TALLINN UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Department of Computer Science

ITT70LT

Tanel Torn 121337IVCMM

SECURITY ANALYSIS OF ESTONIAN I-VOTING SYSTEM USING ATTACK TREE METHODOLOGIES

Master thesis

Supervisor: Vahur Kotkas, MSc

Tallinn 2014

Declaration

I hereby declare that I am the sole author of this thesis. The work is original and has not been submitted for any degree or diploma at any other university.

…….. …….

(Date) (Signature)

Abstract

This thesis provides a security analysis of the Estonian I-voting system using three different attack tree methodologies. The computational models of each methodology are used to analyse which attacks a rational and economically thinking attacker could undertake in order to attack the system. The thesis concentrates on large-scale attacks.

The work shows that based on the assigned parameter values by the author, the Estonian I-voting scheme is reasonably secure against large-scale vote manipulation attacks, but under certain conditions, susceptible to large-scale revocation attacks. The work also shows that the computational models of different attack tree methodologies can produce somewhat different results.

Annotatsioon

Käesolev magistritöö analüüsib Eesti e-valimiste süsteemi turvalisust, kasutades selleks kolme erinevat ründepuude metoodikat. Metoodikate arvutusmudelite abil uuritakse, millised on võimalikud ründed ratsionaalse ning majanduslikult mõtleva ründaja puhul. Töös keskendutakse laiaulatuslikele rünnetele.

Töös näidatakse, et töö autori poolt väärtustatud parameetrite järgi on Eesti e-valimiste süsteem turvaline laiaulatusliku häältega manipuleerimise suhtes, kuid, teatud tingimustel, ebaturvaline rünnete puhul, mille eesmärgiks on e-häälte tühistamine. Samuti selgub tööst, et erinevate ründepuude arvutusmudelite kasutamine võib pakkuda mõnevõrra erinevaid tulemusi.

Table of Contents

Figures

Tables

List of abbreviations

NEC – National Electoral Committee

VMM – Vote Modifying Malware

VCM – Vote Changing Malware

VBM – Vote Blocking Malware

RVM – Re-voting Malware

SVM – Self-voting Malware

VFS – Vote Forwarding Server

VSS – Vote Storage Server

VCA – Vote Counting Application

FVA – Fake Voting Applications

Introduction

The inconspicuous integration between technology and everyday life has led many of us taking the technology for somewhat granted. An average person does not usually wonder why or how something works – knowing that it does, provides to be sufficient. The same thing applies when an average person casts a vote over the Internet and is unaware of the processes executed on the background. The I-voter takes trusts in the system that his or her vote will be received by the server in an unchanged form. Similarly, the regular voter also takes trust in the system by inserting an anonymous vote into the ballot box, as she or he has no means to confirm that his or her vote was correctly counted in the final tally. Therefore, a certain level of trust between the voter and the system is required in both cases.

In order to consolidate the trust of the voter, the security of the system must be of a certain level. To assess the security of a given system, we must use some sort of formal methodologies. One of the ways this can be done is by using the concept of multi-parameter attack trees [4], which allows us to evaluate parameters such as the cost and the success probability of the attack, and based on that information, compute a result which would indicate whether a certain attack is practical or not.

Although the concept of attack trees has been used to analyse the Estonian I-voting system before [28], the computational models of the method have been improved over time and due to the topicality of online voting, it is essential to carry out additional studies using contemporary methods. As each research on the topic either confirms or disputes any previous results, the author hopes to provide a better overview of the actual security level of the system.

The aim of the thesis is not to provide a complete risk analysis of the system with listing every possible threat, but to concentrate on large-scale attacks that could have a significant impact on the election results. Using computational models of the three attack tree methodologies, the author analyses the feasibility of various attacks against the Estonian I-voting system.

Since the results of the computations are only as good as the corresponding input parameters, the work does not strive for perfection, meaning that the author does not expect each numeric value in the thesis to be unconditionally in accordance to real life cases. Instead, the purpose is

to determine the possible weak points of the system, which could be exploited to affect the election results.

In Chapter 1, the author discusses the concept of voting and how it relates to the Internet voting method. It focuses on the requirements that must be met in order to implement a new voting method, and points out the goals and possible benefits of implementing the I-voting system.

In Chapter 2, the author provides an overview of Internet voting in Estonia. The chapter discusses the potential effects of I-voting on the election results and provides a detailed description of the Estonian I-voting system. The chapter also provides a brief comparison of the Estonian system and systems used in other countries.

In Chapter 3, the author provides a security analysis of the Estonian I-voting system. It describes the concept of attack trees and the models used to carry out the analysis. The body of the chapter consists of describing the processes of constructing the attack trees and assigning the parameter values as well as providing the results of the simulations. The author also discusses some of the potential countermeasures to mitigate the risks of large-scale attacks against the I-voting system.

1. Concept of voting

By elections we understand a formal process of delegating power to a small group of individuals by voting. People, who are interested in taking part of selecting the representatives to the government, are presented with a list of possible candidates and from that list, they cast their vote in favour of a specific individual or a political party. If a person should decide that she wants to be part of the delegated power of the people herself, she can list herself as a possible candidate (given that she is eligible) and other people can vote for her. The means by which the people are able to cast their vote (and how votes are gathered) are determined by the implementation and availability of the voting methods in society.

In the 18th century, people in the United States called their votes aloud as the clerk wrote down the votes next to the names of the voters [20]. This provided the voters the means to directly verify how their votes were actually recorded, while the election as a whole was made transparent, as the voting method ensured easily observable election process. The problem with this approach, however, was that it left voters susceptible to bribery and coercion. The vote buyers were able to verify in person that the voter who was intimidated or bribed into voting for a specific candidate, actually voted for that candidate.

In order to safeguard against this kind of behaviour, and thus to ensure the sincere choice of the voter, it is necessary for the voter to be able to cast her vote in secret. In the elections today, the secrecy is usually achieved with the use of secret ballots. When a person goes to voting, she receives an empty ballot, which contains no information about the person herself. To cast the vote in private, the voter steps into the polling booth, where she fills in the ballot with her candidate of preference and puts the ballot into the ballot box. As the ballot box only contains anonymous ballots, no connection can be drawn between the voter and the vote, and the overall goal of secrecy is achieved. The concept of secret ballot itself dates back to Ancient Greece [35] and has now been widely adopted as *de iure* voting method in most (if not all) democratic countries [6, 7, 8, 9]. The idea of the voter being able to cast her vote in private and in total secrecy protects the voter against coercion and bribery. However, the use of secret ballots makes the elections as a whole less transparent, as the observers cannot directly verify that each vote was correctly (as intended) accounted for in the final tally.

It seems that secrecy and verifiability are essentially opposites and the contradiction between these two requirements makes it difficult to develop a system that could, in one hand, be

thoroughly auditable, while on the other hand, maintain voter's right to privacy and secrecy. The development of such a system is made even more difficult as there are other requirements that need to be taken into account as well. According to the paragraph §60 of the Constitution of the Republic of Estonia [7], the following basic principles must be met.

- **Authenticity** – only eligible voters are allowed to vote
- **Freedom** – a voter himself/herself chooses whether or how to vote.
- **Generality** – all citizens have the right to vote
- **Uniformity** – all votes are equal and each voter has but one vote
- **Directness** – the vote is cast directly by the voter
- **Secrecy** – only voter knows how he/she voted

With the information from the Electoral Acts in Estonia [34], we can complement this list with requirements that correspond to the key principles of information security.

- **Confidentiality** – voting results are published after the election
- **Availability** – voters have access to voting methods
- **Integrity** – all votes are taken correctly into account

Therefore, we get a set of rules that each voting method must correspond to in order to be adopted by society. Violating these rules might lead to National Electoral Committee (NEC) declaring the election results as invalid along with the possibility of abandoning the corresponding voting method entirely. Before adopting a new voting method, however, it is necessary to understand the goals of the method and determine whether achieving those goals justifies the implementation of the method. With Internet voting, one of the primary benefits would apparently be the increase of accessibility to the electoral process. Part of the electorate may not be able to vote in the traditional way (or in any other way for that matter) due to several reasons, such as living far from the polling stations with no simple means for transportation or simply by being too busy on the election period. Voting online, however, only takes a fraction of the time and effort from the voter to cast her vote. Increasing the accessibility, in turn, has the potential to increase the overall voter turnout, which would benefit the society as a whole, as more people would be able to take part of the democratic processes.

Internet voting would also help to lower the costs of elections in the long term, as the I-voting system can be re-used in future elections and maintaining the system requires less human resource than sustaining polling stations throughout the country. However, as Internet voting is not yet nearly as popular as traditional voting (see Section 2.2 for statistics), it probably will not start to replace paper voting in any time soon and we cannot talk about the decrease in costs on a state level today, although the decrease in costs for the voter is evident even now.

2. Internet voting in Estonia

This chapter provides an overview of the Estonian Internet voting scheme. It begins by giving a short history to the internet voting project in Estonia (Section 2.1), following a short statistical analysis of the past election results. Section 2.2 describes the system currently in use as well as pointing out the main weaknesses in the system used from 2005 to 2011 and describing the solution implemented in 2013. Finally, a brief overview of the internet voting projects in other countries is given in Section 2.4.

2.1 Beginnings of Internet voting in Estonia

The history of I-voting in Estonia can be traced back to 2001, when the Ministry of Justice ordered an analysis of the realization possibilities of I-voting [24]. The report was unfavourable in nature, completely ruling out the possibility of implementing the system in the Parliamentary Elections of 2002 due to the lack of technological capabilities. However, the authors of the paper did conclude that I-voting is possible (even inevitable) in the future and suggested a research program towards the cryptographic and technological means necessary to implement I-voting in the country as well as any social impacts it might have on the society.

Later that year, another report was ordered by the Ministry of Transport and Communications, also analysing the possibilities of implementing I-voting in Estonia [40]. The latter one was far more positive in nature than its predecessor, concluding that the implementing I-voting is possible even in 2002 as well as giving recommendations on the required organisational and technological means. Still, similarly to the first report, it also saw the development of the I-voting system as a long-term process.

In regard to these two reports, it was decided in 2002 that I-voting would not be implemented before the local government elections of 2005 [26]. In the following year, a group of security experts proposed a scheme for implementing I-voting in Estonia [13]. The security analysis of the scheme stated that while the proposed scheme is simplistic in nature, a mathematically more secure yet more complex scheme would also make its implementation more complex with increasing the number of components and links between them, and ultimately reducing the security of the scheme. It proposed a compromise between the theoretical security and the complexity of its implementation by maintaining the level of similarity between Internet voting and traditional voting, and by the use of existing solutions for digital signatures, simplistic cryptographic protocols and the know-how present in Estonia, while placing a level of trust in

the central servers and in the computers of the voters. The analysis concluded that such of a compromise is reasonable and the proposed I-voting mechanism was believed to be more secure than the regular voting.

2.2 Statistical analysis of I-voting results

The first electronic elections in Estonia took place during the local municipalities elections in 2005. Since then, people of Estonia have been able to cast their vote electronically on seven different occasions and it is very likely that this number will increase in time, as no large-scale incidents have occurred during this period.

Since the implementation of I-voting, the number of voters willing to cast their vote online has seen a significant increase over the years, peaking with 24,3% (56,4% among advance voters) during the Parliamentary elections in 2011 (Figure 1) [37]. Although the percentage of I-voters saw a slight decrease during the local municipalities' elections in 2013 in regards to the Parliamentary elections of 2011, it still shows that a very large portion of Estonian society has accepted the I-voting method as a valid means to cast a vote. It also shows that the tally of the I-votes has a potential to have a significant impact on the outcome of the elections as a whole.

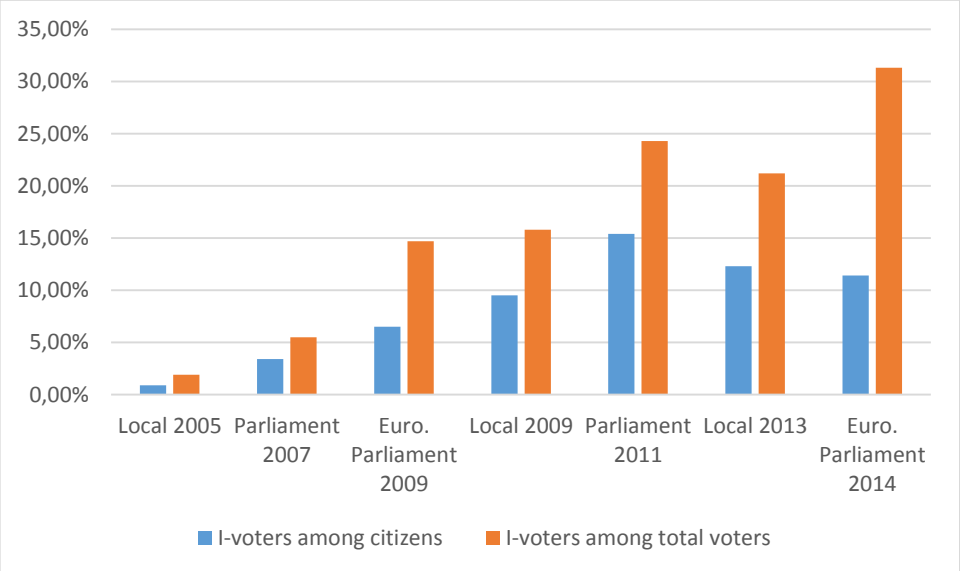

Figure 1. I-voting results by elections

As mentioned in Chapter 1, one of the primary goals of I-voting is to increase accessibility to elections in terms of casting a vote, which in turn can serve as potential means to increase voter turnout. Looking at the statistics, it can be argued that I-voting has indeed had a positive effect on the elections as a whole (i.e. the voter turnout has increased). Table 1 [37] shows that the voter turnout has fluctuated from 36,5% to 63,5% during the period from 2005 to 2014.

Table 1. Estonian I-voting results by elections (NEC)

	Local 2005	**Parliamentary 2007**	**EP 2009**	**Local 2009**	**Parliamentary 2011**	**Local 2013**	**EP 2014**
Eligible voters	1059292	897243	909628	1094317	913346	1086935	902873
Participating voters	502504	555463	399181	662813	580264	630050	329766
Voter turnout	47,40%	61,90%	43,90%	60,60%	63,50%	58,0%	36,50%
I-voters	9317	30275	58669	104413	140846	133808	103151
I-votes counted	9287	30243	58614	104313	140764	133662	103105
I-votes cancelled	30	32	55	100	82	146	46
I-votes invalid	-	-	-	-	-[1]	1	-
Multiple I-votes	364	789	910	2373	4384	3045	2016
I-voters among eligible voters	0,90%	3,40%	6,50%	9,50%	15,40%	12,3%	11,40%
I-voters among participating voters	1,90%	5,50%	14,70%	15,80%	24,30%	21,2%	31,30%
I-votes among advance votes	7,20%	17,60%	45,40%	44%	56,40%	50,5%	59,20%
I-votes cast abroad among I-votes	-	2% 51 states	3% 66 states	2,8% 82 states	3,90% 105 states	4,20% 105 states	4,69% 98 states
I-voting period	3 days	3 days	7 days	7 days	7 days	7 days	7 days
I-voters using mobile-ID	-	-	-	-	2690	11753	11609
I-voters using mobile-ID among I-voters	-	-	-	-	1,90%	8,6%	11,00%
Share of I-votes that were verified by the voter	-	-	-	-	-	3,40%	4,10%

However, in sense of statistics, it is meaningful to look different types of elections separately from each other as the general voter turnout differs in each case. The average voter turnout (during the same period) for European parliament elections (40,2%) is lower than the average turnout for local municipalities elections (55,3%). The latter one, in turn, is lower than the average turnout for parliamentary elections (62,7%). Therefore, it makes sense to group different kinds of elections together and study how the voter turnout has changed correspondingly to each group.

[1] One invalid vote is depicted among cancelled votes

In order to get a complete picture, we must also include the data from the past elections where I-voting was not yet implemented as a method for voting. However, it must be noted that including the data from all of the past elections in Estonia would likely yield in inaccurate results in terms of measuring the effects of I-voting. Estonia regained its independence from the Soviet Union in 1991 and the first elections were held in 1992. The following elections were held in 1993 and 1995. During the first years after a country (re)gains independence, however, people's desire to vote can be expected to be significantly higher than it would be otherwise as the political spirit is at an elevated state. It can be presumed that this was also the case in Estonia during the period from 1991 to 1995.[2] Therefore, it seems reasonable to exclude those years from the data set and only consider the past few elections prior to the I-voting for each group.[3] Table 2 shows the overall voter turnout for each election during the period from 1996 to 2014 [37, 38].

Table 2. Voter turnout for the elections held in the period 1996-2014

Local municipalities elections	**1996**	**1999**	**2002**	**2005**	**2009**	**2013**
Turnout (%)	52,5	49,8	52,5	47,4	60,6	58,0
Parliament elections	**1999**	**2003**	**2007**	**2011**		
Turnout (%)	57,4	58,2	61,9	63,5		
European Parliament elections	**2004**	**2009**	**2014**			
Turnout (%)	26,8	43,9	36,5			

With the data from Table 2, we can compute the average voter turnout for the elections that were held before as well as after the implementation of I-voting. For the local municipalities' elections, the average turnout prior to I-voting is 51,6%, while the average turnout for the elections following I-voting is 55,3%. This shows a 7,2% increase in terms of overall turnout. It can be argued that this number could be considered much higher as there were only 1,9% of I-voters among participating voters in 2005, while the percentage was 15,8% in 2009. This actually makes the I-voting results in 2005 closer to the results in 2002, rather than to results in 2009 (1,9% is closer to 0% than to 15,8%). It is reasonable to assume that the I-voting results in 2005 could not have had any significant impact on the voter turnout, as there simply were not enough people who voted online due to prudence. For the Parliamentary elections before

[2] The period from 1991 to 1995 was chosen intuitively.
[3] The reason for considering only few of the past elections prior to I-voting comes from the fact that there is a very limited number of elections held between the first elections (where people were strongly inclined to patriotism) and elections where I-voting was implemented.

and after I-voting, the average turnouts were 57,8% and 62,7% respectively. This means an 8,5% increase in the turnout, which is arguably quite significant. Following the same reasoning in computing the average turnout for the European Parliament elections, would give us a 50,0% rise in the turnout. However, it must be noted that the first European Parliament elections were held in the same year (2004) when Estonia joined the European Union (EU). Although the majority (66,8%) of Estonian citizens supported the idea of joining the EU [12], it was far from unanimity. It can be argued that the opposition also resulted in low voter turnout during the elections in 2004. The author expects the support for EU to have increased in time, mostly due to the financial support and the sense of security the union provides. Therefore, following similar reasoning of excluding the first elections in Estonia after the country regained independence, the statistics for European Parliament elections cannot be adequately used to assess the impact of Internet voting on voter turnout.

Although, it cannot be claimed that these increases in voter turnout were caused solely by the implementation of Internet voting, it is this author's belief that it is reasonable to assume that Internet voting has had, at least partly, a positive effect on the voter turnout. In this respect, it can be said that the voting method has achieved its primary goal.

