ON VERIFIABLE INTERNET VOTING SYSTEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

KÖKSAL MUŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
CRYPTOGRAPHY

APRIL 2016

Approval of the thesis:

ON VERIFIABLE INTERNET VOTING SYSTEMS

submitted by **KÖKSAL MUŞ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Department of Cryptography, Middle East Technical University** by,

Prof. Dr. Bülent Karasözen
Director, Graduate School of **Applied Mathematics** _____

Prof. Dr. Ferruh Özbudak
Head of Department, **Cryptography** _____

Assoc. Prof. Dr. Murat Cenk
Supervisor, **Cryptography, METU** _____

Examining Committee Members:

Prof. Dr. Ersan Akyıldız
Department of Mathematics, METU _____

Assoc. Prof. Dr. Murat Cenk
Cryptography Department, METU _____

Prof. Dr. Ali Aydın Selçuk
Computer Engineering Department, TOBB-ETÜ _____

Assoc. Prof. Dr. Ali Doğanaksoy
Department of Mathematics, METU _____

Assist. Prof. Dr. Fatih Sulak
Department of Mathematics, Atılım University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: KÖKSAL MUŞ

Signature :

ABSTRACT

ON VERIFIABLE INTERNET VOTING SYSTEMS

Muş, Köksal
Ph.D., Department of Cryptography
Supervisor : Assoc. Prof. Dr. Murat Cenk

April 2016, 51 pages

After the Estonian Parliamentary Elections held in 2011, an additional verification mechanism was integrated into the i-voting system in order to resist malicious voting devices, including the so-called Student's Attack. This mechanism gives voters the opportunity to verify whether the vote they cast is stored in the central system correctly. However, the verification phase ends by displaying the cast vote in plain form on the verification device. Indeed, when applied in wide range, this would even compromise the fairness and the overall secrecy of the elections. In this work, our aim is to investigate this verification phase in detail and to point out that displaying the cast vote in plain form may leak voter privacy. In this respect, we propose an alternative verification mechanism for the Estonian i-voting system to overcome this vulnerability. Not only is the proposed mechanism secure and resistant against corrupted verification devices, so does it successfully verify whether the vote is correctly stored in the system. We also highlight that our proposed mechanism brings only symmetric encryptions and hash functions on the verification device, thereby mitigating these weaknesses in an efficient way. More concretely, it brings only m additional symmetric key decryptions to the verification device, with m denoting the number of candidates. Finally, we prove the security of the proposed verification mechanism and compare the cost complexity of the proposed method with that of the current mechanism.

Keywords : Internet Voting, Privacy, Secrecy, Verifiability, Trust

ÖZ

DOĞRULANABİLİR İNTERNET OYLAMA SİSTEMLERİ HAKKINDA

Muş, Köksal
Doktora, Kriptografi Bölümü
Tez Yöneticisi : Doç. Dr. Murat Cenk

Nisan 2016, 51 sayfa

2011'de yapılan Estonya Parlamento Seçimlerinden sonra, internet oylama sisteminde kötücül yazılım barındıran bilgisayarlara yapılabilen *Öğrenci Atağı'nı (Student's Attack)* engelleyen bir doğrulama mekanizması eklendi. Bu mekanizma seçmenlere oylarının merkezi sistemde doğru kaydedilip kaydedilmediğini doğrulama imkanı sunmaktadır. Fakat, doğrulama adımının sonunda seçmenin oyu açık şekilde ekranda gözükmektedir. Dahası, ataklar geniş ölçekli uygulandığında genel seçim bütünlüğünü ihlal edebilmektedir. Bu çalışmadaki amacımız doğrulama adımını güvenlik açısından detaylı şekilde incelemek ve oyun ekranda açık şekilde görüntülenmesi işleminden kaynaklı seçmen mahremiyet probleminin olup olmadığını ortaya çıkarmaktır. Bu bağlamda, Estonya için bu zayıflığın üstesinden gelebilecek alternatif bir doğrulama mekanizması önermekteyiz. Önerdiğimiz mekanizma kötücül yazılım içeren doğrulama cihazlarına karşı da güçlü olmasının yanında, oyun sisteme doğru yüklenip yüklenmediğini de kontrol etmektedir. Ayrıca, önerilen sistem bu zayıflıkları çözmek için doğrulama cihazına sadece simetrik şifre ve özet fonksiyon maaliyeti getirdiğini de vurgulamak gerekir. Yani, doğrulama cihazına aday sayısı m olmak üzere, m tane ekstra simetrik şifre çözme işlemi yaptırır. Sonuç kısmında önerilen doğrulama sisteminin güvenlik ispatını ve önerilen sistemle kullanılan sistemin işlem maaliyetlerini karşılaştıracağız.

Anahtar Kelimeler : İnternet Oylaması, Mahremiyet, Güvenlik, Doğrulama, Güven

Aileme, özellikle babama

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my supervisor Assoc. Prof. Dr. Murat Cenk for supporting me at the point of giving up and throughout the entire thesis work. Additionally, I would like to express my deepest gratitude to Mehmet Sabır Kiraz for his guidance and insight he provided throughout the thesis work. Although, the current legislation does not allow me to have the chance to be a formal student of him, his ideas and tremendous support had a major influence on this thesis and his motivating attitudes made the thesis an enjoyable academic journey for me. I am also thankful to İsa Sertkaya for sharing his inspiring ideas at the critical moments. Moreover, it is a great pleasure for me to thank Ali Aydın Selçuk and Ersan Akyıldız for motivating me about post-PhD period.

My sincere thanks goes to Ali Doğanaksoy to his endless support and care like a father throughout my life in METU. And, special thanks should be given to Fatih Sulak for his friendship and guidance like a brother from my very first days in METU. Additionally, I am thankful to my wife Sinem Sasmaz Mus for her understanding especially while writing the thesis.

I also would like to thank everyone at IAM-METU, especially Nejla Erdoğdu and Muhiddin Uğuz for their problem solving talents about all administrative problems. Furthermore, I thank to Zack Crist and Tayfun Evyapan for language corrections.

It was also a privilege to being a part of METU Aikido Society and aikido family all over the world to refreshing my mind and giving me to opportunity to understand why I should have another profession.

The last but not least, my special thanks goes to my family, my friends in İstanbul, Ankara and all other parts of the world. They sometimes motivate, sometimes demotivate me, but they help me to be who I am today and also they give me a reason to live for.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

NEC	National Electoral Committee
NIST	National Institute of Standars and Technology
I-voting	Internet voting
E-voting	Electronic Voting
VFS	Vote Forwarding Server
VSS	Vote Storage Server
VFS	Vote Counting Server
VoterApp	Voter Application
VerifApp	Verification Application
HSM	Hardware Security Module
TLS	Transport Layer Security
SSL	Secure Socket Layer
QR code	Quick response Code
IFP	Integer Factorization Problem
DLP	Discrete Logarithm Problem
CA	Certification Authority
M	Message
$SymEnc_{(.)}(.)$	Symmetric Encryption
$SymDec_{(.)}(.)$	Symmetric Decryption
$H(.)$	Hash Function
$\mathcal{V}$	Voter
$pk_{\mathcal{V}}$	Voter Public Key
$sk_{\mathcal{V}}$	Voter Private Key
pk_S	Election Specific Public Key
sk_S	Election Specific Private Key
$voteref$	Vote Reference
E_{asym}	Encrypted Vote
$SignEncVote$	Signed and Encrypted Vote
r	Random Number
CL	Candidate List

c	Candidate number
m	Number of Candidates
Q	Potential Verification Parameter List
q	Verification Parameter

CHAPTER 1

Introduction

Technology is frequently used in daily routines for governmental or banking services via the internet using computers or smart devices. Among these services, internet voting (i-voting) has the potential of increasing election participation, allowing voters, especially handicapped citizens or those citizens living abroad, to cast a vote without going to polling stations on a specific day or time. However, related security issues have not been taken into extensive consideration when the users install applications onto their devices. In particular, by not paying attention to the permissions given to the applications, users turn their smart devices into potential targets for malicious malware that may be used to obtain critical information about users [33].

Estonian i-voting protocol with its verification mechanism present an interesting case because Estonian protocol avoids the additional pre- and post-channels as seen in the Norwegian protocol, in which verification is performed via smart devices. Since 2005 Estonian i-voting system is still being used and the number of i-voters increased in every election. In 2005 local election trial, while only 1.9% of all votes were cast using the i-voting system, more specifically, 5.5%, 14.7%, 15.8%, 24.3%, 21.2%, 31.3% and 30.5% of votes were cast using the i-voting system in the later elections, respectively [12]. These statistics show that the increasing number of citizens prefer to use i-voting system. Accordingly, security concerns related the i-voting system should be considered more seriously.

The i-voting system aims to be at least as secure as traditional paper ballots, meaning that i-voting should meet both cast-as-intended and recorded-as-cast requirements [29, 5]. As mentioned in [20, 18], client-side weaknesses were experienced in both the cast-as-intended and recorded-as-cast mechanisms during Estonia's 2011 parliamentary elections, so called *Student's Attack*. Therefore, after the 2011 election, a verification mechanism was added to the system that gives voters the opportunity to verify their votes stored in the system. The verification mechanism pilot was first tested in the 2013 local elections and then used in the European Parliament Elections and in 2014, and the Parliamentary Elections in 2015. Although using an application on a smart device for voting verification solves the aimed security weaknesses, it may bring with it additional problems related not only to the voter privacy, but also to the secrecy of election results.

The limitations of the system to fulfill the aimed security level is given in [5]. In the

following, we group the important limitations of Estonian i-voting system [11] under the general electronic voting limitations [4] to make the concept more clear:

- **Voter Eligibility:** Eligibility is decided under the legislation. All eligible voters should authenticate before casting vote (authorization of voters) and it is allowed to vote more then once (possibility for electronic re-vote). However, the last vote should be taken into account as voter's choice (one person-one vote). Within a certain time of i-voting period, conventional voting opportunity is given to the voters who prefer to vote manually or want to vote again (supremacy of conventional voting). Obviously, all the *Voter*s should have equal voting opportunity and every vote should have equal weight (uniformity of voting).
- **Secrecy and Privacy:** It should be impossible to determine the voter's intention, voting time, date and device (privacy of the fact of voting). Additionally, the system should not leak any information about the vote and the voter (secrecy of vote).
- **Accuracy :** It is expected that votes are received, stored and tallied accurately by the Central System (controllability of vote counting). Additionally, vote counting process must be repeatable (repeatability of vote counting).
- **Integrity :** Nobody should not be able to change the vote and give falsified votes to the system (prohibition of falsification of votes).
- **Verifiability :** There are two kinds of verifiability. Individual verifiability: every voter should be able to check his/her own vote. Universal verifiability: Everybody is able to check the election results without knowing the intention of voters.
- **Secrecy of voting results :** Nobody should get any idea about the partial results of the election before the voting period ended.
- **Receipt-freeness :** It should not be possible to prove voter's intention (unprovability of voting).
- **Incoercibility :** The system should protect voters from coercion and vote buying issues.
- **Auditability :** The processes before and after the election should be transparent (transparency). And, all the processes should be audited by independent authorized auditors (auditability).
- **Operability :** The election system must be easily operable and stable.
- **Practicality and scalability :** The system should be easy to understand, practical to use and flexible to deploy for large scale elections.

All the aforementioned limitations are not easy to satisfy simultaneously. There are some conflicted limitations and they should be solved by using special techniques or trade off between the limitations must be considered. "Privacy of the vote" aims to release the link between the vote and voter while "individual verifiability" needs a

connection in between. “Privacy of the voter” provided by encrypting the vote using a publickey cryptography while digital signature on the encrypted vote ensures the “secrecy of the vote”. Using this technique provides “individual verifiability” to the system. On the other hand, using double envelope method exterminates the relation between the vote and voter during counting process. Another conflict is between “receipt-freeness” and “verifiability”. While a receipt is required for verification, the system is expected to be receipt-free. We solved this problem by encrypting a voter defined verification parameter using symmetric key cryptography.

1.1 Previous Work

After the 2011 elections in Estonia, Heiberg et al. published a paper discussing new attacks and weaknesses resulting from client and server side weaknesses [18, 20]. The designers of the Estonian i-voting system improved it by adding a verification mechanism. Like in the Norwegian i-voting scheme, using SMS services as a post channel was a possible solution; however, not all citizens may register their mobile numbers. Furthermore, the post channel mechanism was not only rather expensive, it also had various problems, as already seen in the Norwegian election system [32]. After a period of research and analysis, it was agreed that an individual verification mechanism using smart devices without requiring any personal information would be the most suitable verification channel for the Estonian i-voting system [20].

The designers of the Estonian i-voting system claim that it was as reliable and secure as the conventional election [10]. Contrary to their security claims, in [31], Springall et al. reported that the system is plagued by serious procedural and architectural weaknesses enabling client-side attacks that skew the results of the election undetectably bypassing the ID card system and smart device verification mechanism. Additionally, it is claimed that there are several inadequate procedural controls, lax operational security, insufficient transparency and several vulnerabilities in the published code. Moreover, in the same work, Springall et al. implemented a mock election in which they experienced both client and server side attacks. In responses the authors presented their recommendations on how to eliminate inadequate procedural controls and lax operational security weaknesses. In [19], Heiberg et al. researched ways to eliminate transparency weaknesses using an auditing mechanism.

