Planning the next steps for Estonian Internet voting

Sven Heiberg[1], Kristjan Krips[2,4], and Jan Willemson[2,3]

[1] Smartmatic-Cybernetica Centre of Excellence for Internet Voting
Soola 3, 51004, Tartu, Estonia
sven@ivotingcentre.ee
[2] Cybernetica AS, Mäealuse 2/1, 12618, Tallinn, Estonia
{kristjan.krips,jan.willemson}@cyber.ee
[3] STACC, Narva mnt 20, 51009, Tartu, Estonia
[4] Institute of Computer Science, University of Tartu,
Narva mnt 18, 51009, Tartu, Estonia

Abstract. This paper considers the current state of Estonian Internet voting, identifies its shortcomings with respect to the present-day threat landscape, and discusses possible mitigation measures. It turns out that the area requiring the most attention and introduction of new measures is electronic identity. We also propose and analyse an update to the current Estonian individual vote verification protocol allowing to use PC as a verification device in case voting would move to mobile platforms.

1 Introduction

Casting a vote via Internet (i-voting) has been an option in Estonia since 2005. In 2019 Parliamentary elections, about 44% of all the votes were cast via this medium[5]. The system has been a subject of debates and research scrutiny since the beginning of deployment.

The first full security study was composed by a group of Estonian researchers in 2003, and later updated in 2010 [3]. In 2011, several potential problems (e.g. an invalid vote and proof-of-concept vote manipulation malware) surfaced in practice [6]. To counter them, individual verification option was added to Estonian Internet voting in 2013 [9]. In 2014, Springall *et al.* published a study pointing out the need for better verifiability of system-level properties [14]. As a result, in 2017, a completely re-designed IVXV protocol was deployed in Estonia [7].

In 2019, the debate about Internet voting security intensified again in Estonia after a new political coalition was formed. The Minister of Foreign Trade and Information Technology called together a committee that produced a list of open action items to potentially work on[6].

One of the ideas listed was to introduce the option of casting votes from mobile devices. Since this would be quite a significant change in the current

[5] https://rk2019.valimised.ee/en/participation/participation.html
[6] https://www.mkm.ee/sites/default/files/content-editors/e-valimiste_tooruhma_koondaruanne_12.12.2019_0.pdf, in Estonian

Estonian i-voting infrastructure, a separate analysis effort was initiated by the State Information System Authority and State Electoral Office.

The current paper builds on the initial findings gathered during the analysis[7]. Even though the original focus of the study was on mobile voting, it turns out that most of the issues and recommendations are actually more general and hold for the PC-based voting as well. In Section 2, we will first cover the general electronic identity and OS level threats. Section 3 discusses a possible change that introducing mobile voting may bring along for verification. In Section 4, we list and categorise existing and newly proposed mitigation measures. Finally, Section 5 presents the conclusions and sets directions for future work.

2 General risks

2.1 Threat actors

We start our study by identifying the main classes of threat actors.

- **Civil hacktivist seeking publicity.** Such an attacker is not necessarily malicious, but can cause unintended problems as side effects of his activities.
- **Single candidate trying to get more votes.** Such an attacker acting alone has limited resources, and his attacks are not likely to scale too much.
- **Political party trying to increase the number of seats.** Such an attacker has medium level of resources. It may have significant organisational capability, enabling certain attacks (e.g. coercion) to scale quite well.
- **Organization that aims at influencing policy decisions.** Such an attacker may have financial or ideological motives. This category includes large national or international enterprises, and their methods range anywhere from media campaigning and lobbying to direct bribery.
- **Foreign state-level actor interested in gaining more control over the country.** Such an attacker may have significant resources and access to rare technical capabilities (like zero-day attacks against common OS-es).

2.2 eID level risks

The Estonian Internet voting scheme relies heavily on the electronic identity (eID) infrastructure. There are currently three main eID solutions in use in Estonia.

- **ID-card**, first launched in 2002, was historically the first one and is still in wide use. The latest generation of ID-cards also possesses Near Field Communication (NFC) functionality which provides an option of using it in the context of m-voting as well.

[7] `https://www.valimised.ee/sites/default/files/uploads/eng/2020_m-voting-report.pdf`

- **Mobile-ID (mID)**, first launched in 2007, relies on the mobile phone Subscriber Identity Module (SIM) card as the key storage and cryptographic coprocessor.
- **Smart-ID (sID)**, first launched in 2016, is a software-only solution making use of a specific cryptographic scheme [4] where the signature key is split between the mobile device and server.

