irregularities and election fraud.

In the 2002 general election in the USA, the State of Georgia introduced touch-screen based voting machines produced by Diebold. Because of Diebold's strong connections to the White House and the Republican Party, there was a widespread concern—especially among Democrats—that the voting machines might be tampered with in favor of the Republicans [57]. In order to try out the hypothesis of using election predication as a method for detecting election fraud, Alvarez and Katz made predictions of the election results for all 159 counties in Georgia. Their predictions included both the

senate election and the gubernatorial election. For each county, Alvarez and Katz compared the actual election result to a 95 percent confidence interval of the predicted result. The election results were outside the confidence intervals in 35 counties in the gubernatorial election, and in 34 counties in the senate election. Because the republicans did better than predicted in all 35 counties of the gubernatorial election, and the democrats did better in all 34 counties of the senate election, Alvarez and Katz find it unlikely that the voting machines had been systematically fixed. They argue that if someone had systematically tampered with the voting machines, they would in that case have rigged the machines in favor of the Republicans in the gubernatorial election, and in favor of the Democrats in the senate election. Alvarez and Katz find this unlikely, and conclude that the unexpected election results can be explained due to poor tactical political choices.

This may be a reasonable explanation. However, it may also be the case that the forecast model was wrong. Alvarez and Katz based their forecast model on two variables: results from the previous races for governor and senate, and the percentage of the county population that is non-white. In this analysis, Alvarez and Katz were not interested in getting a "true" prediction, only a good forecast. It is not unlikely that using other variables, like rate of economic growth or inflation, would have produced a different forecast.

Another method for predicting election results is the use of a *prediction market* (also know as information market or election stock market). In a prediction market, a group of individuals is used to predict the final election result [58]. The group members are not asked what they are going to vote on election day, but what they think the final election result will be. The basic idea is that every individual possesses some relevant information about the election and the political landscape. By aggregating all individuals' information it is possible to make a good prediction of the final result. Prediction markets have proven to be a good method for predicting election results. Arnesen [58] conducted an experiment with prediction markets before the 2009 Norwegian parliamentary election.[4] Compared to contemporaneous polls, the prediction markets were closer to the final election result in 88 % of the comparisons [58].

Prediction of an election result, either using a prediction market or a variable based forecast model, may be used as an indicator for detecting irregularities. The main benefit of using election predications as an indicator is that it can be realized by the administrative part of the e-voting software. Because of the uncertainty of election prediction in general, this indicator should be used in addition to other indicators.

[4] `http://past2009.uib.no/info/about.php`

Benford's Law Frequencies		
Digit	First digit	Second digit
0	-	.120
1	.301	.114
2	.176	.109
3	.125	.104
4	.097	.100
5	.079	.097
6	.067	.093
7	.058	.090
8	.051	.088
9	.046	.085

Table 4.2: *Benford's Law frequencies for the first and the second digit.*

Test based on Benford's Law

Recently, some research has been done to find other statistical methods—not depending upon (subjective) assumptions about the election result—detecting indications of election irregularities. *Benford's law* (also called the first digit law) states that in many sets of statistical data, the value of the first digit and the second digit follows a specific distribution. Hill describes the phenomenon of Benford's law in [59]. For example, the digit one will occur as the first digit with a probability of 30.1%, meanwhile the digit nine will occur as the fist digit with a probability of only 4.6%. The frequencies of first and second digits according to Benford's Law are presented in Table 4.2.

Benford's Law has been found to have many "real-world applications". For example, several occurrences of financial fraud have been detected using Benford's Law [59, 60]. Benford's Law can also be used to test new mathematical models [59].

The most important question in this setting is whether Benford's Law applies to vote counts, and can be used to indicate irregularities. This is a controversial topic (see [61, 62] and [63, pp. 132–134]).

It is quite clear that vote counts at the precinct level often don't follow the distribution of Benford's Law when looking at the at the first digit. The reason for this may be that precincts often are designed to be of equal size, including the same amount of voters. From this observation we see that if a candidate or a party gets roughly the same percentage of votes in each precinct some digits will be overrepresented [61]. Taking this into account,

Mebane [61] chooses to focus on the second digit when looking at election results.

According to Hill [59] the phenomenon of Benford's Law often occurs when (random) samples of data from different distributions are combined. Mebane [61] claims that vote counts on precinct level can be seen as a mixture of different statistical distributions. He explains this by arguing that voters have to make several decisions, like when and where to vote, and for which party or candidate to vote for, before they cast their vote. Furthermore, it is expected that a small proportion of voters will make "mistakes" when they vote. Mebane states that [64]:

> "Differences in partisanship, economic class, mobilization campaigns, administrative rules, and other details cause the proportion of voters who intend to choose each candidate or ballot question to vary across precincts. If such variations are combined with small and also varying probabilities of 'mistakes' in making or recording each choice, then the resulting precinct vote counts will often follow the 2BL [second digit Benford's Law] distribution."

Mebane argues that systematically manipulations of election results, especially in tied races, will be reflected by a 2BL test [64]. However, he also notes that the 2BL test is not sensitive to all kinds of manipulations. If the amount of manipulation is small and distributed randomly among the precincts, the 2BL test will probably not detect the fraud. Similarly, a vote count that diverges from Benford's Law does not necessarily indicate irregularities like election fraud. Mebane suggest that a Benford's Law test should be used as a supplement to other indicators.

Several elections have been tested against Benford's Law. For example, the presidential elections in the U.S. in 2000, 2004 [64], and 2008 [62], the presidential election in Iran in 2009 [65, 66], and the presidential election in Mexico in 2006 [61]. Because of the uncertainty of the results from these analyses, and the controversies regarding use of Benford's Law in election forensics, more research has to be done before we can say that this is a solid indicator.

The main benefit of a Benford's law test (assuming that precinct level vote counts follows a Benford's Law distribution) is that it does not depend upon predictions of election results, or assumptions of voters' behavior. Furthermore, given vote counts on precinct level, the test is easy and efficient to compute. Thus, this second-digit test can easily be implemented in the administrative software in e-voting systems.

Last-two digits distribution

A quite different approach than Benford's Law has been used to look for non-random patterns in vote counts. To analyze Venezuelan elections between 2006 and 2009, Levin et al. [67] looked at the two last digits in vote counts. Individuals tend to avoid paired numbers (like 11, 22, 33 etc.) if they are asked to write down a random sequence of numbers. If the distribution of the two last digits in a vote count is not uniformly distributed (i.e. paired numbers occur less frequently than one tenth of the time), further investigation should be initiated.

To exemplify this indicator, we analyzed the 2009 Norwegian parliamentary election.[5] Here we looked at the last-two digits in vote counts for every precinct in the four largest cities in Norway (Oslo, Bergen, Stavanger, and Trondheim). By summarizing the occurrence of paired numbers in the two last digits, we found that paired numbers had an average frequency of 0.110, close to the expectation of 0.1. Figure 4.2 shows the frequency of paired numbers for the seven largest parties.

The distribution of the last two digits is not sensitive to all kinds of election fraud. However, the test of the distribution is easy to implement in an electronic voting system, and is likely to detect fake election results made by individuals.

4.4 Protecting the integrity of collected information

Collected information, intended for use as evidence in an investigation process, must be shown to be trustworthy. The main weakness of electronic evidence is that it is easy to manipulate, and the fact that it has been manipulated can be difficult to detect or prove. Different techniques can be used to make manipulation of data more difficult, and possible manipulation easier to detect. This section presents two different approaches to increase the trustworthiness of collected information.

4.4.1 Digital signing of documents

Documents can be signed digitally if a Public Key Infrastructure (PKI) is available. The purpose of signing a document digitally is to prove that the document has not been changed after it was signed. The signature will

[5] `http://www.regjeringen.no/krd/html/valg2009/`

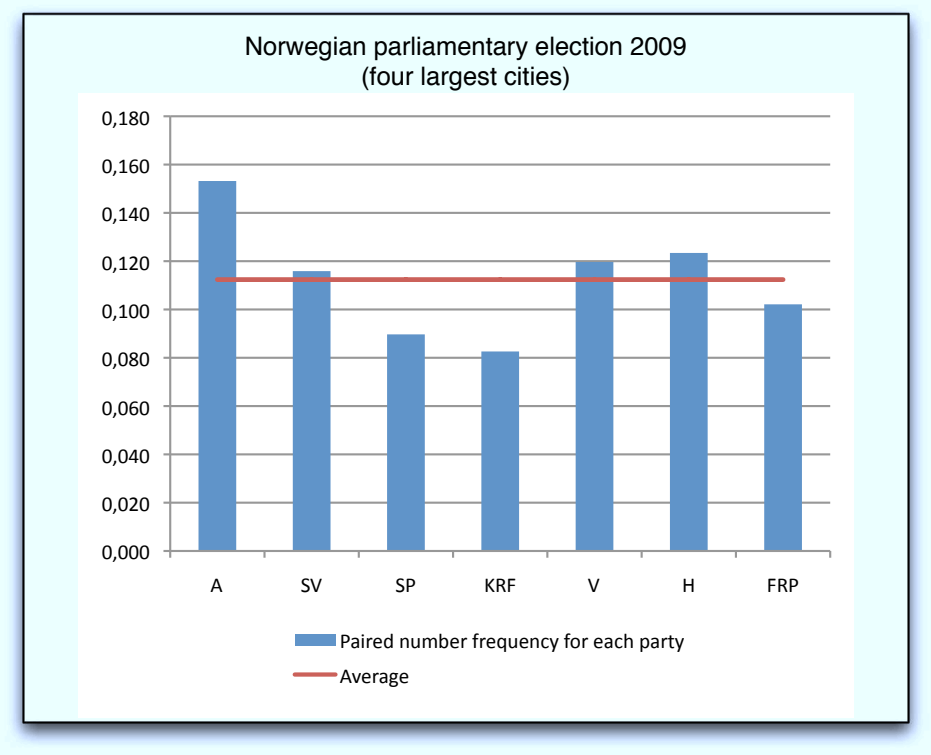

Figure 4.2: *Frequency of paired numbers in the Norwegian parliamentary election in 2009.*

also point to the entity (i.e. an individual or a computer) that signed the document.

In practice, signing a document digitally is a three-step process. First, the signer hashes the document, using a one-way hash-function. Second, the signer makes the "signature" by encrypting the hash-value with his private key. Third, the signature is attached to the document.

Verifying a document's signature is a similar process. First, the verifier hashes the document by using the same hash-function. Second, the verifier decrypts the "signature", using the signer's public key. The signature is valid if the result of the decrypted signature equals the document's hash-value (see [68, Ch. 2 and 13] for more information about digital signatures).

As pointed out by Oppliger and Rytz in [54], the trustworthiness of a digitally signed document depends on several security assumptions. First, we must assume that that the private key—used to sign a document—is not compromised. Second, we must assume that the cryptographic hash function works as intended. Third, we must assume that a document's hash value is unaltered when it reaches the signing device.

4.4.2 Immutable audit logs

User interactions with the election system and communications between system components are typically stored in electronic logs. The main purpose of logging and auditing is to identify policy violations and unwanted behavior in systems. An extensive use of logging and well-established policies for documenting important procedures may also have a preventive effect. Potential adversaries may find it less attractive to try to violate the election system if they are aware of the fact that all interactions are under surveillance.

A well-known problem with regular logging is that an adversary can take control over the log and hide his attacks on the system. To protect the integrity of an audit log, one must deny adversaries the ability to delete, alter, or reorder existing log entries, as well as inserting fake log entries. The term *immutable audit logs* (also known as tamper-resistant logs) denotes technical approaches to make logs more reliable.

Schneier and Kelsey [69] presented a method for creating secure logs to support computer forensics. Their method has been used as a foundation for many subsequently proposed secure logging systems. In this section, we present some of the main concepts in Schneier and Kelsey's logging scheme to give an idea of what a secure logging systems may look like.[6]

The Schneier-Kelsey scheme takes place between an untrusted logging machine $\mathcal{U}$ and a trusted remote server $\mathcal{T}$. Before $\mathcal{U}$ opens a new audit log, it establishes a shared secret key A_0 with $\mathcal{T}$. New secret keys are regenerated for every new log entry L_i by using a one-way hash function: $A_{i+1} = H(A_i)$. Once a secret key A_i is used, $\mathcal{U}$ computes A_{i+1}, and deletes its copy of A_i. Each log entry L_i contains three parts:

1. Log entry data, denoted M_i, $i = 0, 1, 2, 3, \ldots$
2. A chained hash Y_i where:

$$Y_i = H(M_i \| Y_{i-1}) \text{ and } Y_0 = H(M_0).$$

3. A message authentication code (MAC) denoted as Z_i, computed over Y_i with the current secret key A_i:

$$Z_i = MAC_{A_i}(Y_i).$$

To close the log file, $\mathcal{U}$ writes a special final-record message D_f and deletes A_f. In Schneier and Kelsey's complete version of the scheme, the data field

[6] This simplified version of the Schneier-Kelsey scheme is based on Ma and Tsudik's presentation of the scheme in [70].

in every log entry is encrypted. The possibility to encrypt the data field is especially useful if sensitive data are logged (see [69] for further details).

