Estonian Voting Verification Mechanism Revisited

Köksal Muş[1], Mehmet Sabır Kiraz[2], Murat Cenk[3], and İsa Sertkaya[2]

[1]İstanbul Aydın University, [2]TÜBİTAK BİLGEM
[3]Middle East Technical University Turkey
koksalmus@aydin.edu.tr,mehmet.kiraz@tubitak.gov.tr,
mcenk@metu.edu.tr,isa.sertkaya@tubitak.gov.tr

Abstract. After the Estonian Parliamentary Elections held in 2011, an additional verification mechanism was integrated into the i-voting system in order to resist corrupted voting devices, including the so called Student's Attack where a student practically showed that the voting system is indeed not verifiable by developing several versions of malware capable of blocking or even changing the vote. 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. In other words, the device on which the verification is done learns the voter's choice. In this work, our aim is to investigate this verification phase in detail and to point out that leaking the voter's choice to the verification application may harm the voter privacy. Additionally, when applied in a wide range, this would even compromise the fairness and the overall secrecy of the elections. 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 with a negligible cost. More concretely, it brings only m additional symmetric key decryptions to the verification device, where 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, Verifiability, Trust

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 [6].

Estonian i-voting protocol with its verification mechanism present an interesting case because it avoids the additional pre- and post-channels as seen in the Norwegian protocol, in which the verification is performed via smart devices [12, 20]. Since 2005 Estonian i-voting system is still being used and the number of i-voters increases 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 upheld elections successively [33]. 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 [7, 8]. As mentioned in [18, 21], 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 the vote stored in the system via a smart device with a camera and internet connection. 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.

1.1 Contributions

In this work, 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 process on the verification device of which the voter is most probably the owner. In fact, smart phone 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 [22–25, 27–31]. 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 paper is to mitigate the described 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 [21]. In this work, however, our updated verification mechanism ensures the same security level without leaking any information about the vote.

1.2 Organization

The rest of the paper is organized as follows. Related works are discussed in the following section. The necessary preliminaries and underlying cryptographic mechanisms are explained in Section 3. The current Estonian i-voting system along with its components, security and threat models are explained in Section 4. A potential privacy issue is introduced and the proposed voting protocol and security model are given in Section 5. The security analysis of the proposed system is presented in Section 6. Section 7 compares the complexities of Estonian system with our proposed system. Finally, Section 8 concludes the paper.

2 Previous Work

In this section, we will give a brief overview of related work, focusing on only internet voting schemes. After the 2011 elections in Estonia, Heiberg et al. published a paper discussing new attacks and weaknesses resulting from client and server side weaknesses namely Student's Attack [14, 15, 18, 21]. 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 [10, 12]. 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 [21]. It also well-known that the Helios system is end-2-end verifiable which is not sufficient for secure elections since it does not prevent attacks from both corrupted client and servers [2, 13, 16, 17].

The designers of the Estonian i-voting system claim that it was as reliable and secure as the conventional election [12,32]. Contrary to their security claims, in [20], 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 [11], Heiberg et al. researched ways to eliminate transparency weaknesses using an auditing mechanism.

The future brings more security and privacy risks for mobile devices [29–31]. For example, users can be fooled into installing malicious applications on their devices or to grant unauthorized remote access [22–25, 27–31]. 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., [26]). 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 .

3 Preliminaries

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

Underlying Cryptosystem. We are going to denote a symmetric key encryption process as $\mathcal{E}_{\mathsf{sym}} = \mathsf{SymEnc}_k(M)$ and decryption as $M = \mathsf{SymDec}_k(\mathcal{E}_{\mathsf{sym}})$, where k is a secret key and M is a plaintext to be encrypted. We are going to denote the hashing of a message M as $\mathsf{H}(M)$, where M is a message and $H : \{0,1\}^* \rightarrow \{0,1\}^t$. AES-256 and SHA3-256 (where $t = 256$) can be utilized for symmetric encryption and hash function, respectively [3, 4].