Right now, only ID-card and mID are used for i-voting.

Regarding security aspects, we consider the user's personal computing environment to be the weakest point in the e-ID ecosystem (see Section 2.3). All the above e-ID solutions use OS input-output mechanisms to display confirmation codes, enter PINs, etc. While ID-card is theoretically also usable with a PIN-firewalled smartcard reader featuring a separate PIN-pad, such readers are not widely available on the market and hardly anyone is using them in Estonia. If an attacker is able to monitor PIN entry of some legitimate session, he will later be able to enter the same PINs in the session of his choosing.

The most serious implication of this threat is an attacker submitting a vote using a compromised e-ID environment without the voter noticing. This is a problem both in the scenario when the attacker changes the originally submitted vote by re-voting, and also when the voter did not intend to vote at all (which is her legal right in Estonia). To complete such an attack, the attacker would need to implement his own voting client. This is feasible as the protocol description is public, even though not always sufficiently detailed [11].

There are a few aspects of user behaviour that contribute to this problem.

- General low level of digital hygiene, e.g. installing software from untrustworthy locations, carelessly opening email attachments, failure to keep the OS updated, etc. Such failures are often required as presumptions for attackers to launch malware-related attacks. Raising digital hygiene awareness is one key measure in raising the security level of every kind of digital services, including i-voting.
- Usage scenarios where ID-card is left attached to the working terminal for extended periods of time, e.g. as a login token. Even though short periods of legitimate ID-card usage might already be sufficient to implement an attack, the login token scenario has more problems. Namely, it is typically implemented at an OS level by leaving the card's authentication environment open. As a result, applications (including malicious ones) do not need to have access to PIN1 in order to perform authentication.

In general, one of the core problems seems to be that e-ID tokens (be it an ID-card or an mID SIM) are getting too intimately connected to the computing platforms and OSes. On one hand, this connection is convenient for the users, but at the same time it increases the attack surfaces and time windows. Whether the corresponding risk level still remains acceptable depends on the application scenario and threat actor we consider.

In case of electronic voting (both PC and mobile platform based), the integrity risks become significant when the attacks start to scale easily. We estimate that out of the threat actors listed in Section 2.1, high-resource state level attackers have the capacity to attack mobile platforms in a sufficiently scalable manner.

There are several possible mitigation measures to both prevent and detect unauthorised use of voter's e-ID. We will describe and discuss these measures in Section 4.3.

2.3 OS level risks

It is very hard to rationally estimate security level of an operating system or a particular version of it. There are several folk beliefs either based on common knowledge ("A newer version of OS should have less vulnerabilities") or some sort of personal view ("iOS is more secure than Android"), but these beliefs are quite hard to quantify.

Concerning the more updated versions having less vulnerabilities we may look at published vulnerability reports[8]. However, even one critical zero-day flaw may be sufficient for a state-level attacker to implement an attack, so the number of unpatched vulnerabilities is not necessarily a good measure of security.

Claims about the comparative security level of specific OSes (say, Android vs iOS or Linux vs Windows) are even more questionable. In case of open development models (Android, Linux) the attackers have easier time of discovering weaknesses, but at the same time public disclosures also speed up patching. For example, the potential bounties paid out for a fresh Android zero-click exploits are even higher that those of iOS[9]. This may be interpreted as an indication that such Android exploits are more rare. However, as argued by Ross Anderson, open and closed development models produce software of roughly comparable security level in the long run [2].

One way how such argumentation could be backed up is by comparing the number of exploits for open source and closed source software. There is a recent study by RAND Corporation, analysing a rare dataset of exploits based on zero-day vulnerabilities [1]. The dataset contained 74 exploits for open source software and 123 exploits for closed source software. The analysis showed that the survival probability for both classes of exploits was roughly the same, with the average life expectancy of an exploit for closed source software being 6.93 years and 6.51 years for open source software.