Due to this scheme, it is possible for $\mathcal{T}$ to verify the hash chain by going through Y_0 up to Y_f. Because $\mathcal{T}$ knows A_0, $\mathcal{T}$ is able to calculate A_f by hashing A_0 f times. In order to verify the correctness of Z_f, $\mathcal{T}$ can calculate $Z_f = MAC_{A_f}(Y_f)$.

Schneier and Kelsey note that their system does not prevent all possible manipulations of the audit log. However, the scheme will detect most manipulations, like deletions of log entries. In other words, the purpose of the system is to detect manipulations after the fact. Based on this observation they suggest that the log can be written to a "write-only" media, for example a CD-ROM or a paper trail. Furthermore, if $\mathcal{U}$ is likely to be compromised, $\mathcal{U}$ can send its log entries to $\mathcal{T}$ continuously.

Note that one can never guarantee that a logging mechanism will gather all intended information. Developers may have created "back doors" in the system—either deliberately or accidental—where user activity may omit logging [71].

4.4.3 Design considerations for immutable audit logs

When implementing audit logs, several important design considerations must be made [71]. An e-voting system is likely to have a large group of users (voters, election officials, and system operators). Logging all users' interactions with the system, all information exchanges between system components, and all activity within each system component, will create an enormous amount of data. Hence, system owners must carefully consider what to log to fulfill the auditors' requirements.

Furthermore, if we consider the time complexity of generating each log entry, and the additional information added to each log entry in order to ensure integrity, immutable audit logs create some more "overhead" compared to regular logs. Extensive logging may decrease overall system performance. Thus, deciding what to log in regular logs, and what to log in immutable audit logs are important.

One should also be aware of the fact that aggregation of data from different logs may potentially violate users' privacy. All collected data must therefore be considered as a whole.

Finally, in order to make an audit system functional, necessary tools to read the audit log must be provided. For example, in 1996 Anderson [72, pp. 384–385] found that Sun's operating system Solaris did not provide tools to read audit data. Nor was the audit format documented. Anderson states that:

> "The audit facility seemed to have been installed to satisfy the formal checklist requirements of government system buyers, rather than to perform any useful function."

4.5 Dealing with reports of irregularities

According to [6, p. 63], many election officials prefer to avoid careful scrutiny of the election process because the likelihood of irregularities being exposed increases. Such an attitude is somewhat dubious. Even though election officials often have limited resources, it is important that both voters and candidates can trust election officials to do everything within their power to ensure that irregularities and election fraud are detected. If evidence exists suggesting that election officials have neglected indications of irregularities, or not collected the necessary information to prove the correctness of an election, the public's trust in the election will decrease.

An electronic election is an irreproducible event [54]. In other words, it is impossible to reconstruct what exactly happened when each ballot were cast. This makes it often difficult to determine the impact of discovered irregularities in e-voting systems. Most attacks on paper-based elections do not scale very well—in electronic elections they do [73].

Examples of irregularities with unknown impact may be reports of malware on voters' computers changing the voters' ballots, or discovering an anonymous web site buying votes. It is difficult to determine how widespread the malware is, and it may be impossible to determine if vote buyers managed to influence the election result. There is no easy answer on how to react to irregularities like this. However, it is important to consider such scenarios before deploying an e-voting system.

Chapter 5

Trust by User Involvement

Trust but verify.
Ronald Reagan

In most traditional paper-based elections the voters cast their votes and hope the votes will be included—unaltered—in the final tally. Trust in paper-based election systems is often based on layman control, independent election observers, and the fact that the election procedure is relatively transparent and easy to understand. This is not the situation for most e-voting systems. An e-voting system is often too complicated to understand for a regular voter, and layman control becomes difficult. This Chapter presents some techniques and protocols that can be used to verify critical parts of the voting and tabulation phases in an e-voting system.

5.1 Is disclosure of source code a key factor for free and fair elections?

Election systems where source code for the computer systems used to cast, receive, process, and tabulate the votes are not disclosed, are often referred to as *black box voting systems*. Adida [74] states that it is an important principle to not count votes in secret. He argues that using proprietary software, with undisclosed source code, means counting votes in secret.

Many leading e-voting suppliers do not share much information about how their system works [75]. Especially, they do not disclose their source code [44]. The Norwegian E-vote 2011 project has made an important decision by demanding that the software vendor delivering the e-voting system must

make the source code publicly available.[1] Disclosing source code sends an important signal about openness and transparency. However, most voters do not have the necessary skills to make a qualified opinion about the system's suitability as an election system by looking at the source code. It is therefore important to ensure that groups of competent people make an effort to verify the quality of the source code.

According to Arce [76, p. 10], the CTO of Core Security Technologies:

> "Reliable software does what it is supposed to do. Secure software does what it is supposed to do, and nothing else."

It is crucial that an electronic election system does what it is supposed to do, and nothing else. Even though all source code for the election system is made publicly available for inspection, we still have two major challenges to overcome before we can conclude that the software is secure enough to be used for election purposes. First, how does one proceed in order to verify that the software "does what it is supposed to do, and nothing else"? Second, how can one verify that the software in the different election components are based on the source code that is made publicly available?

5.1.1 Analyzing the source code

Reviewing source code is a difficult task. It can be done automaticly, or manually. One example of automatic code reviews is *static code analysis*. Static code analysis reviews the source code based on predefined rules [76]. Doing static code analysis is important, but this is first and foremost an approach to find bugs and to some degree security flaws—not to detect malicious code that is added to the source deliberately.

A manual code review can be done in an effort to detect malicious code or unwanted system behavior. This requires that the source code is easy to read and well documented. Manual code review of complex systems, like an e-voting system, is likely to be very time-consuming and hard. According to Rubin, a manual code review is hard or almost infeasible [44, p. 240]:

> "If a graduate student in computer science can hide bugs in five-line programs that can fool ten other grad students and two senior researchers, imagine what professional programmers can hide in fifty thousand lines of code."

[1] Note that this demand does not apply to the software used to scan paper ballots, nor the authentication mechanism used to authenticate the voters.

5.1.2 Which code is actually running?

A suggested solution to the second challenge—how to decide whether or not the software in use on the election server differs from the source code that is made publicly available—is to compute a hash of the compiled code once it is verified and approved [44]. By generating a new hash of the installed compiled code at a later point in time, election officials and election observers can compare the new hash to the one created at the point of approval. If the hashes are equal, they can conclude that it is the approved compiled code that is actually running on the election server. This sounds like a reasonable solution, but have some practical disadvantages. First, the voters are not able to verify the hashes because they do not have access to the compiled code running on the server. Hence, the voters must trust the election official and the election observers. Second, in an remote e-voting system most voters are not capable of verifying the integrity of the voting client.

5.1.3 Transparency builds trust

Making the source code for e-voting systems publicly available, and inviting voters, candidates, and researchers to inspect the code, is first and foremost an important means for building trust. Kitcat [75] argues that disclosing source code does not solve (what he calls) "the fundamental challenges that e-voting presents". Adida [77] argues that verification of election integrity is more important than disclosure of source code. If the software vendor can prove that the election result is correct, then it is subordinate if they used proprietary software to calculate the result.

5.2 Code voting

So far, this thesis has mainly focused on voters' trust in election officials and the e-voting system. Voting in an uncontrolled environment (via personal computers at home) brings in another important trust aspect—can the voters trust their own computer as a voting client? Malware (like viruses and trojans) infect many computers. It is not unlikely that some adversaries create malware that can affect the voting process. This unwanted program snippet does not disturb the voter, but when the voter casts his vote the malware changes the content of the ballot into something else. The voter gets no indications that something is wrong, but the vote recoded on the server is not in accordance with the voter's intention. At worst, such malware can change the outcome of an election. Furthermore, key loggers, or malware capturing the voters' computer screen, may infringe privacy.

A technique called *code voting* (or pre-encrypted ballots), is a countermeasure against unsecure voting clients [78, 79]. Ahead of the election, voting cards are distributed to the voters. On the voting cards, each voting option (i.e. each political party or candidate) is assigned a random code (see Figure 5.1). All voters will receive a set of personal codes. For example, the code corresponding to "Party A" will be different for two different voters.

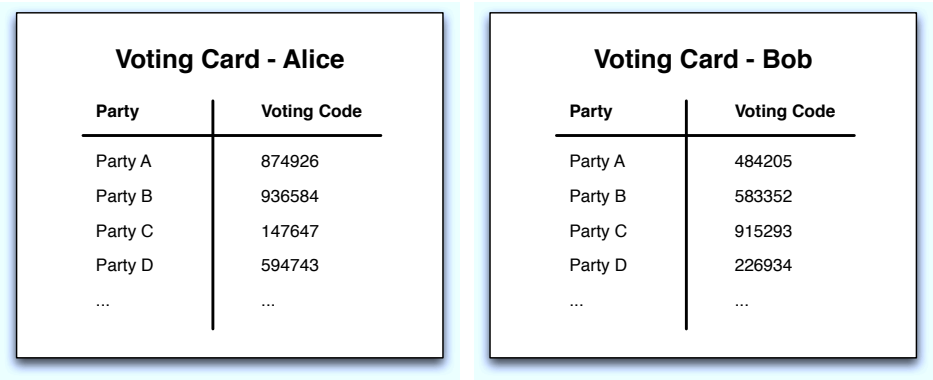

Figure 5.1: *Example of voting cards.*

In order to cast a vote, the voter submits a code from the voting card corresponding to the party or the candidate the voter wants to vote for. An eavesdropper will not be able to determine the voter's choice, unless the eavesdropper has a copy of the voter's voting card.

5.2.1 Verification codes

Even if an adversary is unable to determine the content of a voter's ballot, the adversary still has the opportunity (via malware or by acting as a man-in-the middle) to delete the vote before it reaches the election server. A countermeasure against this kind of DoS attack is to extend the code voting protocol to also include a set of verification codes. Once the election server receives a vote, the server sends a verification code back to the voter. The voter checks that the verification code he received from the election server matches the verification code printed on his voting card. The reception of a wrong verification code indicates that the vote has been changed during transfer to the election server. If no verification code is received, the voter can assume that the vote did not reach the election server at all.

In [80], Ansper et al. discuss different approaches for delivering verification codes to the voters. They suggest using an out-of-band channel like mobile phones and SMSs for sending out verification codes. Studies have shown that SMS is not a perfectly reliable channel [81], and SMS is vulnerable

to a wide range of different attacks like spoofing attacks, and DoS-attacks [82]. Appendix B presents an evaluation of *Encap*, a system developed for transmitting codes (for example one-time passwords) to a user's mobile phone in a secure way. The evaluation was published in [83].

5.2.2 Code generating and distribution

Generation and distribution of voting cards is one of the critical parts of the code voting protocol. First, the codes must have a sufficient length to ensure that an adversary is unlikely to guess the correct code for a party. Oppliger [79] concludes that a four decimal digit code is enough to both make the code hard to guess, and add some redundancy (i.e. a checksum) to the code in order to detect errors. Second, the voting cards must be generated in a "secure environment". It must be infeasible for an adversary to reconstruct a voter's voting codes, or obtain a copy of the voters' voting cards. Finally, it is recommended that voting cards are distributed to the voters via a trusted "out-of-band" channel, for example via postal mail. If the set of voting codes is distributed directly to the voters' computers, malicious software can obtain the codes, and both determine and change the content of the voters' votes before they are sent to the election server.

Oppliger [79] suggests that a keyed one-way hash function, also known as a message authentication code (MAC), can be used to generate the voting codes. Inputs to the hash function are the string M (a concatenation of the reference number for the vote and the voting card's ID), and the candidate (or party) C_i. For example, the voting code VC_i for candidate C_i, is computed by the following hash, truncated to 10 bits,

$$VC_i = (MAC_k(M \parallel C_i))_{10},$$

where $\parallel$ denotes concatenation. The encryption key k for the MAC function is kept secret (for example inside a tamper-resistant hardware module), such that only the election server has access to the key.

5.2.3 Counting code votes

How the votes are tallied in a code voting system depends much upon on how the codes are generated. In order to count the votes from Oppliger's protocol, the tabulation server re-computes the possible voting codes for each voter and compares the voter's values with the code in the received vote. Due to this scheme, protecting the correlation between the voters and their voting cards is tremendously important. Malicious election officials may compromise the

voters' privacy if they manage to trace the voting card reference back to the voter, and at the same time get access to the encryption key.

5.2.4 Analysis of proposed verification code protocol for the Norwegian e-voting project

Unsecure voting clients is a major concern for the Norwegian E-vote 2011 project. A code voting based solution was discarded early in the acquiring process due to bad usability. However, all tenderers were urged to implement solutions that included some sort of verification codes. A "verification-only" solution gives the voters the opportunity to verify that their votes have reached the election server unaltered. Thus, malware capturing the voters' computer screens can only compromise the voters' privacy.