The underlying public key encryption scheme is semantically secure [1] (e.g., Paillier [35], and ElGamal [34]). An election specific public and private key pair (pk_S, sk_S) is generated by the servers of the National Electoral Committee (NEC) in a k of n threshold manner. In other words, it is generated by the cooperation of n independent parties and it is not possible to regenerate the key pair if at least k of them do not cooperate. Furthermore, pk_S is mounted to voter applications $\mathsf{VoterApp}$. 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 an encryption of a message M using the public key of the voter $\mathcal{V}$. Similarly, $\mathsf{Sign}_{sk_{\mathcal{V}}}(M)$ denotes the signature of a user $\mathcal{V}$ on a message M using the private key of $\mathcal{V}$.

[1] Note that semantically secure cryptographic algorithms are basically randomized encryptions meaning that encryption of the same votes is uniformly indistinguishable from each other.

4 Estonian Internet Voting Protocol and its Security Analysis

4.1 Components of the System

The Estonian i-voting mechanism is composed of three main parts: (1) client-side applications (a voter application VoterApp and a verification application VerifApp), (2) a Central System, and (3) Auditing and Counting processes.

A VoterApp is performed by the citizens eligible to vote via their ID cards. VoterApp is already developed and published by NEC, and should be installed beforehand. VerifApp should be installed on a smart device. Note that VerifApp can be developed by any parties, including NEC, political parties, or an open source community. The Central System has three main servers for forwarding, storing and counting phases under NEC responsibility. The Vote Forwarding Server (VFS) is the server that the client-side applications authenticate, send signed and encrypted votes, and obtain the required data for both the voting and the verification stages. The Vote Storage Server (VSS) stores the signed encrypted votes during the voting period. At the end of the election, it removes double votes, cancels ineligible voters, and prepares the votes to be tallied by anonymizing the encrypted votes via a mixnet mechanism (e.g., [5]).

Separated from the rest of the system by an air gap, the Vote Counting Server tallies the anonymized votes and computes the election results. During all these processes, the auditing mechanisms save logs of the Central System events in order to resolve independent auditors' disputes and complaints.

4.2 Estonian I-Voting Protocol

For simplicity of describing the voting protocol, we divide it into two phases: the voting phase and the verification phase. The voting phase begins by VoterApp authenticating 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, the voter $\mathcal{V}$ chooses a candidate $c \in CL$ to cast the vote. VoterApp 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 $r \in_R \{0,1\}^{\kappa}$ is a random number, $\kappa \in \mathbb{N}$. Next, VoterApp sends SignEncVote to VFS, and then receives a vote reference voteref which is a receipt to be used in the verification phase.

During the verification phase, VerifApp receives r and voteref from VoterApp, 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, the voter $\mathcal{V}$ puts an alarm.

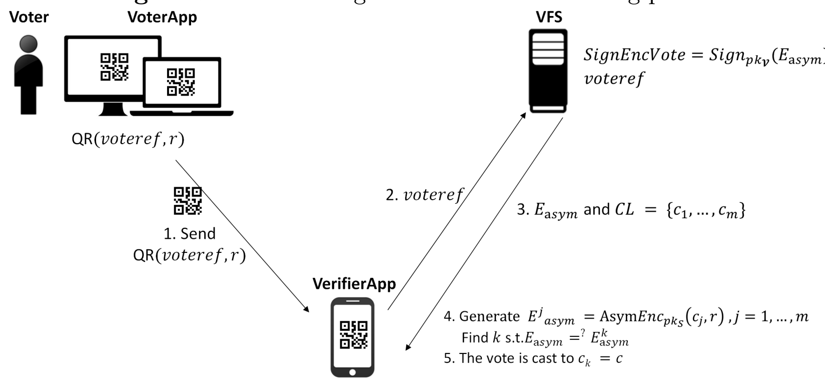
Voting Stage:

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

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

Verification Stage: Note that the verification stage is optional and used only to ensure whether the vote has been correctly stored in VFS. It is also important to note that for security purposes, VerifApp and VoterApp 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.