2.3 Overview of the Estonian I-voting system

During the period from 2005 to 2014, the basic scheme for the Estonian internet voting has remained largely the same with the most significant change – the verification protocol – implemented in 2013. On the conceptual level, the scheme is rather straightforward and resembles a double postal voting system, where a voter would identify herself with an ID document, after which she would receive a paper ballot and two empty envelopes. The voter would write a candidate of preference on the ballot and put it in the envelope, which contains no information about the voter. She would enclose the envelope into the outer envelope, which does contain information about the voter. Both envelopes would then be delivered to the voter's polling division of residence and after the eligibility of the voter is determined, the inner envelope would be taken out from the outer envelope. The outer envelope would be cast aside, while the inner (anonymous) envelope would be put in the ballot box. The purpose of the envelope scheme is to ensure the secrecy of the vote, while recording the vote in the list of voters in the polling district of residence prevents voting more than once.

There are altogether four main parties in the I-voting system, which are represented by differently coloured squares in Figure 2. [14]

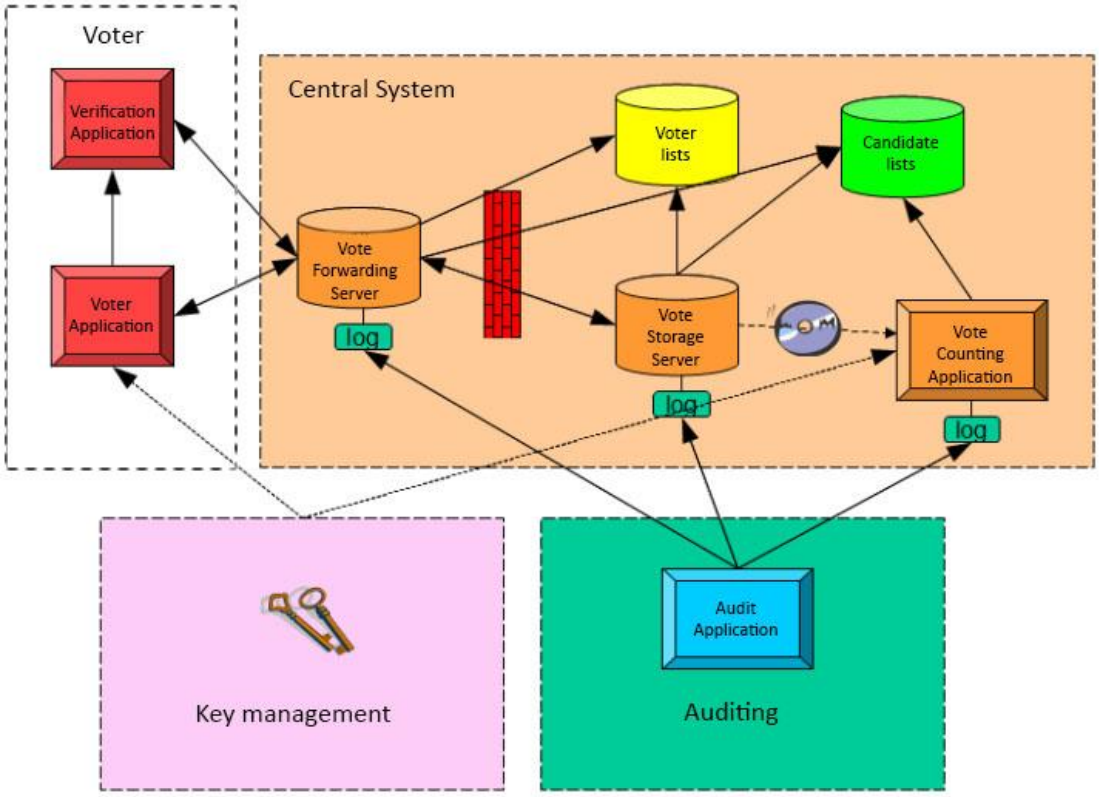

Figure 2. General architecture of the system

Voter encrypts and digitally signs the vote and sends it to the Central System using the Voting Application. Using the Verification Application the voter is enabled to verify that the Central System received the vote correctly.

Central System receives and processes the votes until the composite results of I-voting are output. Central System is a system component that is under the responsibility of the NEC and dependent on the Population Register to provide lists of eligible voters, and on NEC to provide the lists of candidates. It consists of three main components:

Vote Forwarding Server (VFS) – Authenticates the voter with the means of ID-card (or Mobile-ID), displays the candidates of voter's constituency to the voter and receives the encrypted and digitally signed I-vote. The I-vote is immediately sent to the Vote Storage Server and the confirmation received from there is then forwarded to the voter. It ends its work after

the closing of advance polls. It is the only component of the Central System that is directly accessible from the Internet.

Vote Storage Server (VSS) – Receives I-votes from the VFS and stores them. After the closing of advance polls, it removes double votes, cancels the votes by ineligible voters, and receives and processes I-vote cancellations. Finally, it separates inner envelopes from outer envelopes and readies them for the Vote Counting Application.

Vote Counting Application (VCA) – Offline component to which encrypted votes are transmitted with the digital signatures removed. The Vote Counting Server uses the private key of the system, tabulates the votes and outputs the results of I-voting.

Key Management generates and manages the key pair of the system. The public key is integrated into Voter's applications, while the private key is delivered to Vote Counting Application.

Auditing solves disputes and complaints, using logged information from the Central System.

A detailed description on how these parties and components relate to one another is given in the next sub-section (3.1).

2.3.1 Voting process

The I-voting process follows the same principles that are used in the double postal scheme described above, although the process is slightly more complicated. The process is as follows.

At first, the central voting system generates a RSA key pair and publishes the public key with the Voter Application. The key pair is generated in a Hardware Security Module (HSM) without the private component ever leaving the module. Communication with the HSM is carried out by key managers with the necessary physical (key-card) and knowledge-based (PIN-code) authentication devices. At least four members out of seven must be in present in order to activate the module and perform security critical operations.

The voter, using the downloaded Voting Application, authenticates herself to the VFS either by using an ID-card or a Mobile-ID and the VFS verifies the eligibility of the voter by performing a query from the database containing the list of voters (Figure 3).

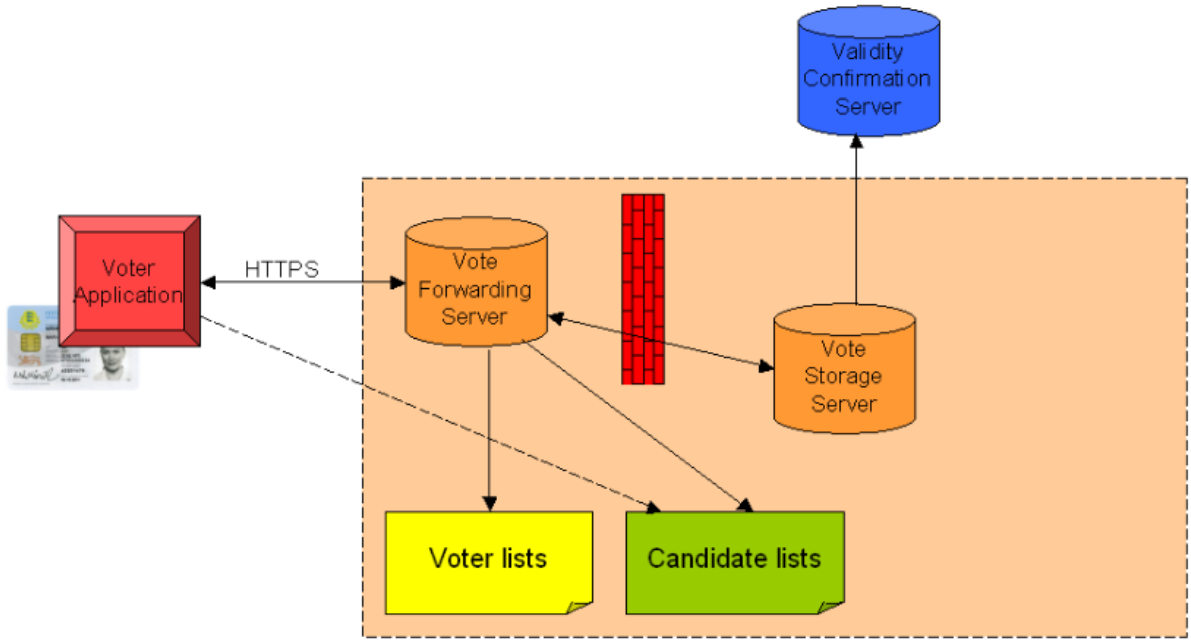

Figure 3. Components of the voting process

If the voter is eligible, the VFS checks from the VSS whether she has already voted and requests a candidate list from the candidate database that corresponds to the voter's constituency. The voter makes her choice and after confirmation, the voter application encrypts the chosen candidate number along with a random number using the public key of the VCA. The purpose of using a random number is to ensure different cryptograms for similar votes. This serves as the inner envelope in the aforementioned concept, where the voter makes her choice and puts the ballot in the envelope that does not contain any information about the voter.

The effect of the outer envelope is achieved by signing the digital ballot using the voter's ID-card (or Mobile-ID) and the resulting complete ballot is sent to the VFS. After the formal correctness of the received vote is verified by the VFS, the vote is forwarded to the VSS. The VSS acquires a certificate confirming the validity of the digital signature from the Validity Confirmation Server and adds it to the signed vote. In case the vote is valid, the VSS sends the VFS a confirmation that the vote has been received and the corresponding message is displayed to the voter. An entry that the vote was received is recorded in the log-file (LOG1). As the voter can vote several times, all previous votes will be automatically revoked and the corresponding entry will be recorded in the log-file (LOG2). After the end of I-voting period, the VFS ends all communication.

After casting the vote, it is possible for the voter to verify her choice by downloading the Verification Application and scanning the control code received from the server to her mobile. Sub-section 2.3.3 describes this process in more detail.

When the I-voting period ends, lists of I-voters by polling station are compiled and sent to polling stations simultaneously with the advance polls envelopes. This is done in order to prevent effectively counting two votes (I-vote and a traditional vote) for the same person. The polling divisions begin preparing the appeals for cancelling of I-votes. The people who have cast an I-vote and voted in advance on paper as well are marked in the appeals. The NEC sends a consolidated list of the lists received to the VSS, which checks the digital signature, saves the cancellation appeal and executes the cancellations. The corresponding entries are recorded in the log-file (LOG2). When the cancellation period ends, the digital signatures are removed from the ballots and the votes are prepared for transfer to the VCA on an external storage medium. All the entries sent to the VCA are recorded in log-file LOG3. Figure 4 illustrates this process visually.

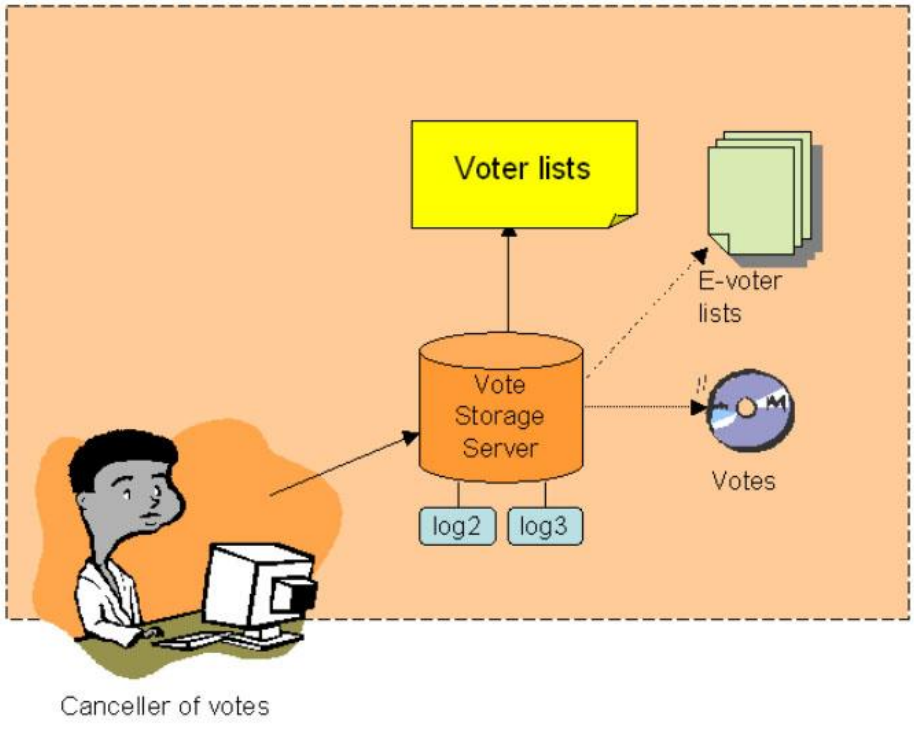

Figure 4. Components of vote storage and cancellation processes

As the last step, the votes are decrypted by constituencies using the private key of the VCA. The decrypted vote is checked against the candidate list to determine if it is possible to vote for the candidate in that constituency. If the candidate number is incorrect, the vote is declared

invalid and the corresponding entry is recorded in log-file (LOG4). The votes that are considered as valid are tabulated by candidates and constituencies, and recorded in log LOG5. Finally, the results of I-voting are added to the results of the regular voting (Figure 5).

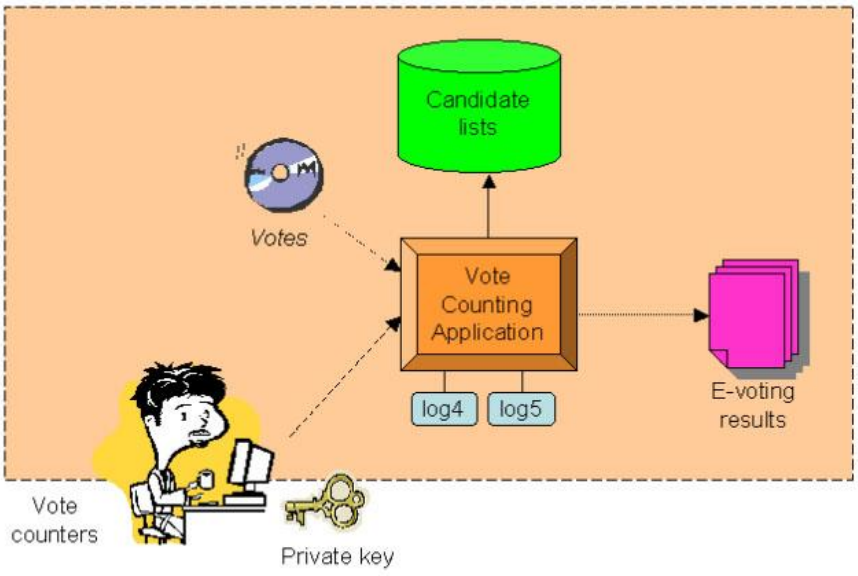

Figure 5. Components of vote counting process

As mentioned above, information about the votes are recorded as entries in different log-files. The audit application makes it possible to determine what happened with the vote cast with a specific personal identification code (PIC) and storing a hash from the encrypted candidate number and a random number ensures that one cannot reconstruct the original value of the vote. Table 3 provides an overview of the entries recorded in log-files.

Table 3. Log-file entries

Log file	**Entries**	**Format**
LOG1	Received votes	PIC, hash(vote)
LOG2	Cancelled votes	PIC, hash(vote), reason
LOG3	Votes to be counted	PIC, hash(vote)
LOG4	Invalid votes	hash(vote)
LOG5	Accounted votes	hash(vote)

The audit application is also able to check the integrity of logs. Entries from LOG2 and LOG3 combined must equal to the content of LOG1. Similarly, entries from LOG4 and LOG5 combined must equal to the content of LOG3.

2.3.2 Shortcomings of the system used in 2005-2011

During the period from 2005 to 2011, the system architecture was similar to the one described above, except it did not allow the voter to verify her vote by alternative channels other than her computer. The problem with this protocol was that manipulations with the votes could remain unnoticed by the voter as she has no real guarantee on how or even if her vote was cast. The confirmation of the vote is displayed on the voter's computer screen, but it is very easy to attack the voter's computer environment and relying on the same computer for verification, does not achieve much against malware.

In the Parliament elections of 2011, the most severe and widely published attack was demonstrated by a student, who wrote a prototype of a malware that was capable of tampering with the elections. Different versions of the malware were able to block and even change the original I-vote [16]. The student made use of the fact that the system used from 2005 to 2011 gave no reliable feedback to the voter concerning how the vote was actually received by the server. An appeal demanding the revocation of all I-votes was also submitted to the NEC [3], which even made it to Supreme Court of Estonia, but was dismissed on the grounds that the person's right to vote was not violated [10].

Although the appeal was dismissed and no indication of the malware being used for other purposes than mere demonstration, it reinforced the need for reliable verification. As the voter's computer cannot be trusted, the verification of the vote must be done via alternative channels.

2.3.3 Proposed verification protocol

The proposed verification system that was implemented in 2013 added a post-channel between the state and the citizen to inform the voter how her vote was received by the server as it uses voters' mobile devices to verify the vote. The central idea is the following [14]:

When the voter makes her choice and sends the signed encryption of the vote and a random number to the voting server, the server sends back a unique code, which is later used to download the correct vote to the mobile device. (Figure 6)

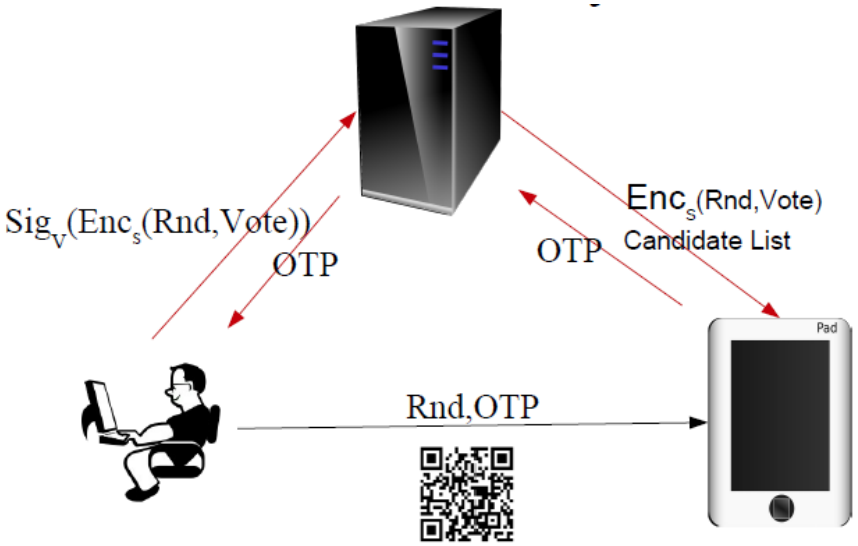

Figure 6. Vote verification process

The unique code *vr* (as in vote reference, denoted as OTP in Figure 6) and a random number *r* that was used to encrypt the vote are displayed to the voter as a QR code. The voter transfers them to her mobile phone by taking a picture of the QR code. The voter then sends the received *vr* to server from her mobile, which would identify the I-vote being verified and the server returns the digitally signed I-vote to the smart device along with a list of candidates she might have voted for.