Acquiring superuser credentials In general, malware has two ways of getting root access to a device. It can either escalate privileges by using an exploit, or abuse the access that an unsuspecting user provides. On a PC, users may choose to run software with root user permissions, but doing the same in Android or iOS is not so easy. While root access gives more freedom to the user, it also breaks

[8] See e.g. `https://www.cvedetails.com/`

[9] `https://zerodium.com/program.html`

the security model of the underlying platform and makes it easier to attack the device. Thus, some vendors are trying to prevent the user from getting root access. E.g. with each new release of iOS, Apple has taken more serious steps to prevent users from getting root access (called *jailbreaking* in iOS community). At the same time, Apple is also working to decrease the motivation of jailbreaking in the first place (e.g. by increasing configurability of the official iOS). As a result, the iOS jailbreaking community has recently decreased[10].

Android rooting, on the other hand, is still happening a lot. It can be classified into hard rooting and soft rooting [15]. The former is done by flashing the device with an executable having root permissions, while the latter is based on exploiting vulnerabilities. Malware applications typically abuse the method from the second category. While there are plenty of vulnerabilities for Android, recent studies show that developing a universal exploit is not common due to the fragmentation of hardware and software [12]. Thus, root exploits are usually tailored either for specific devices, models or operating system versions [5]. However, public sources do not reveal information about zero day vulnerabilities that are stored by governmental entities. The report [1] by RAND corporation revealed that the median lifetime of an exploit based on a zero-day exploit is 5.07 years. Given the long lifetime of the exploits, it is likely that the arsenal of stored exploits is quite large.

3 Verification

One of the problems that arises when Estonia would introduce voting on mobile devices is losing mobile devices as an independent verification platform. Independence of the voting and verification platforms is important for the verification to fulfil its primary goal of detecting whether a vote has been manipulated by a potentially malicious (e.g. malware-infected) voting device [9].

In principle, there are two possible solutions to this problem.

1. Retain verification from the mobile device as the only option, hoping that voters will be using different devices for voting and verification.
2. Allow verifying mobile votes from a PC-based verification app (possibly also allowing verification with mobile devices in parallel).

The first option has the benefit of making use of workflows and apps that the voters are already accustomed to. On the other hand, many voters could perceive the need to grab for yet another mobile device as a superfluous action that gives them little to no added value. Some voters could try to trick the system and verify the vote with the voting device (say, using mirrors to relay

[10] It is hard to find reliable statistics about the actual usage of jailbreaking, but there are several recent posts written by the developers expressing their rapid decline of motivation to continue working on the respective applications, see e.g. `https://www.idownloadblog.com/2019/10/26/coolstar-sileo-development-suspended/` and `https://old.reddit.com/r/jailbreak/comments/7iu0sx/discussion_can_we_please_find_someone_to_help/dr2m6nx/`.

the QR code to the camera, or perhaps finding some esoteric apps that fulfil the same purpose). Behaviour of the voters in this scenario is hard to predict at this point; it would require conducting a dedicated user study.

In order to consider the second option above, we propose using PC-based verification to be used in conjunction with mobile voting.

The current verification scheme (see also [9]) is displayed in Figure 1.

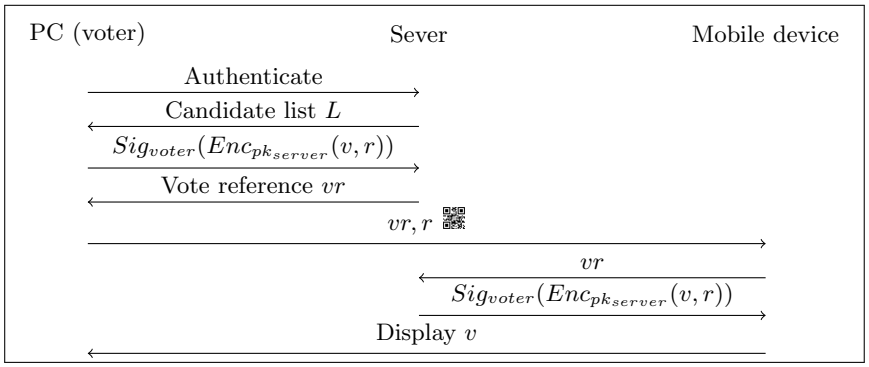

Fig. 1. Present Estonian voting and vote verification protocol

Note that the communication between the voter/PC and the mobile device is close range and optical. After the voting is over, the voting application displays a QR-code containing vote reference vr and encryption randomness r. The voter uses her mobile device to capture and decode the QR-code, downloads the corresponding encrypted vote from the server and decrypts it with the help of r (the latter operation being straightforward for the ElGamal encryption that the IVXV system currently uses). The vote is displayed on the mobile device screen for the user to inspect, again in close range and visually.