The future brings more security and privacy risks for mobile devices [16, 22, 38]. For example, users can be fooled into installing malicious applications on their devices or to grant unauthorized remote access [16, 22, 35, 38, 7, 2, 13, 15]. Hence, an adversary can easily identify the owner of the smart device via private information, such as one’s IMEI number, location, contacts, phone number, emails, and photos. More specifically, IMEI numbers might also be required to be record into a central system beforehand which are used to identify and authenticate the mobile device whenever there is a connection request to a carrier. Those IMEI numbers not recorded into the system can be banned from communicating (e.g., [21]). For these reasons, one should never be able to obtain any information about the intention of a voter from the voting details on the verification device .

1.2 Contributions

In this thesis, we point out an important privacy issue in the verification system of the Estonian i-voting system. The motivation of our attack comes from the fact that all voter details including the real vote are displayed by the verification device. We stress that if the smart device running the verification application is corrupted, then vote privacy can be easily compromised by sniffing the voting verification process on a device of which the voter is most probably the owner. In fact, smartphone users generally install mobile applications without paying attention to potential security or privacy issues. Therefore, assuming the corruption of a smart device is relevant due to the huge number of increase in malwares during the last years [16, 22, 35, 38, 7, 2, 13, 15]. Hence, it is possible for an adversary to acquire an IMEI number and other private information, such as location, contacts, phone number, emails, and photos from smart devices including voting details, thereby compromises the voters' privacy.

The goal of this thesis is to mitigate the current privacy leakage of the Estonian i-voting verification mechanism. In this respect, we propose a new, privacy-preserving, and an efficient verification mechanism even in the case that a corrupted verification device is used. Our proposal is quite practical since only a few additional symmetric encryptions on the verification device is performed. Secondly, the secrecy of the election results may also be violated within a wide range attacks. Specifically, attacker may obtain information about the partial results of the election before it has concluded. Heiberg et al. stated that any sufficiently strong verification mechanism facilitates both vote selling and coercion [20]. In this thesis, however, our updated verification mechanism ensures the same security level without leaking any information about the vote.

1.3 Organization

The rest of the thesis is organized as follows. In Chapter 2, the necessary preliminaries, underlying cryptographic mechanisms and the components of the system are explained in detail. We start with the symmetric key encryption and hash functions. Since we propose a mechanism using symmetric systems, we include the advanced encryption standard. Then, we briefly give the properties of the cryptography hash functions. The basic idea of public key cryptography and digital signatures are also introduced in this chapter. Moreover, we present decryption mix-net, re-encryption mix-net, El-Gamal encryption scheme, TLS, ID card, threshold key management, double envelope voting method, and QR codes in this section.

In Chapter 3, current Estonian i-voting election, security and threat model of the system are explained. More specifically, we present system architecture and participating parties, central system, Estonian I-voting protocol, security analysis of Estonian I-voting system, a new potential privacy issue with the Estonian verification mechanism, and complete set of attack scenarios.

The proposed verification mechanism is explained in Chapter 4. This section begins with a warmup proposal. Then, our main proposal, verification mechanism with addi-

tional symmetric encryption, is described. This section also includes security analysis and complexities of the mentioned systems compared.

Finally, in Chapter 5, updated malicious scenarios are given and thesis is concluded.

CHAPTER 2

Preliminaries

In this section, we will present the general setup and symbols needed for presenting our protocol.

2.1 Symmetric Key Encryption and Hash Functions

Symmetric key cryptography algorithms use the same secret key, assumed to be shared beforehand, for both encryption and decryption processes. Even if key sharing seems to be a drawback in comparison to public key schemes, symmetric key systems are much faster. The security of these systems relies on computationally infeasible key spaces. We are going to denote a symmetric key encryption process with $\mathcal{E}_{\mathsf{sym}} = \mathsf{SymEnc}_k(M)$ and decryption with $M = \mathsf{SymDec}_k(\mathcal{E}_{\mathsf{sym}})$, where k is a secret key and M is a plaintext to be encrypted.

A cryptographic hash function is a one-way function which is used to create fixed length digest or tag for an input data with variable length. We are going to denote the hashing process with $\mathsf{H}(M)$, where M is a message. AES-256 and SHA3-256 can be utilized for symmetric encryption and hash function, respectively [26, 27].

In this thesis, we are going to utilize symmetric key systems and hash functions. Even though any secure symmetric key system and hash function, which comply the key length requirements, are compatible to our proposed protocol, AES-256 and SHA3-256 can be utilized, in which case one would have $SymEnc_k(M) :=$ AES-256$_k(M)$ and $\mathsf{H}(M) :=$ SHA3-256(M).

2.1.1 Advanced Encryption Standard (AES)

In 1997, the National Institute of Standards and Technology (NIST) started a competition and Rijndael by Joan Daeman and Vincent Rijmen was selected as the advanced encryption algorithm (AES). In this algorithm, there are three different key sizes that are 128, 192, and 256 bits. The algorithm with a 128 bits key has 10 rounds, the one with a 192 bits key has 12 rounds, and the one with a 256 bits key has 14 rounds. Note that each round has a round key that is generated by the original key. Moreover, a

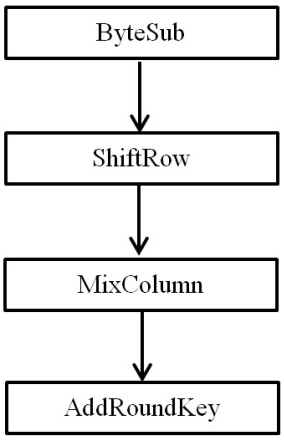
round gets a 128 bits output, and yields a 128 bits output.

AES consists of four basic steps. These steps are defined as the layers and they are the basics of the rounds.The layers are the followings:

- The ByteSub Transformation (BS)
- The ShiftRow Transformation (SR)
- The MixColumn Transformation (MC)
- The AddRoundKey Transformation (ARK)

A round of the AES algorithm is given in Figure 2.1. A round of the algorithm starts with the ByteSub transformation. Then, ShiftRow, MixColumn, and AddRoundKey transformations are applied.

Figure 2.1: A round of the AES algorithm



Now, we explain the details of transformations. First, we explain how to represent the inputs. The algorithm takes a 128 bits input, and it is embedded into a 4×4 matrix of bytes. It is given in the following matrix:

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}.$$

In this matrix, each entry is a byte and totally there are 16 bytes that is equivalent to 128 bits. So, each block of size 128 bits is represented as a 4×4 matrix of bytes. Note that there is a one to one relation between a byte and an element of the finite field $GF(8)$. In AES algorithm, elements of the matrix are seen as the elements of $GF(8)$ where the irreducible polynomial of $GF(8)$ that defines the finite field is

$$x^8 + x^4 + x^3 + x + 1,$$

and the product of the elements of $GF(8)$ is reduced to this reduction polynomial in order to multiply these elements.

The ByteSub transformation takes each element of the state matrix and transform to the another element of the $GF(8)$. The procedure works as follows: First, the entry of the matrix is converted to an element of $GF(8)$. This is a straightforward process and the bits of the byte that is converted are considered as the coefficients of the polynomial that is the representation of the elements of the finite field. Then the inverse of this element in $GF(8)$ is computed and $x^8 + x^7 + x^3 + x^2$ is added to the result.

In the ShiftRow transformation, the elements of the matrix obtained by applying the ByteSub transformation are shifted cyclically to the left with the following order: The first row is not changed. The second row is shifted to the left by one, the third one is shifted to the right by two, and the third one is shifted to the left by three. This is shown as follows:

$$ShiftRow\left(\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}\right) = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,1} & a_{1,2} & a_{1,3} & a_{1,0} \\ a_{2,2} & a_{2,3} & a_{2,0} & a_{2,1} \\ a_{3,3} & a_{3,0} & a_{3,1} & a_{3,2} \end{bmatrix}.$$

The next transformation is the MixColumn. In this transformation, the current state matrix that is the output of the ShiftRow transformation is multiplied from left by the following fixed matrix:

$$\begin{bmatrix} 00000010 & 00000011 & 00000001 & 00000001 \\ 00000001 & 00000010 & 00000011 & 00000001 \\ 00000001 & 00000001 & 00000010 & 00000011 \\ 00000011 & 00000001 & 00000001 & 00000010 \end{bmatrix}.$$

The next step is to add the round key to the output of the MixColumn. This step is called the RoundKey addition. In each round, we use a different key that are derived from the original key. In order to establish the round keys, first we arrange a 4×4 matrix of bytes including the original key. This matrix is then enlarged by adding 40 more columns. Let $W(0), W(1), W(2), W(3)$ the columns obtained form the original key. We obtain the other columns by using the following recursions

$$W(i) = W(i-4) \oplus W(i-1)$$

if i is not 4ℓ for a positive integer ℓ, and

$$W(i) = W(i-4) \oplus T(W(i-1)),$$

if i is 4ℓ for a positive integer ℓ. Note that T is a transformation. If a, b, c, d are the columns of the $W(i-1)$, then T first shifts these to obtain b, c, d, a. Then, these elements are applied to ByteSub. Let the result be e, f, g, h. Lastly, the round constant is computed as follows:

$$r(i) = 00000010^{(i-4)/4}$$

in $GF(2^8)$. Then, we obtain

$$T(W(i-1)) = (e \oplus r(i), f, g, h).$$

By using this method, we obtain the all new columns.

The encryption starts with ARK. Then, a round of encryption consists of ByteSub (BS), ShiftRow (SR), MixColumn (MC), AddRoundKey (ARK) in the given order. Note that in the last round, we apply BS, SR, ARK, i.e., we do not apply MC in the last round. On the other hand, in order to decrypt the cipher text using AES algorithm, the inverse of the each transformation is applied to the cipher text in the reverse order. The inverse of the BS is called InvByteSub (IBS), the inverse of SR is called InvShiftRow (ISR), the inverse of MC is called InvMixCoulumn (IMC) , and the inverse of ARK is itself. The decryption begins with ARK, ISR, IBS. Then we apply ARK, IMC, ISR, IBS in all other rounds. Finally, ARK is applied.

For more details, we refer to [34].

2.1.2 Cryptographic Hash Functions

A cryptographic hash function takes a message of arbitrary length and yields a fixed length output called message digest. There are three main properties of cryptographic hash function. The first one is that it must compute the hash of a message fast. The other property is that it must be one-way function. This means that given a hash value y, it should be difficult to find a message m whose hash value is y. Finally, it must have the strong collision free property. This means that it must be computationally hard to find two messages whose hash values are the same.

There are several popular hash functions that are widely used in cryptographic applications. The well known algorithms are MD5, SHA-1, SHA-2, and SHA-3. Cryptographic hash functions are used in password storage and data integrity check.

2.2 Public Key Encryption and Digital Signature

In public key cryptography, each user has two keys called public and secret. The public key of the user is known by everybody while the secret key must be kept secret by the user. Making the public key known should not reveal any information on the secret key. In order to provide such a relation between these keys, computationally intractable problems in mathematics are used. Two of such hard problems widely used in cryptography are integer factorization problem (IFP) and discrete logarithm problem (DLP). Finding all prime factors of an integer n is called IFP. On the other hand, for a given cyclic group G, its generator g, and an element $h \in G$, DLP is to find the integer k such that $h = g^k$ in G. The multiplicative group of finite fields and a cyclic subgroup of the group of elliptic curve points over finite fields are generally used in DLP based cryptography. Some of the well known public key algorithms based on IFP are RSA and Paillier. On the other hand, ElGamal encryption algorithm is an example for the public key cryptography based on DLP.

Public key systems are not as efficient as the symmetric systems. Therefore, public key

systems are generally used for encryption of small size messages such as a symmetric encryption key or a digital signature. Encryption, decryption, generation and verification of digital signatures with public key algorithms work as follows: Let A and B two people who want to communicate. Assume that A wants to send a message M to B. For encrypting M, A uses the public key of B and encrypts M with the public key of B. In order to decrypt, B uses his own secret key. On the other hand, for authentication, A first reckons the hash value of the message, and then encrypts this hash value with her secret key to construct the digital signature. In order to verify the signature, B decrypts the signature using A's public key, and then compare it with the hash value of the message. If they are equal, the signature is validated. Otherwise, the message has not been sent by A or it has been altered during the transmission.

RSA algorithm is one of the frequently used public key algorithm in practical applications. In this system, each user has a public key pair denoted by (n, e) and a private key d. These parameters are generated as follows: First, two prime numbers p ans q are generated. It is suggested that in order to provide a secure systems they should be at least 1024 bits so that the RSA algorithm becomes 2048 bits. First, n is computed as $p \cdot q$. Then, an integer e is selected in a such way that the multiplicative inverse of it modulo $(p-1)(q-1)$ exists. This e is one of the other components of the public key pair. Then, the private key d is computed as

$$d \equiv e^{-1} \bmod (p-1)(q-1)$$

where d is the multiplicaitve inverse of e modulo $(p-1)(q-1)$. It should be noted that $(p-1)(q-1)$ is denoted by $\phi(n)$ and it is called Euler's phi function. Moreover, for e to be invertible, the greatest common divisor of e and $\phi(n)$ must be one, i.e., they are relatively prime to each other.