Fig. 1. Verification stage of the Estonian i-voting protocol


Voter
VoterApp
VFS
$SignEncVote = Sign_{pk_{\mathcal{V}}}(E_{asym})$
$voteref$
QR($voteref, r$)
1. Send QR($voteref, r$)
2. $voteref$
3. E_{asym} and $CL = \{c_1, \dots, c_m\}$
VerifierApp
4. Generate $E^j{}_{asym} = \mathrm{Asym}Enc_{pk_S}(c_j, r)$, $j = 1, \dots, m$
Find k s.t. $E_{asym} =^? E^k_{asym}$
5. The vote is cast to $c_k = c$

1. (a) VoterApp: Generates a QR code including r and voteref and show on the screen.
   (b) VerifApp: Scans QR code by camera.
2. VerifApp: Sends voteref to VFS.
3. VFS: Sends E_{asym} and CL to VerifApp.
4. VerifApp:
   (a) Computes $E^j_{\mathsf{asym}} = \mathsf{AsymEnc}_{pk_S}(c_j, r)$ for all $j = 1, \cdots, m$.
   (b) Finds ℓ such that $E_{\mathsf{asym}} \overset{?}{=} E^{\ell}_{\mathsf{asym}}$ for some $\ell \in \{1, \cdots, m\}$.
   (c) Shows c_ℓ on the screen.
5. $\mathcal{V}$: Checks $c_\ell \overset{?}{=} c$.
   (a) If $c_\ell = c$, the vote is received and stored VFS without any modification.
   (b) Else, $\mathcal{V}$ puts an alarm (which basically shows that malware is present).

4.3 Security Model of the Estonian I-Voting Protocol

A security analysis of the current i-voting system of Estonia is discussed in detail in [20, 21] which are briefly mentioned in attack scenarios. Estonian i-voting security model assumes that either VoterApp or the device that runs VoterApp is malicious. We note that the assumptions VerifApp and VFS collude maliciously or VerifApp and VoterApp collude maliciously are not realistic since the duty of VerifApp is to independently check the correctness of VFS and VoterApp. Furthermore, as noted in [8], limited number of corrupted voters' devices are accepted as a reasonable risk.

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

- **Manipulation Attacks.** Manipulation attacks consist of modifications to a vote without the knowledge of the voter $\mathcal{V}$. These attacks are aimed to change the vote to either a predetermined or a random candidate. There are basically two variants of manipulation attacks:
  - 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 on VoterApp. This malware is designed so that it undermines VoterApp 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, 20]). But, after, it became a real and efficient attack [18]. So that, a verification mechanism is integrated into the system [21].
  - Ghost Click Attack. A malicious software that runs on the same device as VoterApp 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 [20]. Although this is an interesting attack, it is not included in the scope of this work.
- **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, not only an effective verification mechanism must be set up to prevent such attacks, but also, citizens should be well informed about the process and security concerns about the system [8]. 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 can 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.

Threat Model.

- **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 [20]. As a countermeasure, auditing mechanisms with independent auditors can verify the computations of VFS [11].
- **Malicious VoterApp and its adversarial environment.** During the preparation of signed encrypted vote, a malicious VoterApp can change the voter's intention $c \in CL$ by $c^* \in CL$ 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. The voter $\mathcal{V}$ re-votes again from another device. Note that integrity of the stored data in VFS is ensured after successful verification. Therefore, manipulating the vote using a malicious VoterApp is not possible.
- **Malicious VerifApp and its adversarial environment**: It is possible to sniff data from a smart device using any malicious application installed on the device [22–25, 27–31]. 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 based on the leakage is that using a device's IMEI number allows one to match the vote with the voter $\mathcal{V}$. In other words, the link between the vote and the voter $\mathcal{V}$ 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, the candidate choice of a voter represents the voter's political view. Therefore, when the voter's intention revealed, there is no way to recover. In other words, it is not possible to choose another candidate and vote again. More details about the attacks are given in 5.1.

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

5 A New Potential Privacy Issue and Our Improved Verification System

5.1 No Vote Privacy in the Estonian Verification Mechanism

Recall that VerifApp is integrated into the i-voting system to be resistant against Student's Attack. VerifApp can be freely developed by NEC, by an open source community or even by one who is able to write her own verification software while VoterApp is developed only by NEC. In contrast, most citizens cannot develop or control the trustability of VerifApp even it is downloaded from a known source. Therefore, 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 [22–25, 27–31]. 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 the voter $\mathcal{V}$. 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).

When considered on the individual level, this attack causes privacy leakage. On the other hand, when applied over a wide range by sniffing the plain votes from the VerifApps without trying to find the related identity information, this attack may also promise the reliability and, more importantly, the secrecy of the elections. Because, as explained in [9], 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.