In case of voting with the mobile device, it would in principle be possible to display the QR-code on the mobile device screen and capture it with PC. However, not every PC has a camera, so we can not take this design path. Also, capturing the QR-code with the mobile device from the PC screen is a workflow familiar to the users, so we would like to retain it.

Of course, since the PC in the mobile voting scenario does not know vr and r, we have to change the content of the QR-code. Our proposal is to let the PC generate a one-time cryptographic (say, symmetric) key k and display it on screen as a QR-code. The mobile device will then capture and decode it, and use it to encrypt vr and r. The cryptogram will be sent to the PC that will decrypt its content and run the rest of the verification protocol in the familiar manner.

The resulting voting and verification scheme is is displayed in Figure 2

There are two main differences between the protocols presented in Figures 1 and 2.

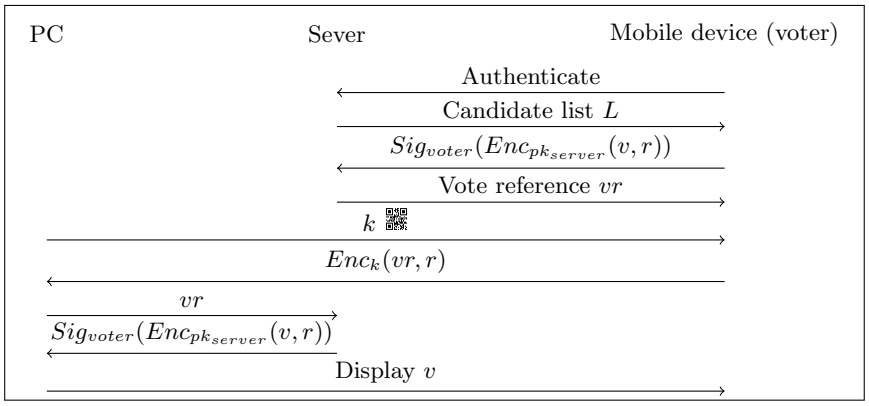

Fig. 2. Proposed Estonian voting and vote verification protocol

First, there is an extra cryptographic key k that aims at protecting vote secrecy by protecting confidentiality of the communication between the mobile device and verification PC. However, the voter has no assurance about the origin of the key – it may have been generated by an adversary in an attempt to breach the verification protocol.

Note that we are considering here the scenario where the verification PC is under the adversarial control. If besides the verification device either the server side or voting device would be malicious as well, we can not obtain meaningful security guarantees for the voter. Thus, it only makes sense to study the situation when the verification PC alone is malicious, but the voting device and server are honest.

Under such a scenario, there are two main kinds of attacks that the attacker can mount.

- Breaching privacy of the vote. This is an inherent risk present with any kind of verification that has to be accepted. This is similar to the present Estonian vote verification.
- Manipulating the verification process (manipulating the keys, delaying messages, etc.) leaving the voter with an impression that she voted for someone else. Note that under the attack model where only the verification PC is malicious, the vote was cast and recorded correctly. Thus, the adversary's activities will efficiently cause voter confusion, mistrust and general havoc. This is also what an attacker can do in the present scheme by manipulating the mobile verification device. There are standard measures designed for such a user experience (essentially, helpdesk will recommend the voter to use different voting and verification devices, and try again).

Thus we conclude that malicious manipulation of the verification device (and the key k along the way) does not make the situation worse compared to the present Estonian vote verification protocol.

The second difference between the two protocols is that there is an extra attack capability potentially gained by the adversary when he only manages to breach the voting device. Unlike the protocol in Figure 1, the protocol in Figure 2 is *active* in the sense that the voting device has to participate in initiating the verification process. Thus, the attacker could dynamically decide which voters to attack depending on whether they start with the verification process or not. For example, malware can delay delivery of the ballot and wait to see if the voting application is closed right after the vote has been cast via the user interface. In such a case it is unlikely that the voter verified the vote and thus malware can drop the vote without the voter noticing it. This kind of an attack could be prevented by introducing a feedback mechanism which notifies the voter once a vote has been successfully cast. This mitigation measure is discussed in Section 4.3.

4 Mitigation measures

In this Section, we are going to elaborate on possible mitigation measures for the risks listed in Section 2. Table 1 summarises the measures and classifies them according to their aim.