To encrypt a message m, we use the public key of the receiver. Since public key of the receiver is known by everybody, we obtain his public key pair. Let it be (n, e). Then, the cipher text c for a message m is computed as follows:

$$c \equiv m^e \bmod n.$$

In order to decrypt, the receiver uses the following decrption:

$$m \equiv c^d \bmod n.$$

In order to use the RSA algorithm in digital signatures, the one who wants to sign a message uses his own private key. To this end, first he computes the hash value h of a message m because it is computationally infeasible to use public key algorithm for long messages. Therefore, the hash value of the message is used rather than the message. Then, he computes

$$s \equiv h^d \bmod n$$

where s is the digital signature of the message. In order to verify the digital signature, the one who wants to verify the digital signature computes $s^e \bmod n$ and checks whether this is equal to the h or not. If they are equal then the digital signture is accepted, otherwise it is rejected.

One of the important notion in cryptography is the semantic security which enables deriving an information about some unknown message from the ciphertext of that message. Direct use of many cryptosystems do not satisfy semantic security. For example, RSA encryption maps a certain plaintext to a certain ciphertext. This property reveals some statistical properties of the ciphertext, and therefore RSA is a semantically insecure algorithm. In order to overcome this security issue, randomization techniques are used in the encryption algorithms. In order to make RSA semantically secure, Optimal Encryption Padding (OAEP) scheme is used . Moreover, Probabilistic Signature Scheme (PSS) is employed for RSA digital signature scheme to provide semantic security. The underlying public key encryption scheme of i-voting systems must be semantically secure (some well-known algorithms are RSA-OAEP [17], Paillier [28], and ElGamal [14]).

2.3 Mix-net Protocols

The aim of using mix-net is to shuffle the encrypted votes of k voters by erasing the relation between the votes and the voters and making the votes anonymous. In Estonian scheme, mix-net is used during double envelope method before sending the encrypted votes to Vote Counting Server (VCS) from Vote Storage Server (VSS). There are two types of mix-net protocols, namely decryption mix-net and re-encryption mix-net. The protocols are begin with an initial encryption E. After, there are several mix phases $mix_1, \cdots, mix_t$ applied where $t \in \mathbb{Z}$ is the number of mix phases which provides robustness. Every round of mix phases are begin with a secret permutation. In decryption mix-nets, mix phases are followed by partial decryption whereas in re-encryption mix-nets, mix phases are followed by re-encryption phases. In re-encryption mix-nets a final decryption phase is applied.

The protocols are given in detail below:

2.3.1 Decryption Mix-net

Every mixing step of decryption mix-net has a public and private key pair (sk, pk), i.e., for every mix step i, mix_i has the key pair (sk_i, pk_i). Using the key pair, in every step, input data is decrypted and then secretly permuted. The permuted data is the input of the next mix step.

The initial encryption E has all the public keys $pk_1, \cdots, pk_t$ of every mix steps. Each input encrypted by the public key pk_i, $i = t$ to 1. In other words, if we denote a ballot by B_i,

$$C_i = E(B_i) = E(pk_1 \cdots E(pk_{t-1}, E(pk_t, B_i))).$$

Note that, decryption mix-net protocol neither verifiable nor robust.

2.3.2 Re-encryption Mix-net

A re-ecryption mix-net begins with an initial encryption phase E, then followed by several mix phases $mix_i, \cdots, mix_t$ which mix by scrambling, re-encrypting, and a final decryption D. El-Gamal Encryption has a property that suits re-encryption. So, in general El-Gamal encryption scheme is used for encryption.

2.3.2.1 El-Gamal Encryption Scheme

Public Key: (p, g, y)
Secret Key: x such that $g^x = y(modp)$
Message to be encrypted: m
Encrypted message: (g^r, my^r), $r \in_R \mathbb{Z}_q$, Q is a large prime dividing $p-1$, g is a generator of the subgroup of elements whose orders divides q and all operations are done modulo p.

Note that, Re-encryption of an encrypted message $(a, b) = (g^r, my^r)$ by a random number $s \in_R \mathbb{Z}_q$ is $(ag^s, by^s) = (g^{r+s}, my^{r+s})$. This re-encryption preserves the message same while encrypted form of the message completely different.

2.3.2.2 El-Gamal Based Re-encryption Mix-net

1. An El-Gamal public-key (p, g, y) is generated in a distributed manner.
2. (a) Encrypt all the ballots $B_1, \cdots, B_k$ by using the El-Gamal encryption with the public key (p, g, y), $C_{(i,0)} = El - Gamal_{(p,g,y)}(B_i)$ for $i = 1, \cdots, k$.
   (b) Send $(C_{(1,0)}, \cdots, C_{(k,0)})$ to bulletin board.
3. For $i = 1$ to t, re-encrypt each ciphertext $(C_{(1,i-1)}, \cdots, C_{(k,i-1)})$ with the step key pair (pk_i, sk_i) and permute the resulting ciphertexts using a secret permutation.
4. After the last step, decrypt the ciphertexts $(C_{(1,t)}, \cdots, C_{(k,t)})$ in a distributed manner where t achieve robustness.

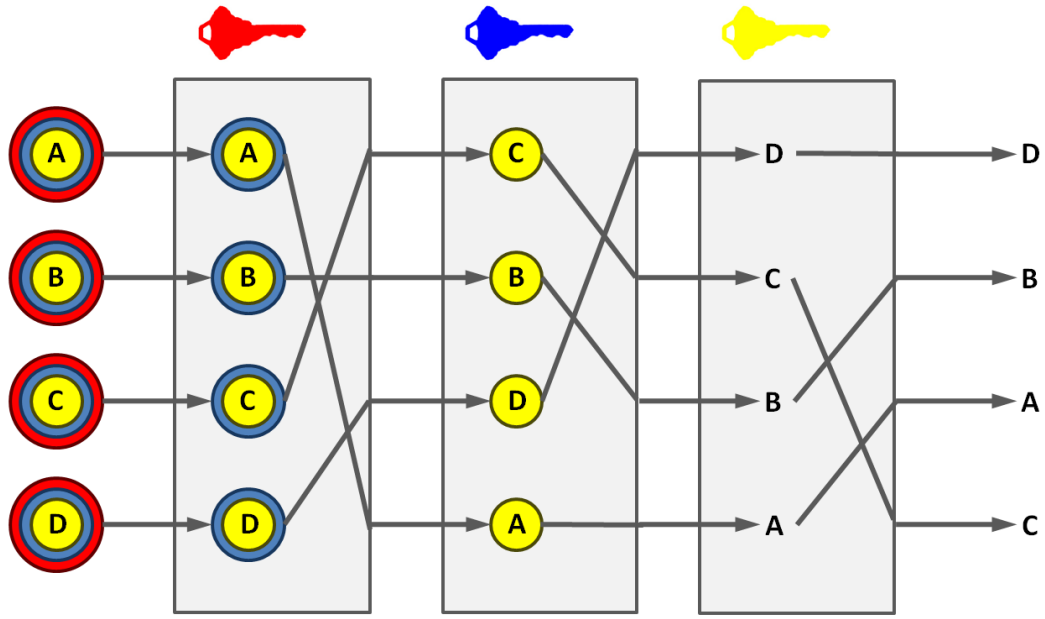
Achieving Verifaibility and Robustness During the protocol, every mix server prove that i has done the correct operation. In other words, each mix_i i required to prove that

$$C_{(j,i)} = El - Gamal(C_{\pi(j),i-1}) \text{ for } j = 1, \cdots, n$$

where π is a permutation and $El - Gamal$ is a El-Gamal encryption with the key (p, g, y). In Figure 2.2, general mix-net protocol is given [1].

Suppose, a ciphertext $C_2 = (\alpha_2, \beta_2) = (g^u, m_2 y^u)$ is a re-encryption of a ciphertext $C_1 = (\alpha_1, \beta_1) = (g^t, m_2 y^t)$.

Figure 2.2: General Mix-net Protocol



Definition 2.1. A tuple (g, y, g^r, y^r) is a DDH tuple if it is of the form (g, g^x, g^r, g^{rx}) [25].

Remark 2.1. c_2 is a re-encryption of c_1 if and only if $(g, y, \frac{\alpha_2}{\alpha_1}, \frac{\beta_2}{\beta_1})$ is a DDH tuple.

Proof. Consider the tuple where $y = g^x$

$$(g, y, \frac{\alpha_2}{\alpha_1}, \frac{\beta_2}{\beta_1}) = (g, g^x, g^{u-t}, \frac{m_2}{m_1}(g^x)^{u-t}).$$

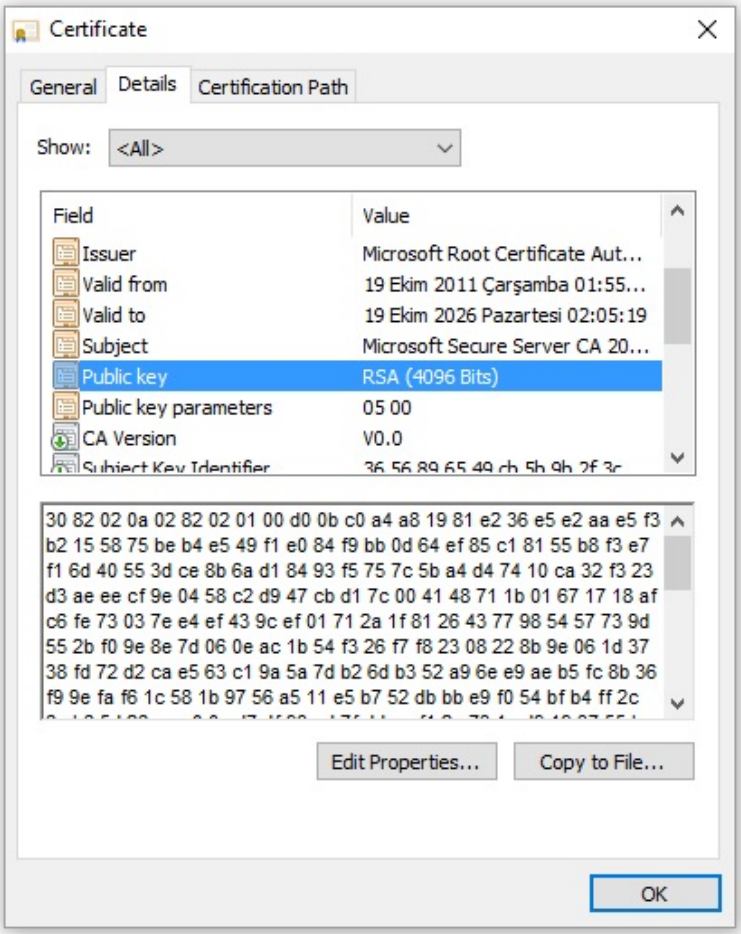
If $m_2 = m_1$, then $(g, g^x, g^r, (g^x)^r)$ is DDH where $r = u - t$ □

For more details, the reader may refer to [8, 9, 23, 3, 30]

2.4 TLS

In order to use the public key cryptography in practical applications, the management of public keys is essential. To this end, a public key infrastructure (PKI) has been developed. In this system, certificates and validation operations are conducted. For each entity or individual, a certification is formed by a certification authority (CA). Certificates include identity information and the public key, and this information is signed by CA. One type of certificate is the X.509 that requires each user has a certificate and Figure 2.3 shows an example of such certificate.

Figure 2.3: An example of X.509 certificate



One can easily obtain the public key of an entity or individual by decrypting this signature using the public key of CA. This system is commonly used in the HTTPS communication securely using Secure Socket Layer (SSL) and its slight modification Transport Layer Security (TLS). These protocols provide secure communication by encrypting messages with symmetric algorithms. In order to establish the common key for this symmetric system, the handshake protocol which employs the public key algorithms is realized. Therefore we may assume that the communication between *VotingApp* and *VFS* in Estonian system is a secure channel and transmitted data are not known to any attacker.

2.5 ID Card

In Estonia, every citizen has an ID card with cryptographic abilities to authenticate and access to the governmental services [6]. In Estonian Scheme, the public and secret

keys of the *Voter*s are provided by national ID-cards where the secret key of the *Voter* sk_V is stored in the ID-card and it is assumed that it is not possible to extract the key from the card. That is why the ID cards are used to authenticate the election server and to sign the ballots using pk_V and sk_V as well.

2.6 Threshold Key Management

Each of the election specific public and private key pair (pk_S, sk_S) of the *Central Server* is generated by independent parties and auditors using m of n threshold cryptography which ensures that at least m of them must cooperate in order to re-generate the same key again. The secret key of the Central System, sk_S, is stored in an Hardware Security Module (*HSM*) isolated from the internet by an air gap and the election specific public key, pk_S, is made public and embedded in a voting application *VotingApp* and a verification application VerifApp.

Similarly, $(pk_\mathcal{V}, sk_\mathcal{V})$ denotes a public and private key pair of a user $\mathcal{V}$. $E_{\mathsf{asym}} = \mathsf{AsymEnc}_{pk_\mathcal{V}}(M)$ denotes a randomized encryption of a message M using the public key of the $\mathcal{V}$. Moreover, $\mathsf{Sign}_{sk_\mathcal{V}}(M)$ denotes the signature of a user $\mathcal{V}$ on a message M using the private key of the $\mathcal{V}$.

At the end of the election, sk_S must be exterminated since it may cause a privacy leakage by disclosing the cast votes.

2.7 Double Envelope Voting Method

Double envelope voting methods is one of the simplified methods which also gives at least the same security level with the traditional envelope method. It is very important to design it similar to the traditional method since the public confidence in the election process should remain strong. That is why in Estonian i-voting, Double Envelope Scheme is designed to be as close as the traditional scheme which ensures the anonymity of the votes and eligibility of the voters.