In the next section, we offer a new protocol for the verification mechanism which eliminates the trust the verification device.

5.2 Enhanced Security Model

Our security model extends the Estonian scheme by eliminating the need to trust the verification device on which VerifApp runs. Namely, the extended scheme gives specific information to VerifApp on which the voted candidate should be recognized only by the voter $\mathcal{V}$ with her own eyes manually. Even VerifApp can not learn the voted candidate using the data which is gathered from the network or its outputs by VerifApp. More formally, the probability of correctly guessing $\mathcal{V}$'s intention using the information given by VerifApp's inputs and outputs should be the same as the conditional probability of guessing $\mathcal{V}$'s intention randomly. We stress that using a mobile device to verify the vote makes the Estonian i-voting scheme vulnerable as in the original system because VerifApp outputs $\mathcal{V}$'s intention explicitly. In order to overcome this vulnerability, each candidate will be displayed on the screen with a verification parameter and to let $\mathcal{V}$ checks these results with her own eyes using the parameters and decides whether the correct parameter and the candidate is matched.

5.3 Our Verification Mechanism

We are now ready to describe our proposal. The actual flow of the protocol is similar to the Estonian i-voting system. In order to make the verification phase more resistant against the attacks described in Section 5.1, we introduce a new parameter q that will only be known to VoterApp and VFS where the size of q is the same as the output size of the hash function H. More concretely, $\mathcal{V}$ 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 encrypt q for transmitting $\mathcal{E}_{\mathsf{sym}} = \mathsf{SymEnc}_{\mathsf{H}(E_{\mathsf{asym}})}(q)$ to VerifApp. During the verification phase, in order to find the correct encryption key, VerifApp will perform a number of decryptions $q_i = \mathsf{SymDec}_{\mathsf{H}(E^i_{\mathsf{asym}})}(\mathcal{E}_{\mathsf{sym}}), i = 1, \cdots, m$ depending on the number of candidates. Hence, it outputs a list $Q = \{q_1, \cdots, q_m\}$ of possible verification parameters with the corresponding candidates. Finally, the voter $\mathcal{V}$ will manually check the output list on the VerifApp's screen with her own eyes to learn the given q during the voting phase on the position of the chosen candidate . The verification phase ends successfully if q exists and its index is the same as the index of the candidate chosen by the voter $\mathcal{V}$ in the list CL.

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 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.

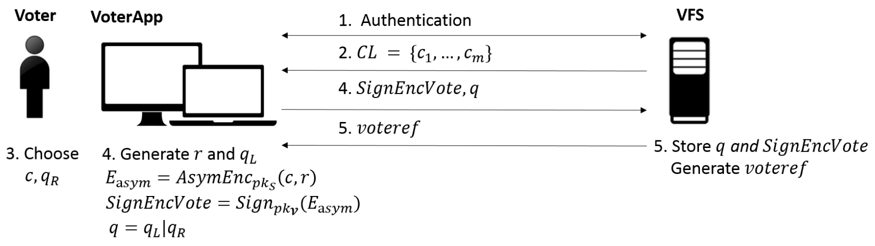
The voting and verification phases are explained in detail below.

Voting Stage:

1. A voter $\mathcal{V}$: Authenticates to VFS through VoterApp using a national ID Card.
2. VFS: Sends $CL = \{c_1, \ldots, c_m\}$ to VoterApp where m is the number of candidates.
3. $\mathcal{V}$: Chooses c from CL and 4 characters of random verification parameter (composed of 4 bytes per character) which is a random value $q_{\mathsf{right}} \in_R \{0,1\}^{32}$.
4. VoterApp:
   (a) Generates a random number $r \in_R \{0,1\}^{\kappa}$ and $q_{\mathsf{left}} \in_R \{0,1\}^{224}$.
   (b) Encrypts c and r by pk_S, $E_{\mathsf{asym}} = \mathsf{AsymEnc}_{pk_S}(c, r)$.
   (c) Signs E_{asym} by $sk_{\mathcal{V}}$, $\mathsf{SignEncVote} = \mathsf{Sign}_{sk_{\mathcal{V}}}(E_{\mathsf{asym}})$.
   (d) Sends SignEncVote and the value $q = q_{\mathsf{left}} || q_{\mathsf{right}}$ to VFS[2].
5. VFS:
   (a) Stores SignEncVote and q.
   (b) Generates voteref.
   (c) Sends voteref to $\mathcal{V}$ for verification phase.