In this section we will discuss some aspects of a verification code scheme, proposed by the Estonian company Cybernetica AS in collaboration with the Norwegian company Computas AS [84].[2]

The voting protocol takes place between the voter, and three servers: the vote collector server (VC), the messenger server (MS), and the tabulation server (TS). Ahead of the election, each voter receives a voting card with a set of verification codes. In addition, two pairs of asymmetric cryptographic keys are established: one for MS and one for TS. The protocol assumes that all voters are members of an existing Public Key Infrastructure (PKI), and are able to encrypt and digitally sign their ballots. The main steps of the protocol are summarized in Figure 5.2. Here follows a more detailed explanation:

1. The voter encrypts two copies of his vote, one encrypted with the MS's public key and one encrypted with the TS's public key.

2. The voter generates a non-interactive zero-knowledge (NIZK) proof, proving that the two encrypted votes correspond to the same candidate, and that the encrypted votes correspond to a valid candidate.

3. Finally, the voter digitally signs the encrypted votes and the NIZK proof, and sends the signed votes and the NIZK proof to the VC.

4. When the VC receives the votes from the voter, it verifies the signatures and the NIZK proof. If the verification succeeds, the VC stores the vote encrypted with the TS's public key, and initiates a proxy oblivious transfer (POT) protocol with the MS

[2]The tabulation process is not described in [84] and will not be considered here.

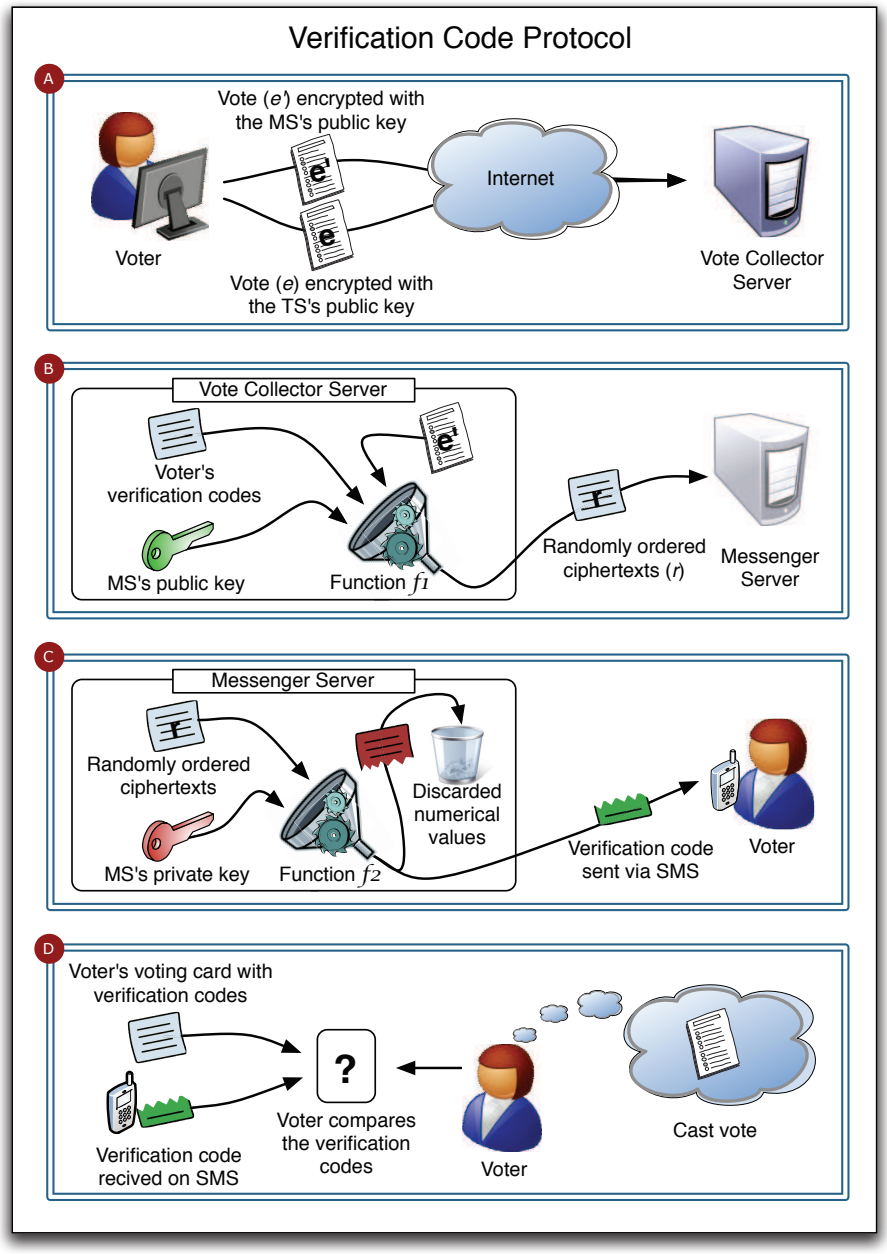

Figure 5.2: *Overview of proposed verification code protocol. In step B the VC uses the voter's verification codes, the MS's public key, and the vote encrypted with the MS's public key as input to the function f_1 to compute a list of randomly ordered ciptetexts r. Furthermore, the VC sends r to the MS. In step C the MS uses his private key and r as input to the function f_2 to compute the correct verification code for the vote, which is sent to the voter's mobile phone.*

5. The purpose of using POT is to make the MS able to calculate the correct verification code, and send it to the voter, without decrypting the voter's ballot. The VC knows the set of verification codes that is given to each voter but is unable to determine which code to return to the voter, because the VC cannot decrypt the votes.

   In the first step of POT, the VC computes a list of randomly ordered ciphertexts r. The function f_1, which is used to generate r, takes the received vote encrypted with the MS's public key, the voter's set of verification codes, and the MS's public key as input. The VC sends r to the MS. See [84] for further details regarding the POT protocol.

6. The MS uses r and the MS's private key as input to the function f_2. The output from f_2 is a list of numerical values. One of the numerical values in the output differs from the rest by being very small. This smallest value is the verification code for the vote. The MS sends the verification code to the voter, e.g., via an out-of-band channel like SMS. Note that the MS does not receive neither the encrypted votes, nor information that makes it possible for the MS to determine which verification code that belongs to which candidate.

7. Finally the VC signs all received votes encrypted with the TS' public key, and sends them to the TS for tabulation.

This protocol seems to be robust against threats (for example insiders) with access to only a limited part of the system, like one of the central servers. On the other hand, if a threat is able to take control over a number of important system components, several attacks become possible. For example, if a threat controls both the VC and the MS (i.e. the threat has access to the encrypted votes and the MS's private key), or a threat in control of the MS's private key is able to eavesdrop on the communication between the voters and the VC, the threat can decrypt the voters' ballots. Similarly, if a threat controls both the voting client and the service sending out the verification codes, he can insert a forged ballot on behalf of the voter, and discard the verification code. Because the verification code for the forged ballot never reaches the voter, the voter will not notice anything. A version of this attack is described by Gjøsteen in [85, Section 2].

The Norwegian e-vote 2011 project will use a different voting protocol with similar properties to the ones described here [85]. Instead of using a POT protocol, the Norwegian system will share the election's private key between the different election servers. The TS gets the key a_1, the VC gets a_2, and the MS gets a_3. The keys are related as $a_1 + a_2 \equiv a_3 \pmod{q}$,

where q is a big prime number. Details about how the verification codes are generated can be found in [85].

A threat with access to both the VC's and the MS's private key, will be able to reconstruct the TS's private key, and decrypt voters' ballots. To mitigate this possible attack, the Norwegian e-voting project plans to physically separate the MS and the VC. Furthermore, the two servers will be operated by different sets of individuals, and no single individual can access any of the servers alone.

5.2.5 Code voting and trust

The main advantage of an election system based on code voting is that voters can use untrusted computers to cast their votes. On the other hand, if the voters' main untrust (or distrust) is directed against election officials in charge of the election servers, code voting is not the right way to go.

5.3 End-to-end verification

In an *end-to-end verification* voting system two different properties are important: *personal verifiability* and *universal verifiability* (first introduced in Section 2.1.1). Personal verifiability is achieved when every voter can verify that his vote was recorded and counted in accordance with his intention. In an election system that supports universal verifiability, everyone can verify that all cast votes were properly counted.

A standard code voting protocol with verification codes satisfies these properties to some extent. The verification codes allows the voter to verify that the vote was recorded as intended, but he is unable to verify that the vote was included in the tally. Universal verifiability is not offered in any of the code voting protocols described above.

A lot of research on end-to-end verifiable e-voting systems have been done during the past decade, and several voting protocols and voting systems have been proposed (see for example [86, 87, 88, 89, 90]).

Common for all these different voting protocols is that every voter gets some sort of receipt after he has cast his vote. When the election is closed and the tabulation phase over, the election officials publish information about the tabulation process. First, information proving that all cast ballots have been properly counted is published. Second, some information about each counted ballot is published. By publishing information about each ballot, the voters can use their receipt to verify that their ballot is included in the tabulation.

The main problem with end-to-end verification is how to protect against coercion and vote selling. If a voter can use his receipt to verify that his vote was included in the tally, so can the coercer. For this reason, Adida concludes that Helios, an e-voting system offering end-to-end verification, is suitable only for "low-coercion elections" like student governments, local clubs, and online groups [89].

5.3.1 Desired properties for ballot receipts

First of all, the receipt should give the voter confidence that his vote was counted. Besides cryptography, usability must be carefully considered when designing the receipts in order to fulfill this property. If the procedure for checking the receipts is badly designed, voters may become uncertain whether or not their votes were counted. Second, the voter should not be able to use the receipt to prove which party he voted for. Third, voters should not be able to generate fake receipts. If it is possible for a voter to present a fake receipt, and then claim that his vote was not included in the tally, he may be able to cast doubt on the entire election result.

5.3.2 Novel receipt protocol

Recently, Raddum [91] has proposed a novel end-to-end voting protocol. The protocol's main goal is to fulfill the properties for ballot receipts listed in the previous section, i.e., offer a high degree of personal verifiability and universal verifiability. The protocol uses the ElGamal public key cryptosystem [92] to encrypt the ballots.

In ElGamal, a multiplicative group G of prime order q generated by g is constructed. A secret key $x \in \mathbf{Z}_q^*$ is chosen at random. The public key $k = g^x$ is published together with g and q. To encrypt a plaintext m into the ciphertext (a, b), a random parameter r is chosen. Encryption is done by calculating $(a, b) = (mk^r, g^r)$. Decryption is done by calculating $m = ab^{-x}$.

One of the properties of ElGamal is that a given ciphertext $(a, b) = (mk^r, g^r)$ can be re-encrypted into (a', b') by generating $(a', b') = (ak^s, bg^s) = (mk^{r+s}, g^{r+s})$. Decryption can still be done in one step, and anyone knowing s can prove that (a', b') is a re-encryption of (a, b) by running a zero-knowledge proof.

Raddum's protocol takes place between the voter, a vote collector server (VC), and a tabulation server (TS). The protocol assumes that all voters are members of an existing PKI, and are able to encrypt and digitally sign their ballots. The main steps of the protocol are summarized below. The receipt generation and the tabulation phases is depicted in Figure 5.3.

1. The voter, with voter ID V_{ID}, encrypts his ballot b by using ElGamal and the election's public key (p, g, k) :

$$E_k(b) = (bk^r, g^r)$$

2. The voter generates a timestamp t, or preferably, receives a timestamp from a trusted server.

3. Finally, the voter cast his vote by sending the encrypted ballot, the timestamp, his voter ID, and a signature of the encrypted ballot and the timestamp, to the VC: $E_k(b), t, V_{ID}, Sig_{ID}(E_k(b) \parallel t)$. Voters are allowed to cast multiple votes by repeating these three steps, but only the last vote cast will be counted.

4. The VC stores all received votes in a table T.

5. When the voting phase is over, the VC sorts and re-encrypts all received votes. During this process the last received vote form every voter is re-encrypted and stored in a table R. All other votes are re-encrypted and stored in a table S.

6. The VC publishes the set of encrypted ballots in T and the re-encrypted ballots in R and S. Voter IDs, signatures, and timestamps for the encrypted and the re-encrypted ballots are not published.

7. The re-encrypted ballots listed in R act as receipts. For each voter, the VC sends a digitally signed copy of the voter's re-encrypted ballot stored in R to the voter. E-mail or SMS can be used to distribute the receipts.

8. When the voter receives the receipt, he can check that his receipt is listed in R.

9. Finally, the re-encrypted ballots stored in R are sent to the TS. Because the VC knows which re-encrypted ballot belongs to which voter, the ballots are sent through a mix-net before they are decrypted. The mix-net removes any correlation between the voters and the ballots.