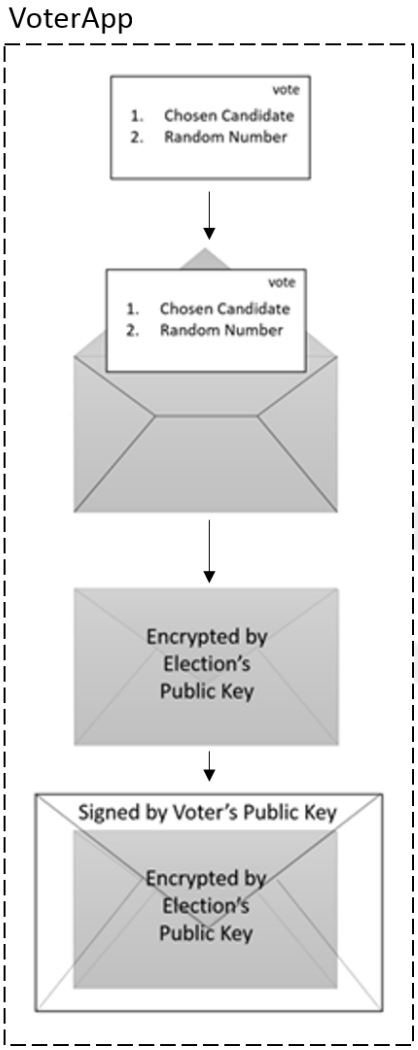
In the former method,

1. The ballot is put into a blank inner envelope without any identity info.
2. Inner envelope is placed in an outer envelope which has the identity info of the voter.
3. The envelope is sent to the voter's polling station.
4. First the eligibility of the voter is checked.
5. If the voter is eligible, the inner and outer envelopes are separated.
6. Then the inner envelope is put into the ballot box.

7. The ballot box is shuffled.

8. Anonymous inner envelopes are opened and counted.

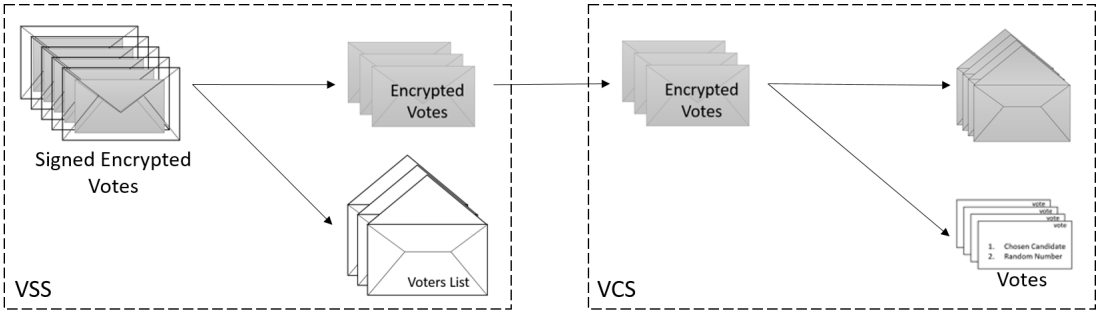
Figure 2.4: Double Envelope Vote Generation in Voter Application



On the other hand, double envelope voting scheme requires two key pairs to protect the vote secrecy and voter privacy. The first key pair is the election's public-secret key pair. Public key of the election embedded in the voter application will be used to encrypt the vote to ensure the security of the vote while election's secret key is stored in a secure *HSM* in the *NEC*'s *VCS*. The other key pair is the public-secret keys of the voter. Public key of the voter will be used to sign the vote to prove the identity of the voter while the public key of the voter already public and also stored in election committee's server.

Double envelope voting method is used during the ballot generation and during the voter verification. The ballot first encrypted by election's public key to ensure the security then signed by voter's secret key to identify the voter. So, during counting

Figure 2.5: Double Envelope Vote Counting in *VCS*



process, it is easy to find who the voter, even though it is not possible to find what the vote is as in the double envelope method.

As shown in Figure 2.4 and 2.5, the Double Envelope Voting Method can be expressed as follows:

1. Voting application encrypts the vote consisting chosen candidate and random number chosen by the voter by election's public key.
2. Voting application signs the encrypted vote by voter's secret key.
3. Signed and encrypted vote sends to the Vote Storage Server (*VSS*) through VFS.
4. First the eligibility of the voter is checked by public key of the Voter. If the voter is eligible, only the last vote is used for the rest of the counting process.
5. After signature verification, encrypted vote is stored in *VSS* anonymously.
6. Anonymous encrypted votes are shuffled by a permutation such as Mixnet Operation [24].
7. Shuffled encrypted anonymous votes are transmitted to *VCS* using a media such as DVD.
8. Anonymous encrypted votes decrypted by the election specific secret key.
9. Finally, the votes are counted.

2.8 Quick Response (QR) Codes for Verifiability

QR codes are commonly used for storing all type of data in a small area with a high-speed reading capacity and error correcting capacity using Reed-Solomon Codes.

QR codes are established by black and white dots namely modules. The number of modules contained in the *QR code* decides the data capacity of that code. So, the size of the *QR code* must be decided based on the requirement of the data stored. There are

several number of different *QR code* types depending on where it will be used; it is also called Module Configuration. Reader may refer to the standards defined by *ISO/IEC 18004:2015* [37].

2.8.1 QR Code Types.

There are 5 main types of *QR codes* for different purposes:

Figure 2.6: QR Code of Types 1 and 2. The Figure is taken from [36]

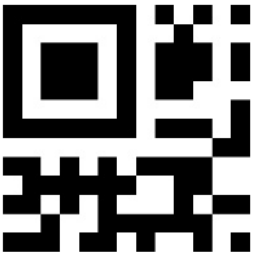
1 *QR Code Type 1 and 2*: It is the most frequently used *QR code* type which can be readable from any direction through position detection patterns located at the three corners of the code. There are two types with the same appearance shown in 2.6. The main difference is the capacity of the code they can store.

   (a) *QR Code Type 1*: The original QR code type. The largest version can store 1 to 167 numerals, which is version 14 and has 73 by 73 modules.

   (b) *QR Code Type 2*: It is the improved version of model 1 since the capacity demand of model 1 is not enough for some cases. There are 40 different module configurations from the minimum sized version 1 containing 21 by 21 modules to the minimum sized version 40 containing 177 by 177 modules which is capable of storing 7,089 numerals. Every version contains 4 more modules than previous version, i.e., version 2 contains 25 by 25 modules. Currently, the term *QR code* refers to model 2. The standardization of *QR code* is defined in *ISO/IEC18004*.

2 *Micro QR Code*: Though it does not support the readable from any direction property, this type requires smaller coding area than *QR Code*. The main weakness of *Micro QR code* is it can store at most 35 numerals. The standardization of *Micro QR code* is defined in *JIS X 0510* and shown in 2.7.

Figure 2.7: Micro QR Code. The Figure is taken from [36]



Micro QR Code

3 *iQR Code*: This type of code allows to store data smaller than *Micro QR code* and larger than *QR Code Type 2*. *iQR codes* do not necessarily a square shapes, it may be in a rectangular shape. In other words, the size of the code can be arranged up to the size of the free space where it will be placed. A sample of iQR code is shown in 2.8. The largest size of *iQR code* can store 40,000 numerals and has 422 by 422 modules.

4 *SQRC*: This type of codes looks the same with *QR Codes* can store secret and public part separately. Even though the code can only readable by specific types of scanners, it cannot guarantee the security of the stored data. A sample of SQR code is shown in 2.9.

5 *Frame QR Code*: In the center of the code, there is a canvas area in which you can place every type of shape of graphics that does not interfere the readability of the code. This type of codes are better to use in promotional tools. A sample of Frame QR code is shown in 2.10.

2.8.2 Error Correcting Capability

QR Codes have different usage purposes. So, there is an error correction capability to restore the data even if the code is dirty or damaged. Error correction capability is the implementation of Reed-Solomon Code to the data which is intended to send correctly. There are four different error correction levels, namely L, M, Q and H. For the purpose of the i-voting, it is better to use the lowest error correction level L can be adjusted to the needed data capacity. In Table 2.1 you can find the error correction levels for total codewords according to the operating environment.

Figure 2.8: iQR Code. The Figure is taken from [36]

iQR Code

Table 2.1: QR Error Correction Capability

level L	Approx 7 %
level M	Approx 15 %
level Q	Approx 25 %
level H	Approx 30 %

2.8.3 Deciding the Version of QR Code

It is important to decide the version number of QR code up to the required data capacity. You can find the steps about how to determine the size of the QR code in the following:

1. Choose a type of the data to be coded.
2. Choose a data correction level up to the environment the QR code will be used.
3. Find the required size of the data at the intersection of the data correction level and data type in the table.

As an example of determining the version number of 56 characters is given in Table 2.2.

Figure 2.9: SQRC. The Figure is taken from [36]

Figure 2.10: Frame QR Code. The Figure is taken from [36]

Table 2.2: Table of QR Code Version Number [36]

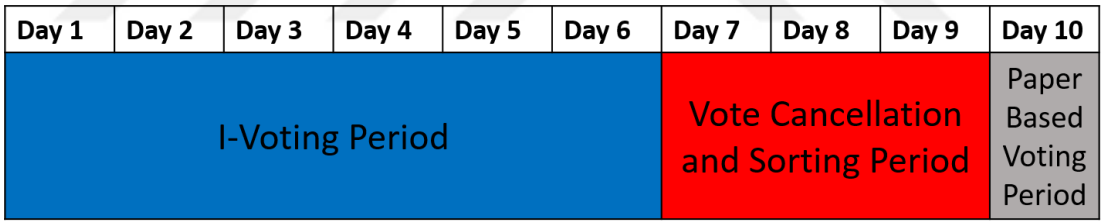
Version	Modules	ECC Level	Data bits	Numeric	Alphanumeric	Binary
1	21x21	L	152	41	25	17
		M	128	34	20	14
		Q	104	27	16	11
		H	72	17	10	7
2	25x25	L	272	77	47	32
		M	224	63	38	26
		Q	176	48	29	20
		H	128	34	20	14
⋮	⋮	⋮	⋮	⋮	⋮	⋮
10	57x57	L	2192	652	395	271
		M	1728	513	311	213
		Q	1232	364	221	151
		H	976	288	174	119
⋮	⋮	⋮	⋮	⋮	⋮	⋮
40	177x177	L	23648	7089	4296	2953
		M	18672	5596	3391	2331
		Q	13328	3993	2420	1663
		H	10208	3057	1852	1273

CHAPTER 3

Estonian Internet Voting Protocol and its Security Analysis

Estonian elections begin with announcing the voter lists and candidate lists one and half weeks before the voting period starts. Voting period is divided into i-voting and paper ballot periods. I-voting period is seven-day long. During i-voting period, voters are allowed to vote as many times as they want. In other words, the voters can change or replace their votes. Before the paper ballot period, multiple votes and the votes of ineligible voters are excluded from the lists. For the paper ballot period, i-voter lists are sent to the polling divisions to replace the paper ballots by i-votes. When the voting period is over, all the votes are tallied and results are announced [11]. You can find the voting schedule of the elections in Figure 3.1.

Figure 3.1: Election Schedule



3.1 System Architecture and Participating Parties

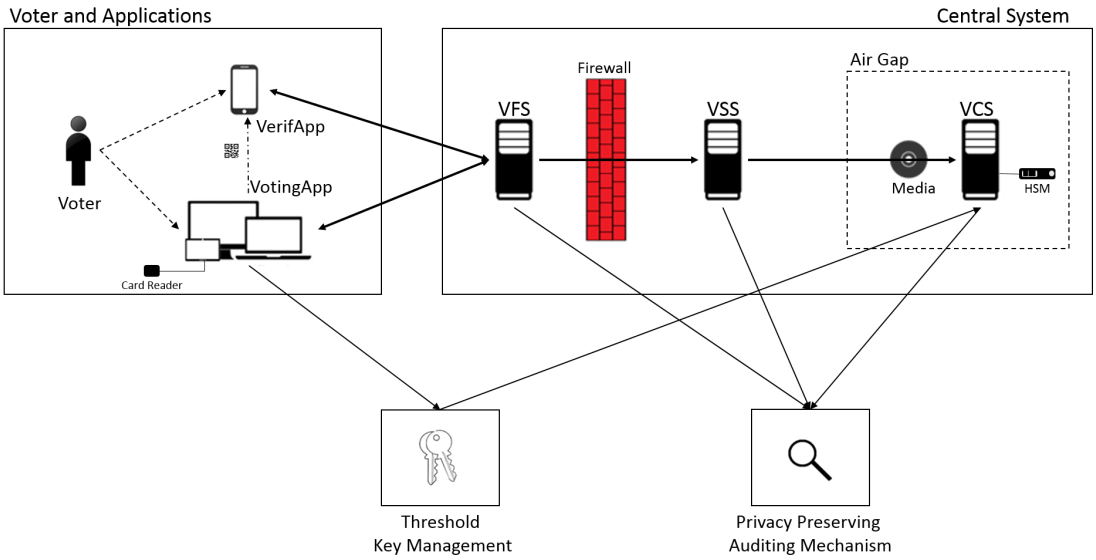
Estonian i-voting has been distributed to four parties to realize the system, which are explained briefly in the following subsections:

3.1.1 Voter, Voting Application (*VotingApp*), and Verification Application (*VerifApp*)

1. **Voter** ($\mathcal{V}$): Voters denoted by $\mathcal{V}$ are the citizens of Estonia who are allowed to vote in the elections according to electoral legislation.