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

Fig. 2. The voting phase of the proposed protocol


Voter
VoterApp
VFS
1. Authentication
2. $CL = \{c_1, \ldots, c_m\}$
4. $SignEncVote, q$
5. $voteref$
3. Choose c, q_R
4. Generate r and q_L
$E_{asym} = AsymEnc_{pk_S}(c, r)$
$SignEncVote = Sign_{pk_V}(E_{asym})$
$q = q_L | q_R$
5. Store q and $SignEncVote$
Generate $voteref$

Verification Stage:

1. (a) VoterApp: 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 $q \overset{?}{=} q_\alpha \in Q = \{q_1, \cdots, q_m\}$ and $c \overset{?}{=} c_\beta \in \{c_1, \cdots, c_m\}$, where $\alpha, \beta \in \{1, \cdots, m\}$ [3].
   (a) Checks $\alpha \overset{?}{=} \beta$.
   (b) If $\alpha = \beta$, the vote is received and stored correctly in VFS.
   (c) Else, $\mathcal{V}$ puts an alarm (which basically shows that malware is present).

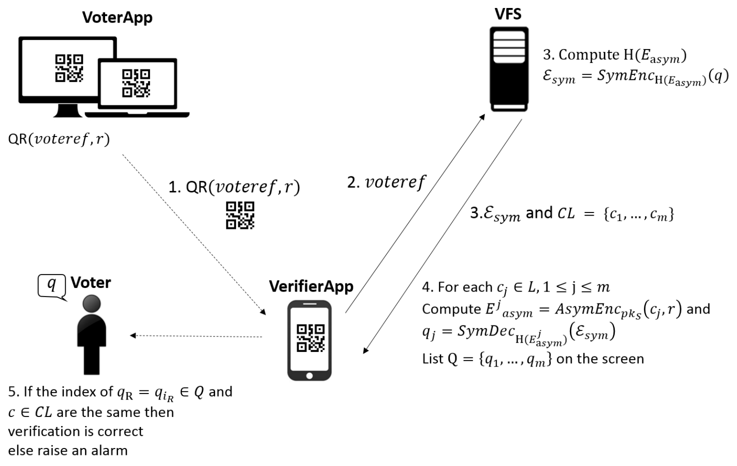
6 Security Analysis of Our Verification Mechanism

We next show the correctness and prove the security of our mechanism.

Theorem 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).*

[3] For usability concerns, 32-bit integers q is viewed to the voter as 4 characters.

Fig. 3. The Verification phase of the proposed protocol


VoterApp
VFS
3. Compute $\mathrm{H}(E_{asym})$
$\mathcal{E}_{sym} = SymEnc_{\mathrm{H}(E_{asym})}(q)$
QR($voteref, r$)
1. QR($voteref, r$)
2. $voteref$
3.$\mathcal{E}_{sym}$ and $CL = \{c_1, \ldots, c_m\}$
q
Voter
VerifierApp
4. For each $c_j \in L, 1 \leq j \leq m$
Compute $E^j{}_{asym} = AsymEnc_{pk_S}(c_j, r)$ and
$q_j = SymDec_{\mathrm{H}(E^j_{asym})}(\mathcal{E}_{sym})$
List Q = $\{q_1, \ldots, q_m\}$ on the screen
5. If the index of $q_R = q_{i_R} \in Q$ and
$c \in CL$ are the same then
verification is correct
else raise an alarm

Proof. During the verification phase, the VerifApp first obtains r and voteref from VoterApp. 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 voter $\mathcal{V}$ votes for the candidate $c_\beta \in CL$ where $\beta \in \{1, \cdots, m\}$. 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 $q_i = \mathsf{SymDec}_{\mathsf{H}(E^i_{\mathsf{asym}})}(\mathcal{E}_{\mathsf{sym}})$. Since the random value of the underlying asymmetric encryption r is given to the VerifApp, E^i_{asym} becomes deterministic. Therefore, VerifApp computes the correct value q successfully. Finally, the voter $\mathcal{V}$ searches for q (i.e., the last four characters of q which were shown on the screen of the voter computer) on the verification device and finds the index $\beta \in \{1, \cdots, m\}$ where $q_\beta = q$.