4.1 Awareness measures

Increase digital hygiene It is important to raise the general awareness level of digital hygiene. For example, it would have a significant positive impact if many citizens would regularly update their software to patch existing vulnerabilities. While such action is necessary, it won't be possible to educate every voter. In addition, state level actors are able to bypass antivirus software and have access to exploits built on top of zero day vulnerabilities [1].

Promote verification Currently, the rate of verifiers is about 4-5%[11], but the more there are, the smaller attacks we are able to detect [9]. In case individual verification would be more widespread, it would also act more as a preventive measure. When an attacker wants to change the election outcome, the attack should be executed silently. Thus, widespread individual verification can reveal if votes get dropped or changed by malware, and thereby deter such attacks from attackers who have to prevent detection. However, the current vote verification system is not able to detect malware that casts a re-vote which overwrites voter's original choice. The following mitigation measures also address the issue of preventing such malware from succeeding. As a possible new detection mechanism, establishing a feedback channel can also be considered (see Section 4.3).

[11] `https://www.valimised.ee/en/archive/statistics-about-internet-voting-estonia`

Table 1. Classification of mitigation measures based on their effect to i-voting.

	Prevention	**Detection**	**Recovery**
Increase digital hygiene	●	●	
Promote verification	◐[1]	●	
Introduce a feedback channel	◐[2]	●	
Do not support legacy mobile operating systems	◐[3]		
Obfuscation	◐[4]		
Add freshness notification to vote verification	◐[1]	●	
Prevent ID-card from being in the reader when not used	●		
Promote the usage of PIN-pad based ID-card readers	●		
Require both ID-card signature and mobile-ID/Smart-ID signature	●		
Analyse i-voting logs		●[5]	
Allow to re-vote on during i-voting period	●[6]		●[7]
Allow to re-vote on paper on election Sunday	●[6]		●[7]
Postpone i-voting			●[8]
Fall back to paper voting after a large scale attack			●[8]

● = measure is effective ◐ = measure is partially effective

[1] In case individual verification is widespread, the motivation for some types of attacks falls.
[2] A feedback channel may stop an attacker who wants to invisibly interfere.
[3] An attacker is able to run his own voting client on legacy operating systems.
[4] Client side restrictions can be bypassed if adversary has full control over the voting device.
[5] This is a system-wide measure to detect anomalies.
[6] The option of the voter re-voting limits the coercer's capability to ensure that coercion was successful.
[7] This is an individual recovery measure for voters who were coerced.
[8] This is a system-wide measure to recover from a malfunction or from an attack.

Prevent ID-card from being in the reader when not used Discourage the scenarios where it is required to leave the ID-card in the reader for extended periods of time, and practices where the card's authentication environment is left open on the OS level. In case voter's device is infected with malware and the voter is not using a PIN-pad-based ID-card reader, the malware could re-vote and thus overwrite the voter's initial choice.

Promote the usage of PIN-pad based ID-card readers Target e-ID solutions with better separated authentication factors. E.g. on the regular PC platforms, make use of PIN-pad-equipped ID-card readers. Without such a reader, malware could issue a re-vote right after the voter has voted as the ID-card is

still in the reader. Currently individual verification would not detect such an attack. For usage with mobile devices as terminals, NFC cards with integrated displays and PIN-pads could be utilised. In case individual verification would provide some integrity guarantees as described in Section 4.3, the NFC based vote signing could be a step forward. While the majority of smartphone users rely on Smart-ID and mobile-ID for daily interactions, the NFC based vote casting could offer a way to prevent malware from re-voting by more strongly separating the e-ID token and the main computing platform.

4.2 Existing measures

Analyse i-voting logs Log analysis can reveal anomalies which can be used to identify attacks. For example, it is possible to monitor when and how many times people vote, which e-ID tools and OSes they use, whether and when they verify their votes, etc. [8].

Allow to re-vote during i-voting period Allowing the voter to overwrite her vote by casting a new i-vote is a measure designed to prevent coercion. The rationale is that if the coercer knows that the voter can easily change her vote, his motivation to coerce (say, to pay for a vote) decreases. It is also possible to go to the polling station during the advance voting period to vote on paper. To enable this, the i-voting period currently ends two hours before the advance paper voting period.

Allow to re-vote on paper on election Sunday If the voter was coerced in the end of the i-voting and she was unable to attend the polling station during the extra two hours of advance voting period, there was no way to cast a re-vote up to 2019. However, this will change in 2021 when the i-voters will have the option to re-vote on paper during the election Sunday as well.