Coercion resistance and verifiability

Note that in the protocol described above, the voters do not receive their receipts before the voting phase is over. The purpose of sending a receipt to a voter, is to give the voter confidence that one of his cast votes is listed in

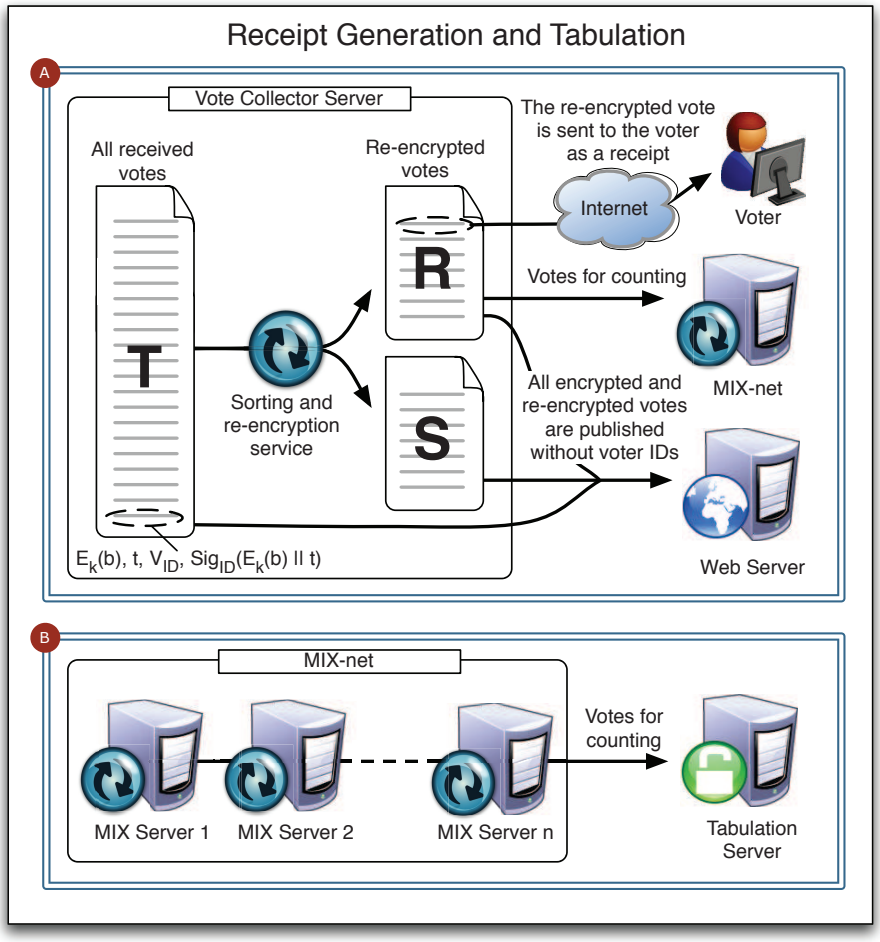

Figure 5.3: *Overview of the receipt generation and tabulation phases. All received votes, stored in list T are re-encrypted and divided into the two lists R and S. List R contains the votes that shall be counted. Each voter receives a copy of his re-encrypted vote, stored in R. The re-encrypted votes in R are sent through a Mix-net before they tabulated on the TS.*

R. A coercer cannot check whether the voter has cast a new vote after being coerced, because the receipt does not reveal any information about which vote from T VC has included in R.

However, a dishonest VC can cheat in many different ways. VC can for example put all votes from one voter in S, or put a vote that is not the most recent one in R. So how can a voter be sure that his receipt is a re-encryption of his most recently cast vote? A solution is to use a set of trusted auditors to verify that VC behaves correctly. Based on timestamps, signatures and voter IDs, the auditors can identify the votes in T that should be placed in R.

All mixing nodes in the mix-net between R and TS can produce universally verifiable proofs of correctness. Thus, if the voters trust the auditors, the remaining part of the mixing and decryption procedure can be verified and trusted. Readers interested in verifiable mix-nets are referred to [93, 94, 95].

Because the voters have to trust the auditors, Raddum's protocol is a trade-off between coercion resistance and universal verifiability. However, the protocol defines clearly what the auditors must control to verify the elections integrity. And the protocol is rather easy to explain—which is important to build trust.

Suggested improvements

As mentioned earlier in this Chapter, usability must be considered when designing a voting protocol with receipts. The main usability drawback of Raddum's protocol is that the receipts (i.e. a re-encrypted ballot represented as a hexadecimal value) are sent back to the voters via e-mail or SMS. First, e-mail and SMS are not perfectly reliable channels. It is likely that some voters will not receive their receipts. Second, the process of checking that a specific hexadecimal value is stored in R may be too complicated for many voters.

A simple improvement is to assign a reference number to each re-encrypted ballot, and send both the re-encrypted ballot and the reference number to the voter. The voter can then easily search for the reference number in R and compare the re-encrypted ballot published in R to the copy he received.

A better solution is to post the voters' IDs next to the re-encrypted ballots in R, and not send a re-encrypted copy of the voter's ballot back to the voter. A voter who wants to check that one of his cast ballots is listed in R, can search for his voter ID. Moreover, by using this solution, every voter can verify that the re-encrypted ballots stored in R is assigned with different voter IDs. The encrypted and re-encrypted ballots stored in T and S will still be published without voter IDs. Voters have to keep their voter IDs secret if they do not want to disclose whether they have voted in the election.

5.4 Does end-to-end verification increase the voters' level of trust?

Election systems offering end-to-end verification are not widely used today. In 2009 the Université catholique de Louvain elected its president using the e-voting system Helios [96], and Princeton University used Helios for their

undergrad government election in 2009. In November 2009, the voters in Takoma Park in Maryland,[3] elected mayor and city council with a system called ScantegrityII [97].[4]

The main focus in these elections has been on technical details—like proving the correctness' of the election results. Thus, more research have to be done before we can draw any conclusions about how end-to-end verifiable elections influence the voters' level of trust.

5.4.1 Complex election systems

Several verifiable voting protocols publish all decrypted ballots after the election [90]. Making the ballots publicly available seems fine for elections where the voters chooses one out of n possible options. In some elections, publishing all ballots may be impossible due to the problem of coercion and vote selling [98]. For example, in Norwegian municipality elections, a voter can add candidates form other party lists to his ballot [99]. The voter is also able to reorder the listed candidates on the ballot. If a vote buyer, or a coercer, dictates a special combination of write-in candidates and a special reordering of candidates, it will be possible identify the voter's ballot after the election. Further research has to be done in designing user-friendly end-to-end verifiable e-voting systems for these complex election systems.

[3] `http://www.scantegrity.org/takoma/`

[4] ScantegrityII is a paper-based voting system offering end-to-end verification.

Chapter 6

Summary and Conclusions

The ignorance of one voter in a democracy impairs the security of all
—John F. Kennedy

6.1 Summary

A necessary requirement for free and fair elections is that voters, candidates, and election officials trust the e-voting system in use. As pointed out in Chapter 1, trust is situational and varies over time. Building and preserving trust in an e-voting system must therefore be a continually ongoing process. Roughly, this process can be divided into two phases. Phase one is building trust in the e-voting system before it is deployed, and phase two consists of building (or preserving) trust in the system (and the election result) during and after an election. In this thesis, we have studied both phases in depth.

Chapter 2 focused on the importance of stating high-level, non-functional security and privacy requirements during the first phase. The purpose of these non-functional requirements is twofold. First, emphasizing security and privacy requirements especially relevant for e-voting systems is important for building trust in a system. We argued that it is crucial to achieve public acceptance of the selected non-functional requirements early in the development process. If this is not possible, it is more likely that the process of developing an e-voting system will fail at a later stage. Second, stating well-defined requirements makes it easier for stakeholders to discuss and develop low-level, technical requirements for a particular implementation of an e-voting system.

Furthermore, Chapter 2 also presented a technique for analyzing an e-voting system based on the selected security and privacy requirements. Chapter 3 then applied this analysis technique to the e-voting system used during the 2009 rectorial election at the University of Bergen (UiB). The results of the analysis clearly shows why it is important to analyze an e-voting system *before* it is deployed.

Both Chapters 3 and 4, describing how UiB's election board handled the situation when an entire election system was put into doubt, illustrate the importance of maintaining trust during phase two. Suspicion of an improper election result is likely to decrease the voters' level of trust. Chapter 4 discussed the importance of gathering necessary information to support a forensic analysis of an election. It is also important that election officials and system operators are capable of detecting fraud and irregularities themselves. Different techniques for detecting indications of fraud and other irregularities in an e-voting system were also explored in Chapter 4.

Using proprietary software with undisclosed source code to cast and count votes is a controversial topic. Chapter 5 discussed the importance of disclosing source code for an e-voting system to build trust.

Usually, a voter's role in an election is completed once the ballot is cast. Chapter 5 presented some e-voting protocols where the voters can verify that their ballots are included in the tabulation phase. Especially, a novel coercion resistant voting protocol was described in Section 5.3.2. This new voting protocol makes it possible for voters to verify that their votes are included in the tabulation phase, and election auditors can verify the correctness of the election result. Finally, some usability improvements to the protocol were suggested.

6.2 Conclusions

While it is impossible to build a 100 % secure computer system [5], it is possible to build a system that can be highly trusted by most stakeholders. The most important discovery made while working on this thesis is that building trust in an e-voting system is not only a matter of building a good e-voting system with a well designed user interface and reliable voting protocols—it is first and foremost a matter of providing trustworthy information to all stakeholders about the election system and its behavior before, during, and after an election.

Providing trustworthy information is especially important if irregularities or abnormal situations occur. Information must be gathered in order to make it possible to explain why a problem occurred, and determine the dimension

of the problem. Because an election is an irreproducible event, such information must be gathered throughout the entire election phase. Starting to gather the information needed for an investigation after the fact is usually too late.

6.3 Further Work

We believe that voters are likely to have a high degree of trust in an election result if they are involved in the later phases of the election, as in the case for the protocols described in Chapter 5. So far these voting protocols are not widely used. It is therefore difficult to determine the effect these voting protocols will have on voters' level of trust in different types of elections. A topic for further research could therefore be to explore this assumption. Furthermore, researches have until now mainly focused on the technical details of these protocols. Improving the usability of end-to-end verifiable e-voting protocols has yet to be done.

Researchers working with e-voting systems that facilitates voting in an uncontrolled environment should try to develop new countermeasures against coercion and vote selling. The ability to re-vote electronically or re-vote on paper, such as in the Norwegian and the Estonian e-voting systems, are not suitable for all elections. This countermeasure may also weaken over time, especially if the popularity of e-voting cause a reduction of polling stations where voters can cast their paper ballots.

Finally, the E-vote 2011 project in Norway may potentially give researchers an unique opportunity to study different aspects of e-voting in an uncontrolled environment. According to the project's work schedule, three non-binding Internet elections are about to be held in three different municipalities in 2010.[1] Here, it would be interesting to study how voters and election officials behave if they are exposed to attacks during an election. Based on results from such research, election officials could be trained to better handle real attacks and irregularities in the system.

[1] `http://www.regjeringen.no/upload/KRD/Prosjekter/e-valg/Tidslinje_evalg.pdf`

Appendix A

Evidence Gathering in an E-voting System

An electronic election is an irreproducible event [54]. To support audits and forensics, evidence must be gathered before, during, and after the election. This appendix suggests which data to collect for remote e-voting systems. The list of data was motivated by Ernst & Young's investigation of the rectorial election at UiB and the overviews in [6] and [51].

A.1 Data to collect before the election

System description

Any e-voting system should have a detailed system description. This will make it possible for auditors to verify that the e-voting system in use complies with its description.

Record of all e-voting personnel

One should maintain a record of all individuals with access to any part of the e-voting system. This includes both individuals with physical access to hardware, and individuals with access to software and databases. The record must keep track of when and where each individual had access to the system, which parts of the system the individual had access to and why such access was given.

Logging of system activity

Logs recording system activity, before, during, and after the election must be established.

Physical security

How election equipments are physically secured, and how policies for access control are being enforced, should be documented.

Inventory of all hardware

All hardware equipments included in the e-voting system should be described with name of the vendor, serial number, time and date of purchase, and source of delivery.

Inventory of all software

All election specific software, as well as operating systems and other software installed on the election systems' hardware, should be specified. The inventory should be as detailed as possible, specifying vendors, build numbers, and version numbers.

Configuration settings

When new hardware and software are installed, all initial configuration settings should be recorded. All succeeding changes in configuration settings should be documented and explained.

Hardware and software updates

Log all updates, replacements, and changes to hardware equipment and software after the point of installation. This applies to both election specific software, as well as for operating systems and other software installed on the hardware in use. One should also document which individuals deployed the update, and which individuals requested the update. Preferably, the motivation for all updates should be explained.

Test results

All parts of the election system should be tested for accuracy and usability before the election. Every test should be documented. The documentation

should include a description of which parts of the system that are tested, the type of tests carried out, a description of test data used, and problems observed during testing.