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 2 (Privacy against malicious VoterApp and its adversarial environment). *Our verification mechanism described in Section 5.3 is secure against malicious* VoterApp *or adversarial environment of* VoterApp.

Proof. Suppose VoterApp (or its adversarial environment) cheats and tries to send the vote for another predefined candidate. Below, we show that the voter $\mathcal{V}$ can easily detect this adversarial action via VerifApp.

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

$$q' = q'_{\mathsf{left}} || q_{\mathsf{right}} = \mathsf{SymDec}_{\mathsf{H}(E'_{\mathsf{asym}})}$$

where $E'_{\mathsf{asym}} = \mathsf{AsymEnc}_{pk_S}(c^*, r^*)$ and $q = q_{\mathsf{left}}||q_{\mathsf{right}}$. Namely, a malicious VoterApp can fool a voter by only showing the correct value q_{right} which is in the position of the chosen candidate c.

In our system, for usability concerns, a voter chooses only four characters during the voting phase. However, a malicious VerifApp may guess r^* in such a way that $q' = q'_{\mathsf{left}}||q_{\mathsf{right}} = \mathsf{SymDec}_{\mathsf{H}(E'_{\mathsf{asym}})}(\mathcal{E}_{\mathsf{sym}})$ where $E'_{\mathsf{asym}} = \mathsf{AsymEnc}_{pk_S}(c^*, r^*)$. However, this probability is $1/m \cdot \frac{1}{2^{32}}$ (choose the correct candidate of the voter and the corresponding value r^*). Because voting phase is independent for each voter, this probability goes to negligible for only 3 voters (i.e., $\frac{1}{2^{96}}$).

In the most general case, if a voter chooses the full length of q, then 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 infeasible because of the underlying encryption scheme. An attacker may also simply guess q^* which will be correct with probability at most $\frac{1}{2^k}$ where k is the length of q. □

Theorem 3 (Privacy against malicious VerifApp and its adversarial environment). *Our verification mechanism described in Section 5.3 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 $\mathcal{E}_{\mathsf{sym}}$. 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. Hence, VerifApp leaks no information about the choice of the $\mathcal{V}$. □

Theorem 4 (Privacy against malicious VFS). *Our verification mechanism described in Section 5.3 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 [8]. □

7 Complexity Analysis

In Table 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 1. Comparison of Computational Cost of the Verification Mechanisms where SymEnc, AsymEnc, Sign denote symmetric, asymmetric encryptions and digital signature.

	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	✓

7.1 Usability and Optimization Improvement for the Verification q

The size of q is important due to privacy and usability concerns. Firstly, because VerifApp computes $q_i = \mathsf{SymDec}_{\mathsf{H}(E^i_{\mathsf{asym}})}(\mathcal{E}_{\mathsf{sym}}), i = 1, \cdots, m$ and outputs the list $Q = \{q_1, \cdots, q_m\}$, each q_i must be random of equal size where $q_j = q$ for some $j = 1, \cdots, m$. Otherwise, a malicious VerifApp can easily detect the vote from only the size of the q value. However, it not usable and impractical for a voter to generate a random q of t-length bits in practice because, for example, if AES-256 and SHA3-256 were used as the encryption function then the voter must also generate a random q of size 256-bits. Therefore, instead of searching the value q itself, it is only sufficient to check the last 32 bits of q and $q_1, \cdots, q_m$. In other words, $q_{\mathsf{left}} \in \{0,1\}^{224}$ can be generated by VoterApp while $q_{\mathsf{right}} \in_R \{0,1\}^{32}$ can be provided by the $\mathcal{V}$ and it is sufficient to compare the last 32 bits q_{right}. In this way, a voter can easily compare the q on VoterApp to the q_j's on VerifApp. Hence, the protocol will be run as defined in Section 5.3, but both VoterApp and VerifApp will only display the last 32 bits of the verification parameters of q and $q_1, \cdots, q_m$. A voter has to confirm with her own eyes that the original q (i.e., the last 32 bits as 4 letters) exist in Q.