2. **Voting Application** (***VotingApp***): The applications that $\mathcal{V}$s use for voting purposes on their PCs via a smart card reader and the application is published by

Figure 3.2: General Architecture



National Electoral Committee (*NEC*). This application can be downloaded before the elections for the related operating systems with the election specific public key pk_S and TLS certificate to establish a secure connection with the Vote Forwarding Server *VFS*. Every $\mathcal{V}$ uses her national ID card to identify herself to the polling commission through this application.

3. **Verification Application (*VerifApp*)**: *VerifApp* is to check the cast vote by an application installed on a smart device having an internet connection and a camera. The details of voting and verification process will be explained in the next subsection. Note that, although *VotingApp* is published by election authority, *VerifApp* may be published by any person or organization.

3.1.2 Central System

Central system has three main parts that are forwarding, storing and counting phases under the responsibility of *NEC*.

1. Vote Forwarding Server (*VFS*): It is the only server in the *Central System* that is accessible by outside applications. The $\mathcal{V}$ accesses the *VFS* to authenticate, gets the required data for voting and verification stages, receives the signed votes etc. In other words, it is the communication channel of the *Central System* with the $\mathcal{V}$s. *VoterAPP* and *VFS* authenticates via secure *TLS* connection.

2. Vote Storage Server (*VSS*): After voting and verification process, the votes, voter identity and all related info about the votes are stored in the storage server after checking the eligibility of the votes and voters. In other words, *VSS* is the storage

of the votes during the voting period of the election. At the end of the period, it removes double votes and cancels ineligible voters. After that, votes are prepared for the counting server.

3. Vote Counting Server (*VCS*): It is an offline server of the *Central System*. Election specific sk_S are stored in this server by a *HSM* to provide security to the keys and counting phase. Since it is the only server which has the sk_S, it is separated from the *VFS* and *VSS* by an *air gap*. Therefore, the votes are transferred to *VCS* by a DVD after removing digital signatures from the votes, i.e., anonymized votes are transfered to *VCS* on a media. In this way the relation between the voters and votes are eliminated.

3.1.3 Privacy Preserving Auditing Mechanism

There is an auditing mechanism in which all the processes and results of the election can be checked by an independent committee while the privacy of the voters are preserved. Since it is out of scope of this thesis, the issue will not be mentioned here. The reader can find the details in [19].

3.1.4 Key management

The election specific keys are generated by m of n threshold key management as explained in 2.6. Besides, a voter $\mathcal{V}$ keys are managed by national ID cards 2.5. Details can be found in [6].

3.2 Estonian I-Voting Protocol

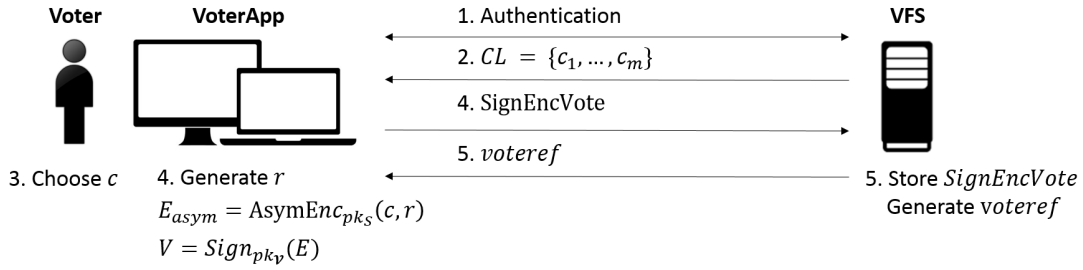
For simplicity of describing the voting protocol, we divide it into voting and verification phases which are explained in detail below.

3.2.1 Voting Stage:

The voting phase begins by *VotingApp* authenticating the *VFS* via a *TLS connection*. A voter $\mathcal{V}$ receives the related candidate list $CL = \{c_1, \cdots, c_m\}$ where c_i's are candidates' unique identity values and m denotes the number of candidates. Next, $\mathcal{V}$ chooses a candidate $c \in CL$ to cast the vote. *VotingApp* generates a signed and encrypted vote $\mathsf{SignEncVote} = \mathsf{Sign}_{sk_\mathcal{V}}(E_{\mathsf{asym}})$ where $E_{\mathsf{asym}} = \mathsf{AsymEnc}_{pk_S}(c, r)$ and $r \in \{0,1\}^k$ is a random number, $k \in \mathbb{N}$. Next, *VotingApp* sends $\mathsf{SignEncVote}$ to *VFS*, and then receives a vote reference *voteref* which is a receipt to be used in the verification phase.

1. A voter $\mathcal{V}$: Authenticates to *VFS* through *VotingApp* using a national ID Card.

Figure 3.3: Voting stage of the Estonian i-voting protocol



2. *VFS*: Sends $CL = \{c_1, \ldots, c_m\}$ to *VotingApp* where m is the number of candidates.
3. $\mathcal{V}$: Chooses c from CL
4. *VotingApp*:
   (a) Generates a random number r.
   (b) Encrypts c and r by pk_S, $E_{\text{asym}} = \text{AsymEnc}_{pk_S}(c, r)$.
   (c) Signs E_{asym} by $sk_{\mathcal{V}}$, i.e. $\text{SignEncVote} = \text{Sign}_{sk_{\mathcal{V}}}(E_{\text{asym}})$.
   (d) Sends SignEncVote to *VFS*.
5. *VFS*:
   (a) Stores SignEncVote.
   (b) Generates *voteref*.
   (c) Sends *voteref* to *VotingApp*.

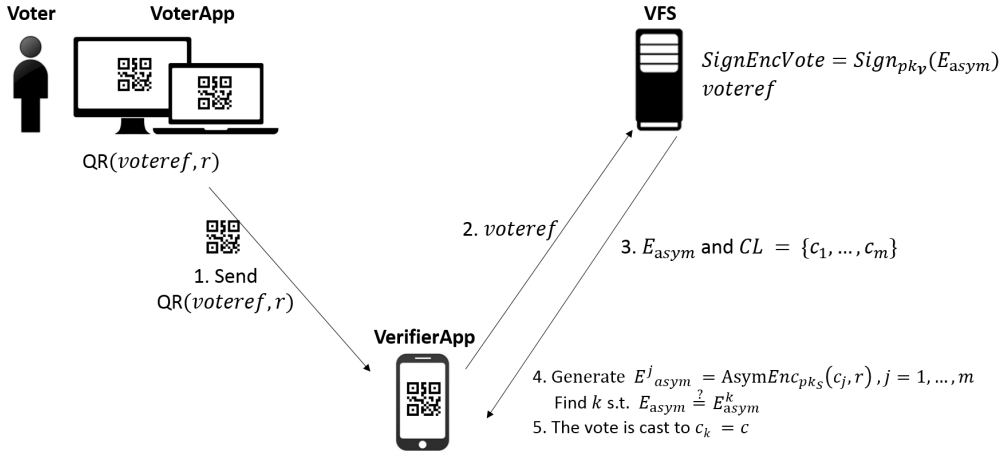
3.2.2 Verification Stage:

Note that the verification stage is optional and only ensures whether the vote has been correctly stored in *VFS*. It is also important to note that for security purposes, *VerifApp* and *VotingApp* should not be installed on the same device. Additionally, it should be noted that *VerifApp* scans the *QR code* by camera instead of obtaining it via an internet connection.

During the verification phase, *VerifApp* receives r and *voteref* from *VotingApp*, request the related data from *VFS* by the *voteref*, and computes the vote. Finally, *VerifApp* shows the recorded vote on the screen. If the voter confirms the correctness of the cast vote then the voting procedure ends successfully, otherwise, $\mathcal{V}$ puts an alarm.

1. (a) *VotingApp*: Generates a *QR code* including r and *voteref* and show on the screen.
   (b) *VerifApp*: Scans *QR code* by camera.

Figure 3.4: Verification stage of the Estonian i-voting protocol



2. *VerifApp*: Sends *voteref* to *VFS*.
3. *VFS*: Sends E_{asym} and CL to *VerifApp*.
4. *VerifApp*:
   (a) Computes $E^j_{\text{asym}} = \text{AsymEnc}_{pk_S}(c_j, r)$ for all $j = 1, \cdots, m$.
   (b) Finds k such that $E_{\text{asym}} \stackrel{?}{=} E^k_{\text{asym}}$ for some $k \in \{1, \cdots, m\}$.
   (c) Shows c_k on the screen.
5. Voter $\mathcal{V}$: Checks $c_k \stackrel{?}{=} c$.
   (a) If $c_k = c$, the vote is received and stored correctly to *VFS*.
   (b) Else, $\mathcal{V}$ puts an alarm (which basically shows that malware is present).

In the next section, we will show that there is a practical and realistic attack that may become a real issue.

3.3 Security Analysis of Estonian I-voting System

A security analysis of the current i-voting system of Estonia is discussed in detail in [20, 31]. Estonian i-voting security model assumes that either *VotingApp* or the device that runs *VotingApp* is malicious. We note that the assumptions *VerifApp* and *VFS* collude maliciously or *VerifApp* and *VotingApp* collude maliciously are not realistic since the duty of *VerifApp* is to independently check the correctness of *VFS* and *VotingApp*. Furthermore, as noted in [5], limited number of corrupted voters' devices are accepted as a reasonable risk.

The main attack scenarios are about ballot integrity, the reliability of the voting system, and coercion resistance.

- **Manipulation Attacks.** Manipulation attacks are consist of modifications to a vote without the knowledge of the $\mathcal{V}$. These attacks can be performed by changing the vote to either a predetermined or a random candidate.

  Student's Attack. In Estonia's 2011 parliamentary elections, the *Student's Attack* exposed that neither ballot integrity and secrecy in the election was guaranteed [18]. The attack is based on installing a malware to the device that runs *VotingApp*. This malware is designed so that it undermines *VotingApp* and while the vote is being cast, it silently diverts or cancels the intended vote. At first, this attack was not considered as a thread (see [18, 31]), but then, a verification phase is integrated into the system [20]. In the papers [18, 31], it is also mentioned that there is still a *manipulation attack* risk if both *VotingApp* and *VerifApp* are malicious and work in a coordinated manner. In contrast, it is also stated that it is very hard to install malicious applications to different devices and if the attack realized in large scale, it can be detected easily. In both papers, it is concluded that, there is no known realistic manipulation attack risk to Estonian i-voting system.

  Ghost Click Attack. A malicious software that runs on the same device as *VotingApp* can obtain the *PIN code* of the *ID card* during the voting process. When the *ID card* is reinserted, the malware may re-vote silently without being detected [31]. Although this is an interesting attack,it is not included in the scope of this thesis.

- **Reliability Issues.** By decreasing voters' confidence in the i-voting system, an election can be held questionable without there being any violation to vote secrecy. Therefore, an effective verification mechanism must be set up to prevent such attacks [5]. In fact, fairness and the secrecy of the elections should be satisfied, that is in any paper based or electronic voting scheme, the election's partial or total results would not be revealed before the tallying process.

- **Coercion Attacks.** An i-voting system must prevent a voter from being able to prove to a coercer how he voted. Therefore, the system should provide receipt-freeness or coercion resistance. Allowing vote updates (i.e., re-voting) is the countermeasure to prevent such attacks. In the counting phase of Estonian i-voting scheme, the last vote is accepted as the voter's intention. Even if a voter is enforced to vote a specific candidate, coercer cannot be sure about whether re-voting has been done after or not. This makes vote buying and coercion inefficient since coercer can never be sure about whether voter re-vote again or not.

3.4 A New Potential Privacy Issue with the Estonian Verification Mechanism

Beside the above-mentioned attacks, we would like to emphasize that there is a potential privacy weakness about the verification mechanism. Even though adding a verification mechanism to the system in 2011 solved Student's attack risk, it may yield an unforeseen privacy weaknesses. In this thesis, we are focusing a new privacy issue

in which sniffing random number and encrypted vote data during verification phase is easy and instantly the cast-vote may be deduced.

Privacy measures are different than coercion measures. For coercion or vote buying, the coercer directly communicates the $\mathcal{V}$ and she knows who the coercer is and also aware of the coercion. But for the privacy issues, an attacker can get the data and deduce the chosen candidate while $\mathcal{V}$ does not perceive herself attacked. Another problem is that, coerced *Voters* may re-vote before the i-voting phases ended or in paper ballot phase. Despite that, choice of the candidate is up to $\mathcal{V}$'s own will and once it is revealed, there is no way to obscure it. Moreover, sniffing cast-votes wide range gives idea about the election results, this contradicts the *secrecy of the election results* property explained in Section 1. Additionally, for passive listening, it is enough to get data using any other malicious application installed on the device. Therefore, Estonian i-voting system needs additional countermeasures against privacy leakage especially for *VerifApp*. Considering all possible attack scenarios, Estonian Election System fulfills sufficient security level. In contrast, there are still some privacy weaknesses in the system which is explained in Section 3.