The Verification Application cannot decrypt the I-vote but it knows the random number *r* used in encrypting the vote and the public key of the I-voting system that the data is encrypted with. The Verification Application will then create cryptograms to all the candidates in the candidate list using the random number. Once it finds the cryptogram that matches the I-vote received from the server, the voter's choice is confirmed.

2.4 Internet voting projects in other countries

Although there are many countries with I-voting projects, two of the most successful countries (besides Estonia) could be considered to be Norway and Switzerland. The next two sections will give a brief overview of the I-voting projects in both of these countries.

2.4.1 Norway

The first pilot project for internet voting in Norway was conducted in 2011 during the local government elections [19]. Internet voting was available only during the advance voting period and voters in 10 municipalities were allowed to cast their vote electronically. A total of 27 557 voters, or 16.4% of the eligible internet voters, chose to vote via Internet. [19]. As this was the

first time the Norwegians could cast their vote electronically, this number is surprisingly high. In comparison to the Estonian local government elections of 2005 (when the I-voting method was first introduced), the percentage of I-voters was only 1,9%. This could be partly related to the fact that the average voter turnout in Norway is significantly higher than in Estonia [42]. For example, the average turnouts for the Norwegian Parliamentary Elections in 2009 and 2013 were 76,37% and 78,23% respectively, while in Estonia, those numbers were 61,90% and 63,50% for the elections held in 2007 and 2011. The statistics indicate that Norwegian people are more interested in taking part of the elections than Estonians. This might also mean that they are more open to new voting methods. Two years later, during the Parliamentary elections of 2013, the percentage of the votes cast over the Internet in the municipalities piloting Internet voting was already as high as 36% [31]. Although these numbers show a rapidly rising popularity of I-voting among the Norwegian people, it must be noted that I-voting is currently available only in certain parts of Norway.

The Norwegian I-voting system architecture is similar to the Estonian one, although it holds some differences. For a detailed description, the author refers to [19] and [39]. The main components of the Norwegian system are:

Voting Application manages all interactions between the internet voting system and the voter. It provides the user (voter) interface, downloads the Java-based secure voting client that encrypts the ballots and sends them to the Vote Collector Server.

ID-portal authenticates the registered voters

Vote Collector Server (VCS) contains the electronic ballot box.

Return Code Generator (RCG) calculates and transmits the return codes via SMS.

The ministry servers consists of cleansing, mixing and counting servers. The cleansing server is responsible for the cleansing process to eliminate multiple ballots from the same voter, ballots from voters that also voted by paper ballot, and ballots from voters not on the voter list. The mixing server mixes the ballots. The counting server decrypts the mixed ballots and tallies the election results.

The I-voting process It starts by the voter logging on to a web-based voting application, where she authenticates herself by means of an electronic ID. If the authentication is successful, the system automatically checks the eligibility of the voter. If the voter is eligible, the voter is

presented with a list of candidates to choose from. Unlike the system implemented in Estonia, the Norwegian method also allows the voter to cast a blank ballot. After the choice is made and confirmed, the vote is encrypted and signed with the voter's digital signature. Since personal ID-cards in Norway do not have digital signatures, the voting system assigns one to each voter taking part in the pilot [39]. Finally, the vote is transmitted to server to be stored in the electronic ballot box

After the vote has been cast, the voter receives a SMS containing the return code corresponding to the vote. This return code can be compared to the voter's individual poll card, which has been previously sent to her by mail, and which contains a list of all the available candidates and their corresponding return codes. These return codes are individually calculated per voter prior to the election, which makes it possible for the voter to verify that her vote was cast as intended.

Figure 7 illustrates the verification process visually. [39]

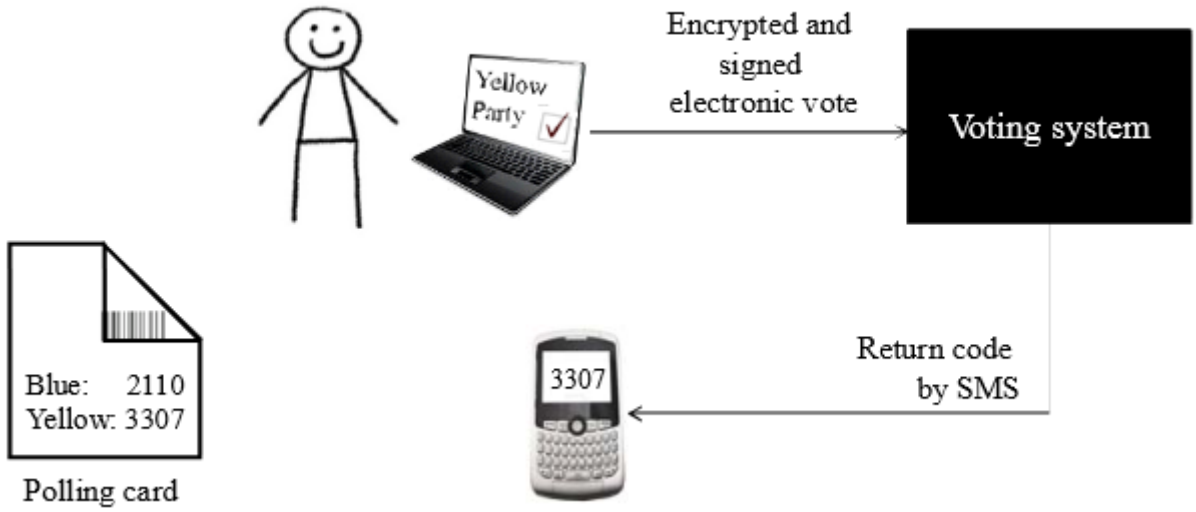

Figure 7. Vote verification process of the Norwegian system

What makes the Norwegian I-voting system different from the Estonian one is the use of polling cards and the SMS-based post-channel for verification. The main drawback of the Norwegian system is its unintuitive user interface as the voter must compare digital codes from the SMS to the ones on the polling card (rather than simply be presented the name of the candidate) in order to verify the vote. This might in turn have an effect on the voter turnout. Additionally, as the polling cards are distributed to voters by mail and the return codes are sent via SMS, the implementation of the system can be rather costly.

2.4.2 Switzerland

The Swiss internet voting project is one of the oldest in Europe as it began in 2000, when three cantons – Geneva, Zurich and Neuchâtel – of the Confederation volunteered to implement Internet voting. At that time, Geneva was the only Swiss canton, who had a computerized and centrally maintained voter registry [30] and in 2004, after some preliminary tests in the year before, it introduced Internet voting for the first time for cantonal and federal ballots. Four municipalities were offered internet voting as a valid voting option and 21,8% of the eligible voters chose to cast their vote online. The average turnout for these municipalities reached 58,35% [33], while the turnout for the whole canton was 57,1% [25]. As of June 2013, there have been 30 different electoral opportunities to vote online in the canton of Geneva, though most of them having involved referendums, rather than election of representatives or parties. There have been four occasions, when all eligible voters of the Geneva canton have been granted the option to vote online. On these four occasions, approximately 240 000 citizens were able to cast their vote via Internet with the average percentage of I-voters being 18,3% [25].

The Geneva's I-voting system makes use of voting cards that are sent out to the voters by mail prior to the elections. The voting cards are only valid for the upcoming ballot and essentially mark the person's right to vote. In order to cast a ballot, the voter first connects to the Internet voting website and SSL communication is established between the browser and the website. The voter is then asked to identify herself to the ballot management system by entering a unique identification number located on her voting card. A cryptographic hash function is applied on the voting card number to produce an imprint, which is then sent to the voting system. The system then uses a correspondence table drawn up at the stage of the print files generation to determine the voting card number based on the imprint.

After accepting the legal notices and filling out the ballot, the voter enters her birth date and the password from their voting card. Additionally, the voter selects her municipality of origin from a random list. The completed ballot paper is the sent, via the secure channel, to the voting server. The server decrypts and checks the authenticity of the voter's ballot paper, and performs a syntax check on the ballot paper to confirm the integrity of the vote. Finally, the server sends the voter confirmation of the date and time of her vote along with a control code, which corresponds to the control code on the voting card, confirming the server received the vote.

What makes the Swiss I-voting scheme significantly different from the Estonian and Norwegian schemes is the lack of post-channel verification protocol in order for the voter to verify how her

vote was actually received by the server. It also relies on voting cards for identification. This makes them different from the polling cards in Norway, which are only used to verify how the vote was actually cast. From the voter's point of view, it does not matter much as both systems require the use of external cards in the voting process and the usage is ultimately rather similar.

3. Security analysis of the Estonian I-voting system

This chapter provides a security analysis of the Estonian I-voting system using three different attack tree methodologies. It begins by describing the concept of attack trees (Section 3.1) and follows with giving an overview of the computational models of each methodology (Section 3.2). In Section 3.3, the author describes the formulated attack trees and the process of assigning parameter values. The chapter is concluded with Section 3.4, which describes the simulation process and its results.

3.1 Concept of attack trees

Attack trees provide a formal and a methodical approach to security analysis by using conceptual diagrams to describe how an asset or a system might be attacked. The analysis begins by identifying a primary threat, also called the root node, which directly inflicts damage on the system. The primary threat is divided into sub-attacks (sub-nodes) until such a state is reached, where it becomes impractical to further divide the resulting attacks. Such attacks, also called leaf nodes, describe an attack on the lowest level and are connected with each other through refinement nodes. These refinement nodes function similarly to logical AND and OR. In order to reach the goal of an AND-node (conjunctive), all of its child nodes need to be executed, whereas in order to reach the goal of an OR-node, at least one of its child nodes needs to be executed. An example of an attack tree is shown in Figure 8.

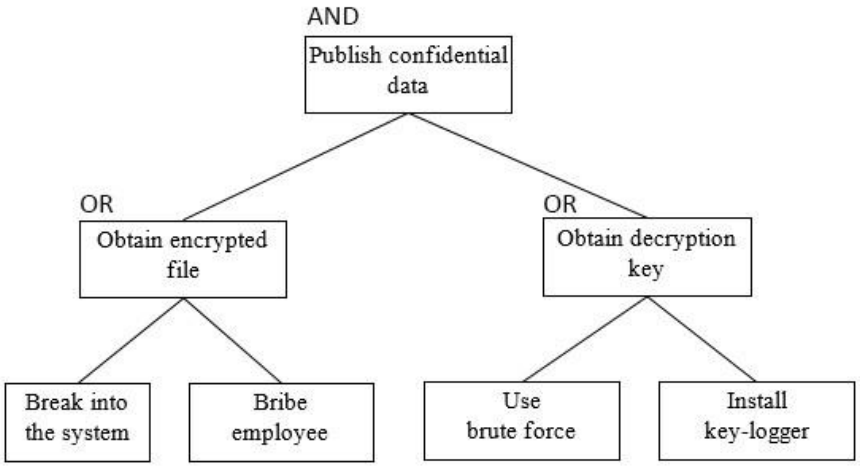

Figure 8. Example attack tree

In order to materialize the primary threat and publish confidential data in the example above, the attacker must obtain the encrypted file as well as decrypt it (because the root node itself is

an AND-node). There are two options to obtain the encrypted file: either by breaking into the system and stealing it, or by bribing an employee. Similarly, the file can be decrypted either by using a key-logger, or by brute forcing the password.

Figure 8 shows the possible attack vectors of the attacker, but it does not speculate on which attack(s) the attacker actually decides to execute. In order to get a sense of the potential behaviour of the attacker, some sort of parameters must be associated with the nodes and computational algorithms are needed to perform a quantitative security assessment.

The attack tree method derived from the concept of threat trees, which were used for tasks like fault assessment and vulnerability analysis, and was adapted to information security by Bruce Schneier in 1999 [36, 4]. Since then, the attack tree method has seen a significant development as new models have been developed and the old ones improved [23].

3.2 Models used in the analysis

The following sub-sections (3.2.1 to 3.2.3) describe the three different attack tree methodologies used to realize the security analysis of the Estonian internet voting system in the present thesis. Although the methodology proposed by Buldas *et al.* [4], and described in sub-section 3.2.1, could also be considered as a parallel model (due to the fact that all attacks in this model are executed simultaneously), for the sake of distinction, it is denoted as BLPSW[4] model, and with the Parallel model in Sub-section 3.2.2, the author refers to the parallel model described in [22].

3.2.1 BLPSW model

The models prior to the one proposed in [4] only considered one specific parameter for the nodes at a time, such as the cost or probability of the attack [36, 27]. In real life, however, it is reasonable to assume that the decision making process of the attacker is more complicated as there are many parameters the attacker considers simultaneously before concluding whether to launch the attack or not. This led the authors of [4] to develop a model that adopts a game-theoretic approach to attacker's decision making process and supports the use of multi-parameter nodes in the construction of attack trees.

[4] BLPSW comes from the initials of the authors.

The model views the attack as a game played by a rational attacker, meaning she is only willing to play the game if it is profitable for her. The model uses following parameters in order to decide about the profitability of the attack:

- Gains – gains of the attacker, in case the attack is successful
- Costs – the cost of the attack
- p – success probability of the attack
- π^+ – expected penalty, in case the attack was successful
- π^-– expected penalty, in case the attack was not successful

Here, the expected penalties can be computed as $\pi^+ = q^+ \cdot Penalties^+$ and $\pi^- = q^- \cdot Penalties^-$, where q^+ denotes the probability of getting caught in case the attack was successful, while q^- denotes the probability of getting caught in case the attack was not successful. Similarly $Penalties^+$ and $Penalties^-$ denote the expected penalties the attacker must pay if she is caught, in case the attack was successful or not. The overall value of the game, i.e. the expected outcome for the attacker, can be computed as:

$$\text{Outcome} = -\text{Costs} + p \cdot (\text{Gains} - \pi^+) - (1-p) \cdot \pi^-$$

It is assumed that the attacker behaves in a rational way, meaning that she is only willing to attack if $\text{Outcome} > 0$.

The parameters of the refinement (non-leaf) nodes are computed with the following routines:

For an OR-node with child nodes with parameters $(\text{Costs}_\text{i}, p_i, \pi_i^+, \pi_i^-)(i = 1{,}2)$ the parameters $(\text{Costs}_\text{i}, p_i, \pi_i^+, \pi_i^-)$ are computed as follows:

$$(\text{Costs}, p, \pi^+\pi^-) = \begin{cases} (\text{Costs}_1, p_1, \pi_1^+, \pi_1^-), \text{if Outcome}_1 > \text{Outcome}_2 \\ (\text{Costs}_2, p_2, \pi_2^+, \pi_2^-), \text{if Outcome}_1 \leq \text{Outcome}_2 \end{cases},$$

where $\text{Outcome}_i = p_i \cdot \text{Gains} - \text{Costs}_i - p_i\pi_i^+ - (1-p_i) \cdot \pi_i^-$.

For an AND-node with child nodes with parameters $(\text{Costs}_\text{i}, p_i, \pi_i^+, \pi_i^-)(i = 1{,}2)$ the parameters $(\text{Costs}_\text{i}, p_i, \pi_i^+, \pi_i^-)$ are computed as follows:

$$\text{Costs} = \text{Costs}_1 + \text{Costs}_2, \quad p = p_1 \cdot p_2, \quad \pi^+ = \pi_1^+ + \pi_2^+,$$

$$\pi^- = \frac{p_1(1-p_2)(\pi_1^+ + \pi_2^-) + (1-p_1)p2(\pi_1^- + \pi_2^+) + (1-p_1)(1-p2)(\pi_1^- + \pi_2^-)}{1 - p_1p_2}$$

The formula for π^- represents the average penalty of an attacker, assuming that at least one of the two child-attacks was not successful. It is convenient to subsequently denote the cost of the attack and the expected penalties for the node i as $\text{Expenses}_i = \text{Costs}_i - p_i\pi_i^+ - (1 - p_i) \cdot \pi_i^-$.

3.2.2 Parallel model

The model described in the previous section provided simple, but effective semantics for computing the expected outcome for the attacker. The model, however, had several drawbacks. Most notably, it was inconsistent with the framework proposed by Mauw and Oostdijk [27], which stated that the semantics of the tree should remain unchanged when the underlying Boolean formula is transformed to an equivalent one. For example, let us suppose we have an attack tree $T_1 = A \vee (B \wedge C)$ having parameters $\text{Gains} = 5000$, $p_A = 0.5$, $p_B = 0.5$, $p_C = 0.4$, $\text{Expenses}_A = 1500$, $\text{Expenses}_B = 1000$ and $\text{Expenses}_C = 500$. According to the computational rules of [4], we get $\text{Outcome}_1 = 1000$. Converting the underlying Boolean formula to conjunctive normal form (CNF), we get $T_2 = A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$. As the parameters for the resulting tree T_2 remain the same, we get $\text{Outcome}_2 = -500$. The expected outcomes are different, although the corresponding Boolean formulas are equivalent. This comes from the fact that when transforming the formula to CNF in the example above, the model of [4] creates a duplicate node of the leaf A and instead of having just one node throughout the computational process, we would actually have A_1 and A_2.

There also were problems with the computational model regarding the OR-nodes. It was assumed that the attacker would only choose to execute one attack in an OR-node (the one with the highest outcome), whereas in real life, it would make sense for the attacker to launch several attacks, if the success probabilities are high, while expected expenses are low. Additionally, the decision in the OR-node was made comparing the outcomes of the child nodes using the global Gains parameter. This means that each successful sub-attack would give the attacker the full outcome, which is clearly an overestimation [22].

The proposed model by Jürgenson and Willemson [22] overcomes these issues by providing more accurate semantics. In the model, the attacker first constructs an attack tree and evaluates the parameters of its leaves. The attacker then considers all potential attack suites (subsets) $\sigma \subseteq \mathcal{X} = \{X_i : i = 1, \dots, n\}$, evaluates the outcome for suites that materialise the root attack and launches the attack suite with the highest outcome.

The attack tree can be viewed as a Boolean formula F composed of the set of variables $\sigma \subseteq \mathcal{X} = \{X_i : i = 1, \ldots, n\}$, which correspond to the elementary attacks. Satisfying assignments $\sigma \subseteq \mathcal{X}$ of this formula correspond to the attack suites sufficient for materialising the root attack and the exact outcome of the attack can be computed with the following formula:

$$O = max\{O_\sigma : \sigma \subseteq \mathcal{X}, F(\sigma := true) = true\}$$

Here, O_σ denotes the expected outcome of the attacker if she decides to try the attack suite σ and $F(\sigma := true)$ denotes evaluation of the formula F, when all of the variables of σ are assigned the value $true$, while all other variables are assigned the value $false$. The expected outcome O_σ of the attack suite is computed as follows:

$$O_\sigma = p_\sigma \cdot g - \sum_{x_i \in \sigma} E_i$$

Here, p_σ denotes the success probability of the attack suite σ, g denotes the expected gains and E_i denotes the expenses of carrying out the leaf attack, which can be computed with:

$$E_i = c_i + p_i \cdot \pi_i^+ + (1 - p_i) \cdot \pi_i^-$$

It must be noted that as the full suite σ is used to mount the attack, it may contain redundancy as there may be subsets $\rho \subseteq \sigma$ that are sufficient for materialising the root attack. Therefore, the elementary attacks in the $\sigma \setminus \rho$ that do not contribute to materialising the root attack, affect the success probability of p_ρ with $(1 - pj)$, and the success probability of the entire attack suite σ can be computed with:

$$p_\sigma = \sum_{\substack{\rho \subseteq \sigma \\ \mathcal{F}(\rho := true) = true}} \prod_{X_i \in \rho} p_i \prod_{X_j \in \sigma \setminus \rho} (1 - p_j)$$

The downside of this model is the increase in computational complexity and therefore, this model is impractical to be used with very large attack trees.