Postpone i-voting This is a legal measure that can be executed when a large scale attack is detected.

Fall back to paper voting after a large scale attack This is a legal measure that allows to cancel i-voting in case a large scale attack is detected that can not be mitigated by other means. That way the voters can be asked to vote on paper during the election Sunday. This is also one of the reasons why i-voting should be limited to the advance voting period.

4.3 Newly proposed measures

Introduce a feedback channel A feedback channel (say, an SMS or email) can be used to notify the voters about their act of voting. This measure would be useful in multiple scenarios. For example, the voter would be able to detect

re-voting malware or malware that drops votes based on a prediction on whether the voter is going to verify the vote. In the latter case, the voter could detect vote dropping attacks even without using the verification system, which would make it difficult for an attacker to avoid detection. This is relevant e.g. when considering the proposed verification scheme for m-voting discussed in Section 3. Similarly to individual verification, the feedback channel is mainly a measure to detect interference. However, as a side effect, it can also deter an attacker in the fear that the attack to be revealed. Again, similar to individual verification, we can hope that this deterrence will also act as an efficient prevention measure.

Introducing a feedback has actually been considered before in Estonia, and the main reason why it has not been implemented this far is the fear of making coercion attacks (e.g. vote buying/selling) easier. Thus, before taking a decision on whether to introduce such a measure or not, a wider analysis including also legal aspects should be conducted.

However, from the technical point of view we make the following observations about the potential coercion-enabling risk.

- Even if the coercer observes a voter during the voting session and demands to see her feedback channel (say, mailbox) during this session, the voter can still re-vote later.
- We assume that it is hard for the coercer to maintain physical access at many victims at the same time (most importantly, during the last minutes of the voting period). However, it is possible to demand virtual presence, say, in the form of e-mailbox passwords. To counter this threat, we can use an email redirection service that the voter can privately configure. In Estonia, there is the official @eesti.ee email redirection service that can be used for this purpose. Every citizen has an official government-supplied email address of the form `personalcode@eesti.ee` and is expected to redirect the emails from there to his/her personal email account.
- If the coercer is trying to get a control of all the digital channels of a voter, there must be sufficient evidence of this attempt so that the voter can turn to the law enforcement. However, the main rationale behind making use of the @eesti.ee redirection service is to lower the coercer's incentive to control the voter's main mailbox, since this gives the coercer no guarantee of detecting a revote.
- If the coercer is willing to go as far as ceasing all the e-ID means from the voter in an attempt of blocking her option of logging onto the @eesti.ee redirection service, he can use the same approach to block the voter's revoting ability already with the present system. However, the voter is still able to cast a paper ballot in case she has access to a passport, driver's licence or any other valid ID. Thus, from this point of view, introducing the notification feedback channel does not open significant new attack vectors.

Of course, in order for the feedback channel to be an efficient measure, care has to be taken in implementation. For example, it should be difficult for a piece of malware operating in the user's voting environment to block the feedback

channel. If mobile voting would be introduced, we have to take into account that people would probably vote and read SMSes from the same device. This would render SMS as a potential feedback channel weaker since malware operating on the mobile device could cast a vote without the user knowing, and also block the SMS that notifies the voter about the vote being cast on her behalf.

A possible drawback of the feedback channel measure is also the possibility for an attacker to generate havoc by sending out a lot of fake notifications. A possible countermeasure would be to include a statement signed with a key of the election organiser. In any case, also the legal impacts to voting freedom need to be assessed before such a measure can be implemented.

Add freshness notification to vote verification Estonian i-voting system gives voters the option to use individual verification. This means that the voters can check whether their vote reached the voting system. The existing implementation allows to verify the vote during a limited time window, which has historically been set between half an hour and an hour. Thus, after casting a vote, the voter has up to an hour to take a smartphone with a verification application and check whether her ballot reached the voting system. It is important to note that the voter is not able to check whether the ballot that reached the voting system will be counted in the tally as such an ability would also make vote selling easier.

The current verification system is optimised for being coercion resistant and thus verification does not reveal if a re-vote has been cast. Now, imagine what could happen when a voting device would be infected and controlled by malware. As noted in Section 2.2, malware can use voter's e-ID if it is directly connected to the infected device, by recording and re-using the PIN codes. The voter is physically not sufficiently fast to remove the ID-card from the card reader to prevent malware from accessing it (which can be done in a fraction of a second). Verifying the previous vote would still succeed with the current set-up.