Training sessions

It is recommended that all training of election personnel is logged and documented. If a problem classified as misfeasance occurs during the election, it may be useful to check if improper training caused the problem.

Changes to the electoral roll

Tampering with the electoral roll is a potential source for election fraud. For example, by adding fake identities to the electoral roll, an adversary may vote multiple times. Thus, all changes to the electoral roll (before and during the election) should be logged. The log must contain information about the individuals who requested and performed the change, as well as documentation justifying the change.

Generating encryption keys

Most e-voting protocols use some kind of encryption scheme to encrypt the votes. Routines for generating encryption (and decryption) keys, and the actual process of generating these keys, should be documented.

A.2 Data to collect during the election

Opening and closing of the voting phase

Necessary information to verify that the voting phase was opened and closed in accordance with predefined routines should be collected.

Voter behavior

Different statistics about voters' behavior during the election may be collected. However, it is extremely important that the collected information do not violate the voters' privacy. It is natural record the time and date of each cast vote. It is also a good idea to log the number of votes cast from the same computer.

An overview showing the number of voters who voted multiple times can be created, if the election systems allow voters to cast multiple votes. On

the other hand, publicly disclosing *which voters* voted multiple times, is not recommended. A coercer can use such information to verify that voters voted in accordance with his intensions.

System performance

All hardware and software failures during the election must be documented with exact time and date, and preferably a description of the system's state. Especially, undocumented system behavior or error messages should be recorded, even if they occur only once. All unexpected decreases in system performance during the election, as well as system components acting unresponsive, must be documented. Preferably, election systems should not be patched during the voting phase. However, deployment of system changes, in order to correct any problems, should be well documented.

A.3 Data to collect after the election

Tabulation procedure

The tabulation process is a critical phase in all elections. Detailed information must be recorded to make auditors able to verify that the tabulation process was conducted in accordance with predefined procedures.

Some voting protocols produce mathematical proofs that can be used to verify the integrity of the tabulation process. Examples of such protocols are presented in Chapter 5.

Failures during the tabulation

Discrepancy in vote totals, or failures in equipment used for counting the votes should be documented.

Election results

Overviews of election results should distinguish between early votes, absentee votes, and votes cast on election day. An overview should also provide detailed statistics about ballot corrections. In other words, all numbers used to calculate the final result must be made available.

Invalid ballots

A well-designed user interface in the voting client should manage to prevent voters from casting invalid ballots. However, invalid ballots may occur due to technical problems, or hacking attempts, and must be documented.

Test results

It is recommended that the election system is tested for accuracy after the election. Documentation of post-election tests must be made in accordance with pre-election tests in order to simplify comparison.

Appendix B

Security Analysis of Mobile Phones Used as OTP Generators

The Norwegian company Encap has developed protocols enabling individuals to use their mobile phones as one-time password (OTP) generators. The following paper presents an analysis of three of Encap's protocols, and a system-level test of an online bank utilizing Encap's solution. The paper suggests some countermeasures to thwart the vulnerabilities discovered in the analyses.

A short-version of this paper was presented at the Workshop in Information Security Theory and Practices (WISTP 10), Passau, Germany, April 12-14 [83].

Security Analysis of Mobile Phones Used as OTP Generators

Håvard Raddum, *Lars Hopland Nestås*, and *Kjell Jørgen Hole*
Department of Informatics
University of Bergen
(Havard.Raddum@ii.uib.no, lma029@student.uib.no, Kjell.Hole@ii.uib.no)

Abstract

The Norwegian company Encap has developed protocols enabling individuals to use their mobile phones as one-time password (OTP) generators. An initial analysis of the protocols reveals minor security flaws. System-level testing of an online bank utilizing Encap's solution then shows that several attacks allow a malicious individual to turn his own mobile phone into an OTP generator for another individual's bank account. Some of the suggested countermeasures to thwart the attacks are already incorporated in an updated version of the online banking system.

1 Introduction

There are many services on the Internet needing strong user authentication. Examples are online banks and e-government services, in particular public health services containing sensitive medical information. User authentication is often achieved utilizing a two-factor authentication technique based on something the user knows, i.e. a static password, and something the user has, i.e. a one-time password (OTP) generator or a list of OTPs. The static password is usually typed into a login page or used to turn on a hardware-based OTP generator.

The Norwegian company *Encap* has developed a system enabling an individual to use his mobile phone as an OTP generator when authenticating to a web-based service. The phone runs a Java MIDlet, which communicates with a server to generate OTPs. This paper describes the three main protocols utilized by Encap's system in 2009 and analyze their security.

Perhaps because the protocols were analyzed earlier [1], the authors' initial analysis only revealed two minor flaws in the protocol designs. We found that a cryptographic key generation should be upgraded in accordance with "best practice," and a few steps in the protocol specifications could be simplified without losing any security benefits.

Early in 2009, an online bank deployed Encap's solution as part of their customer authentication. System-level testing revealed that malicious software, or *malware*, on a customer's PC can steal sensitive information when a customer activates his mobile phone as an OTP generator. An attacker can then use this information to turn his own mobile phone into an OTP generator for the customer's account.

Further testing revealed that a modified version of the malware attack can be directed against *all* customers of the online bank, including those who do not wish to use their mobile phones as OTP generators. We also found that a similar attack can be instigated using social engineering and a malicious proxy. Because the discussed attacks allow an attacker to use his own mobile phone to generate OTPs for a customer's account, the attacker can access the account whenever he wants until it is closed by the bank. We present countermeasures to thwart the described attacks.

Current authentication solutions do not protect against attacks modifying transactions or injecting false transactions into an established session between a client and a server [2, 3]. We outline how a possible new control feature can be added to Encap's solution to stop these attacks.

The rest of the paper is organized as follows. Section 2 presents the protocols, Section 3 analyses the protocol designs, Section 4 describes attacks on an online bank utilizing the protocols, Section

5 presents the new control feature, and Section 6 concludes the paper.

2 Protocols enabling phones to generate OTPs

This section describes the three main protocols used by Encap's system during 2009. Initially, the user runs a protocol to download the Encap client (Java MIDlet) to his mobile phone. Then, the Encap client executes a protocol to register with both Encap's server and a service provider utilizing Encap's system for user authentication. After successful execution of the download and activation protocols, the user can run the authentication protocol an unlimited number of times.

2.1 General assumptions

Before describing the individual protocols, we make a few general assumptions. A protocol is aborted if a protocol step fails, e.g. to verify data. All communication channels between the parties of the protocols are protected with SSL/TLS, except for the communications between the user and his PC and mobile phone, as well as an SMS starting the Encap client on the phone. An Encap white paper [1] discusses the secure deletion of secret information on a phone after it has displayed a new OTP. We assume that both the secure deletion and the Java platform on the phone work as intended.

2.2 The download protocol

The first step a user takes to turn his mobile phone into an OTP generator is to download the Java MIDlet to the phone. This process is specified in a download protocol, although in practice this protocol seems to be embedded in the activation protocol (see next section). For clarity, we describe the download protocol separately.

The download protocol takes place between four parties: the *user*, the user's *mobile phone*, the user's *PC*, and the *Encap server* (ES). The complete protocol is depicted in Figure 1. In the following, we describe the main steps of the protocol.

1. The user enters the number of his mobile phone in a web page on the PC, and sends a request to ES to download the client software to the phone.
2. ES connects to the phone, asking for a *user agent* describing the phone's capabilities.
3. When ES receives the user agent from the mobile phone, it responds with a Java Application Descriptor (JAD) file and a URL to download the client software appropriate for the particular phone.
4. The phone downloads the MIDlet and lets the user install it.

There are very few security considerations for the download protocol. If a user is allowed to download Encap's client without authenticating to the download server, then an attacker can more easily fool the user into installing a rogue client on his phone. The rogue client can behave exactly like the real client, except that it covertly sends cryptographic keys or other secret information back to a server controlled by the attacker. At the time of writing, the online bank utilizing Encap's solution authenticates all users before they can download the client. However, the security risk will increase if it becomes possible to download the client from many service providers without any form of authentication.

2.3 The activation protocol

After the user has downloaded the client software onto his mobile phone, he must activate the phone as an OTP generator before it can be used for authentication to a web-based service. The activation protocol takes place between five parties: the *user*, the user's *mobile phone*, the user's *PC*, the *ES*,

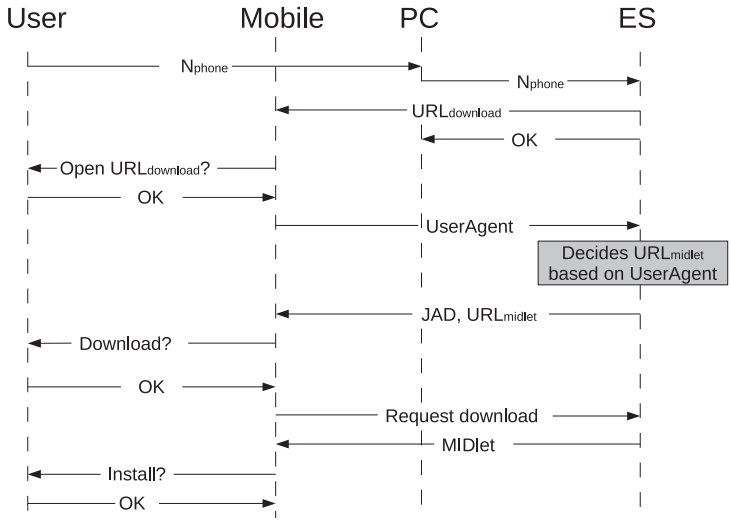

Figure 1: The download protocol.

and the *service provider* (SP). Figure 2 shows the complete activation protocol. We remark that the depicted session ID, ID_s, is not considered in this paper, but included to provide a complete presentation of the protocol. The main steps of the protocol are summarized below.

1. The user authenticates himself to SP using credentials already known to SP. (The authentication procedure may involve a 'traditional' hardware-based OTP generator.)

2. When the user asks to activate his mobile phone as an OTP generator, SP redirects the user's browser to ES with a URL that contains an activation request and a *Secure Object*.[1]

3. ES verifies that the Secure Object comes from SP, and gets the user's phone number.

4. ES sends an *activation code* to the user's PC and an SMS message to the user's phone asking it to start the client software.

5. The mobile phone asks the user to enter the activation code, available on his PC, and transmits the code to ES.

6. ES verifies that the activation code is the same as the one sent to the PC, and sends a challenge to the mobile phone together with an encryption key K_0. (The role of K_0 is explained in Section 2.5.)

7. The user chooses a personal identification number (PIN) and enters it on the mobile phone, which generates a *security code* and a *response*. The response is the encryption of the challenge using the security code as key. The security code and response are sent to ES, and ES stores the security code.

[1]The exact content of the Secure Object in the URL redirecting the user's PC to ES is not known to us, but we assume it contains information enabling strong authentication of the SP to ES, and we know it contains information to identify the user.

8. ES verifies that the response and the security code correspond to the challenge, and if so, the user has activated the mobile phone as an OTP generator for use with SP.

The activation protocol's main goal is to ensure that only the legitimate user's mobile phone is activated as the OTP generator for the SP. The protocol contains several steps to achieve this goal. First, the user must authenticate himself to the SP using an already trusted authentication mechanism. Second, the Secure Object authenticates the SP to ES. Third, the activation code sent by ES to the user's PC is sent back to ES from the user's mobile phone.

These steps should ensure that the PC and the mobile phone are in the same location, or at least that there exists a communication link between the person using the PC and the holder of the phone. Since the person using the PC is authenticated and has transferred the activation code to the phone, we can assume that this person really wants to activate the mobile phone as an OTP generator.

2.4 The authentication protocol

The OTP-based authentication protocol takes place between five parties: the *user*, the user's *mobile phone*, the user's *PC*, the *ES*, and *SP*. The complete authentication protocol is depicted in Figure 3. The main steps of the protocol are described below.

1. The user enters the identity he shares with SP on its login page.
2. SP asks the user for an OTP, and sends a request to ES to generate an OTP for the user.
3. ES first sends an SMS to the user's mobile phone to start the client software. It then sends a challenge to the phone together with two encryption keys K_i and K_{i+1}, whose role will be explained in Section 2.5.
4. The user enters his PIN on the phone, and the phone computes the same security code generated at the time of activation. The phone then encrypts the challenge with the security code as key and sends the ciphertext as a response to ES.
5. ES verifies that the response from the mobile phone corresponds to the challenge, and sends an OTP to the phone.
6. The user enters the OTP on the SP's login page, and SP contacts ES to verify that the OTP is indeed the correct one for this user.

The authentication protocol's main goal is to ensure that only the legitimate user can obtain an OTP from ES. The goal is achieved mainly because the phone's response to ES' challenge is the encryption of the challenge using the key (security code) made during activation. The correct generation of this key requires the correct PIN, which only the person who activated the mobile phone is supposed to know. This person was in turn authenticated at the time of activation, hence we can be confident that he is the legitimate user.