3.2.3 Serial model

The previous two models can be considered parallel models as the elementary attacks take place simultaneously and the success or failure of such attacks are not considered by the attacker's decision making process. In practice, however, the attacker is able to order her actions and take alternative routes based on the results of previously executed attacks. She might stop trying altogether if some critical subset of elementary attacks has already failed or succeeded [21].

The model proposed in [21] can be considered as a serial model as it extends the basic parallel model by introducing temporal order to the elementary attacks and gives the attacker the possibility to skip some of them or stopping the attack before all of the elementary attacks have been tried. It also generalizes the attack tree approach to accommodate arbitrary rooted directed acyclic graphs (RDAG-s).

As the complexity of computation algorithms with this approach is much higher than in the parallel models described above, this chapter provides an abridged description of the model and the author refers to [21] for a more detailed one. In the Serial model, the attacker takes the following steps:

1. Creates an attack RDAG with the set of leaf nodes $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$.
2. Selects a subset $S \subseteq \mathcal{X}$ materialising the primary threat and considers the corresponding sub-tree.
3. Selects a permutation α of S.
4. Based on the sub-tree and permutation α, computes the expected outcome.
5. Maximises the expected outcome over all the choices of S and α.

Since only one subset S and the corresponding subtree are relevant in step 4, it can be assumed with any loss of generality that $S = \mathcal{X}$ and the attacker's behaviour for permutation α is described in Algorithm 1 (Figure 9).

Algorithm 1 Perform the attack

Require: The set of elementary attacks $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$, permutation $\alpha \in S_n$ and a monotone Boolean formula $\mathcal{F}$ describing the attack scenarios

```
for i := 1 to n do
    Consider X_α(i)
    if success or failure of X_α(i) has no effect on the success or failure of the root node then
        Skip X_α(i)
    else
        Try to perform X_α(i)
        if the root node succeeds or fails then
            Stop
        end if
    end if
end for
```

Figure 9. Algorithm 1 of the Serial model

The expected outcome of the attack based on permutation α can be computed with:

$$O_\alpha = p_\alpha \cdot g - \sum_{X_i \epsilon X} p_{\alpha,i} \cdot E_i$$

Here, p_α denotes the success probability of the primary threat and $p_{\alpha,\mathrm{i}}$ the probability that the node X_i is encountered during Algorithm 1.

For better describing the computational process, let us have an attack tree with the leaf nodes $X_1, \ldots, X_n$ with the corresponding success probabilities $p_1, \ldots, p_n$ that are independent from one another. Considering the permutation $\alpha \epsilon S_n$, we introduce two new parameters to each node Y, namely $Y.t$ and $Y.f$, which denote the probabilities that the node has been proven to be either $true$ or $false$. The parameters for each leaf node are initially set $Y.t = p_i$ and $Y.f = 1 - p_i$, whereas the parameters for all other nodes are set $Y.t = Y.f = 0$. These values will be incrementally adjusted until we have $R.t = p_\alpha$ for the root node R. The parameters for the parent nodes are computed using the parameters of the child nodes. For an AND-node A with children B and C, we set:

$$A.t = B.t \cdot C.t$$

$$A.f = B.f + C.f - B.f \cdot C.f$$

For an OR-node A with children B and C, we set:

$$A.t = B.t + C.t - B.t \cdot C.t$$

$$A.f = B.f \cdot C.f$$

Algorithm 2 (Figure 10) is used to calculate the probabilities $p_{\alpha,i}$.

Algorithm 2 Computing the probabilities $p_{\alpha,i}$

Require: An attack tree with leaf set $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$ and a permutation $\alpha \in S_n$

Ensure: The probabilities $p_{\alpha,i}$ for $i = 1, 2, \ldots, n$

1: **for all** $Z \in \{X_1, \ldots, X_n\}$ **do**
2: $Z.t := 0,\ Z.f := 0$
3: **end for**
4: **for** $i := 1$ to n **do**
5: Find the path $(Y_0, Y_1, \ldots, Y_m)$ from the root $Y_0 = R$ to the leaf $Y_m = X_{\alpha(i)}$
6: $p_{\alpha,\alpha(i)} := \prod_{j=1}^{m}(1 - Z_j.a)$, where Z_j is the sibling node of Y_j and

$$a = \begin{cases} t, & \text{if } Y_{j-1} \text{ is an OR-node,} \\ f, & \text{if } Y_{j-1} \text{ is an AND-node} \end{cases}$$

7: $X_{\alpha(i)}.t = p_{\alpha(i)}$
8: $X_{\alpha(i)}.f = 1 - p_{\alpha(i)}$
9: Update the parameters of the nodes $Y_{m-1}, Y_{m-2}, \ldots, Y_0$ according to formulae (5)–(8)
10: **end for**

Figure 10. Algorithm 2 of the Serial model

Similarly to the parallel model described in the previous section, a major drawback of the serial model is the immense increase in computational complexity.

3.3 Constructing the attack trees

There is no general agreement on what the primary threat should be for online voting as various researchers label the root attack differently. The author, himself, is most inclined towards the categorization proposed in [17], which will also be used in this thesis. Following this classification, three primary attacks are identified:

Manipulation Attack attempts to change the outcome of the elections by altering the voting results. In the context of the present thesis, this refers to effectively changing a certain number of votes in favour of the attacker's candidate of preference without the attack being discovered.

Revocation Attack attempts to change the election outcome by cancelling unsuitable voting results. In the context of the thesis, this attack assumes that a proportion of the I-voters will to vote again on the Election Day after the I-voting results are revoked. This attack benefits those parties, who receive proportionally less I-votes than regular votes.

Reputation Attack attempts to decrease voter confidence in the voting method and discredit Internet voting as a whole. As a result, a fraction of the people not willing to vote online, might

not vote at all. Additionally, this attack may assume that vote distribution is different among the potential voters and I-voters.

This thesis aims to focus on the Manipulation and Revocation attacks, and more specifically on potential malware attacks. The concrete attack trees corresponding to the primary threats are described in more detail in the following sub-sections (3.3.1 to 3.3.3).

In order to assign concrete values to the parameters, the author used the help of various cyber-related acquaintances (friends and lectors), concrete computations, extensive Internet research and in some cases also intuition. Some of the values were also determined with a reference to the values given in [354]. At times, the author also rounded some of the values to get even numbers. When assigning intuitive values it is reasonable to follow at least some sort of a methodical approach. Instead of directly coming up with a numeric value for the probability of the attack, it is easier to divide the attacks into verbally labelled groups that correspond to the likelihood of a given attack. The author divided the attacks into six verbal groups and assigned each of them a range of numeric values (Table 4).

Table 4. Verbally labelled groups of attack probabilities

Likelihood	**Probability range**
Unrealistic	$p < 0.001$
Very unlikely	$0.001 < p \leq 0.05$
Unlikely	$0.05 < p \leq 0.1$
Feasible	$0.1 < p \leq 0.5$
Likely	$0.5 < p < 0.8$
Highly likely	$p \geq 0.8$

This way, the specific value to the attack is assigned in two steps. First, the attack is assigned to one of the six verbally labelled groups and then a numeric value from the corresponding range is determined.

For assigning the parameter values for potential penalties, the following approach was adopted. Since there is no empirical data that would suggest what the potential penalties would be in case of a large-scale election fraud in Estonia, it is very difficult to assess the numeric values. The actual penalties are likely to depend on many factors, such as the volume and severity of the act committed, the competence of legal representation, and other factors. In order to come up with some estimates, the attacks in this thesis are compared to the paragraphs of the Estonian

Penal Code and maximum penalties are used in computations[5]. This should give us rough estimates of what the penalties would actually be. Since most of the penalties for the attacks described in the thesis include actual prison time, we have to calculate that time into money. In order to get the base amount, we can use the (normal) hourly income of the attacker. Assuming that normally (with a legal job) the attacker would make €15 per hour, we can compute the base cost of spending one year in prison as $15 \cdot 24 \cdot 365 = €131\,400$. However, we must also consider the cost of legal representation, protracted court cases, the possible involvement of multiple people as well as long-term loss of trust. Therefore, the base amount is multiplied with a factor of 5. Five years in prison, for instance, would then be equal to $131\,400 \cdot 5 \cdot 5 = €3\,285\,000$. Since we have to consider two types of potential penalties, namely when the attack is successful and when it is not, the author multiples the corresponding values with factors of 0.8 and 0.2 respectively. Therefore, five years of imprisonment would make $Penalties^{+} = 3285000 \cdot 0.8 = €2\,628\,000$ and $Penalties^{-} = 3285000 \cdot 0.2 = €657\,000$.

Due to length constraints and the main focus of the thesis, the parameter evaluation process is described in detail only for the Malware sub-attack of the Manipulation Attack. The process is similar for all other attacks.

3.3.1 Manipulation Attack

Manipulation Attack assumes that the attacker is able to effectively change a certain number of votes without the attack being discovered. Here, this number is set to 5000 votes[6]. As for the expected gains, the authors of [28] used a value of 100 million Estonian kroons. Due to appreciation of everyday life, let us assume this value to be risen to 10 million Euros.

Successful Manipulation Attack can be carried out either by attacking the voter and her computer (and mobile) environment, or by attacking the Central System (Figure 11).

[5] The justification for taking the maximum sentences is that the paragraphs in the Penal Code define maximum penalties, but do not specify the extent of the crimes. For example, strictly according to the paragraphs, the maximum penalty for infecting one computer and infecting 10000 computers is the same. The author considers large-scale election fraud as one of the most severe crimes and therefore uses the maximum penalty values.

[6] The number was chosen, because it is assured to gain one additional seat in the Parliament with approximately 5000 votes. Also, changing 5000 votes will certainly fall into a category of large-scale election fraud.

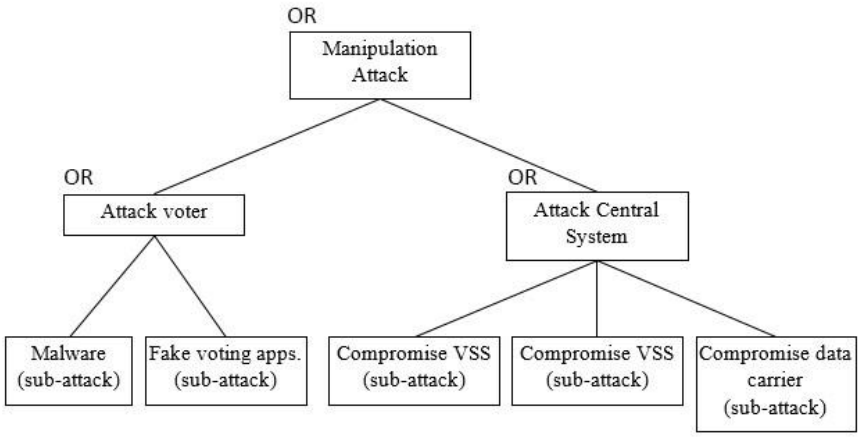

Figure 11. Attack tree for the Manipulation Attack

There are two ways to attack the voter. The attacker can either develop and distribute malware that is capable of modifying the original vote, or develop fake voting applications and get enough voters to download them. As for attacking the Central System, the attacker can either compromise the Vote Storage Server, the Vote Counting Application or the data carrier that is used to transfer the list of votes from VSS to VCA. The full attack tree for the Manipulation Attack is presented in Appendix 1.

The author lists altogether three types of possible malware. Each of them must be able to effectively modify 5000 votes in favour of attacker's candidate of preference. The basic tree structure is the same for all three malware types, consisting of three sub-nodes: *development*, *distribution* and *avoiding detection*. The first two are self-explanatory, the last one is necessary to influence the overall success probabilities of each malware attack vector (described in more detail below).

The probability of a successful development process of the malware is assumed to be 95% irrespectively from the malware type, but as the complexity level of each type varies, the development costs are different for each case. The probabilities of getting caught for developing malware is assumed to be 5% in all cases.

After the developing process, the attacker needs to distribute the malware to voters' computers (and mobile devices), where it can start changing the behaviour of the application(s). Since online voting period only lasts for seven days, it is not practical for the attacker to start addressing the distribution issues only after the final product of malware is finished. It is assumed that the attacker distributes the voting malware during the online voting period via previously obtained botnet. Botnet is built by the attacker herself or purchased from external

parties. Here, *buying botnet* does not presume an already existing botnet in Estonia, but rather that someone is willing to build and sell it to the attacker. It is assumed that the attacker is significantly less likely to be caught if she buys the botnet instead of creating it herself. The respective probabilities in case the attack is successful are assumed to be 5% (buy) and 50% (create). In case the attack is not successful, the respective probabilities are 1% and 10%. These values are the same for each type of malware.

According to [15], a suggested setup for building a botnet is expected to cost $595 (~€430) for the first month of operations and an additional $225 (~€160) per month for sustaining the operations. In order to calculate the cost of the botnet needed for a specific type of voting malware, the author assumes that 1500 machines are added monthly to the botnet on average. The author also assumes it takes the attacker about one month to research the subject and establish the basis of the botnet. For additional months, it is assumed it takes the attacker 40 hours of work per month to sustain the operation. The cost of creating a botnet can be then calculated as $cost = 430 + 160 \cdot 50 + \frac{N}{1500} \cdot (160 + 40 \cdot 50)$, where N denotes the number of required machines in the botnet.

For the cost of buying a botnet, the author refers to [29], which states that a botnet with 100 000 nodes was put on sale with $36000 (~€26000). For the sake of simplicity, the author assumes that the relation between the size of the botnet and its cost is linear. Assuming it takes approximately one month to find the right seller, the cost of the botnet for each type of voting malware can be computes as $cost = \frac{N}{100000} \cdot 26000 + 20 \cdot 8 \cdot 50$, where N denotes the number of required machines in the botnet.

The attacker is also required to take steps to ensure the malware will not be discovered, as the act of large-scale vote manipulation must remain secret. As the detection probability varies from the type of malware used, this is regulated with the node *Avoid detection* in the attack tree and it contributes to the success probability of the overall sub-attack.

As for potential penalties, according to §216[1] of the Estonian Penal Code [11], development of malicious malware for the purposes of carrying out a computer-related crime is punishable up to three years of imprisonment. For the distribution penalties, in addition to §216[1], §208 also applies, which states that the dissemination of spyware, malware or computer viruses is punishable up to five years of imprisonment when a significant damage is caused. We must also include §163, which states that causing the destruction, damaging, elimination or falsification

of election or voting documents, or incorrect counting of votes is punishable up to one year of imprisonment. Therefore, for developing such malware, the penalties are $Penalties^{+} = 0.8 \cdot 3 \cdot 657000 = €1\,576\,800$ and $Penalties^{-} = 0.2 \cdot 3 \cdot 657000 = €394\,200$. For distributing such malware, the penalties are $Penalties^{+} = 0.8 \cdot (3 + 5 + 1) \cdot 657000 = €4\,730\,400$ and $Penalties^{-} = 0.2 \cdot (3 + 5 + 1) \cdot 657000 = €1\,182\,600$. It is assumed that the same potential penalties apply for each type of malware.

The three types of malware are listed in Table 5.

Table 5. Types of malware

Malware type	**Vote Modifying Malware**	**Re-voting Malware**	**Self-voting Malware**
Abbreviation	VMM (VCM/VBM)	RVM	SVM
Vote modification type	Change/block	Change	Change
Vote modification instant	During the legitimate voting process	After the legitimate voting process	Any time during the online voting period
Official voting application required	YES	YES	NO
Official verification application required	YES	NO	NO
Learning of the PIN-codes of ID-card	Not needed	During the voting process	Before and during the voting period

Vote Modification Malware (VMM) activates itself when the victim launches the voting application and starts the voting process. It must be able to bypass the verification protocol, meaning that both the voters' computer and her mobile device must be infected. In order to know, which vote to "verify", the infected voting application must send the corresponding information to the infected verification application.

When the voter confirms her choice, the malware either changes the vote (VCM) or blocks it entirely (VBM). The latter one can be considered slightly easier (i.e. less costly) to develop as the application does not have to actually send the vote to the Central System, and blocking the vote can be done as simply as cutting off the connection to the VFS. On the other hand, it is also less effective as it would require significantly more computers to be infected in order to produce effectively similar results. For example, let us propose a situation, where candidate A receives 100 votes, while candidate B receives only 40. In order to alter the I-voting result in such a way that A and B would receive the same number of votes, it is necessary to block 60

votes for candidate A. However, if we would change the votes (from A to B) instead of blocking them, it is only necessary to modify 30 votes.

In real life, of course, there are several different candidates and the number of votes required to be blocked depends on the number of votes a certain party received. In order to produce similar results as changing 5000 votes in favour of a certain party, the necessary number of blocked votes can be computed with the following formula:

$$V_{block} = N - \frac{V_p}{V_{p+}} \cdot N$$

Here, N denotes the number of total I-votes, V_p denotes the number of votes a certain party p received and V_{p+} denotes the number of votes party p received plus 5000 votes. In the Parliamentary elections of 2011, there were four parties that obtained seats in the Parliament [38]. Table 6 shows the number and percentage of I-votes each party received. It also shows how many votes need to be blocked for different parties in order to achieve similar results as changing 5000 votes.

Table 6. Required number of blocked votes to produce similar results as changing the votes

Party	**RE**	**IRL**	**SDE**	**KE**
Actual results (votes)	52015	35735	25332	13892
Actual results (%)	36,95	25,39	18,00	9,87
Results with 5000 changed votes in favour of the party (votes)	57015	40735	30332	18892
Results with 5000 changed votes in favour of the party (%)	40,50	28,94	21,55	13,42
Number of votes required to be blocked in order to achieve the same results as changing 5000 votes	12344	17278	23204	37255

As it can be seen from Table 6, for each party, the attacker must block at least twice as many votes to achieve the effect of 5000 changed votes. The Central Party (KE), for instance, would need to infect more than seven times as many devices using Vote Blocking Malware instead of

Vote Changing Malware. Therefore, changing the votes instead of blocking them, proves to be significantly more effective in terms of altering the results of the election.[7]

According to the author of the proof-of-concept malware in 2011 (see Sub-section 2.3.2 for more details), developing such a malware is very simple and it only takes a few hours [32][8]. However, the aforementioned malware did not handle the situation, where the voter can verify her vote via alternative channels nor did it take any measures to avoid getting detected. In order to successfully carry out the Manipulation Attack, the developed malware needs to address both of these issues.