3.4.1 Privacy Attacks

Recall that *VerifApp* is integrated into the i-voting system to be resistant against *Student's Attack*. *VerifApp* can be freely developed by a *NEC*, by an open source community or even by one able to write her own verification software. In contrast, most citizens cannot develop or control the trustability of *VerifApp* even it is downloaded from a known source. It is not realistic to assume that *VerifApp* is honest. The Estonian i-voting system is designed in this manner and *VerifApp* explicitly outputs the voters' intention in plain form.

Even if *VerifApp* itself were to work properly, any malicious application running on the same device may sneak into the processes in an attempt to monitor inputs and outputs. Therefore, it would be sufficient to have any privileged application running on the same device [16, 22, 35, 38, 7, 2, 13, 15]. As soon as an adversary manages to install such a privileged application, the only thing that needs to be done is to grab a screen shot of the device while *VerifApp* displays the output to $\mathcal{V}$. After that, the cast vote will also be known by the adversary. Furthermore, since IMEI numbers and other private information (e.g, contacts, phone number, location, emails, photos) can be obtained by a malicious application loaded on the verification device, an adversary can obtain $\mathcal{V}$'s identity. Hence, privacy of $\mathcal{V}$ can be easily compromised. We highlight that this privacy issue may lead to coercion or vote buying (e.g., an adversary can force voters to install a malicious application on their smart devices for later check the their actual votes).

On the one hand, when considered on the individual level, this attack causes privacy leakage. On the other hand, when applied over a wide range, this attack may also promise the reliability and, more importantly, the secrecy of the elections. Because, as explained in [11], in any election the results would not be exposed before the counting process has been partially or totally completed. Therefore, the possibility of a

corrupted verification device must be included in the design criteria of the Estonian i-voting system, the current Estonian i-voting system requires additional countermeasures to guard against this privacy leakage.

3.5 Complete Set of Attack Scenarios

In Table 3.1, we show the complete list of possible scenarios in the case of malicious parties with possible countermeasures. More concretely, we analyze the scenarios as follows:

- **Malicious *VFS*.** In this case, there is a full privacy leakage where *VFS* may leak all stored information as a result of potential *Insider Attacks* [31]. As a countermeasure, auditing mechanisms with independent auditors can verify the computations of *VFS* [19].
- **Malicious *VotingApp* and its adversarial environment.** During the preparation of signed encrypted vote, a malicious *VotingApp* can change the voter's intention $c \in CL$ by c^* where $c \neq c^*$ and then sends encrypted form of c^* instead of c. During the verification, *VerifApp* will reveal that the ballot does not reflect the voter's own will as intended and puts an alarm. $\mathcal{V}$ re-votes again from another device.
- **Malicious *VerifApp* and its adversarial environment**: It is possible to sniff data from a smart device using any malicious application installed on the device [16, 22, 35, 38, 7, 2, 13, 15]. Leaking verified votes from the smart devices gives an opportunity to get some idea about the election results before the election ended. This attack violates one of the important limitation secrecy of election results of the Estonian i-voting system. Another attack is that using a device's IMEI number allows one to match the vote with $\mathcal{V}$. In other words, the link between the vote and $\mathcal{V}$ is revealed. The attack disrupts the privacy of $\mathcal{V}$ which is one of the main limitation of the Estonian i-voting system. Note that, when the voter's intention revealed, there is no way to recover.

Table 3.1: Possible Attacks and Countermeasures for Different Malicious Scenarios

If Malicious	**Potential Attacks**	**Countermeasures**
VFS	Privacy Leakage	Privacy Preserving Auditing
	Insider Attack	Privacy Preserving Auditing
VotingApp	Manipulation Attack	*VerifApp*
	Reputation Attack	*VerifApp*
	Ghost Click Attack	**No Countermeasure**
VerifApp	Reputation Attack	Re-voting
	Coercion	**Solved in this thesis**
	Privacy Attack	**Solved in this thesis**
VFS and *VotingApp*	Manipulation Attack	Privacy Preserving Auditing
VerifApp and *VotingApp*	Manipulation Attack	**Assumption**
VerifApp and *VFS*	Full privacy Leakage	**Assumption**

CHAPTER 4

Our Proposed Verification System

Our security model extends the Estonian scheme and eliminates the need to trust the verification device on which the *VerifApp* runs. Namely, an adversary may also obtain any information that the *VerifApp* gathers from the network or its outputs. More formally, the probability of correctly guessing the voter's intention should be the same as the conditional probability of guessing the voter's intention correctly given in *VerifApp*'s inputs and outputs. We stress that this additional assumption makes the Estonian i-voting scheme vulnerable because *VerifApp* outputs the voter's intention explicitly. In order to overcome this vulnerability, a cost efficient solution will be to output a masking value for each candidate and to let voters check these results with their own eyes and decide whether to confirm the vote or not only possible by utilizing *VotingApp* and *VerifApp* together.

The aim of the verification mechanism update is to solve the trustability problem of verification mechanism. Therefore, we assume that *VotingApp*, *VFS*, and *VerifApp* may be malicious one by one. On the other hand, the assumptions that *VerifApp* and *VFS* are malicious and that collude and *VerifApp* and *VotingApp* are malicious and collude are not realistic since the duty of *VerifApp* is to check the correctness of *VFS* and *VotingApp* for each vote cast. Accepting the collusion means that using the *VerifApp* is meaningless in the system. Additionally, assuming that *VotingApp* and *VFS* are malicious but do not simultaneously collude is not different than the assumption that *VotingApp* and *VFS* are malicious one by one, which is already considered.

4.1 Warmup: Verification Mechanism with Fake Votes

In the protocol given here has an importance to understand the idea of getting rid of privacy leakage. In the Estonian protocol, random number, r, used during ballot creation in *VotingApp* and is also sent to *VerifApp* for verification. It yields a privacy weakness since an attacker sniffing *VerifApp* may get r and other parameters E_{asym} and CL to learn the voter intention. This weakness, also mentioned in Section 3.3, can be tackled by masking r by sending some additional fake random numbers with it. More concretely, these fake random numbers are sent to anonymously render the actual vote against a malicious *VerifApp*. In order to do that, at the end of the voting stage of Estonian protocol, *VFS* generates and sends m different random numbers to

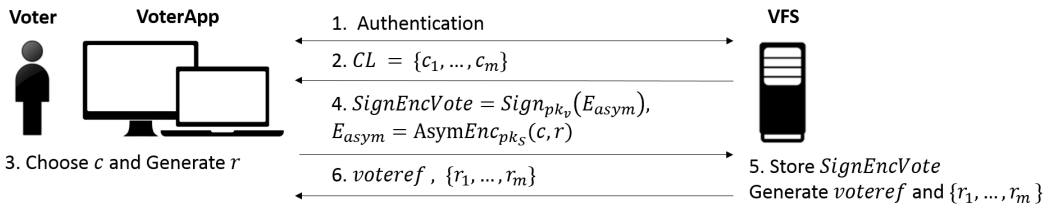
the *VotingApp*. *VotingApp* permutes all these random numbers, including voter's random number, at a predetermined place to mask the random number and sends to the *VerifApp*. In verification stage, *VerifApp* will receive m verified encrypted fake votes generated in *VFS* by using the fake random numbers and randomly selected candidates from the list CL where only one of them is the real vote. After required computations, the *VerifApp* shows candidates in a different order on *verification device*'s screen. The $\mathcal{V}$ checks only the predetermined place of the real vote manually and will not take the others into consideration. In this way, *VerifApp* does not learn the real cast-vote. By the use of fake votes in verification phase, protocol overcomes the privacy leakage.

The cost of this new verification mechanism is $m^2 + m$ public key encryption which takes considerably more than the current Estonian system. There are some ways to make the verification phase more efficient, such as using less fake votes, or Elliptic Curve Cryptography algorithms can be considered as the encryption algorithm to decrease the complexity and time requirement of the system to applicable levels. Since this is just a warm-up scheme, we will not mention about such efficiency tricks. As a result, this system overcomes a serious privacy leakage by increasing complexity of the verification phase since privacy is one of the main concerns of reliability of an election system.

We are now ready to present our electronic voting protocol.

4.1.1 Voting Stage:

Figure 4.1: Voting Stage



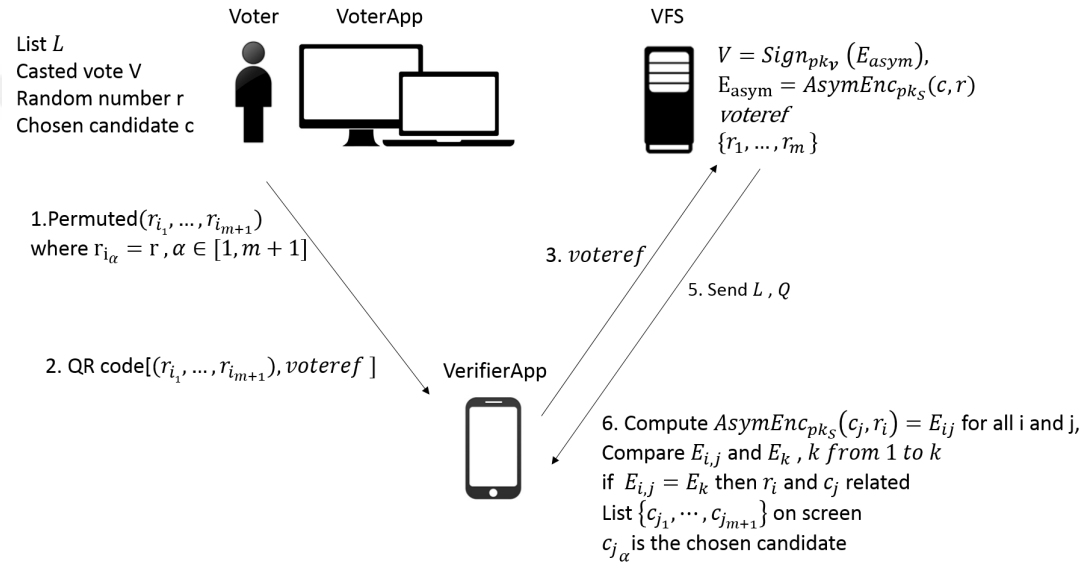
1. A Voter $\mathcal{V}$: Authenticate to *VFS* through *VotingApp* using an ID Card.
2. *VFS*: Send $CL = \{c_1, \cdots, c_m\}$ where m is the number of candidates.
3. $\mathcal{V}$: Choose c from CL.
4. *VotingApp*:
   (a) Generate a random number r.
   (b) Encrypt c and r by election specific public key pk_S, $E_{\mathsf{asym}} = \mathsf{Enc}_{pk_S}(\mathsf{r}, c)$.
   (c) Sign E_{asym} by $\mathcal{V}$'s private key $sk_{\mathcal{V}}$, $V = \mathsf{Sign}_{sk_{\mathcal{V}}}(E_{\mathsf{asym}})$.
   (d) Send V to the *VFS*.

5. *VFS*

6. (a) Store V.
   (b) Generate Vote Reference *voteref*.
   (c) Generate m random numbers $r_1, \cdots, r_m$.
   (d) Send *voteref* and $r_1, \cdots, r_m$ to *VotingApp*.

4.1.2 Verification Stage:

Figure 4.2: Verification Stage



1. $\mathcal{V}$: Choose a number $1 \leq \alpha \leq m+1$.

2. *VotingApp*:

   (a) Permute $P\{r, r_1, \cdots, r_m\}$ s.t. $P_\alpha = \mathsf{r}$.
   (b) Generate a QR code $\mathsf{QR}(r_{j_1}, \cdots, r_{j_{m+1}}, \mathsf{voteref})$.

3. *VerifApp*:

   (a) Scan the QR code.
   (b) Send *voteref* to the *VFS*.

4. *VFS*:

   (a) Match r_{j_i} and c_i, Encrypt $E^{i,j}_{\mathsf{asym}} = Enc_{pk_S}(c_i, r_{j_i})$ for $i = 1, \cdots, m$.
   (b) Send CL, E_{asym} and $E^{i,j}_{\mathsf{asym}}, i = 1, \cdots, m$ to the *VerifApp*.

5. *VerifApp*:

(a) Compute $E^{i,j}_{\mathsf{asym}} = Enc_{pk_S}(c_i, r_j)$ where $i, j = 1, \cdots, m$.

(b) Generate a list Out s.t.,

i. Check $E^{i,j}_{\mathsf{asym}} \stackrel{?}{=} E^{k}_{\mathsf{asym}}$ for some $k \in [1, m+1]$

ii. If $E^{i,j}_{\mathsf{asym}} = E^{k}_{\mathsf{asym}}$, $Out_k = c_k$.

iii. Show Out on screen.

6. $\mathcal{V}$:

(a) If $Out_\alpha = c$, the vote is verified

(b) Else, put an alarm (which basically shows that malware is present).

Note that, this solution has costly public key operations which cannot be used on resource-constrained mobile devices. In the next section we propose another mechanism which significantly reduces the verification cost from $m^2 + m$ asymmetric encryption to only m asymmetric and m symmetric encryptions, in other words, it only requires m symmetric decryptions in addition to the mentioned Estonian i-voting system in Section 3.

4.2 Our Main Proposal: Verification Mechanism with Additional Symmetric Encryptions

We are now ready to describe our proposal. The actual flow of the protocol is similar to the Estonian i-voting system. However, to make the verification phase resistant against the attacks described above, we introduce a new t-bits random verification parameter $q \in \{0,1\}^t$ that will only be known to *VotingApp* and *VFS*[1].