2.5 Generation of security code and responses

The hash function SHA-1 and the encryption algorithm AES with a 16-byte key are used to generate the security code and the responses. Hashing is denoted by $H(\cdot)$ and encryption with key K is denoted by $E_K(\cdot)$.

The security code, SC, is computed by the following hash, truncated to 16 bytes,

$$SC = H(PIN||IMEI||CR||SPID)_{16}\,, \tag{1}$$

where $||$ denotes concatenation of the following strings:

- PIN is a secret number with at least four digits entered by the user.

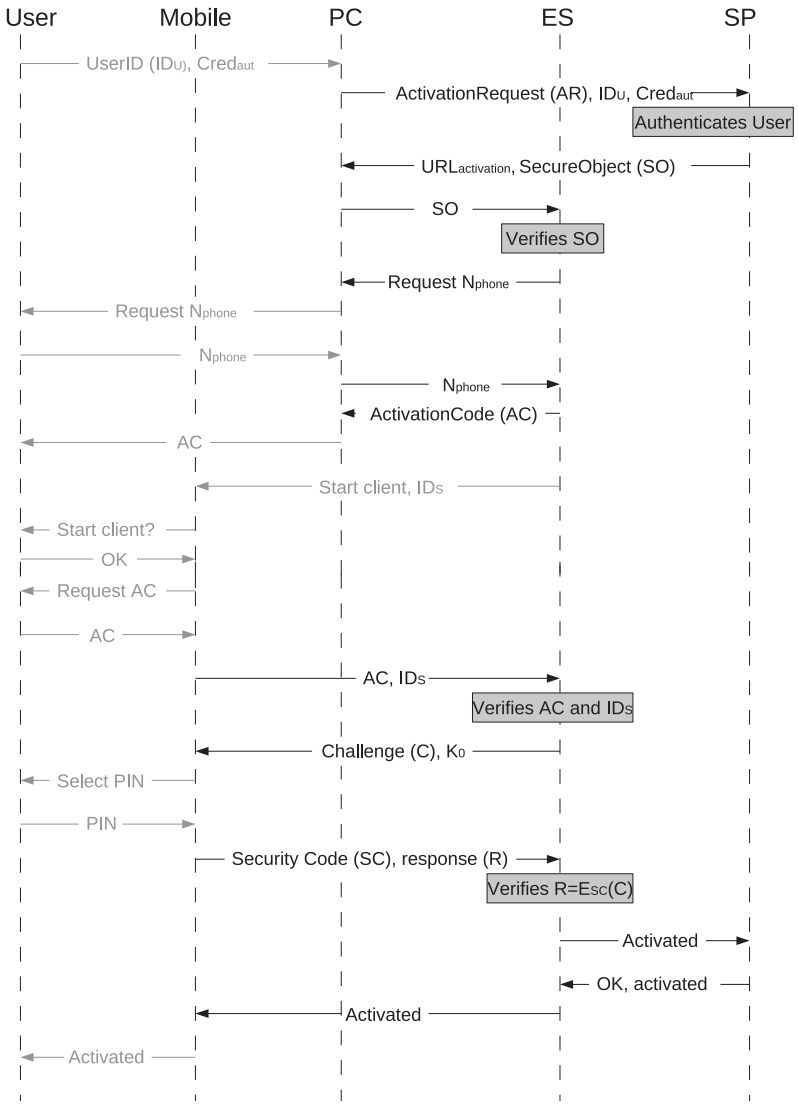

Figure 2: Activation protocol. Black channels are protected by SSL/TLS, gray channels are not.

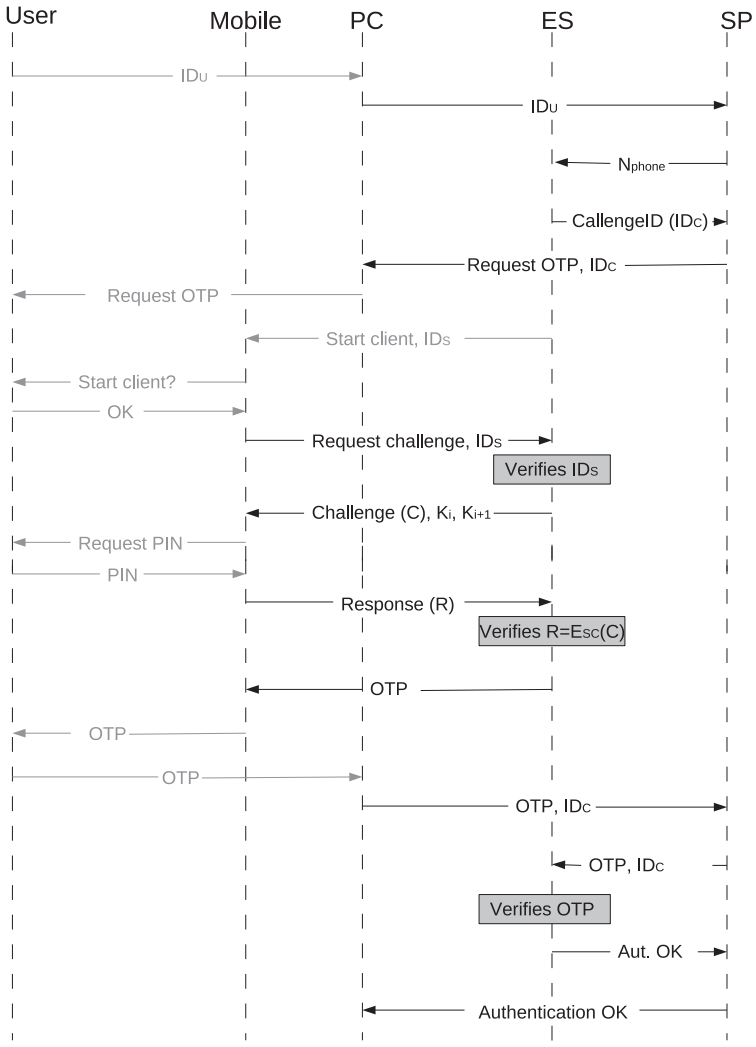

Figure 3: OTP-based authentication protocol. Black channels are protected by SSL/TLS, gray channels are not.

- $IMEI$ is a 14 digit code uniquely identifying the mobile phone where the Encap client is installed.
- CR, or *client reference*, is a 40-byte random string generated on the mobile phone during the activation protocol.
- $SPID$ is a public value identifying the SP to whom the user wishes to authenticate.

The client reference (CR) needs to be stored on the mobile phone for later use. It is only stored in encrypted form. During the activation protocol it is encrypted using AES with the key K_0 which is sent by ES together with the challenge.

When the client reference is needed in the authentication protocol it is first decrypted using K_i, and when it goes back into storage it is encrypted using K_{i+1}. The keys K_i and K_{i+1} are sent from ES together with the challenge in the authentication protocol.

The generation of a 16-byte response, R, to a challenge, C, is defined by the expression

$$R = E_{SC}(C)\,,$$

where SC is the 16-byte security code defined by (1) and C is a 16-byte challenge received from ES.

3 Possible improvements to the protocol designs

Our analysis of the protocol designs suggested two minor changes. The generation of the security code should be upgraded in accordance with "best practice," and a few steps in the protocol specifications could be simplified without losing any security benefits.

3.1 Improved security code generation

The security code defined by (1) is generated in the activation protocol. The code is used as a shared secret key between the ES and the client on the mobile phone. Currently, the security code is generated by the client alone and then transferred to ES. When generating a shared secret key between two parties, it is preferable that no single party can control the value of the key. Instead, it is suggested to use an established key exchange protocol, for instance Diffie-Hellman key exchange [4].

The following changes are needed to implement Diffie-Hellman in the existing activation protocol: The ES first picks a random value a, and when ES sends the challenge it includes $g^a \bmod p$, g, and p, where p is a large prime and g is a generator of a large subgroup of Z_p^*. Next, the client computes the SHA-1 hash as before, but the hash digest is treated as a "random" number b by the client (instead of a security code). The client then computes the security code as $g^{a*b} \bmod p$, and sends the response together with $g^b \bmod p$ to ES. Finally, the ES computes the security code as $g^{b*a} \bmod p$ and verifies the response. The ES must store the value a together with the security code in order to allow the client to generate the exact same security code in future executions of the authentication protocol.

3.2 Encryption of client reference

The expression (1) for the security code contains a random 40-byte client reference (denoted CR). The purpose of the client reference seems to be to increase the entropy of the input to the hash function in (1). The client reference needs to be stored on the mobile phone for future use. It is specified that the client reference should only be stored in encrypted form, with keys to encrypt and decrypt supplied by ES.

At first there seems to be some added protection from this encryption: An attacker who gets hold of a user's mobile phone can determine the IMEI number of the phone and read the memory where the client reference is stored. The SPID is publicly known. If the client reference was stored in cleartext, then the attacker could record these values, and exhaustively try all different PINs to

generate the set of possible security codes. Determining the correct security code would then be no harder than guessing the user's PIN. However, this approach is not available to the attacker since the client reference is encrypted before it is stored. Thus, the attacker needs the decryption key before being able to generate the (relatively small) set of possible security codes.

Unfortunately, the ES supplies the needed decryption key *before* any authentication takes place. If an attacker gets hold of a user's mobile phone and is able to read the encrypted client reference, all he needs to do is to follow the authentication protocol to get the decryption key from ES. Hence, the encryption of the client reference does not add to the security of the scheme.

According to Encap, another reason for introducing pairs of keys K_i, K_{i+1} to repeatedly decrypt and re-encrypt the client reference is to ensure that no two phones can obtain the same sequence of OTPs from the Encap server. If an attacker can guess the PIN and copy the IMEI, client reference, and SPID from a legitimate user's phone to his own phone, then the attacker can try to obtain the same sequence of OTPs as the legitimate user.

If two phones encrypt the same client reference with the same key K_{i+1}, then when they later ask the Encap server for K_{i+1} to decrypt the client reference, as well as K_{i+2} to re-encrypt the client reference, only one of the phones will receive K_{i+1} and K_{i+2}. (The other phone will receive K_{i+2} and K_{i+3}.) Hence, only one of the phones will be able complete the authentication protocol because decryption fails in the other. However, there is no need to apply AES to achieve this goal. Since the keys K_i are random bit strings, they could simply be XOR-ed with the client reference. The use of XOR instead of AES simplifies the activation and authentication protocols.

4 Attacks on the phone activation in an online bank

While we can continue to study the the protocol designs to determine attacks, system-level analysis and testing, preferably on a real-world implementation, make it easier to ascertain the practicality and impact of suggested attacks. This is particularly true when it is possible to develop proof of concepts. Early in 2009, an online bank deployed Encap's solution as part of their customer authentication procedure. One of the authors opened an account with the bank and used his own PC together with browser-based tools to study authentication related messages. (No attempt was made to penetrate the bank's central infrastructure.)

We discuss malware on PCs in general, before describing how tailored malware can give an attacker control over customers' bank accounts when the customers activate their mobile phones as OTP generators. It is then shown how a modified version of this attack can be directed against customers who never planned to use their mobile phones to generate OTPs. The essentially same attack can also be instigated using e-mail phishing and a Man-in-the-Middle (MitM) proxy. Finally, we introduce countermeasures to stop the attacks.

4.1 The danger of malware

If an attacker is able to introduce malware on a user's PC, then the attacker can, e.g., steal sensitive information, display bogus web pages, and redirect or spoof internet traffic. The attacker can essentially take control over the user's PC. Having a PC infected with malware is indeed a real risk [5, 6]. In particular, typical banking trojans steal usernames, passwords, and OTPs using techniques such as form grabbing, screenshots and video capturing, key logging, and traffic sniffing. The *Haxdoor.KI* trojan attacked Nordea's Swedish bank customers in 2006 and caused financial losses of at least 8 million SEK [7, 8].

4.2 Malware-based replay attack on customers activating mobile phones

The goal of the following *malware-based replay attack* is to collect a victim's username and password, and to generate the victim's OTPs on a mobile phone of the attacker's choice. An attacker can modify existing malware similar to the trojan Haxdoor.KI to carry out this attack.

First, the malware captures the victims' username and password when he logs on to his online bank. Second, when the victim starts the implemented activation protocol, the malware captures the URL containing the Secure Object. The activation procedure is not secured against replay attacks occurring inside a time window of a few minutes, nor is it tied to one particular IP address or SSL session. Consequently, the malware need not disrupt the user's activation process, but can just wait until the user has completed the activation and then transmit the URL to the attacker's PC. The attacker enters the URL, containing the Secure Object, into a browser. Finally, the attacker submits his own phone number to download, install, and activate the Encap MIDlet on his mobile phone.