The author of this thesis assumes that a skilled attacker is able to finish developing malware before the third day of the I-voting period. Here, it must be noted that the attacker can make preparations and start code writing before the official voting applications are made available. Let us assume that the attacker begins researching the topic and writing preliminary code four weeks before the I-voting period. This means 8 hours of work for 20 days. Additionally, let us assume that on the first two days of the I-voting period, the attacker works with double workload. This makes the development costs to be $cost = (20 + 2 \cdot 2) \cdot 8 \cdot 50 = €9600$. Here (and throughout the thesis), the hourly rate of the attacker is assumed to be €50. As mentioned before, developing malware that simply blocks the vote might be considered less costly. Hence, 20% of the computed cost is deducted for the Vote Blocking Malware.

Given that the malware has only five days to modify enough votes, let us compute, how many computers should be infected in order to change 5000 votes. Let x be the number of devices necessary to infect in order to change 5000 votes. At first glance, it is tempting to set $x = 5000$. However, in order to change 5000 votes, it is not sufficient to infect only 5000 devices, as there are many factors that must be taken into account. First of all, it is reasonable to assume that some of the infected machines belong to citizens that are not eligible to vote and therefore their computers are not used to cast a vote (at least by themselves). Let us suppose that out of the infected machines, 90% belong to eligible voters. Secondly, it can be assumed that among eligible voters, there are those, who use other voting methods or do not vote at all. The voter turnout for the Parliamentary elections in 2011 was 63,5%, out of which 24,3% voted online (Table 1). The above will contribute to x with factors 0.9, 0.635 and 0.243. Thirdly, it does not

[7] Here, only I-votes are considered. If we were to consider paper votes as well, the required number of blocked votes would be significantly higher.
[8] The author is inclined to consider it as a slight understatement.

make sense to consider votes that are already cast in favour of the attacker's preference. Let us assume that party A would normally receive 10% of the overall Internet votes. This makes the percentage of suitable votes (i.e. votes that are to be changed) to be 90% and it will contribute to x with the factor of 0.9. Fourthly, as the attacker must infect both the computers and mobile devices of the voters', it is reasonable to assume that the pools for infected computers and infected mobile devices will likely differ from each other. Let us suppose that they overlap by 70%, therefore contributing to x with the factor of 0.7. Finally, it is also possible that the infected machine was used to cast the vote before the infection took place. Since it was previously stated that the author assumes the actual vote modification will take place only in the last five days of the I-voting period, it contributes to x with the factor of $\frac{5}{7}$. Of course, it must also be noted that several people may use the same computer to cast the vote. The author assumes that there are approximately 1,5 people per computer.[9] Following this reasoning, we can compute x as following:

$$x = \frac{5000 \cdot 7}{0.9 \cdot 0.635 \cdot 0.243 \cdot 0.7 \cdot 0.9 \cdot 5 \cdot 1.5} \approx 53\,000$$

Therefore, in order to change 5000 votes over the period of five days, the attacker would need to infect about 53000 computers. From this, a general formula can be derived to calculate the relation between the number of days the malware is active and the required number of infected machines:

$$x = \frac{N \cdot P}{\prod_{i \in \mathcal{I}} f_i \cdot d}$$

Here, x denotes the required number of infected machines, N denotes the number of votes that are to be changed, P denotes the I-voting period, $\prod_{i \in \mathcal{I}} f_i$ denotes the product of different factors that affect the number of required infected machines, such as percentage of eligible voters or voter turnout (different factors affect different types of malware) and d denotes the number of days the malware is active.

Figure 12, illustrates this relation visually for each type of malware.

[9] There are no statistics available by the Statistics Estonia for the average number of computers per household in Estonia. However, it is known that the average size of a household is 2,3 people [A39]. Considering that not every person of a household is allowed to vote and as there are households with several computers, the author concludes a rough estimate of 1,5 to be used in the calculations.

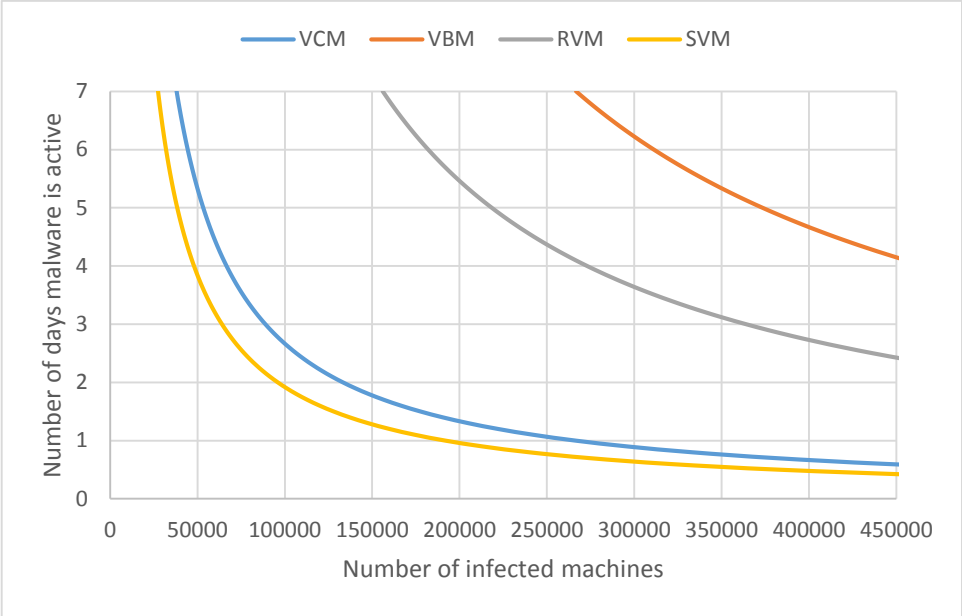

Figure 12. Relation between required number of infected machines and malware's operating period

As seen from Figure 12, following the same computational logic, in order to produce the same results with Vote Blocking Malware, the attacker would need to infect about 40% of the entire electorate. Considering that compromising mobile devices requires similar number of infections, it seems that distributing malware on such a scale without detection is unrealistic in the Estonian context. The author assumes the probability of infecting 53 000 computers is about 5% and the probability of infecting the same number of mobile devices 0,1%. According to the assumptions made in Section 3.3 between the size of the botnet and its price, the cost of creating such a botnet is $cost = 8430 + \frac{53000-1500}{1500} \cdot 2160 \approx €82\,500$ and the cost of buying such botnet is $cost = \frac{53000}{100000} \cdot 26000 + 8000 \approx €22\,000$.

Although the success probability of distribution also depends on the type of the malware (Vote Blocking Malware requires more infections than Vote Changing Malware), it is not actually necessary to produce sub-trees for the mobile and PC malware distribution processes. Since we are using the example of party A, it is necessary to infect approximately seven times as many devices and as the success probability of distributing VCM is already rather low, the probability of distributing VBM is effectively close to zero.[10] Instead of duplicating the attacks in the

[10] Here, the relation between the size of the botnet and the probability of infecting additional machines is assumed to resemble a logistic sigmoid function. This means that the larger the botnet is, the more difficult it is to gain additional bots.

malware *distribution* branch, we can produce similar result by lowering the success probability of development costs of VBM since they are connected with an AND-node.

As for the detection probability, NEC has in its disposal the help of CERT-EE along with the help of Estonian Cyber Defence League and a network of volunteers who monitor the Internet activity with a peaked interest during the voting period. The author assumes that a large-scale network of infected machines is detected with a probability of 95%.[11]

Re-voting Malware activates itself when the victim launches the voting application and starts the voting process (similarly to VMM). However, RVM does not immediately change the vote, but merely records the PIN-codes of user's ID-card. Sometime later, when the voter re-inserts her ID-card into the computer, the malware secretly casts another vote in the background. Since the victim is not aware of this process, the malicious vote is not verified by the user.

It is also possible that the victim leaves her ID-card into the computer after the voting process. In this case, the malware would wait for a certain period of time (e.g. 30 minutes) to allow the user to verify her original vote and only then would cast a new vote. If the malware would vote again immediately after the voting process, the verification of the original vote could fail and thus compromise the attack. RVM is similar to the malware proposed by a group of experts in the security analysis of the Estonian I-voting scheme in [A40].

The benefit of RVM is that the attacker does not need to infect mobile devices and can, therefore, cut back on the development costs. Still, the RVM is likely to be slightly more complicated than the PC part of the VCM. Assuming that RVM is 40% more costly to develop than the PC part of VCM, which in turn forms 50% of the overall development costs of VCM, the cost of RVM is $cost = \frac{9600}{2} \cdot (1 + 0{,}4) = €6720$.

Computing the approximate number of machines required to be infected in order to change 5000 votes is similar as shown above, although there are some differences. As there is no need to infect mobile devices, the factor of $0{,}7$ does not contribute to x. There is, however, the need for the ID-card to be in the computer, either by the voter re-inserting it during the voting period, or simply by not removing it after the voting process. Let us suppose that 10% of the people re-uses the ID-card sometime later during the voting period, and 10% will not remove the card

[11] It must be noted that the botnet would have to avoid large-scale detection throughout the infection period.

from computer after casting the vote.[12] As these percentages may overlap, this contributes to the x with the factor of $0{,}1 + 0{,}1 - 0{,}1 \cdot 0{,}1 = 0{,}19$. Additionally, there are bound to be people who will use Mobile-ID for authentication and digital signing, ruling out the possibility of malware learning the PIN-codes from the ID-card. In 2011, 1,9% of the voters used Mobile-ID (Table 1). Since the Parliamentary Elections of 2011 was the first time when Mobile-ID could be used in voting, the percentage of users was expectedly low. In the European Parliamentary elections of 2014, this percentage was already as high as 11% (Table 1) and the author expects it to be at least 10% for the Parliamentary Elections of 2015. Therefore, the infected machines required to change 5000 votes, can be computed as:

$$x = \frac{5000 \cdot 7}{0.9 \cdot 0.635 \cdot 0.243 \cdot 0.9 \cdot 5 \cdot 1.5 \cdot 0.19 \cdot 0.9} \approx 218\,000$$

Infecting more than 200 000 machines without large-scale detection is arguably unrealistic in the context of Estonia. Since it was previously assumed that the probability of infecting 53 000 machines is about 5%, the probability of infecting 218 000 machines is presumably less than 0,05%.[13] The cost of creating such botnet is $cost = 8430 + \frac{218000-1500}{1500} \cdot 2160 \approx €320\,000$ and the cost of buying such botnet is $cost = \frac{218000}{100000} \cdot 26000 + 8000 \approx €65\,000$.

Self-voting Malware (SVM) activates itself as soon as the computer becomes part of the botnet and starts collecting the PIN-codes of voters' ID-cards. SVM does not require for the official Voter Application to be downloaded to the voter's computer, as it is capable of casting a vote without it. During the online voting period, the malware uses previously collected PIN-codes of ID-cards to cast a malicious vote secretly in the background without the knowledge of the voter. Since it is not required to modify an existing program, the development costs of SVM can be considered lower than the other types of malware. The author assumes it costs 50% less than the PC part of VCM, which makes the development costs to be $costs = \frac{9600}{2} \cdot (1 - 0{,}5) = €2400$.

Since the Voting Application is not required, the necessary number of infected machines does not depend on the overall voter turnout nor on the percentage of I-voters. The required number is affected by the eligible voters (factor of 0,9), suitable votes (0,9), the number of average people per computer (1,5), the percentage of PIN-codes the malware learns before and during

[12] During the 30 minute window.
[13] Again, the relation is presumed to resemble a logistic sigmoid function.

the voting period (the author assumes the factor to be 0,6) and the percentage of voters using the ID-card during the voting period (assuming 25% of people using the ID-card for Internet voting or for other purposes during the election period). This makes the necessary number of infected machines to be:

$$x = \frac{5000}{0.9 \cdot 0.9 \cdot 1.5 \cdot 0.6 \cdot 0.25} \approx 27\,000$$

The author assumes that the probability of creating a botnet of such magnitude is 30%. The cost of creating such botnet is $cost = 8430 + \frac{27000-1500}{1500} \cdot 2160 \approx €45\,000$ and the cost of buying such botnet is $cost = \frac{27000}{100000} \cdot 26000 + 8000 \approx €15\,000$.

As it follows, SVM is less costly to develop and more feasible to distribute than VMM and RVM. However, the probability of detection is virtually 100%. First of all, the malware does not address the situation where the voter tries to cast her first vote after the malware has already voted. Since the Voter Application is not modified, the voter would be informed of a previously cast vote. This would indicate that someone else voted on behalf of the voter. Secondly, and more importantly, people who cast their vote over the internet (intentionally or otherwise) are not allowed to vote on the Election Day. Since the malware also votes on behalf of those people who specifically plan to vote on the Election Day, the act of foul play is certainly discovered. In 2011, 56.9% of the voters cast their vote on the Election Day [38]. Therefore, SVM is not suitable for carrying out the Manipulation Attack. It is, however, quite useful in the Revocation Attack (see Sub-section 3.3.2).

As it can be seen from Figure 11, the Manipulation Attack can be also carried out by other attacks besides malware:

Fake voting applications refers to the attacker developing fake versions of both voting applications as well as getting the voters to download them. As previously stated, changing the votes is much more effective than simply blocking them. Therefore, it is assumed that the function of the fake Voting Application is to change votes instead of blocking them. For getting the voters to download the fake Voting Application, the attacker can either upload the application to a fake website or replace the official application on the NEC website. The author considers the latter one less likely. Although the official website is bound to have more visitors, it can be assumed that the fact it is compromised, would be discovered relatively quickly. For

getting the voters to download the fake Verification Application, the attacker needs to upload the fake application to the official app stores or to other markets. Since the official app store already contains the real application, the attacker needs to replace it or upload a similar one, hoping that at least some of the voters would use the fake one instead of the original.

Compromising VSS refers to changing the list of votes in the Vote Storage Server. The attacker would need to develop malicious code that is able to alter the stored votes. Here, developing malicious code may refer to writing code that would change the votes in the list, or simply replacing the list of votes with another one. This is expected to be done directly before the votes are transferred to VCA. In order to insert the code into the server, the attacker has the options of bribing either the software or the server administrator, and hacking into the server. According to [354], 0,33% of the people are corruptible with approximately 30 000 Euros. However, in order for the whole attack to succeed, the act of vote modification must remain secret. The author considers the likelihood of that to be very unlikely since there are proper audit measures in place. Therefore, the probability of a successful bribing attack (as well as several other Central System attacks for the same reason) is multiplied with a factor of 0,01. As for hacking into the server, the attacker needs to get access to the internal network first as VSS is not available from the outside web.

Compromising VCA refers to using malicious code to change the vote counting algorithm or compromising the counting results in any other way. The attack vector is similar to the attack vector of compromising VSS with the exception that the attacker is not able to hack into the VCA from the web, since it is offline at all times.

Compromising data carrier refers to modifying the voter list in the transferring phase from VSS to VCA. The attacker would need to create an alternative list of votes and write them to the fake transportation device. The attacker would then need to switch the original transportation device with the fake one, which can be done by bribing a NEC's worker or by infiltrating as a participant of the elections (such as an employee or an observer).

3.3.2 Revocation Attack

The basic idea behind the Revocation Attack is to get the I-voting results revoked and hope that a proportion of I-voter would not vote again on the Election Day. This attack only makes sense if the preferred candidate receives proportionally less I-votes than regular votes. For example, in the Parliamentary Elections of 2011, the Central Party (KE) received 23,32% of the total

votes, but only 9,87% of the I-votes [38]. Since the percentage of I-votes is lower than the percentage of votes received in total, the Central Party could possibly benefit from the revocation of I-votes. Let us suppose that the I-voting results are revoked and 10% of the I-voters will not vote again on the Election Day. Table 7 shows the gain in percentage of total votes for each party if the I-votes would have been revoked and 10% of I-voters would not have voted again on the Election Day.

Table 7. Potential effects of the Revocation Attack on the election results of 2011

Party	**RE**	**IRL**	**SDE**	**KE**
Total votes	164255	118023	98307	134124
Total votes (%)	28,56	20,52	17,09	23,32
I-votes (%)	36,95	25,39	18,00	9,87
Total votes with 10% of I-voters not voting again after the revocation of I-votes (%)	28,35	20,40	17,07	23,66
Gain in percentage of total votes (percentage points)	-0,21	-0,12	-0,02	0,34

As seen from the table above, the Central Party would have seen an 0,34 percentage point increase in the received votes from total votes. In order to achieve a similar percentage of the total votes with the Manipulation Attack, the party would have change $134124 \cdot (\frac{23,66}{23,32} - 1) \approx 2000$ votes. As this forms 40% from the 5000 votes required in the Manipulation Attack, it is assumed that $Gains_{Revocation} = 0,4 \cdot Gains_{Manipulation}$. Figure 13 illustrates the relationship between the I-voters who would not vote again and the total votes received by a party. This data is also based on the Parliamentary Elections of 2011.

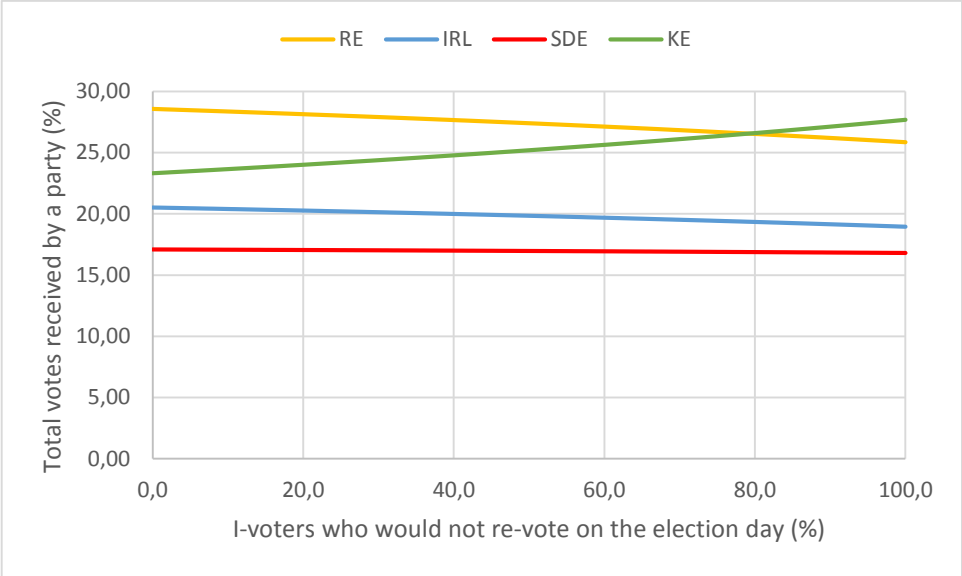

Figure 13. Relation between missed out re-votes and total votes received by party (2011)

As it can be seen from Figure 13, in order for the Central Party to get the same results as the Reform Party, it would take approximately 80% of I-voters not voting again after the revocation of the I-votes. Getting 80% of I-voters not to re-vote might prove to be infeasible to accomplish in real life, but it does show that a successful Revocation Attack might have a quite significant impact on the voting results. It must be noted, however, that this data assumes the I-votes are revoked after the online voting period (but before the Election Day). It is very likely that the revocation of I-votes would take place sometime during the election period (in which case the gain in percentage of total votes would be lower), but since it is very difficult to assess the exact moment, the author uses the worst case scenario.