The $\mathcal{V}$ chooses q as a verification parameter and securely sends it to *VFS* during the voting phase. Furthermore, during the verification phase, *VFS* computes $\mathsf{H}(E_{\mathsf{asym}})$ and uses it as a symmetric key to transmit $\mathcal{E}_{\mathsf{sym}} = \mathsf{SymEnc}_{\mathsf{H}(E_{\mathsf{asym}})}(q)$ to *VerifApp*. *VerifApp* will perform a number of decryptions depending on the number of candidates in order to find the correct encryption key. Hence, it outputs a list $Q = \{q_1, \cdots, q_m\}$ of possible verification parameters with the corresponding candidates. Finally, the $\mathcal{V}$ will manually check the output list on the *VerifApp*'s screen with her own eyes to learn the given q on the position of the chosen candidate during the voting phase. The verification phase ends successfully if q exists and its index is the same as the index of the candidate chosen by the $\mathcal{V}$ in the list CL. More formally, $\mathsf{VerifApp}$ obtains r and $\mathsf{voteref}$ from *VotingApp* and CL and $\mathcal{E}_{\mathsf{sym}} = \mathsf{SymEnc}_{\mathsf{H}(E_{\mathsf{asym}})}(q)$ from *VFS* where $E_{\mathsf{asym}} = \mathsf{AsymEnc}_{pk_S}(c, r)$. Based on this information, $\mathsf{VerifApp}$ computes $q_i = \mathsf{SymDec}_{\mathsf{H}(E^i_{\mathsf{asym}})}(\mathcal{E}_{\mathsf{sym}})$ for all $c_i \in CL$ where $E^i_{\mathsf{asym}} = \mathsf{AsymEnc}_{pk_S}(c_i, r)$.

Here, the trick is to encrypt q with the hash of the encrypted vote as the symmetric key (i.e., $\mathsf{H}(E_{\mathsf{asym}})$). In the verification phase, *VerifApp* tries all candidates in order to

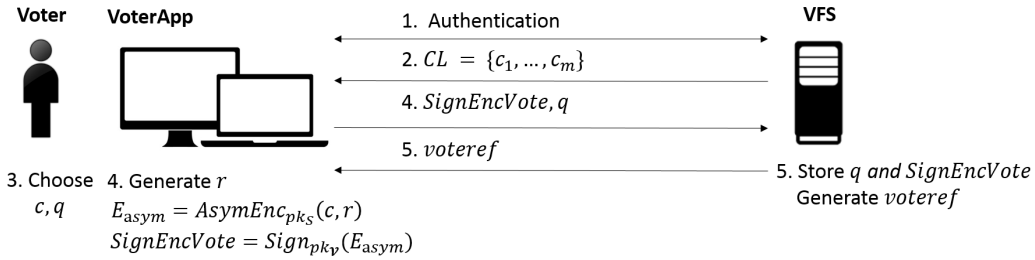
[1] The length of t depends on the hash function, and 256 bit can be used because of AES256 encryption algorithm.

generate the possible keys (hash of the possible encrypted votes). In order to complete the verification successfully, the index of the chosen candidate and the index of q in Q should match. Otherwise, either an alarm would be raised, or the voting procedure should be re-started, or the verification phase should be run in another device. It should be noted that manually finding the correct verification parameter with eyes is an important security measure so as not to reveal a voter's intention to *VerifApp*. Otherwise, if the chosen candidate is displayed in plain form as in the Estonian i-voting scheme, adversaries may obtain a proof of their vote which may lead vote buying or coercion problems.

The voting and verification phases explained in detail below.

4.2.1 Voting Stage:

Figure 4.3: The voting phase of the proposed protocol

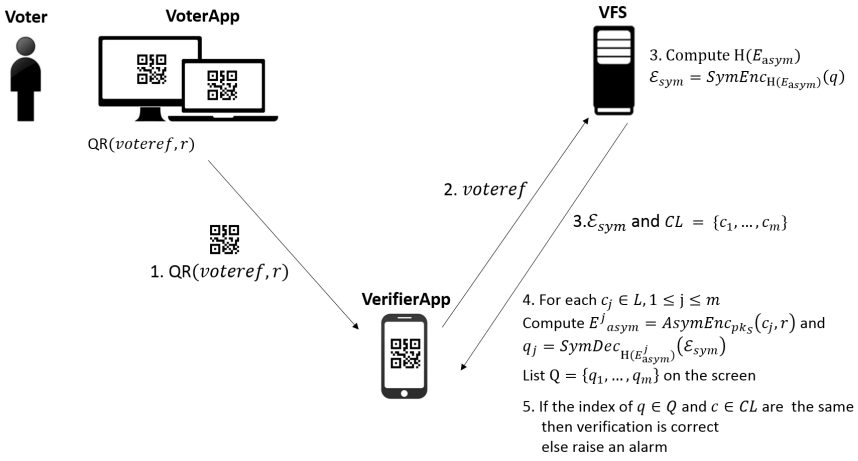


1. A voter $\mathcal{V}$: Authenticates to *VFS* through *VotingApp* using a national ID Card.
2. *VFS*: Sends $CL = \{c_1, \ldots, c_m\}$ to *VotingApp* where m is the number of candidates.
3. $\mathcal{V}$: Chooses c from CL and a t-bits random verification parameter $q \in \{0,1\}^t$.
4. *VotingApp*:
   (a) Generates a random number $r \in \{0,1\}^k$.
   (b) Encrypts c and r by pk_S, $E_{\text{asym}} = \mathsf{AsymEnc}_{pk_S}(c, r)$.
   (c) Signs E_{asym} by $sk_{\mathcal{V}}$, $\mathsf{SignEncVote} = \mathsf{Sign}_{sk_{\mathcal{V}}}(E_{\text{asym}})$.
   (d) Sends SignEncVote and q to *VFS*[2].
5. *VFS*:
   (a) Stores SignEncVote and q.
   (b) Generates *voteref*.
   (c) Sends *voteref* to the $\mathcal{V}$ for verification phase.

[2] We note that q can also be signed by the voter to ensure its source and correctness.

4.2.2 Verification Stage:

Figure 4.4: The Verification phase of the proposed protocol



1. (a) *VotingApp*: Generates a *QR code* including r and *voteref* and show on the screen.
   (b) *VerifApp*: Scans the *QR code* by the camera.
2. *VerifApp*: Sends *voteref* to *VFS*.
3. *VFS*:
   (a) Computes $\mathsf{H}(E_{\mathsf{asym}})$.
   (b) Encrypts $\mathcal{E}_{\mathsf{sym}} = \mathsf{SymEnc}_{\mathsf{H}(E_{\mathsf{asym}})}(q)$.
   (c) Sends the ordered m-tuple CL and $\mathcal{E}_{\mathsf{sym}}$.
4. *VerifApp*:
   (a) For each $c_j \in CL$, $j = 1, \cdots, m$, computes;
      i. $E^j_{\mathsf{asym}} = \mathsf{AsymEnc}_{pk_S}(c_j, r)$.
      ii. The hash value $\mathsf{H}(E^j_{\mathsf{asym}})$.
      iii. $q_j = \mathsf{SymDec}_{\mathsf{H}(E^j_{\mathsf{asym}})}(\mathcal{E}_{\mathsf{sym}})$.
   (b) Shows the ordered m-tuple $Q = \{q_1, \cdots, q_m\}$ on the screen.
5. $\mathcal{V}$: Finds the indices of $q \stackrel{?}{=} q_\alpha \in Q = \{q_1, \cdots, q_m\}$ and $c \stackrel{?}{=} c_\beta \in \{c_1, \cdots, c_m\}$, where $\alpha, \beta \in \{1, \cdots, m\}$.
   (a) Checks $\alpha \stackrel{?}{=} \beta$.
   (b) If $\alpha = \beta$, the vote is received and stored correctly in *VFS*.
   (c) Else, the $\mathcal{V}$ puts an alarm (which basically shows that malware is present).

4.3 Security Analysis of Our Verification Mechanism

Now we are ready to prove the correctness and security of our mechanism.

Theorem 4.1 (Correctness)**.** *The verification phase of the proposed protocol ensures both the properties recorded-as-cast (the vote is stored in VFS) and cast-as-intended (the vote reflects the intention of the voter).*

Proof. During the verification phase, the *VerifApp* first obtains r and voteref from VotingApp. Note that, it is trivial to obtain the encrypted vote E_{asym} from the $\mathsf{SignEncVote} = \mathsf{Sign}_{sk_{\mathcal{V}}}(E_{\mathsf{asym}})$. Next, VerifApp requests the relevant data according to the *voteref* from *VFS*. *VFS* responds with the ordered m-tuple candidate list $CL = \{c_1, \cdots, c_m\}$ and $\mathcal{E}_{\mathsf{sym}} = \mathsf{SymEnc}_{\mathsf{H}(E_{\mathsf{asym}})}(q)$. Now suppose that the $\mathcal{V}$ votes for the candidate $c_\beta \in CL$. Then, *VerifApp* completes the verification phase by computing $E^i_{\mathsf{asym}} = \mathsf{AsymEnc}_{pk_S}(c_i, r)$ for each $c_i \in CL$ and outputs the list $Q = \{q_1, \cdots, q_m\}$ where $q_i = \mathsf{SymDec}_{\mathsf{H}(E^i_{\mathsf{asym}})}(\mathcal{E}_{\mathsf{sym}})$.

Finally, the $\mathcal{V}$ searches for q (which was on the screen of the voter computer) on the verification device and finds the index $\beta \in \{1, \cdots, m\}$ where $q_\beta = q$. Note that because of the encryption scheme this index is unique with very high probability. The decryption of E_{sym} will only result $q = q_\beta$ at the index of c_β.

Therefore, this scheme guarantees that the ballot reflects the intention of the $\mathcal{V}$ (i.e., it has been cast-as-intended) and is correctly stored in *VFS* (i.e., it has been recorded-as-cast). □

Theorem 4.2 (Privacy against malicious *VotingApp* and its adversarial environment)**.** *Our verification mechanism described in Section 4.2 is secure against malicious VotingApp or adversarial environment of VotingApp.*

Proof. Suppose *VotingApp* or its adversarial environment cheat and try to send the vote for another predefined candidate. However, the $\mathcal{V}$ can easily detect this adversarial action via VerifApp. Therefore, the verification stage is ended with an alarm and the $\mathcal{V}$ re-votes from another *VotingApp*.

Suppose a malicious *VotingApp* tries to find the appropriate parameters to mock the *VerifApp* by finding a value r^* for the candidate $c^* \neq c \in CL$ satisfying

$$\mathsf{AsymEnc}_{pk_S}(c^*, r^*) = \mathsf{AsymEnc}_{pk_S}(c, r) = E_{\mathsf{asym}}.$$

For the verification, *VerifApp* will compute the value $q = \mathsf{SymDec}_{\mathsf{H}(E_{\mathsf{asym}})}(\mathcal{E}_{\mathsf{sym}})$ and gets the same verification parameter q which is in the position of the chosen candidate c. Therefore, the $\mathcal{V}$ will not realize that her vote is cast to another candidate c^*. The computational cost of finding r^* for certain c, r and c^* such that $\mathsf{AsymEnc}_{pk_S}(c^*, r^*) = \mathsf{AsymEnc}_{pk_S}(c, r) = E_{\mathsf{asym}}$ is at most 2^k encryptions where k denotes the length of r (which is infeasible since $k >> 80$). Alternatively, an attacker may guess r^* with probability of $\frac{1}{2^k}$ which is negligible. □

Table 4.1: Computational Costs.

	VoterApp	VFS	VerifApp	Security Against Corrupted Verification Device
Estonian Verification	1 AsymEnc + 1 Sign	$\emptyset$	m AsymEnc	✗
This Paper	1 AsymEnc + 1 Sign	1 SymEnc	m AsymEnc + m SymEnc	✓

Theorem 4.3 (Privacy against malicious *VerifApp* and its adversarial environment)**.** *Our verification mechanism described in Section 4.2 is secure against malicious VerifApp or adversarial environment of the VerifApp and does not leak any information about the $\mathcal{V}$ and her intention.*

Proof. For the verification phase, *VerifApp* receives the parameters *voteref*, r, $CL = \{c_1, \cdots, c_m\}$ and C. Additionally, it computes the lists $Q = \{q_1, \cdots, q_m\}$ and $\{E^1_{\mathsf{asym}}, \cdots, E^m_{\mathsf{asym}}\}$ where m is the number of candidates. It can be easily checked if the received parameters or the computed values reveal any information about the $\mathcal{V}$'s intention. Therefore, an attacker sniffing *VerifApp* can only learn whether a voter already has checked her vote.

Note that *VerifApp* never learns which q_j is the correct q since the $\mathcal{V}$ checks the index of verification parameter, which reveals the intention of the $\mathcal{V}$, by her own eyes from the ordered list Q. Therefore, an attacker sniffing *VerifApp* should guess the correct q with probability of $\frac{1}{m}$ which is no more than guessing the candidate randomly. So, *VerifApp* leaks no information about the choice of the $\mathcal{V}$. □

Theorem 4.4 (Privacy against malicious *VFS*)**.** *Our verification mechanism described in Section 4.2 is secure against a malicious VFS.*

Proof. Our proposed verification mechanism uses the same voting phase with the Estonian i-voting system. The only difference is our mechanism is to the random verification parameter q which gives no information about the vote. Therefore, the proof of the security of malicious *VFS* is exactly the same with the Estonian scheme which is explained in [5]. □

4.4 Complexity Analysis

In Table 4.1, the complexity analysis of the proposed verification mechanism is tabulated. The cost of ridding the privacy leakage in the verification phase using symmetric key cryptography requires almost the same cost as the current Estonian i-voting scheme, specifically 1 extra symmetric encryption for *VFS* and m extra symmetric encryptions for *VerifApp*.