This attack was tested on an online bank account belonging to one of the authors (we did not try it on other customers' accounts). We found that an attacker's activation of a mobile phone as an OTP generator automatically overrides any previous activation made by the user. The attacker then has an OTP generator that enables him to log into the user's account. Moreover, the user is not able to log in anymore since his OTP generator is no longer accepted by the Encap system.[2]

4.3 Malware attack on all customers

The malware-based replay attack can be modified to obtain a *malware-based impersonation attack* targeting *any* customer in an online bank utilizing Encap's solution—assuming that all customers have the option to activate their mobile phones as OTP generators. In this case, the malware just waits for a customer to log on to the online bank. The malware then sends a request to activate a mobile phone as an OTP generator without the customer realizing what is going on. When the URL with the Secure Object is returned, it is forwarded to the attacker's PC instead of redirecting the user's browser to ES. The attacker utilizes the URL to activate his own mobile phone as OTP generator for the user's account. This attack is deemed practical, especially since there already exist malware that steals information and manipulate client-server communication, e.g. see [9].

After an attack is completed, the malware can delete itself to make it more difficult for the bank to determine how an attacker is able to remove money from the customer's account. In fact, the customer may initially get the blame since the bank server receives the correct username, fixed password, and OTP each time the attacker logs on to the customer's account.

4.4 Phishing attack on all customers

An attack similar to the malware-based impersonation attack can be carried out without client-side malware. The SSL protocol is supposed to provide strong server authentication in client-server systems. While the cryptography in SSL is strong, poor usability still results in weak authentication in practice. Because SSL cannot thwart *phishing attacks*, i.e. combinations of social engineering and MitM attacks, customers can be tricked into connecting to proxy servers under the control of attackers [10, 11]. This can happen because customers are unable (or unwilling) to verify server-side public-key certificates used by SSL.

To initiate an attack, an attacker can generate phishing e-mails asking customers to log on to a MitM proxy masquerading as the customers' online bank. There is ample evidence showing that many individuals receiving phishing e-mails enter their login credentials at fake web sites [11].

Once a customer has connected to the MitM proxy, it forwards messages in both directions between the customer's PC and the bank's central infrastructure. The attacker has complete control of the communication because the proxy can read all messages, change their contents, and create fake messages. In particular, the proxy records the username and password transmitted by the tricked customer. The proxy can then generate a fake request to activate a mobile phone as an OTP generator and records the returned URL. The URL is used by the attacker to activate his own phone as an OTP generator for the tricked customer's account.

[2] We remark that a similar replay attack can be instigated using e-mail phishing to trick a customer into accessing a MitM proxy. A more general phishing attack is described later in this section.

4.5 Attack comparisons

The reader should note that while the malware-based replay attack can only occur when a customer activates his phone, the impersonation attacks simply require that the customer logs on to his account. We also remark that real-time MitM attacks have been used to bypass 'traditional' hardware-based OTP generators by forwarding user generated OTPs to online banks [12]. These attacks give access to an account only once. Our attacks are different because they let an attacker generate as many OTPs as he wants on his own phone. He can therefore access an account *whenever* he desires until the account is closed by the bank.

4.6 Countermeasures

We suggest three countermeasures to thwart the described attacks. To protect against the malware-based replay attack, the activation process needs to be secured against the replay of old requests. The ES must therefore ensure that each Secure Object is only used once. Also, it should not be possible to just activate another mobile phone as OTP generator for an account, if there already exists a mobile phone activated for that account. A manual process should be introduced to handle this situation.

To also protect against the malware-based impersonation attack and the similar phishing attack, there is a need for a tighter control over the transition from an old OTP generator to a new phone-based OTP generator. At the time of writing, an attacker who gets hold of a valid URL containing a Secure Object need not have an old OTP generator for the account under attack to make his own mobile phone become an OTP generator for the account. A solution here is to let the user enter an OTP from the old OTP generator into the mobile phone, instead of the activation code provided by the activation protocol. The ES must then verify this OTP with the SP. This additional step ties the holder of an existing OTP generator to the mobile phone that is about to be activated.

It should be noted that it is difficult to completely defend against the impersonation attacks since an attacker can create a fake web page to ask a customer for the extra OTP required by the suggested countermeasure [12]. The same technique will not work for the replay attack, since the ES now rejects any earlier received activation request.

5 An enhanced solution

While we have introduced countermeasures to thwart malware and phishing attacks aiming to move the OTP generation from a customer's phone to an attacker's phone, the current Encap solution still does not protect against more 'traditional' phishing and malware attacks modifying or spoofing transaction requests from a client to an online bank's central infrastructure. In this section, we outline how Encap's solution can be augmented to help thwart these attacks, using an online bank as an example.

5.1 Added malware and phishing protection

An online bank may use the mobile phone channel provided by Encap's system to ask a customer to confirm a transaction request sent by his PC. The transaction is then executed only if the bank receives a confirmation from the customer via the phone channel. This extra control feature need not be used for all transactions. A possible solution is to only activate the control feature for outgoing payments over a certain limit selected by the customer or set by the bank. It should of course not be possible for client-side malware to change the value of this limit.

As long as the customer's mobile phone and PC are different devices communicating with the bank server over separate channels, no malware on the PC can modify information in the phone channel or on the phone itself. The malware can still harvest information about the customer's account and transactions, but it cannot manipulate or send fake transaction requests to the bank without the customer being able to notice the malicious activity thanks to the confirmation requests

sent to his phone. Of course, if the customer ignores a confirmation request, or the attacker is able to install malware on both the phone and the PC, then the control feature fails.

While confirmation requests can be sent to a phone in cleartext via SMSs, it is preferable to use a more secure SSL connection. The client portion of the control feature can be included in the existing MIDlet or implemented separately.

6 Summary

The Norwegian company Encap has developed a system allowing individuals to use their mobile phones as OTP generators. We suggested two minor changes to Encap's protocol designs, one to bring the activation protocol's key generation in line with "best practice," and one to simplify the designs without reducing the security.

A third party was responsible for integrating Encap's product into the evaluated online bank. The integration enables several practical attacks. The described client-side malware and phishing attacks on the customer authentication in the online bank are possible because the defense against replay of old activation requests is insufficient, and because the link between the previously used OTP generator and the new phone-based OTP generator is too weak. Encap received an early version of this paper with recommendations to implement the suggested countermeasures to thwart possible future attacks. The authors have since been informed that Encap and the third party have implemented some of the described countermeasures. The details of the implemented countermeasures are not known to us.

The seriousness of the attacks shows how important a system-level analysis and testing can be to determine the level of security provided by protocols in a real system. Since the Encap solution is new, it should be further scrutinized for weaknesses. The authors believe it is particularly important to study how the Encap solution should be integrated into existing web-based services. It may also be interesting to further study our suggestion on how to use the mobile phone to detect modification or spoofing of transaction requests from a customer's PC.

Acknowledgements

The authors would like to thank Encap for providing us with information about their system and for answering all our questions. Thank you also to Vidar Drageide for assistance during the testing of the attack scenarios.

References

[1] A. M. Hagalisletto and A. Riiber, "Using the Mobile Phone in Two-Factor Authentication," Encap white paper;
`www.encap.no/admin/userfiles/file/iwssi2007-05.pdf`

[2] B. Schneier, *The Failure of Two-Factor Authentication*, blog entry, March 15, 2005;
`www.schneier.com/blog/archives/2005/03/the_failure_of.html`.

[3] B. Schneier, *More on Two-Factor Authentication*, blog entry, April 12, 2005;
`www.schneier.com/blog/archives/2005/04/more_on_twofact.html`

[4] RFC 2631, *Diffie–Hellman Key Agreement Method*, June 1999;
`tools.ietf.org/html/rfc2631`.

[5] M. Ståhlberg. "The Trojan Money Spinner," presented at the *Virus Bulletin Conference*, Vienna, Austria, September 2007;
`www.f-secure.com/weblog/archives/VB2007_TheTrojanMoneySpinner.pdf`.

[6] Finjan Malicious Code Research Center, *Cybercrime Intelligence Report,* no. 3, 2009;

[7] F-Secure Virus Descriptions; www.f-secure.com/v-descs/haxdoor_ki.shtml.

[8] T. Espiner, *Swedish Bank Hit by 'Biggest Ever' Online Heist*, ZD Net UK, January 19, 2007; news.zdnet.co.uk/security/0,1000000189,39285547,00.htm.

[9] L. O. Murchu, *Banking in Silence*, January 14, 2008; www.symantec.com/connect/blogs/banking-silence.

[10] A. Jøsang, B. AlFayyadh, T. Grandison, M. AlZomai, and J. McNamara, "Security Usability Principles for Vulnerability Analysis and Risk Assessment," presented at the *Twenty-Third Annual Computer Security Applications Conference* (ACSAC), Miami Beach, FL, USA, Dec. 10–14, 2007; www.acsac.org/2007/papers/45.pdf.

[11] R. Dhamija, J. D. Tygar, and M. Hearst, "Why Phishing Works," CHI 2006, Montral, Qubec, Canada, April 22–27, 2006; people.seas.harvard.edu/∼rachna/papers/why_phishing_works.pdf.

[12] K. J. Hole, A. N. Klingsheim, L.-H. Netland, Y. Espelid, T. Tjstheim, and V. Moen, "Risk Assessment of a National Security Infrastructure," *IEEE Security & Privacy*, January/February 2009; www.nowires.org/Papers-PDF/RiskEvaluation.pdf.

Bibliography

[1] OSCE, "Charter of Paris for a New Europe," 1990, URL http://www.osce.org/documents/mcs/1990/11/4045_en.pdf.

[2] J. Elklit and P. Svensson, "What Makes Elections Free and Fair?" *Journal of Democracy*, 8(3):pp. 32–46, July 1997.

[3] M. Volkamer, *Evaluation of Electronic Voting: Requirements and Evaluation Procedures to Support Responsible Election Authorities*, Springer-Verlag, October 2009.

[4] Komunal- og Regionaldepartementet, "Prosjektdirektiv for e-valg 2011," February 2009.

[5] B. Schneier, *Schneier on Security*, Wiley Publishing, Inc., 2008.

[6] R. Celeste, D. Thornburgh, and H. Lin, *Asking the Right Questions About Electronic Voting*, National Academies Press, March 2006.

[7] S. Marsh and M. R. Dibben, "The Role of Trust in Information Science and Technology," *Annual Review of Information Science and Technology*, 37(1):pp. 465–498, 2003.

[8] S. Marsh and M. R. Dibben, *Trust, Untrust, Distrust and Mistrust - An Exploration of the Dark(er) Side*, volume 3477 of *Lecture Notes in Computer Science*, pp. 17–33, Springer-Verlag, May 2005.

[9] L. Loeber, "E-Voting in the Netherlands; from General Acceptance to General Doubt in Two Years," in R. Krimmer and R. Grimm, editors, *Electronic Voting*, volume 131 of *Lecture Notes in Informatics*, pp. 21–30, GI, 2008.

[10] R. Gonggrijp, W.-J. Hengeveld, A. Bogk, D. Engling, H. Mehnert, F. Rieger, P. Scheffers, and B. Wels, "Nedap/Groenendaal ES3B Voting Computer – A Security Analysis," Technical report, The "We do not trust voting computers" foundation, October 2006, URL http://wijvertrouwenstemcomputersniet.nl/other/es3b-en.pdf.

[11] OSCE/ODIHR, "Final Report on the 22 November 2006 Parliamentary Elections in The Netherlands," 2006.

[12] OSCE/ODIHR, "Needs Assessment Mission Report on the 14 September 2009 Parliamentary Elections in Norway," 2009.

[13] "Final Report: International Election Observation Mission on the Norwegian Parliamentary election of September 12, 2005," 2005.

[14] A. S. Patrick, P. Briggs, and S. Marsh, "Designing Systems that People Will Trust," in L. F. Cranor and S. Garfinkel, editors, *Security and Usability*, chapter 5, pp. 75–99, O'Reilly, 2005.

[15] K. Vollan, "Observing Electronic Voting," Report 15, NORDEM, 2005.

[16] Ministry of Local Government and Regional Development, "Electronic Voting – Challenges and Opportunities," February 2006.

[17] J. Nou, "Privatizing Democracy: Promoting Election Integrity through Procurement Contracts," *Yale Law Journal*, 2009.

[18] Inter-Parliamentary Council, "Declaration on Criteria for Free and Fair Elections," Paris, March 26, 1994, URL http://www.ipu.org/Cnl-e/154-free.htm.

[19] M. Bishop, *Introduction to Computer Security*, Addison-Wesley Professional, 2004.

[20] G. Røsland, *Remote Electronic Voting*, Master's thesis, University of Bergen, 2004.

[21] A. R. Olsson, "E-röstning – En lägesrapport," Technical Report 35/2001, IT-kommissionens observatorium för IT, demokrati, och medborgarskap, 2001.

[22] A. Jones and D. Ashenden, *Risk Management for Computer Security: Protecting Your Network & Information Assets*, Butterworth-Heinemann, 2005.

[23] Ministry of Local Government and Regional Development, "E-vote 2011, System Requirements Specification," October 2009.

[24] Bergens Tidende, "Nei til nettvalg," November 16, 2009, URL http://www.bt.no/nyheter/lokalt/Nei-til-nettvalg-966575.html.