As an illustration, we present a sample shortening operation in Table 2 for the parameter q ='ThEParameterWillBelongToremember'. Let $\ell = 4$ and the position numbers are chosen as 8-bit words at the positions p_1, p_2, p_3 and p_4 respectively 29, 30, 31 and 32. Then, shortened value 'mber' is shown on the screen instead of whole q. In Table 3, Step 5 of the 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 $i = 1, \cdots, m$.

Table 2. Shortened verification parameter on the screen on of the voter computer.

Position #	1	2	3	4	···	25	26	27	28	29	30	31	32
q	T	h	E	P	···	r	e	m	e	m	b	e	r
Shortened q					···					m	b	e	r

Table 3. Original and shortened verification values.

c_i	q_i	t-**bits** q	**Shortened**
c_1	q_1	qxvsEgaKMXpwApGDsNnPaNhjTJYtqven	qven
⋮	⋮	⋮	⋮
c_β	q_β	ThEParameterWillBelongToremember	mber
⋮	⋮	⋮	⋮
c_m	q_m	RatqPKvgtTAFectHpOeteDoPYKbTkApp	kApp

8 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 election 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 plain form which may cause a privacy weakness and compromise the fairness of the election. 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 VoterApp and trusting a device running VerifApp.

In this paper, we have proposed a new verification mechanism that does not disclose any information about a voter's intention, if used by corrupted verification devices. Hence, our proposed verification system is strong against the aforementioned privacy weaknesses. 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.

Acknowledgment
The authors would like to thank the anonymous reviewers for their contributions and comments.

References

1. Gritzalis, D.A., Secure Electronic Voting, Advances in Information Security, Springer US, 2012.

2. Dawid Gawel and Maciej Kosarzecki and Poorvi L. Vora and Hua Wu and Filip Zagorski, Apollo - End-to-end Verifiable Internet Voting with Recovery from Vote Manipulation, Cryptology ePrint Archive, Report 2016/1037, 2016.
3. National Institute of Standards and Technology, FIPS 197: Advanced Encryption Standard (AES), Federal Information Processing Standards Publication Series, 2001.
4. 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.
5. 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. 339353, USENIX Association, Berkeley, CA, USA, 2002.
6. 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. 961987, 2014.
7. 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/WOTE10, pp. 116, USENIX Association, Berkeley, CA, USA, 2010.
8. Ansper, A. and Buldas, A. and Jurgenson, A. and Oruaas, M. and Priisalu, J. and Raiend, K. and Veldre, A. and Willemson, J. and Virunurm, K: E-voting Concept Security, Analysis and Measures, 2016.
9. Estonian National Electoral Committee: 2010 E-Voting System, General Overview-2010, `http://vvk.ee/public/dok/General_Description_E-Voting_2010.pdf`. (last access: 30 April 2016).
10. Stenerud, Ida Sofie Gebhardt and Bull, Christian: When Reality Comes Knocking Norwegian Experiences with Verifiable Electronic Voting, Electronic Voting, volume 205 of LNI, pp.21–33, GI, 2012.
11. Heiberg, Sven and Parsovs, Arnis and Willemson, Jan: Log Analysis of Estonian Internet Voting 2013-2015, IACR Cryptology ePrint Archive, 2015.
12. Meter, Christian: Design of Distributed Voting Systems, Department of Computer Science, Heinrich-Heine-University Dsseldorf, September, 2015.
13. Benaloh, Josh and Rivest, Ronald and Ryan, Peter Y A. and Stark, Philip and Teague, Vanessa and Vora, Poorvi: End-to-end verifiability. In: CoRRabs/1504.0 (2015). http://arxiv:org/abs/1504:03778.
14. Estonian National Electoral Committee: Statistics-Internet Voting- Voting methods in Estonia. http://www:vvk:ee/voting-methodsin-estonia/engindex/statistics/. Version: April 2015.
15. Halderman, John: Independent Report on E-voting in Estonia. https://estoniaevoting:org/. Version: September 2015.
16. Halderman, J. Alex and and Teague, Vanessa: The New South Wales iVote System: Security Failures and Verification Flaws in a Live Online Election, Springer International Publishing, 35-53, isbn:978-3-319-22270-7, September 2015.
17. Gjosteen, Kristian: The Norwegian internet voting protocol. In: EVoting and Identity 7187 LNCS, pp. 118. DOI 10.1007/9783642327476_1. ISBN 9783642327469. 2013.
18. Heiberg, Sven and Laud, Peeter and Willemson, Jan: The Application of I-voting for Estonian Parliamentary Elections of 2011, Lecture Notes in Computer Science series, 3rd International Conference on E-voting and Identity, Tallinn, volume 7187, Springer-Verlag, 2012.