Revocation Attack proposed in this thesis consists of conducting a violation of the election rules and persuading NEC to revoke the election results. As the successfulness of the latter depends on the extent of the violation, it is listed for each of the major sub-attacks in the attack tree.

There are several options to conduct the violation of the election rules. The attacker can attack either the integrity or availability of the system, or the privacy of the voters. The attack vector for attacking the integrity of the system is similar to the Manipulation Attack. The difference is that the attacker does not need to address the issues of vote verification and the attack being discovered. Therefore, the corresponding nodes have been excluded from the attack tree. This also makes the parameter values (costs and success probabilities) more favourable to the attacker. Here, it is also assumed that the attacker is required to change 1000 votes instead of 5000 in order to have a valid claim to have the Internet votes revoked. In addition to the

aforementioned attacks, there is also a possibility of compromising the candidate list in the VFS. This was excluded from the Manipulation Attack, because the detection probability is extremely high.[14]

For attacking the privacy of the voters and the secrecy of the results, the attacker needs to get the votes from the VSS and decrypt them. The attack vector for getting the votes from the VSS is similar to the Compromising VSS Attack (see previous section). In order to decrypt the votes, the attacker needs to obtain the private key of the server or using cryptanalysis to breach the cryptographic security measures. However, even if the attacker is able to steal and decrypt the votes, it may not be enough to get the I-voting results revoked, as the votes themselves were not altered. Therefore, the probability of convincing NEC is rather low.

The attacker can also attack the availability of Internet voting by performing distributed denial of service (DDoS) on the NEC's servers. The author assumes that the servers must be overloaded during at least 80% of the online voting period in order to have any sort of a claim to get the results revoked. As with previous attack, this may also not be enough to convince NEC. The full attack tree for the Revocation Attack is presented in Appendices 2 and 3.

3.3.3 Reputation Attack

Reputation Attack in this thesis consists of discrediting the voting method by launching a public campaign against I-voting to attack the voters' confidence (legal), or by attacking the availability of the system with the hope of people losing interest in casting their vote via Internet (illegal).

In order to organize a meaningful anti-campaign, the trustworthiness of the system must be discredited. This can be done by pointing out problems with previous elections, finding contradictions with the law, or by developing a proof-of concept attack. All of these claims might have an effect on people's attitude towards Internet voting. As soon as the trustworthiness of the system has been discredited, the attacker would need to gain public's attention by advertising against I-voting. This can be done via internet or by physical media (e.g. leaflets or billboards). The attacker could also organize public events such as public lectures and seminars. In order for the attack to have a substantial effect, the attacker would also need to gain support

[14] The voters are very likely to immediately detect that the candidate list is not a valid one.

of IT security related experts, public figures or general public. The full attack tree for Reputation Attack is presented in Appendix 4.

As previously stated in Section 3.3, one of the assumptions of the Reputation Attack can be that vote distribution is different among the potential voters and I-voters. However, this assumption has never been proven and the I-voter cannot be distinguished from the regular voter with statistical means [41]. Although Table 7 clearly shows that the distribution of I-votes is different from the distribution of regular votes, it only suggests that Internet voting is more favourable among voters of candidate A than among voters of candidate B. This, however, makes no indications whether candidate A would receive more votes in total as people who I-vote merely change their voting method, not their political preference.

Although I-voting itself does not give any advantages to a specific party, it is possible that voters whose confidence in the voting method has been decreased may not vote online, and as a consequence, may not vote at all. However, it is not clear how it would affect the election results, as the attack is likely to influence voters of different candidates differently. If I-voting is more popular among voters of candidate A, they might be less susceptible to this attack than voters of candidate B. As there is no statistical data that would suggest how an anti-campaign could affect the outcome of the elections, it is very difficult to assess the parameters of the attack tree. Another possible aim of the attack could be to achieve cancelling of I-voting entirely for future elections, making it similar to the Revocation Attack. The difference with Reputation Attack, however, is that voters would also be able to cast their vote during the advance voting period. Therefore, the effect of the Reputation Attack would likely to be lower.

The effect of the anti-campaign is also likely to be largely dependent on the financial investment. In order to achieve better results, the more costly the attack is going to be. Therefore, the parameters of the Reputation Attack would require function values. Unfortunately, this is something the contemporary attack tree models are not capable of and thus, the author decided to exclude the exact computations for the Reputation Attack from this thesis.

3.4 Simulation and results

For the simulations of BLPSW model, the author used an application of CoCoViLa, which itself is a model-based software development platform [5]. For the Parallel and Serial models, the author used a prototype computer tool [1, 2]. As stated in Sections 3.2.2 and 3.2.3, the parallel

and serial computational models proposed in [22] and [21] respectively cannot be used in practical analysis of trees with large number of leaf nodes.[15] Therefore, it becomes necessary to divide the attacks proposed in Section 3.3 into smaller chunks. Using all three models, the author analysed each of the major sub-attack separately from each other. For the Manipulation Attack, this includes each type of the Malware attack, Fake Voting Application attack and each attack against the Central System. For the Revocation Attack, the attacks are the same as for the Manipulation Attack with addition of attacks against privacy and availability. Using the BLPSW and Parallel models, the author analysed Malware and Central System attacks as wholes. The entire attack trees (Manipulation and Revocation) are analysed only with the BLPSW model.

The computations for each attack suite and with each model for the Manipulation Attack were done in 20 steps, setting the value of $Gains$ from 1 000 000 to 20 000 000 with the increment of 1 000 000. The expected outcomes for Malware and Central System attacks with $Gains = \{1\ 000\ 000; 10\ 000\ 000; 20\ 000\ 000\}$ along with the best possible attack vector are presented in Table 8. Complete results for the Manipulation Attack are presented in Appendix 5.

Table 8. Computational results for Malware and Central System attacks (Manipulation Attack)

Attack	Identifier	AT Model	Outcome (Gains = 1M)	Outcome (Gains = 10M)	Outcome (Gains = 20M)	Best attack suite (Gains = 10M)
Vote modifying malware	M1.1.1	BLSPW	-195 275 €	-195 233 €	-195 185 €	ML1, ML4, ML6, ML7
Vote modifying malware	M1.1.1	Parallel	-68 281 €	-68 280 €	-68 280 €	ML1, ML4, ML6, ML7
Vote modifying malware	M1.1.1	Serial	-1 991 €	-1 910 €	-1 818 €	ML7, ML6, ML4, ML3, ML1
Re-voting malware	M1.1.2	BLSPW	-221 353 €	-220 925 €	-220 450 €	ML8, ML10, ML11
Re-voting malware	M1.1.2	Parallel	-151 400 €	-150 972 €	-150 497 €	ML8, ML10, ML11
Re-voting malware	M1.1.2	Serial	-6 841 €	-6 414 €	-5 939 €	ML11, ML10, ML8
Self-voting malware	M1.1.3	BLSPW	-169 500 €	-169 500 €	-169 500 €	ML12, ML14, ML15
Self-voting malware	M1.1.3	Parallel	-110 173 €	-110 173 €	-110 173 €	ML12, ML14, ML15
Self-voting malware	M1.1.3	Serial	0 €	0 €	0 €	-
Compromise VSS	M2.1	BLSPW	-2 660 790 €	-2 637 030 €	-2 610 630 €	ML31, ML32
Compromise VSS	M2.1	Parallel	-420 978 €	-397 218 €	-370 818 €	ML31, ML32
Compromise VSS	M2.1	Serial	-150 943 €	-150 941 €	-150 939 €	ML34, ML35, ML31
Compromise VCA	M2.2	BLSPW	-2 660 790 €	-2 637 030 €	-2 610 630 €	ML25, ML26
Compromise VCA	M2.2	Parallel	-420 978 €	-397 218 €	-370 818 €	ML25, ML26
Compromise VCA	M2.2	Serial	-349 225 €	-325 465 €	-299 065 €	ML25, ML26
Compromise data carrier	M2.3	BLSPW	-5 184 550 €	-5 181 580 €	-5 178 280 €	ML39, ML41
Compromise data carrier	M2.3	Parallel	-1 038 374 €	-1 035 404 €	-1 032 104 €	ML39, ML41
Compromise data carrier	M2.3	Serial	-353 542 €	-350 572 €	-347 272 €	ML41, ML39

[15] For example, using the serial model, the author let the computations for the Malware Attack (M1.1) ran for more than 8 hours before cancelling the process.

As it follows from Table 8 and Appendix 5, the expected outcomes for the Manipulation Attack were non-positive for each attack suite. This means that a rational (economically thinking) attacker would not try to undertake a large-scale vote manipulation attack. Although each attack suite proved to be unprofitable, the simulations show that attacking the voter and his or her computer environment proves to yield better results than attacking the Central System. This is mainly due to high probability of being detected and getting caught when trying to get access to the Central System. The results also show that using malware would likely achieve better results than using fake voting applications.

Figure 14 illustrates the comparison of expected outcomes according to the three computational models when using different malware types (VMM, RVM and SVM) to launch the Manipulation Attack with $Gains = €10\ 000\ 000$.

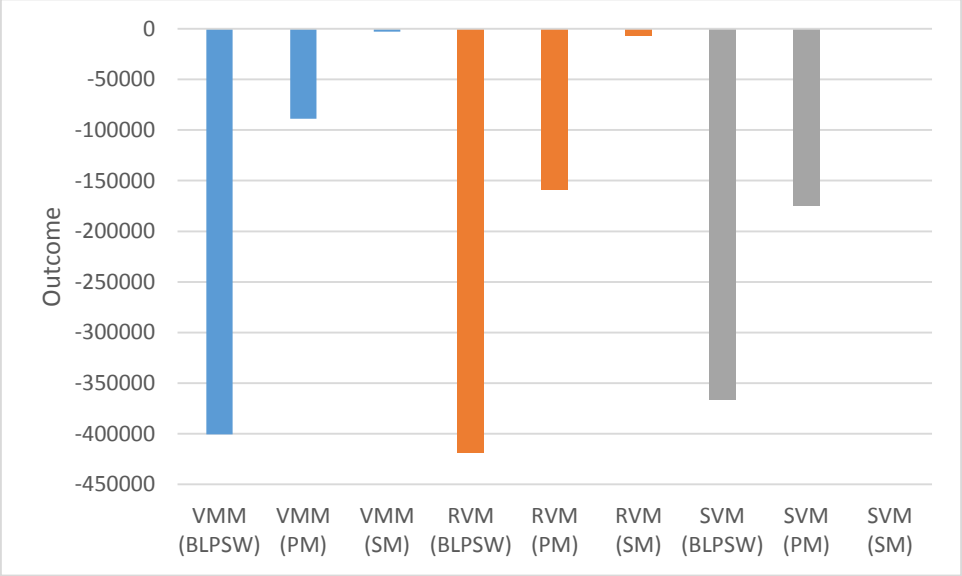

Figure 14. Computational results for Malware attacks by attack tree model (Manipulation Attack)

As it can be seen from the figure above, the models provide numerically very different results. The Serial model (SM) produces a higher outcome than the Parallel model (PM), which in turn gives a better result than the BLPSW model. This is a recurrent characteristic in the computations. The best outcome for the Manipulation Attack is achieved by carrying out the *Self-voting Malware attack (M1.1.3)*, which produces an outcome value of €0 using the serial model. This is because the probability of *Avoid detection* node is set to zero (the reasons for

this are presented in Section 3.3), which, according to the Serial model, renders any additional attacks pointless to launch.

For the Revocation Attack, the results are vastly different. With all three models and $Gains = €4M$, each malware attack is profitable. The best outcome is achieved by *Self-voting malware (R1.1.1.1.3 + R1.1.2)* attack, which produces the outcomes of €2 371 804 (BLPSW), €2 428 765 (Parallel) and €2 608 410 (Serial). This is also the best outcome of the entire *Revocation Attack (R)* using the BLPSW model. Similarly to Self-voting Malware, *Vote modifying malware (R1.1.1.1.1 + R1.1.2)* attack also produces very profitable results with the expected outcome values being €1 459 764 (BLPSW), €1 586 011 (Parallel) and €2 020 150 (Serial). (Table 9)

Table 9. Computational results for Malware and Central System attacks (Revocation Attack)

Attack	Identifier	AT Model	Outcome (Gains = 1M)	Outcome (Gains = 4M)	Outcome (Gains = 10M)	Best attack suite (Gains = 40M)
Vote modifying malware	R1.1.1.1.1 + R1.1.2	BLSPW	91 764 €	1 459 764 €	4 195 764 €	RL2, RL4, RL19
Vote modifying malware	R1.1.1.1.1 + R1.1.2	Parallel	212 294 €	1 586 011 €	4 886 087 €	RL2, RL3, RL4, RL19
Vote modifying malware	R1.1.1.1.1 + R1.1.2	Serial	309 882 €	2 020 150 €	6 042 070 €	RL19, RL4, RL3, RL2, RL1
Re-voting malware	R1.1.1.1.2 + R1.1.2	BLSPW	-315 316 €	-132 916 €	231 884 €	RL5, RL7, RL19
Re-voting malware	R1.1.1.1.2 + R1.1.2	Parallel	-73 605 €	108 795 €	665 940 €	RL5, RL7, RL19
Re-voting malware	R1.1.1.1.2 + R1.1.2	Serial	13 472 €	195 872 €	845 045 €	RL19, RL7, RL5
Self-voting malware	R1.1.1.1.3 + R1.1.2	BLSPW	319 804 €	2 371 804 €	6 475 804 €	RL8, RL10, RL19
Self-voting malware	R1.1.1.1.3 + R1.1.2	Parallel	376 765 €	2 428 765 €	6 532 765 €	RL8, RL10, RL19
Self-voting malware	R1.1.1.1.3 + R1.1.2	Serial	446 555 €	2 608 410 €	7 122 810 €	RL19, RL8, RL10, RL9
Compromise VFS	R1.2.1	BLSPW	-2 603 760 €	-2 600 010 €	-2 592 510 €	RL20, RL23, RL24
Compromise VFS	R1.2.1	Parallel	-207 520 €	-203 770 €	-196 269 €	RL20, RL23, RL24
Compromise VFS	R1.2.1	Serial	-68 511 €	112 582 €	939 232 €	RL24, RL20, RL21, RL22
Compromise VSS	R1.2.2	BLSPW	-2 509 680 €	-2 039 430 €	-1 098 930 €	RL25, RL26, RL30
Compromise VSS	R1.2.2	Parallel	-877 298 €	-407 048 €	613 904 €	RL25, RL26, RL30
Compromise VSS	R1.2.2	Serial	-130 333 €	692 894 €	2 903 210 €	RL30, RL25, RL28, RL26, RL27
Compromise VCA	R1.2.3	BLSPW	-2 509 680 €	-2 039 430 €	-1 098 930 €	RL31, RL32, RL34
Compromise VCA	R1.2.3	Parallel	-877 298 €	-407 048 €	613 904 €	RL31, RL32, RL34
Compromise VCA	R1.2.3	Serial	-365 103 €	216 606 €	1 787 240 €	RL34, RL31, RL32, RL33
Attack privacy/secrecy	R2	BLSPW	-4 682 835 €	-4 666 500 €	-4 633 830 €	RL44, RL46, RL48
Attack privacy/secrecy	R2	Parallel	-13 247 374 €	-13 247 374 €	-13 247 374 €	RL44, RL45, RL47, RL48
Attack privacy/secrecy	R2	Serial	-129 055 €	-129 055 €	-36 868 €	RL39, RL45, RL43, RL42, RL48, RL47
Attack availability	R3	BLSPW	-65 429 €	-65 426 €	-65 421 €	RL49, RL50, RL51
Attack availability	R3	Parallel	-28 412 €	-28 409 €	-28 404 €	RL49, RL50, RL51
Attack availability	R3	Serial	-3 103 €	-3 100 €	-3 094 €	RL51, RL50, RL49

As for attacking the Central System, with all three models and $Gains = €4M$, the unprofitable attacks are *Compromise data carrier (R1.2.4)*, *Attack privacy/secrecy (R2)* and *Attack availability (R3)*. *Compromise VFS (R1.2.1)*, *Compromise VSS (R1.2.2)* and *Compromise VCA (R1.2.3)* attacks are profitable only with the serial model with the outcome values being €112 582, €692 894 and €216 606 respectively. As these are lower than the outcome values for malware attacks, the author assumes that the attacker is more likely to attack the voter and his or her computer environment than the Central System. Complete results for the Revocation Attack with $Gains$ = {1 000 000; 4 000 000; 10 000 000} are presented in Appendix 6. Table 10 shows the profitability of attacks by model for $Gains$ = 4 000 000.

Table 10. Profitability of attacks by model (Revocation Attack)

Attack	**Identifier**	**BLPSW**	**Parallel**	**Serial**
Revocation attack	R	YES	n/a	n/a
Malware	R1.1.1.1 + R1.1.2	YES	YES	n/a
Vote modifying malware	R1.1.1.1.1 + R1.1.2	YES	YES	YES
Re-voting malware	R1.1.1.1.2 + R1.1.2	NO	YES	YES
Self-voting malware	R1.1.1.1.3 + R1.1.2	YES	YES	YES
Fake voting applications	R1.1.1.2 + R1.1.2	NO	NO	n/a
Attack Central System	R1.2	NO	n/a	n/a
Compromise VFS	R1.2.1	NO	NO	YES
Compromise VSS	R1.2.2	NO	NO	YES
Compromise VCA	R1.2.3	NO	NO	YES
Compromise data carrier	R1.2.4	NO	NO	NO
Attack privacy/secrecy	R2	NO	NO	NO
Attack availability	R3	NO	NO	NO

As seen from the table above, attacking the central system components (R1.2.1, R1.2.2, and R1.2.3) are unprofitable with BLPSW and Parallel models, but profitable with the Serial model. This shows that using different models, one can produce different results. Unfortunately, as stated before, the Parallel and Serial models could not have been used to analyse larger (sub)trees. However, based on the information from Table 10, it is reasonable to assume that the entire Revocation Attack is profitable also with the Parallel and Serial models since some of its sub-attacks, which realize the root attack, are profitable. The same applies for *Malware (R1.1.1.1 + R1.1.2)* and *Attack Central System (R1.2)* with the Serial model.

Following the results of the simulations, the author is inclined to conclude that the Estonian I-voting system is reasonably secure against large-scale vote manipulation attacks. The main

bottleneck for these attacks seems to be the large-scale distribution of malware in the Estonian context as well as keeping the act of manipulation secret.

As for the Revocation Attack, many of the sub-attacks are profitable mostly due to the fact that the act of vote modification does not have to remain secret. In addition, the distribution process of malware is easier as there are fewer number of machines required to be infected. In that sense, it can be said that the I-voting system is susceptible to large-scale revocation attacks. However, as stated before, the Revocation Attack described in this thesis only works (in such a scale) under the assumption that I-voting is available during the entire advance voting period and I-votes are revoked only after the online voting period (but before the Election Day). This leaves the I-voters the option to cast their vote again only on the Election Day. However, relying only on a single day to have all of the I-voters to be able to cast a vote in the polling station may not be sufficient. There are bound to be people who simply do not have the time or the means to vote at the polling station on the Election Day. This is, however, what the Revocation Attack relies on.