[25] Asker og Bærums Budstikke, "Ikke nettvalg i Bærum," November 4, 2009, URL http://www.budstikka.no/sec_nyheter/article270703.ece.

[26] D. Hubbard, *The Failure of Risk Management*, John Wiley & Sons, Inc., 2009.

[27] S. Hernan, S. Lambert, T. Ostwald, and A. Shostack, "Uncover Security Design Flaws Using The STRIDE Approach," *MSDN Magazine*, (November), 2006, URL http://msdn.microsoft.com/en-us/magazine/cc163519.aspx.

[28] D. Stuttard and M. Pinto, *Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws*, John Wiley & Sons, Inc., 2007.

[29] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns : Integrating Security and Systems Engineering (Wiley Software Patterns Series)*, John Wiley & Sons, Inc., March 2006.

[30] B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering: Using UML, Patterns and Java, Second Edition*, Prentice-Hall, Inc., 2003.

[31] T. Tjøstheim, T. Peacock, and P. Y. A. Ryan, "A Model for System-Based Analysis of Voting Systems," presented at the *Fifteenth International Workshop on Security Protocols*, April 2007.

[32] D. W. Jones, "Threats to Voting Systems," A position paper for the *NIST workshop on Threats to Voting Systems*, October 2005.

[33] D. Jefferson, A. D. Rubin, B. Simons, and D. Wagner, "A Security Analysis of the Secure Electronic Registration and Voting Experiment (SERVE)," January 2004, URL http://www.servesecurityreport.org/.

[34] A. C. Hobbs, *Locks and Safes: The Construction of Locks*, Virtue & Co. 1853.

[35] Det sentrale valgstyret ved Universitetet i Bergen, "Rektorvalget 2009. Protokoll," April 2009.

[36] K. J. Hole, "Alvorlige svakheter ved e-valgsystemet til UiB," May 15, 2009, URL http://nyheter.uib.no/?modus=vis_leserbrev&id=43736.

[37] "Passwords," February 2009, URL http://it.uib.no/?mode=show_page&link_id=157720&sublink_id=157721.

[38] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems (5th Edition)*, Addison-Wesley Longman Publishing Co., Inc., 2006.

[39] T. Evensen, "Valgdata ved rektorvalget UiB 2009," May 2009, URL http://www.uib.no/filearchive/notat.pdf.

[40] Ernst & Young, "Rapport fra gjennomgang av valgavvikling," May 2009.

[41] På Høyden, "Ni stemmer forkasta ved valet," April 3, 2009, URL http://nyheter.uib.no/?modus=vis_nyhet&id=43404.

[42] "Valgreglement for Universitetet i Bergen," URL http://regler.uib.no/regelsamling/show.do?id=184.

[43] G. Grendstad, "Usikker prosedyre, sikkert resultat," *Bergens Tidende*, May 16, 2009.

[44] A. D. Rubin, *Brave New Ballot: The Battle to Safeguard Democracy in the Age of Electronic Voting*, Morgan Road Books, September 2006.

[45] På Høyden, "Eksternrapport klar i løpet av veka," May 12, 2009, URL http://nyheter.uib.no/?modus=vis_nyhet&id=43688.

[46] G. Grendstad, "Universitetsvalgene," May 8, 2009, URL http://nyheter.uib.no/?modus=vis_leserbrev&id=43670.

[47] T. Tungodden and G. Grendstad, "Uriktige påstander om ulovlig UiB-valg," May 11, 2009, URL http://www.uib.no/adm/nyheter/2009/05/uriktige-paastander-om-ulovlig-uib-valg.

[48] Bergens Tidende, "IT-studenter ble rektorkandidater," May 15, 2009.

[49] På Høyden, "Testa hol i systemet," May 15, 2009, URL http://nyheter.uib.no/?modus=vis_nyhet&id=43747.

[50] På Høyden, "Rektorvalget er gyldig," May 19, 2009, URL http://nyheter.uib.no/?modus=vis_nyhet&id=43771.

[51] M. Bishop, S. Peisert, C. Hoke, M. Graff, and D. Jefferson, "E-Voting and Forensics: Prying Open the Black Box," in *Proceedings of the 2009 Electronic Voting Technology Workshop/Workshop on Trustworthy Computing (EVT/WOTE '09)*, August 2009.

[52] Det sentrale valgstyret ved Universitetet i Bergen, "Klage over Rektorvalget 2009," May 2009.

[53] Ernst & Young, "Tillegg til 'Rapport fra gjennomgang av valgavvikling'," May 2009.

[54] R. Oppliger and R. Rytz, "Digital Evidence: Dream and Reality," *IEEE Security and Privacy*, 1(5):pp. 44–48, 2003.

[55] Estonian National Electoral Committee, "Elektroonilise hääletamise statistika," URL http://www.vvk.ee/index.php?id=10610.

[56] R. M. Alvarez and J. N. Katz, "The Case of the 2002 General Election," in R. M. Alvarez, T. E. Hall, and S. D. Hyde, editors, *Election Fraud*, chapter 9, pp. 149–161, Brookings Institution Press, 2008.

[57] K. Zetter, "Did E-Vote Firm Patch Election?" October 2003, URL http://www.wired.com/politics/law/news/2003/10/60563.

[58] S. Arnesen, "Informasjon, motivasjon, prediksjon –Eit forsøk med prediksjonsmarknad før Stortingsvalet 2009," Presented at *Norsk statsvitskapleg fagkonferanse*, January 2009.

[59] T. Hill, "The First Digit Phenomenon," *American Scientist*, 86(4):pp. 358–363, Jul-Aug 1998.

[60] M. J. Nigrini, "I've got your Number," *Journal of Accountancy*, pp. 79–83, May 1999.

[61] W. R. Mebane, "Election Forensics: Vote Counts and Benford's Law," Prepared for presentation at the 2006 Summer Meeting of the Political Methodology Society, July 2006.

[62] J. Deckert, M. Myagkov, and P. C. Ordeshook, "The Irrelevance of Benford's Law for Detecting Fraud in Elections," 2010, URL http://www.vote.caltech.edu/drupal/node/327.

[63] The Carter Center, "The Venezuela Presidential Recall Referendum: Comprehensive Reports," 2005.

[64] W. R. Mebane, "The Second-Digit Benford's Law Test and Recent American Presidential Elections," in R. M. Alvarez, T. E. Hall, and S. D. Hyde, editors, *Election Fraud*, chapter 10, pp. 162–181, Brookings Institution Press, 2008.

[65] W. R. Mebane, "Note on the Presidential Election in Iran," June 2009.

[66] B. F. Roukema, "Benford's Law Anomalies in the 2009 Iranian Presidential Election," *Submitted to the Annals of Applied Statistics*, 2009.

[67] I. Levin, G. A. Cohn, P. C. Ordeshook, and R. M. Alvarez, "Detecting Voter Fraud in an Electronic Voting Context: An Analysis of the Unlimited Reelection Vote in Venezuela," in *Proceedings of the 2009 Electronic Voting Technology Workshop/Workshop on Trustworthy Computing (EVT/WOTE '09)*, August 2009.

[68] C. Adams and S. Lloyd, *Understanding PKI: Concepts, Standards, and Deployment Considerations*, Addison-Wesley Longman Publishing Co., Inc., 2002.

[69] B. Schneier and J. Kelsey, "Secure Audit Logs to Support Computer Forensics," *ACM Trans. Inf. Syst. Secur.*, 2(2):pp. 159–176, 1999.

[70] D. Ma and G. Tsudik, "A New Approach to Secure Logging," *Trans. Storage*, 5(1):pp. 1–21, 2009.

[71] "Implementing a Trusted Information Sharing Environment. Using Immutable Audit Logs to Increase Security, Trust, and Accountability," Technical report, Markle Foundation, 2006.

[72] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, John Wiley & Sons, Inc., 2001.

[73] D. S. Wallach, "Voting System Risk Assessment Via Computational Complexity Analysis," *William & Mary Bill of Rights Journal*, 17:pp. 325–249, December 2008.

[74] B. Adida, "Source Code and Voting: What's Really on that Machine," October 2009, URL http://benlog.com/articles/2009/10/29/source-code-and-voting-whats-really-on-that-machine/.

[75] J. Kitcat, "Source Availability and E-Voting: an Advocate Recants," *Commun. ACM*, 47(10):pp. 65–67, 2004.

[76] B. Chess and J. West, *Secure Programming with Static Analysis*, Addison-Wesley Professional, 2007.

[77] B. Adida, "Theory and Practice of Cryptography, Google Tech Talks," December 2007, URL http://www.youtube.com/watch?v=ZDnShu5V99s.

[78] D. Chaum, "SureVote: Technical Overview," in *Proceedings of the Workshop on Trustworthy Elections (WOTE'01)*, August 2001.

[79] R. Oppliger, "Addressing the Secure Platform Problem for Remote Internet Voting in Geneva," May 2002, URL http://www.geneve.ch/evoting/english/doc/rapports/rapport_oppliger_en.pdf.

[80] A. Ansper, S. Heiberg, H. Lipmaa, T. A. Øverland, and F. Laenen, "Security and Trust for the Norwegian E-Voting Pilot Project E-valg 2011," in *NordSec '09: Proceedings of the 14th Nordic Conference on Secure IT Systems*, pp. 207–222, Springer-Verlag, 2009.

[81] X. Meng, P. Zerfos, V. Samanta, S. Wong, and S. Lu, "Analysis of the Reliability of a Nationwide Short Message Service," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pp. 1811 –1819, May 2007.

[82] W. Enck, P. Traynor, P. McDaniel, and T. La Porta, "Exploiting Open Functionality in SMS-Capable Cellular Networks," in *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pp. 393–404, ACM, 2005.

[83] H. Raddum, L. H. Nestås, and K. J. Hole, "Security Analysis of Mobile Phones Used as OTP Generators," in *WISTP*, volume 6033 of *Lecture Notes in Computer Science*, pp. 324–331, Springer-Verlag, 2010.

[84] S. Heiberg, H. Lipmaa, and F. V. Laenen, "On E-Vote Integrity in the Case of Malicious Voter Computers," Cryptology ePrint Archive, Report 2010/195, 2010.

[85] K. Gjøsteen, "Analysis of an Internet Voting Protocol," March 2010, URL http://www.regjeringen.no/upload/KRD/Kampanjer/valgportal/e-valg/Nyheter/core.pdf.

[86] A. C. Yao, "Protocols for Secure Computations," in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pp. 160–164, IEEE Computer Society, 1982.

[87] D. Chaum, "Elections with Unconditionally – Secret Ballots and Disruption Equivalent to Breaking RSA," in *Lecture Notes in Computer Science on Advances in Cryptology (EUROCRYPT'88)*, pp. 177–182, Springer-Verlag, 1988.

[88] J. Benaloh, "Simple Verifiable Elections," in *Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop (EVT'06)*, USENIX Association, 2006.

[89] B. Adida, "Helios: Web-Based Open-Audit Voting," in *Proceedings of the 17th conference on Security symposium*, pp. 335–348, USENIX Association, 2008.

[90] Z. Golebiewski, M. Kutylowski, and F. Zagorski, "Verifiable Internet Voting State of the Art," April 2004.

[91] H. Raddum, "Coercion-Resistant Receipts in Electronic Elections," Unpublished report, April 2010.

[92] T. El Gamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," in *Proceedings of CRYPTO 84 on Advances in Cryptology*, pp. 10–18, Springer-Verlag, 1985.

[93] D. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *Commun. ACM*, 24(2):pp. 84–90, 1981.

[94] J. Groth, "A Verifiable Secret Shuffle of Homomorphic Encryptions," in *Proceedings of PKC '03, Lecture Notes in Computer Science Series*, pp. 145–160, Springer-Verlag, 2005.

[95] C. A. Neff, "A Verifiable Secret Chuffle and its Application to E-Voting," in *Proceedings of the 8th ACM conference on Computer and Communications Security*, CCS '01, pp. 116–125, ACM, 2001.

[96] B. Adida, O. Pereira, O. D. Marneffe, and J. jacques Quisquater, "Electing a University President using Open-Audit Voting: Analysis of Real-Eorld use of Helios," in *In Electronic Voting Technology/Workshop on Trustworthy Elections (EVT'09)*, USENIX Association, 2009.

[97] D. Chaum, R. T. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A. Ryan, E. Shen, A. T. Sherman, and P. L. Vora, "Scantegrity II: End-to-End Verifiability by Voters of Optical Scan Elections through Confirmation Codes," *Trans. Info. For. Sec.*, 4:pp. 611–627, December 2009.

[98] J. Benaloh, "Administrative and Public Verifiability: Can We Have Both?" in *Proceedings of the conference on Electronic voting technology (EVT'08)*, pp. 1–10, USENIX Association, 2008.

[99] Ministry of Local Government and Regional Development, "The Main Features of the Norwegian Electoral System," URL http://www.regjeringen.no/en/dep/krd/kampanjer/election_portal/the-norwegian-electoral-system.html.