19. Maaten, Epp: Towards Remote E-Voting, Estonian case, Electronic Voting in Europe, LNI Series, volume 47, pp. 83–100, GI, 2004.
20. Springall, Drew and Finkenauer, Travis and Durumeric, Zakir and Kitcat, Jason and Hursti, Harri and MacAlpine, Margaret and Halderman, J. Alex: Security Analysis of the Estonian Internet Voting System, Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS '14 series, pp.703–715, ACM, 2014.
21. Sven Heiberg and Jan Willemson: Verifiable Internet Voting in Estonia, 6th International Conference on Electronic Voting, Verifying the Vote, EVOTE 2014, Lochau/Bregenz, Austria, pp. 1–8, 2014.
22. IPhone and Android Apps Breach Privacy, `http://online.wsj.com/article/SB10001424052748704694004576020083703574602.html` (last access: 30 April 2016).
23. Egele, M., Kruegel, C., Kirda, E., Vigna, G.: PiOS: Detecting Privacy Leaks in iOS Applications, 18th Annual Network and Distributed System Security Symposium, 2011.
24. William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, Anmol N. Sheth, TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones, Proceedings of the 9th USENIX conference on Operating systems design and implementation, p.1–6, Vancouver, BC, Canada, 2010.
25. Mahaffey, K., Hering, J.: App Attack: Surviving the Explosive Growth of Mobile Apps, 2010.
26. ICTA-Information and Communication Technologies Authority: CEIR-Central Equipment Identity Register, `http://www.mcks.gov.tr/en/KonuDetay.php?BKey=47` (last access: 30 April 2016)
27. Bartlomiej Uscilowski: Security Response- Mobile Adware and Malware Analysis, `http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/madware_and_malware_analysis.pdf` (last access: 30 April 2016).
28. Ori Bach: Mobile Malware Threats in 2015, Fraudsters Are Still Two Steps Ahead, `https://securityintelligence.com/mobile-malware-threats-in-2015-fraudsters-are-still-two-steps-ahead/` (last access: 30 April 2016).
29. Felt, Adrienne Porter and Finifter, Matthew and Chin, Erika and Hanna, Steve and Wagner, David: A Survey of Mobile Malware in the Wild, 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.
30. A. K. Jain and D. Shanbhag: IT Professional: Addressing Security and Privacy Risks in Mobile Applications, volume 14, number 5, pp. 28–33, 2012.
31. Zhu, Hengshu and Xiong, Hui and Ge, Yong and Chen, Enhong: Mobile App Recommendations with Security and Privacy Awareness, Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14 series, pp.951–960, ACM, New York, USA, 2014.
32. Estonian Internet Voting Committee: Using ID-card and mobil-ID, May 2014: `https://www.valimised.ee/eng/kkk` (last access: 30 April 2016).
33. National Electoral Committee: Statistics about Internet Voting in Estonia, `http://www.vvk.ee/voting-methods-in-estonia/engindex/statistics` (last access: 30 April 2016).

34. El Gamal, Taher: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, Proceedings of CRYPTO 84 on Advances in Cryptology, Springer-Verlag New York, Inc., 1985.
35. Paillier, Pascal and Pointcheval, David: Efficient Public-Key Cryptosystems Provably Secure Against Active Adversaries, Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security, Advances in Cryptology, ASIACRYPT '99, pp. 165–179, Springer-Verlag, 1999.
36. Fujisaki, Eiichiro and Okamoto, Tatsuaki and Pointcheval, David and Stern, Jacques: RSA-OAEP Is Secure under the RSA Assumption, August 19–23, 2001 Proceedings, Advances in Cryptology — CRYPTO 2001, 21st Annual International Cryptology Conference, pp.260–274 Springer Berlin Heidelberg, Santa Barbara, California, USA, 2001.