In order to safeguard against this threat, the voters should be provided enough time to cast their vote again in the traditional way. A timely detection of an ongoing attack against the system is of a vital importance, since it would enable the voters to re-vote during the advance voting period as well as on the Election Day. Additionally, the possible gain in percentage of total votes would lower, as there would be fewer I-votes. Another possibility is to hold a repeat voting, although it would mean additional costs. It must also be emphasized that the Revocation Attack only benefits those parties who receive proportionally less I-votes than regular votes. Propagating the practical security of the I-voting system could balance vote distribution by voting method, thus mitigating the impact of large-scale revocation attacks.

Summary

In this thesis, the author analysed the security of the Estonian I-voting system using three different attack tree methodologies. The author identified three primary threats and studied the feasibility of the attacks using the computational models of each of the methodologies. The work focused mostly on large-scale vote manipulation and the attempts to get I-voting results revoked. Reputation attacks against the system were excluded from the concrete computations due to the limitations of contemporary attack tree models.

Due to the complexity of the computational models of two of the more recent attack tree methodologies used in this thesis, the author could not have used them to analyse the attack trees as wholes in a reasonable amount of time. Instead, the author split the attack trees into sub-attacks that would all materialize the primary threat, and analysed them separately from each another.

Based on the results of the computations, the author concluded that, under the assigned parameter values, the Estonian I-voting scheme is secure against large-scale vote manipulation attacks since all attacks rendered out to be unprofitable for the attacker. The analysis also showed that a large-scale revocation attack might have a sound impact on the election results and the attack itself could be profitable for the attacker under certain conditions. The computations showed that the most probable attack vector is the use of malware in voters' computers. The author also made some suggestions on how to mitigate or avoid the impact of the revocation attacks. Lastly, the work showed that different methodologies produce different results.

References

1. AForest. Available at http://research.cyber.ee/~alexander/ (27.05.2014)
2. Andrusenko, A. Ründepuude metoodika ja seda toetav tarkvaraline raamistik : Master Thesis. Master Thesis. Tallinn, Tallinn University of Technology, 2010.
3. Appeal no. 14-11/406 to NEC, March 5 2011. [WWW] In Estonian, http://www.vvk.ee/valimiste-korraldamine/vabariigi-valimiskomisjon-yld/kirjad (27.05.2014)
4. Buldas, A., Laud, P., Priisalu, J., Saarepera, M., Willemson, J. Rational Choice of Security Measures via Multi-Parameter Attack Trees. – Critical Information Infrastructures Security First International Workshop, 235-248. CRITIS 2006, LNCS 4347, 2006.
5. CoCoViLa. Available at http://www.cs.ioc.ee/cocovila/ (27.05.2014)
6. Constitution of Belgium. Available at http://home.scarlet.be/dirkvanheule/compcons/ConstitutionBelgium/ConstitutionBelgium.htm (27.05.2014)
7. Constitution of Estonia. (Passed 28.06.1992, redaction in force from 22.07.2011) – Riigi Teataja, 1992, 26, 349.
8. Constitution of France. Available at http://www.legislationline.org/documents/action/popup/id/8808/preview (27.05.2014)
9. Constitution of Ireland. (Passed 1.07.1937, redaction in force from 1.11.2013)
10. Decision of Supreme Court 3-4-1-10-11, March 31 2011. [WWW] In Estonian, http://www.vvk.ee/valimiste-korraldamine/vabariigi-valimiskomisjon-yld/kirjad (27.05.2014)
11. Estonian Penal Code. (Passed 06.06.2001, in force from 01.04.2014) – Riigi Teataja I 2001, 61, 364.
12. Euroopa Liit: küsimus ja vastus. [WWW] http://valitsus.ee/UserFiles/valitsus/et/riigikantselei/euroopa/EL%20ja%20Euroala%20k%C3%BCsimused%20ja%20vastused%20viidetega%282%29.pdf (27.05.2014)
13. E-voting concept security: analysis and measures. Estonian National Electoral Commitee, EH-02-02, 2010.
14. E-Voting System, General Overview. Estonian National Electoral Committee, 2013.
15. Goncharov, M. Underground Market 101. – Security Forum 2013, 2013. [WWW] https://www.securityforum.at/wp-content/uploads/2012/02/Sec.Con_.Max_.Goncharov.Underground.Market.101.pdf (27.05.2014)
16. Heiberg, S., Laud, P., & Willemson, J. The Application of I-voting for Estonian Parliamentary Elections of 2011. – Proceedings of VOTEID 2011, 2011, 208-223.
17. Heiberg, S., Willemson, J. Modeling Threats of a Voting Method. – Design, Development, and Use of Secure Electronic Voting Systems. Information Science Reference, 2014.
18. Households. Statistics Estonia. [WWW] http://www.stat.ee/households (27.05.2014)
19. Internet Voting Pilot Project, Local Government Elections, 12 September 2011. OSCE/ODIHR Election Expert Team Report, 2012.

20. Jones, D. W., Simons, B. Broken Ballots. Will Your Vote Count? Center for the Study of Language and Information, 2012.
21. Jurgenson, A., Willemson, J. Serial model for attack tree computations. – International Conference on Information Security and Cryptololgy. ICISC 2009, LNCS, 2009.
22. Jürgenson, A., Willemson, J. Computing exact outcomes of multi-parameter attack trees. – On the Move to Meaningful Internet Systems, 1036-1051. Volume 5332 of LNCS. Springer, 2008.
23. Kordy, B., Piètre-Cambacédès, L., Schweitzer, P. DAG-Based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees. CoRR abs/1303.7397, 2012.
24. Lipmaa H., Mürk, O. E-valimiste realiseerimisvõimaluste analüüs (An analysis of the possibility to organise e-voting), 2001. Analysis ordered by Estonian Ministry of Justice. In Estonian.
25. List of all eEnabled official ballots conducted in Geneva since the start of the internet voting project. Republic and State of Geneva. [WWW] http://www.ge.ch/evoting/doc/list_of_GVA_ballots.pdf (27.05.2014)
26. Local Government Council Election Act. (Passed 27.03.2002, redaction in force from 01.04.2013) – Riigi Teataja I 2002, 36, 220.
27. Mauw, S., Oostdijk, M. Foundations of attack trees. – International Conference on Information Security and Cryptology, 186-198. ICISC 2005. Volume 3935 of LNCS. Springer, 2005.
28. Mägi, T. Practical Security Analysis of E-voting Systems : Master Thesis. Tallinn, Tallinn University of Technology, 2007.
29. Namestnikov, Y. The economics of Botnets. [WWW] https://www.securelist.com/en/analysis/204792068/The_economics_of_Botnets (27.05.2014)
30. Online voting: challenges and outcomes. Republic and State of Geneva. [WWW] http://www.geneve.ch/evoting/english/presentation_projet.asp (27.05.2014)
31. Parliamentary Elections, 9 September 2013. OSCE/ODIHR Election Assessment Mission Final Report, 2013.
32. Pihelgas, P. Interview with M. Kärmas, 2011. Pealtnägija, e. 422.
33. Press release by the Geneva State Chancellery. Republic and State of Geneva. [WWW] http://www.geneve.ch/evoting/english/communiques_20040926.asp (27.05.2014)
34. Riigikogu Election Act. (Passed 12.06.2002, redaction in force from 01.04.2014) – Riigi Teataja I 2002, 57, 355.
35. Saalfeld, T. On Dogs and Whips: Recorded Votes. – Parliaments and Majority Rule in Western Europe, 531. New York: St. Martin's Press, 1995.
36. Schneier, B. Attack trees: Modeling security threats. – Dr. Dobb's Journal, 1999, 24 (12), 21-29.
37. Statistics about Internet Voting in Estonia. Estonian National Electoral Committee. [WWW] http://www.vvk.ee/voting-methods-in-estonia/engindex/statistics/ (27.05.2014)
38. Statistikakogumik "Valimised Eestis 1992-2011" (Collection of statistics "Elections in Estonia 1992-2011). Tallinn : Estonian National Electoral Committee, 2010.

39. Stenerud, S. G., Bull, C. When Reality Comes Knocking Norwegian Experiences with Verifiable Electronic Voting. Norwegian Ministry of Local Government and Regional Development, 2012.
40. Tammet, T., Krosing, H. E-valimised Eesti Vabariigis: võimaluste analüüs (E-voting in Estonia: feasibility study), 2001. Analysis ordered by Estonian Ministry of Transport and Communications. In Estonian.
41. Vassil, K. Postimehe valimisstuudio. [WWW] http://poliitika.postimees.ee/2793128/postimehe-valimisstuudios-arutati-e-valimiste-teemat (27.05.2014)
42. Voter turnout data for Norway. International IDEA. [WWW] http://www.idea.int/vt/countryview.cfm?CountryCode=NO (27.05.2014)

Appendices

Appendix 1. Attack tree for the Manipulation Attack

Identifier	Attack	Type	p	cost	q+	penalties+	q-	penalties-	π+	π−	Expenses
M	**Manipulation attack**	**ROOT**									
M1	Attack voters' environment	OR									
M1.1	Malware	OR									
M1.1.1	**Vote modifying malware**	**AND**									
M1.1.1.1	Develop malware	OR									
M1.1.1.1.1 (ML1)	*Vote changing malware*	*LEAF*	*0,95*	*9600*	*0,05*	*1576800*	*0,05*	*394200*	*78840*	*19710*	*108150*
M1.1.1.1.2 (ML2)	*Vote blocking malware*	*LEAF*	*1,00E-07*	*7680*	*0,05*	*1576800*	*0,05*	*394200*	*78840*	*19710*	*106230*
M1.1.1.2	Distribute malware	AND									
M1.1.1.2.1	Compromise voters' computers	OR									
M1.1.1.2.1.1 (ML3)	*Create botnet*	*LEAF*	*0,05*	*82500*	*0,5*	*2365200*	*0,1*	*591300*	*1182600*	*59130*	*1324230*
M1.1.1.2.1.2 (ML4)	*Buy botnet*	*LEAF*	*0,05*	*22000*	*0,05*	*2365200*	*0,01*	*591300*	*118260*	*5913*	*146173*
M1.1.1.2.2	Compromise voters' mobile devices	OR									
M1.1.1.2.2.1 (ML5)	*Create mobile botnet*	*LEAF*	*0,001*	*82500*	*0,5*	*2365200*	*0,1*	*591300*	*1182600*	*59130*	*1324230*
M1.1.1.2.2.2 (ML6)	*Buy mobile botnet*	*LEAF*	*0,001*	*22000*	*0,05*	*2365200*	*0,01*	*591300*	*118260*	*5913*	*146173*
M1.1.1.3 (ML7)	*Avoid detection*	*LEAF*	*0,1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
M1.1.2	**Re-voting malware**	**AND**									
M1.1.2.1 (ML8)	*Develop malware*	*LEAF*	*0,95*	*6720*	*0,05*	*1576800*	*0,05*	*394200*	*78840*	*19710*	*105270*
M1.1.2.2	Compromise voters' computers	OR									
M1.1.2.2.1 (ML9)	*Create botnet*	*LEAF*	*0,0005*	*320000*	*0,5*	*4730400*	*0,1*	*1182600*	*2365200*	*118260*	*2803460*
M1.1.2.2.2 (ML10)	*Buy botnet*	*LEAF*	*0,0005*	*65000*	*0,05*	*4730400*	*0,01*	*1182600*	*236520*	*11826*	*313346*
M1.1.2.3 (ML11)	*Avoid detection*	*LEAF*	*0,1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
M1.1.3	**Self-voting malware**	**AND**									
M1.1.3.1 (ML12)	*Develop malware*	*LEAF*	*0,95*	*4800*	*0,05*	*1576800*	*0,05*	*394200*	*78840*	*19710*	*103350*
M1.1.3.2	Compromise voters' computers	OR									
M1.1.3.2.1 (ML13)	*Create botnet*	*LEAF*	*0,3*	*45000*	*0,5*	*4730400*	*0,1*	*1182600*	*2365200*	*118260*	*2528460*
M1.1.3.2.2 (ML14)	*Buy botnet*	*LEAF*	*0,3*	*15000*	*0,05*	*4730400*	*0,01*	*1182600*	*236520*	*11826*	*263346*
M1.1.3.3 (ML15)	*Avoid detection*	*LEAF*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
M1.2	Fake voting applications	AND									
M1.2.1	Develop fake apps	AND									
M1.2.1.1 (ML16)	*Develop fake Voting App.*	*LEAF*	*0,95*	*3840*	*0,1*	*1576800*	*0,1*	*394200*	*157680*	*39420*	*200940*
M1.2.1.2 (ML17)	*Develop fake Verification App.*	*LEAF*	*0,95*	*3840*	*0,1*	*1576800*	*0,1*	*394200*	*157680*	*39420*	*200940*
M1.2.2	Distribute fake Voting App.	OR									
M1.2.2.1	Use fake website	AND									
M1.2.2.1.1 (ML18)	*Develop fake website*	*LEAF*	*0,95*	*800*	*0,1*	*4800*	*0,1*	*1200*	*480*	*120*	*1400*
M1.2.2.1.2	Get voters to visit fake website	OR									
M1.2.2.1.2.1 (ML19)	*E-mail*	*LEAF*	*0,001*	*8045*	*0,005*	*2628000*	*0,005*	*657000*	*13140*	*3285*	*24470*
M1.2.2.1.2.2 (ML20)	*Network attacks*	*LEAF*	*0,005*	*10000*	*0,5*	*2628000*	*0,1*	*657000*	*1314000*	*65700*	*1389700*
M1.2.2.1.2.3 (ML21)	*Social media*	*LEAF*	*0,002*	*400*	*0,005*	*2628000*	*0,005*	*657000*	*13140*	*3285*	*16825*
M1.2.2.2	Replace app. On NEC web server	OR									
M1.2.2.2.1 (ML22)	*Bribe server admin*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
M1.2.2.2.2 (ML23)	*Bribe SW developer*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
M1.2.2.2.3 (ML24)	*Exploit configuration error*	*LEAF*	*0,005*	*15000*	*0,5*	*4730400*	*0,1*	*1182600*	*2365200*	*118260*	*2498460*
M1.2.3	Distribute fake Verification App.	OR									
M1.2.3.1	From official appstore	OR									
M1.2.3.1.1 (ML25)	*Upload similar*	*LEAF*	*0,05*	*150*	*0,01*	*2102400*	*0,01*	*525600*	*21024*	*5256*	*26430*
M1.2.3.1.2	Replace original	OR									
M1.2.3.1.2.1 (ML26)	*Bribe server admin*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
M1.2.3.1.2.2 (ML27)	*Bribe SW developer*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
M1.2.3.1.2.3 (ML28)	*Exploit configuration error*	*LEAF*	*0,005*	*15000*	*0,5*	*4730400*	*0,1*	*1182600*	*2365200*	*118260*	*2498460*
M1.2.3.2 (ML29)	*From other markets*	*LEAF*	*1,00E-05*	*150*	*0,01*	*2102400*	*0,01*	*525600*	*21024*	*5256*	*26430*
M1.2.4 (ML30)	*Avoid detection*	*LEAF*	*0,005*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
M2	Attack Central System	OR									
M2.1	**Compromise VSS**	**AND**									
M2.1.1 (ML31)	*Develop malicious code*	*LEAF*	*0,50*	*12000*	*0,05*	*1576800*	*0,05*	*394200*	*78840*	*19710*	*110550*
M2.1.2	Insert code into server	OR									
M2.1.2.1 (ML32)	*Bribe server admin*	*LEAF*	*0,0033*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
M2.1.2.2 (ML33)	*Bribe SW developer*	*LEAF*	*0,0033*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
M2.1.2.3	Get access to server	AND									
M2.1.2.3.1 (ML34)	*Get access to internal network*	*LEAF*	*0,005*	*15000*	*0,5*	*2628000*	*0,1*	*657000*	*1314000*	*65700*	*1394700*
M2.1.2.3.2 (ML35)	*Exploit configuration error*	*LEAF*	*0,00005*	*15000*	*0,5*	*4730400*	*0,1*	*1182600*	*2365200*	*118260*	*2498460*
M2.2	**Compromise VCA**	**AND**									
M2.2.1 (ML36)	*Develop malicious code*	*LEAF*	*0,50*	*12000*	*0,05*	*1576800*	*0,05*	*394200*	*78840*	*19710*	*110550*
M2.2.2	Insert code into server	OR									
M2.2.2.1 (ML37)	*Bribe server admin*	*LEAF*	*0,0033*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
M2.2.2.2 (ML38)	*Bribe SW developer*	*LEAF*	*0,0033*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
M2.3	**Compromise data carrier**	**AND**									
M2.3.1	Get access to device	OR									
M2.3.1.1 (ML39)	*Bribe worker*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
M2.3.1.2 (ML40)	*Infiltrate as participant*	*LEAF*	*0,05*	*20000*	*0,9*	*4204800*	*0,9*	*1051200*	*3784320*	*946080*	*4750400*
M2.3.2 (ML41)	*Compromise device*	*LEAF*	*0,001*	*4000*	*0,8*	*4204800*	*0,2*	*1051200*	*3363840*	*210240*	*3578080*