Table 4.2: Shortened verification parameter on the screen on of the voter computer.

Position #	1	2	3	4	$\cdots$	18	19	20	21	22	$\cdots$	28	29	30	31	32
q	T	h	E	P	$\cdots$	e	l	o	n	g	$\cdots$	e	m	b	e	r
Shortened q		h					l		n		$\cdots$			b		

Table 4.3: Original and shortened verification values.

c_i	q_i	t-bits q	Shortened
c_1	q_1	qxvsEgaKMXpwApGDsNnPaNhjTJYtqven	xnav
$\vdots$	$\vdots$	$\vdots$	$\vdots$
c_β	q_β	ThEParameterWillBelongTorememberr	hlnb
$\vdots$	$\vdots$	$\vdots$	$\vdots$
c_m	q_m	RatqPKvgtTAFectHpOeteDoPYKbTkApp	aeeA

4.4.1 Usability and Optimization Improvement for the Verification q

The size of q is important due to security concerns. On the other hand, during the verification stage, the decryptions q_j of $\mathcal{E}_{\mathsf{sym}} = \mathsf{SymEnc}_{\mathsf{H}(E_{\mathsf{asym}})}(q)$ for each $j = 1, \ldots, m$ are listed in $Q = \{q_1, \cdots, q_m\}$ where each q_i denotes a t-bits number. A voter has to confirm the original q in Q with her own eyes.

In the large scale, finding the index of q in the verification device screen may be impractical. Instead of searching the value q itself, it is also possible to build a mechanism that sufficiently supports comparing only a predefined subset of bit positions of q and $q_1, \cdots, q_m$. In this way, a voter can easily compare the q on VotingApp to the q_j's on VerifApp. More concretely, the protocol will run as defined in Section 4.2, but both *VotingApp* and *VerifApp* will only display the values depending on the $\ell < t$ of different predefined positions $p_i \in_R \{1, \cdots, t\}$ for $i = 1 \ldots \ell$ of q and $q_1, \cdots, q_m$.

As an illustration, we present a sample shortening operation in Table 4.2 for the parameter $q =$'**Th**EParameterWillBe**l**o**n**gTorememb**e**r'. Let $\ell = 4$ and the position numbers are chosen as 8-bits words at the positions p_1, p_2, p_3 and p_4 respectively 2, 19, 21 and 30. Then, shortened value 'hlnb' is shown on the screen instead of q itself. In Table 4.3, step 5 of verification phase is presented as an example which shows the values on *VerifApp* screen where c_i is the i-th candidate and q_i is the related verification value for $i = 1, \cdots, m$.

CHAPTER 5

Conclusion

Internet voting schemes are evolving and being used practically which aim to guarantee at least the same security level offered by classical paper ballot voting systems. After the 2011 elections in Estonia, a verification phase was integrated into the Estonian system to check whether the vote has been correctly recorded. However, this phase eventually displays the cast vote in a plain form which may cause a privacy weakness and compromise the secrecy of the election results. The designers of the Estonian scheme overlooked the fact that the device running *VerifApp* may also be compromised. Thus, there is no difference between trusting a device running *VotingApp* and trusting a device running *VerifApp*. Moreover, the Estonian system has a weakness that *VerifApp* may leak voter's identity and intention if the verification device is malicious.

In this thesis, we have proposed a new verification mechanism that does not disclose any information about a voter's intention, if it is used by corrupted verification devices. Hence, our proposed verification system is strong against the aforementioned privacy weaknesses. Additionally, the system is strong against the people see verification device's screen, i.e. they cannot figure out the chosen candidate in the election with high probability then guessing it randomly. The proposed verification mechanism prevent the system from estimating election results even if some corrupted verification devices are used. Initially, we proposed a warm-up system that is strong against privacy leakage though it requires more time and more computation power. At the same time warm-up system can be considered as a system to understand the underlying idea of the proposed system. The second proposed system, on the other hand, provides a much more efficient system by decreasing the computation time and power consumption.This system yields a remarkably more efficient way to reach better security level.

In Table 5.1, possible attacks and countermeasures for different malicious scenarios for our updated verification mechanism are presented. As it is seen from the table, the proposed systems are the countermeasures for the coercion and the privacy attack when the *VerifApp* is malicious.

Table 5.1: Possible Attacks and Countermeasures for Different Malicious Scenarios for Our Updated Verification Mechanism

If Malicious	**Potential Attacks**	**Countermeasures**
VFS	Privacy Leakage	Privacy Preserving Auditing
	Insider Attack	Privacy Preserving Auditing
VotingApp	Manipulation Attack	*VerifApp*
	Reputation Attack	*VerifApp*
	Ghost Click Attack	**Future Work**
VerifApp	Reputation Attack	Re-voting
	Coercion	**Updated *VerifApp*** (proposed in this thesis)
	Privacy Attack	**Updated *VerifApp*** (proposed in this thesis)
VFS and *VotingApp*	Manipulation Attack	Privacy Preserving Auditing
VerifApp and *VotingApp*	Manipulation Attack	Assumption
VerifApp and *VFS*	Full privacy Leakage	Assumption

As a future work, investigating a mechanism that resists the Ghost Click Attack and does not rely on additional post-channel communication would be very interesting.

REFERENCES

[1] Mix network, 19 April 2016, https://en.wikipedia.org/wiki/Mix_network#/media/File:Decryption_mix_net.png.

[2] Iphone and android apps breach privacy, October 2013, http://online.wsj.com/article/SB10001424052748704694004576020083703574602.html.

[3] B. Adida, Lecture 17: Introduction to electronic voting, 19 April 2016, http://courses.csail.mit.edu/6.897/spring04/L17.pdf.

[4] K. M. R. Alam and S. Tamura, Electronic voting - scopes and limitations, pp. 525–529, IEEE, 2012, ISBN 978-1-4673-1153-3.

[5] A. Ansper, A. Buldas, A. Jurgenson, M. Oruaas, J. Priisalu, K. Raiend, A. Veldre, J. Willemson, and K. Virunurm, E-voting concept security: Analysis and measures.

[6] E. C. Authority, Id-card, http://www.id.ee/index.php?id=30470.

[7] O. Bach, Mobile malware threats in 2015: Fraudsters are still two steps ahead, 13 April 2016, https://securityintelligence.com/mobile-malware-threats-in-2015-fraudsters-are-still-two-steps-ahead/.

[8] D. Boneh and P. Golle, Almost entirely correct mixing with applications to voting, in *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002*, pp. 68–77, 2002.

[9] D. Chaum and T. P. Pedersen, Wallet databases with observers, in *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, pp. 89–105, 1992.

[10] E. I. V. Committee, Using id-card and mobil-id, https://www.valimised.ee/eng/kkk.

[11] E. N. E. Committee, 2010 e-voting system. general overview-2010, January 2016, http://vvk.ee/public/dok/General_Description_E-Voting_2010.pdf.

[12] N. E. Committee, Statistics about internet voting in Estonia, 20 April 2016, http://www.vvk.ee/voting-methods-in-estonia/engindex/statistics.

[13] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, Pios: Detecting privacy leaks in ios applications, in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*, 2011.

[14] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, in *IEEE*, 4-31, pp. 469–472, Transactions on Information Theory., 1985.

[15] W. Enck, P. Gilbert, B. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. Sheth, Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones, in *9th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2010, October 4-6, 2010, Vancouver, BC, Canada, Proceedings*, pp. 393–407, 2010.

[16] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, A survey of mobile malware in the wild, in *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '11, pp. 3–14, ACM, New York, NY, USA, 2011.

[17] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern, RSA-OAEP is secure under the RSA assumption, in *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pp. 260–274, 2001.

[18] S. Heiberg, P. Laud, and J. Willemson, The application of i-voting for Estonian parliamentary elections of 2011, in A. Kiayias and H. Lipmaa, editors, *3rd International Conference on E-voting and Identity, Tallinn, Sep 29th-30th, 2011*, volume 7187 of *Lecture Notes in Computer Science*, Springer-Verlag, 2012, ISBN 978-3-642-32746-9.

[19] S. Heiberg, A. Parsovs, and J. Willemson, Log analysis of Estonian internet voting 2013-2015., IACR Cryptology ePrint Archive, 2015, p. 1211, 2015.

[20] S. Heiberg and J. Willemson, Verifiable internet voting in Estonia, in *6th International Conference on Electronic Voting: Verifying the Vote, EVOTE 2014, Lochau / Bregenz, Austria, October 29-31, 2014*, pp. 1–8, 2014.

[21] ICTA-Information and C. T. Authority, Ceir-central equipment identity register, 19 April 2016, `http://www.mcks.gov.tr/en/KonuDetay.php?BKey=47`.

[22] A. K. Jain and D. Shanbhag, Addressing security and privacy risks in mobile applications, IT Professional, 14(5), pp. 28–33, Sept 2012.

[23] M. Jakobsson, A. Juels, and R. L. Rivest, Making mix nets robust for electronic voting by randomized partial checking, IACR Cryptology ePrint Archive, 2002, p. 25, 2002.

[24] M. Jakobsson, A. Juels, and R. L. Rivest, Making mix nets robust for electronic voting by randomized partial checking, in *Proceedings of the 11th USENIX Security Symposium*, pp. 339–353, USENIX Association, Berkeley, CA, USA, 2002, ISBN 1-931971-00-5.

[25] M. Matsui, *Advances in Cryptology - ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009, Proceedings*, LNCS sublibrary: Security and cryptology, Springer, 2009, ISBN 9783642103650.

[26] National Institute of Standards and Technology, FIPS 197: Advanced Encryption Standard (AES), Federal Information Processing Standards Publication Series, 2001.

[27] National Institute of Standards and Technology, FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, Federal Information Processing Standards Publication Series, 2015.

[28] P. Paillier and D. Pointcheval, Efficient public-key cryptosystems provably secure against active adversaries, in K.-Y. Lam, E. Okamoto, and C. Xing, editors, *Advances in Cryptology - Proceedings of ASIACRYPT '99*, volume 1716 of *LNCS*, pp. 165–179, Springer, Singapore, 1999.

[29] S. Popoveniuc, J. Kelsey, A. Regenscheid, and P. Vora, Performance requirements for end-to-end verifiable elections, in *Proceedings of the 2010 International Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'10, pp. 1–16, USENIX Association, Berkeley, CA, USA, 2010.

[30] R. Rivest, Lecture 18: Mix-net voting systems, 19 April 2016, `http://courses.csail.mit.edu/6.897/spring04/L18.pdf`.

[31] D. Springall, T. Finkenauer, Z. Durumeric, J. Kitcat, H. Hursti, M. MacAlpine, and J. A. Halderman, Security analysis of the Estonian internet voting system, in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pp. 703–715, ACM, 2014, ISBN 978-1-4503-2957-6.

[32] I. S. G. Stenerud and C. Bull, When reality comes knocking Norwegian experiences with verifiable electronic voting., in *Electronic Voting*, volume 205 of *LNI*, pp. 21–33, GI, 2012, 1617-5468.

[33] G. Suarez-Tangil, J. E. Tapiador, P. Peris-Lopez, and A. Ribagorda, Evolution, detection and analysis of malware for smart devices, IEEE Communications Surveys and Tutorials, 16(2), pp. 961–987, 2014.

[34] W. Trappe and L. C. Washington, *Introduction to Cryptography with Coding Theory (2Nd Edition)*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2005, ISBN 0131862391.

[35] B. Uscilowski, Security response- mobile adware and malware analysis, October 2013, `http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/madware_and_malware_analysis.pdf`.

[36] D. Wave, Types of qr codes, `http://www.qrcode.com/en/index.html` (Accessed on: 30 April 2016).

[37] D. Wave, Qr code standard iso/iec 18004:2015, 2000, http://www.iso.org/iso/catalogue_detail.htm?csnumber=62021 (Accessed on: 30 April 2016).

[38] H. Zhu, H. Xiong, Y. Ge, and E. Chen, Mobile app recommendations with security and privacy awareness, in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pp. 951–960, ACM, 2014, ISBN 978-1-4503-2956-9.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Muş, Köksal
Nationality: Turkish
Date and Place of Birth: 09 September 1982, İstanbul
Marital Status: Married
Phone: +90 505 249 0759

EDUCATION

Degree	Institution	Year of Graduation
M.S.	Department of Cryptography, Institute Of Applied Mathematics, METU	2009
B.S.	Department Of Mathematics, Yildiz Technical University, İstanbul	2004
High School	Beylerbeyi Haci Sabanci High School, İstanbul	1999

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2006-2011	Mathematics Department, METU	Research Assistant
2013-2015	METU Cryptography Department, Institute of Applied Mathematics	Senior Researcher

PUBLICATIONS

K. Muş, M.S. Kiraz, M. Cenk, İ. Sertkaya, A Potential Privacy Leakage in the Estonian Voting Verification Mechanism, Submitted.

A. Doganaksoy, B. Ege, K. Mus, Extended Results for Independence and Sensitivity of NIST Randomness Tests, 3^{rd} National Cryptology Symposium, Ankara,Turkey, 2008.