Appendix 2. Attack tree for the Revocation Attack

Identifier	Name	Type	p	cost	q+	penalties+	q-	penalties-	π+	π−	Expenses
R	**Revocation attack**	**ROOT**									
R1	*Attack integrity*	*OR*									
R1.1	Attack voter	AND									
R1.1.1	Compromise voters' computers	OR									
R1.1.1.1	Malware	OR									
R1.1.1.1.1	**Vote modifying malware**	**AND**									
R1.1.1.1.1.1	Develop malware	OR									
R1.1.1.1.1.1.1 (RL1)	*Vote changing malware*	*LEAF*	*0,95*	*4800*	*0,05*	*1576800*	*0,05*	*394200*	*78840*	*19710*	*103350*
R1.1.1.1.1.1.2 (RL2)	*Vote blocking malware*	*LEAF*	*0,95*	*3840*	*0,05*	*1576800*	*0,05*	*394200*	*78840*	*19710*	*102390*
R1.1.1.1.1.2	Compromise voters' computers	OR									
R1.1.1.1.1.2.1 (RL3)	*Create botnet*	*LEAF*	*0,60*	*21500*	*0,5*	*4730400*	*0,1*	*1182600*	*2365200*	*118260*	*2504960*
R1.1.1.1.1.2.2 (RL4)	*Buy botnet*	*LEAF*	*0,60*	*10500*	*0,05*	*4730400*	*0,01*	*1182600*	*236520*	*11826*	*258846*
R1.1.1.1.2	**Re-voting malware**	**AND**									
R1.1.1.1.2.1 (RL5)	*Develop malware*	*LEAF*	*0,95*	*6720*	*0,05*	*1576800*	*0,05*	*394200*	*78840*	*19710*	*105270*
R1.1.1.1.2.2	Compromise voters' computers	OR									
R1.1.1.1.2.2.1 (RL6)	*Create botnet*	*LEAF*	*0,08*	*69000*	*0,5*	*4730400*	*0,1*	*1182600*	*2365200*	*118260*	*2552460*
R1.1.1.1.2.2.2 (RL7)	*Buy botnet*	*LEAF*	*0,08*	*19500*	*0,05*	*4730400*	*0,01*	*1182600*	*236520*	*11826*	*267846*
R1.1.1.1.3	**Self-voting malware**	**AND**									
R1.1.1.1.3.1 (RL8)	*Develop malware*	*LEAF*	*0,95*	*4800*	*0,05*	*1576800*	*0,05*	*394200*	*78840*	*19710*	*103350*
R1.1.1.1.3.2	Compromise voters' computers	OR									
R1.1.1.1.3.2.1 (RL9)	*Create botnet*	*LEAF*	*0,90*	*14000*	*0,5*	*4730400*	*0,1*	*1182600*	*2365200*	*118260*	*2497460*
R1.1.1.1.3.2.2 (RL10)	*Buy botnet*	*LEAF*	*0,90*	*9500*	*0,05*	*4730400*	*0,01*	*1182600*	*236520*	*11826*	*257846*
R1.1.1.2	Fake voting applications	AND									
R1.1.1.2.1 (RL11)	*Develop fake Voting App.*	*LEAF*	*0,95*	*3840*	*0,1*	*788400*	*0,1*	*394200*	*78840*	*39420*	*122100*
R1.1.1.2.2	Distribute fake Voting App.	OR									
R1.1.1.2.2.1	Use fake website	AND									
R1.1.1.2.2.1.1 (RL12)	*Develop fake website*	*LEAF*	*0,95*	*800*	*0,1*	*4800*	*0,1*	*1200*	*480*	*120*	*1400*
R1.1.1.2.2.1.2	Get voters to visit fake website	OR									
R1.1.1.2.2.1.2.1 (RL13)	*E-mail*	*LEAF*	*0,001*	*8045*	*0,005*	*2628000*	*0,005*	*657000*	*13140*	*3285*	*24470*
R1.1.1.2.2.1.2.2 (RL14)	*Network attacks*	*LEAF*	*0,005*	*10000*	*0,5*	*2628000*	*0,1*	*657000*	*1314000*	*65700*	*1389700*
R1.1.1.2.2.1.2.3 (RL15)	*Social media*	*LEAF*	*0,002*	*400*	*0,005*	*2628000*	*0,005*	*657000*	*13140*	*3285*	*16825*
R1.1.1.2.2.2	Replace app. On NEC web server	OR									
R1.1.1.2.2.2.1 (RL16)	*Bribe server admin*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
R1.1.1.2.2.2.2 (RL17)	*Bribe SW admin*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
R1.1.1.2.2.2.3 (RL18)	*Exploit configuration error*	*LEAF*	*0,005*	*15000*	*0,5*	*4730400*	*0,1*	*1182600*	*2365200*	*118260*	*2498460*
R1.1.2 (RL19)	*Convince NEC*	*LEAF*	*0,80*	*3000*	*0*	*0*	*0*	*0*	*0*	*0*	*3000*
R1.2	Attack Central System	OR									
R1.2.1	**Compromise VFS**	**AND**									
R1.2.1.1 (RL20)	*Develop malicious code*	*LEAF*	*0,50*	*5000*	*0,05*	*1576800*	*0,05*	*394200*	*78840*	*19710*	*103550*
R1.2.1.2	Insert code into server	OR									
R1.2.1.2.1 (RL21)	*Bribe server admin*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
R1.2.1.2.2 (RL22)	*Bribe SW admin*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
R1.2.1.2.3 (RL23)	*Exploit configuration error*	*LEAF*	*0,005*	*15000*	*0,5*	*4730400*	*0,1*	*1182600*	*2365200*	*118260*	*2498460*
R1.2.1.3 (RL24)	*Convince NEC*	*LEAF*	*0,5*	*3000*	*0*	*0*	*0*	*0*	*0*	*0*	*3000*
R1.2.2	**Compromise VSS**	**AND**									
R1.2.2.1 (RL25)	*Develop malicious code*	*LEAF*	*0,50*	*12000*	*0,05*	*1576800*	*0,05*	*394200*	*78840*	*19710*	*110550*
R1.2.2.2	Insert code into server	OR									
R1.2.2.2.1 (RL26)	*Bribe server admin*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
R1.2.2.2.2 (RL27)	*Bribe SW admin*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
R1.2.2.2.3	Get access to server	AND									
R1.2.2.2.3.1 (RL28)	*Get access to internal network*	*LEAF*	*0,5*	*15000*	*0,5*	*2628000*	*0,1*	*657000*	*1314000*	*65700*	*1394700*
R1.2.2.2.3.2 (RL29)	*Exploit configuration error*	*LEAF*	*0,005*	*15000*	*0,5*	*4730400*	*0,1*	*1182600*	*2365200*	*118260*	*2498460*
R1.2.2.3 (RL30)	*Convince NEC*	*LEAF*	*0,95*	*3000*	*0*	*0*	*0*	*0*	*0*	*0*	*3000*
R1.2.3	**Compromise VCA**	**AND**									
R1.2.3.1 (RL31)	*Develop malicious code*	*LEAF*	*0,50*	*12000*	*0,05*	*1576800*	*0,05*	*394200*	*78840*	*19710*	*110550*
R1.2.3.2	Insert code into server	OR									
R1.2.3.2.1 (RL32)	*Bribe server admin*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
R1.2.3.2.2 (RL33)	*Bribe SW admin*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
R1.2.3.3 (RL34)	*Convince NEC*	*LEAF*	*0,95*	*3000*	*0*	*0*	*0*	*0*	*0*	*0*	*3000*
R1.2.4	**Compromise data carrier**	**AND**									
R1.2.4.1	Get access to device	OR									
R1.2.4.1.1 (RL35)	*Bribe worker*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
R1.2.4.1.2 (RL36)	*Infiltrate as participant*	*LEAF*	*0,05*	*20000*	*0,9*	*4204800*	*0,9*	*1051200*	*3784320*	*946080*	*4750400*
R1.2.4.2 (RL37)	*Compromise device*	*LEAF*	*0,1*	*4000*	*0,8*	*4204800*	*0,2*	*1051200*	*3363840*	*210240*	*3578080*
R1.2.4.3 (RL38)	*Convince NEC*	*LEAF*	*0,95*	*3000*	*0*	*0*	*0*	*0*	*0*	*0*	*3000*

Appendix 3. Attack tree for the Revocation Attack (continued)

Identifier	Name	Type	p	cost	q+	penalties+	q-	penalties-	π+	π−	Expenses
R2	*Attack privacy/secrecy*	*AND*									
R2.1	**Get votes from the VSS**	**OR**									
R2.1.1	Use malicious code	AND									
R2.1.1.1 (RL39)	*Develop malicious code*	*LEAF*	*0,50*	*12000*	*0,05*	*1576800*	*0,05*	*394200*	*78840*	*19710*	*110550*
R2.1.1.2	Insert code into server	OR									
R2.1.1.2.1 (RL40)	*Bribe server admin*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
R2.1.1.2.2 (RL41)	*Bribe SW admin*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
R2.1.1.2.3	Get access to server	AND									
R2.1.1.2.3.1 (RL42)	*Get access to internal network*	*LEAF*	*0,50*	*15000*	*0,5*	*2628000*	*0,1*	*657000*	*1314000*	*65700*	*1394700*
R2.1.1.2.3.2 (RL43)	*Exploit configuration error*	*LEAF*	*0,005*	*15000*	*0,5*	*4730400*	*0,1*	*1182600*	*2365200*	*118260*	*2498460*
R2.1.2 (RL44)	*Bribe employee*	*LEAF*	*0,33*	*30000*	*0,7*	*3153600*	*0,4*	*788400*	*2207520*	*315360*	*2552880*
R2.2	**Decrypt votes**	**OR**									
R2.2.1	Obtain private key	OR									
R2.2.1.1 (RL45)	*Steal*	*LEAF*	*1,00E-04*	*4000*	*0,6*	*2628000*	*0,2*	*657000*	*1576800*	*131400*	*1712200*
R2.2.1.2 (RL46)	*Bribe key manager*	*LEAF*	*0,33*	*30000*	*0,7*	*2628000*	*0,4*	*657000*	*1839600*	*262800*	*2132400*
R2.2.2 (RL47)	*Use cryptanalysis*	*LEAF*	*0,01*	*10000000*	*0,005*	*2628000*	*0,005*	*657000*	*13140*	*3285*	*10016425*
R2.3 (RL48)	*Convince NEC*	*LEAF*	*0,05*	*3000*	*0*	*0*	*0*	*0*	*0*	*0*	*3000*
R3	*Attack availability*	*AND*									
R3.1 (RL49)	*Rent botnet*	*LEAF*	*0,95*	*800*	*0,01*	*1576800*	*0,01*	*394200*	*15768*	*3942*	*20510*
R3.2 (RL50)	*Perform DDoS for 5 days*	*LEAF*	*0,0001*	*2500*	*0,01*	*3153600*	*0,01*	*788400*	*31536*	*7884*	*41920*
R3.3 (RL51)	*Convince NEC*	*LEAF*	*0,01*	*3000*	*0*	*0*	*0*	*0*	*0*	*0*	*3000*

Appendix 4. Attack tree for the Reputation Attack

Identifier	Name	Type
D	**Reputation attack**	**ROOT**
D1	Anti-campaign	AND
D1.1	**Question trusthworthiness**	**OR**
D1.1.1 (DL1)	*Point out problems with previous elections*	*LEAF*
D1.1.2 (DL2)	*Find contradictions with constitution*	*LEAF*
D1.1.3 (DL3)	*Develop proof-of-concept attack*	*LEAF*
D1.2	**Get public's attention**	OR
D1.2.1 (DL4)	*Use internet/social media*	*LEAF*
D1.2.2 (DL5)	*Use physical media*	*LEAF*
D1.2.3 (DL6)	*Organize public events*	*LEAF*
D1.3	**Get support**	OR
D1.3.1 (DL7)	*Involve experts*	*LEAF*
D1.3.2 (DL8)	*Involve public figures*	*LEAF*
D1.3.3 (DL9)	*Involve general public*	*LEAF*
D2	Attack public system	OR
D2.1	DDoS	AND
D2.1.1 (DL10)	*Rent botnet*	*LEAF*
D2.1.2 (DL11)	*Perform DDoS for 5 days*	*LEAF*
D2.2 (DL12)	*Defacement*	*LEAF*

Appendix 5. Complete results of the Manipulation Attack

Attack	Identifier	AT Model	Outcome (Gains = 1M)	Outcome (Gains = 10M)	Outcome (Gains = 20M)	Best attack suite (Gains = 10M)
Manipulation attack	M1	BLSPW	-366 696 €	-366 696 €	-366 696 €	ML12, ML14, ML15
Malware	M1.1	BLSPW	-366 696 €	-366 696 €	-366 696 €	ML12, ML14, ML15
Malware	M1.1	Parallel	-88 851 €	-88 851 €	-88 851 €	ML1, ML4, ML6, ML7
Vote modifying malware	M1.1.1	BLSPW	-400 491 €	-400 449 €	-400 401 €	ML1, ML4, ML6, ML7
Vote modifying malware	M1.1.1	Parallel	-88 851 €	-88 851 €	-88 851 €	ML1, ML4, ML6, ML7
Vote modifying malware	M1.1.1	Serial	-2 739 €	-2 723 €	-2 631 €	ML7, ML6, ML4, ML3, ML1
Re-voting malware	M1.1.2	BLSPW	-418 569 €	-418 141 €	-417 666 €	ML8, ML10, ML11
Re-voting malware	M1.1.2	Parallel	-159 494 €	-159 067 €	-158 592 €	ML8, ML10, ML11
Re-voting malware	M1.1.2	Serial	-7 650 €	-7 223 €	-6 748 €	ML11, ML10, ML8
Self-voting malware	M1.1.3	BLSPW	-366 696 €	-366 696 €	-366 696 €	ML12, ML14, ML15
Self-voting malware	M1.1.3	Parallel	-174 918 €	-174 918 €	-174 918 €	ML12, ML14, ML15
Self-voting malware	M1.1.3	Serial	0 €	0 €	0 €	ML15, ML12, ML13
Fake voting applications	M1.2	BLSPW	-446 535 €	-446 535 €	-446 535 €	ML16, ML17, ML22, ML26, ML30
Fake voting applications	M1.2	Parallel	-322 262 €	-322 262 €	-322 262 €	ML16, ML17, ML18, ML21, ML29, ML30
Attack Central System	M2	BLSPW	-2 661 780 €	-2 646 930 €	-2 630 430 €	ML31, ML32
Attack Central System	M2	Parallel	-281 589 €	-281 588 €	-281 586 €	ML31, ML32
Compromise VSS	M2.1	BLSPW	-2 661 780 €	-2 646 930 €	-2 630 430 €	ML31, ML32
Compromise VSS	M2.1	Parallel	-281 589 €	-281 588 €	-281 586 €	ML31, ML32
Compromise VSS	M2.1	Serial	-87 581 €	-87 580 €	-87 579 €	ML34, ML31, ML35
Compromise VCA	M2.2	BLSPW	-2 661 780 €	-2 646 930 €	-2 630 430 €	ML36, ML37
Compromise VCA	M2.2	Parallel	-411 229 €	-396 379 €	-379 879 €	ML36, ML37
Compromise VCA	M2.2	Serial	-235 427 €	-220 577 €	-204 077 €	ML36, ML37
Compromise data carrier	M2.3	BLSPW	-5 184 550 €	-5 181 580 €	-5 178 280 €	ML39, ML41
Compromise data carrier	M2.3	Parallel	-1 038 374 €	-1 035 404 €	-1 032 104 €	ML39, ML41
Compromise data carrier	M2.3	Serial	-353 542 €	-350 572 €	-347 272 €	ML41, ML39

Appendix 6. Complete results of the Revocation Attack

Attack	Identifier	AT Model	Outcome (Gains = 1M)	Outcome (Gains = 4M)	Outcome (Gains = 10M)	Best attack suite (Gains = 4M)
Revocation attack	R	BLSPW	319 804 €	2 371 804 €	6 475 804 €	RL8, RL10, RL19
Malware	R1.1.1.1 + R1.1.2	BLSPW	319 804 €	2 371 804 €	6 475 804 €	RL8, RL10, RL19
Malware	R1.1.1.1 + R1.1.2	Parallel	376 765 €	2 456 380 €	6 957 100 €	RL2, RL4, RL8, RL10, RL19
Vote modifying malware	R1.1.1.1.1 + R1.1.2	BLSPW	91 764 €	1 459 764 €	4 195 764 €	RL2, RL4, RL19
Vote modifying malware	R1.1.1.1.1 + R1.1.2	Parallel	212 294 €	1 586 011 €	4 886 087 €	RL2, RL3, RL4, RL19
Vote modifying malware	R1.1.1.1.1 + R1.1.2	Serial	309 882 €	2 020 150 €	6 042 070 €	RL19, RL4, RL3, RL2, RL1
Re-voting malware	R1.1.1.1.2 + R1.1.2	BLSPW	-315 316 €	-132 916 €	231 884 €	RL5, RL7, RL19
Re-voting malware	R1.1.1.1.2 + R1.1.2	Parallel	-73 605 €	108 795 €	665 940 €	RL5, RL7, RL19
Re-voting malware	R1.1.1.1.2 + R1.1.2	Serial	13 472 €	195 872 €	845 045 €	RL19, RL7, RL5
Self-voting malware	R1.1.1.1.3 + R1.1.2	BLSPW	319 804 €	2 371 804 €	6 475 804 €	RL8, RL10, RL19
Self-voting malware	R1.1.1.1.3 + R1.1.2	Parallel	376 765 €	2 428 765 €	6 532 765 €	RL8, RL10, RL19
Self-voting malware	R1.1.1.1.3 + R1.1.2	Serial	446 555 €	2 608 410 €	7 122 810 €	RL19, RL8, RL10, RL9
Fake voting applications	R1.1.1.2 + R1.1.2	BLSPW	-141 881 €	-137 549 €	-128 885 €	RL11, RL12, RL15, RL19
Fake voting applications	R1.1.1.2 + R1.1.2	Parallel	-160 591 €	-115 034 €	2 113 410 €	RL11, RL12, RL15, RL16, RL19
Attack Central System	R1.2	BLSPW	-2 509 680 €	-2 039 430 €	-1 098 930 €	RL25, RL26, RL30
Comromise VFS	R1.2.1	BLSPW	-2 603 760 €	-2 600 010 €	-2 592 510 €	RL20, RL23, RL24
Comromise VFS	R1.2.1	Parallel	-207 520 €	-203 770 €	-196 269 €	RL20, RL23, RL24
Comromise VFS	R1.2.1	Serial	-68 511 €	112 582 €	939 232 €	RL24, RL20, RL21, RL22
Compromise VSS	R1.2.2	BLSPW	-2 509 680 €	-2 039 430 €	-1 098 930 €	RL25, RL26, RL30
Compromise VSS	R1.2.2	Parallel	-877 298 €	-407 048 €	613 904 €	RL25, RL26, RL30
Compromise VSS	R1.2.2	Serial	-130 333 €	692 894 €	2 903 210 €	RL30, RL25, RL28, RL26, RL27
Compromise VCA	R1.2.3	BLSPW	-2 509 680 €	-2 039 430 €	-1 098 930 €	RL31, RL32, RL34
Compromise VCA	R1.2.3	Parallel	-877 298 €	-407 048 €	613 904 €	RL31, RL32, RL34
Compromise VCA	R1.2.3	Serial	-365 103 €	216 606 €	1 787 240 €	RL34, RL31, RL32, RL33
Compromise data carrier	R1.2.4	BLSPW	-6 102 610 €	-6 008 560 €	-5 820 460 €	RL35, RL37, RL38
Compromise data carrier	R1.2.4	Parallel	-1 469 703 €	-1 371 693 €	-1 175 673 €	RL35, RL37, RL38
Compromise data carrier	R1.2.4	Serial	-590 642 €	-492 632 €	-296 612 €	RL38, RL37, RL35
Attack privacy/secrecy	R2	BLSPW	-4 682 835 €	-4 666 500 €	-4 633 830 €	RL44, RL46, RL48
Attack privacy/secrecy	R2	Parallel	-13 247 374 €	-13 247 374 €	-13 247 374 €	RL44, RL45, RL47, RL48
Attack privacy/secrecy	R2	Serial	-129 055 €	-129 055 €	-36 868 €	RL39, RL45, RL43, RL42, RL48, RL47
Attack availability	R3	BLSPW	-65 429 €	-65 426 €	-65 421 €	RL49, RL50, RL51
Attack availability	R3	Parallel	-28 412 €	-28 409 €	-28 404 €	RL49, RL50, RL51
Attack availability	R3	Serial	-3 103 €	-3 100 €	-3 094 €	RL51, RL50, RL49