However, the existing individual vote verification mechanism can be easily extended so that it would also provide a partial integrity check. The verification system could notify the voter during verification whether the given vote was overwritten or not. If the voter performs this verification after she has removed the ID-card from the possibly malicious device and does not use it any longer during the i-voting period, the voter can be sure that malware has not abused access to the ID-card. The verification time window is short and is probably not suitable for re-voting in case the initial vote was given under coercion. The coerced voter can re-vote later after the verification time-window has passed as then the coercer can not check whether the coerced vote was overwritten. In case coercion takes place during the last hour of the i-voting period, the coerced voter can fall back to casting a re-vote on paper (see Section 4.2).

Until ID-card's NFC interface is not used for other activities on a mobile device (nor over a regular smart card reader), the voter can be sure that malware does not have access to the ID-card. This measure only works when the voter

is careful and when malware can not rely on mobile e-ID solutions (i.e. mID or sID) to cast a (re)vote.

Require both ID-card signature and mobile-ID/Smart-ID signature The idea is to force the vote casting to depend on two independent devices. The vote should be accepted only if the timestamps of both signatures are within a certain time-limit. This measure would lower usability of electronic voting, but it may be an acceptable trade-off with increased resistance against malware attacks.

4.4 Other possible measures

Do not support legacy mobile operating systems It is possible to try to restrict the official voting client so that it would run only on up-to-date operating systems. However, the effectiveness of this measure depends on the capabilities and attack goals of the attacker.

The problem is that a really determined and resourceful attacker can develop a voting client also for an old and vulnerable platform where he can potentially run it without the user knowledge. This is doable as the voting protocol is open even though not always documented the best way [11]. If an e-ID utility is also accessible without the user knowledge, the attacker can mount an attack against vote integrity. Efficient measures against this threat include increasing the general level of digital hygiene and establishing a feedback channel as described above.

However, not supporting legacy OSes by the official voting client has a positive effect on vote privacy. If the voter only has access to the voting client on an up-to-date OS, it will be harder for an attacker to develop and deploy malware that would attempt to, say, read the user's screen during the voting session.

Obfuscation Obfuscation and malware detection measures only work against some attackers. State level actors and researchers have the capability to reverse engineer the voting application to detect which measures are used. Once the measures are known, they can be bypassed, assuming that the attacker has root access to the device. A good example of bypassing obfuscation and malware detection measures is given by Specter *et al.* in case of Voatz [13].

5 Conclusions and future work

In this paper we reviewed the current state of Estonian Internet voting, identified its shortcomings with respect to the present-day threat landscape, and discussed possible mitigation measures. Even though the original motivation of the research was the question about feasibility and the associated risks of mobile voting, the conclusions are more general and hold for PC-based i-voting as well.

The most serious attack vectors against Estonian Internet voting system include malicious unauthorised use of e-ID devices (ID-card, mobile-ID). With such an access, the attacker can cast a re-vote and thereby overwrite the choice of the voter. One of the strongest measures suggested against such a threat is end-to-end (E2E) verifiability that would allow every voter to verify that her vote has been correctly counted in the final tally. Unfortunately, such a strong notion of verifiability potentially conflicts with voter privacy and coercion-resistance.

For example, the (to-date the most comprehensive) report by Kiniry *et al.* studies a number of proposed E2E voting schemes and concludes that "No usable E2E-VIV protocol in existing scientific literature has receipt freedom when the voting computer is untrusted." [10]. Currently, the Estonian Internet voting scheme does not provide full E2E verifiability, but instead balances the verifiability and coercion-resistance requirements using a combination of individual verification [9], server-side auditability [7] and an option of re-voting. However, the search for a better balance is on-going and the question of introducing some form of E2E verifiability without increasing the coercibility level of the protocol too much is one of the main directions of future research.

There are still residual risks that E2E verifiability does not address. For example, if a citizen never intended to vote, but due to hostile take-over of her e-ID, the attacker manages to submit a vote on her behalf, the voter would not learn about this fact even if there is strong E2E verifiability in place. Thus, we propose to add an independent notification channel. The question which channel is the optimal one (also considering the implications on coercion-resistance) is still open and needs future study. This includes the need for additional legal analysis on such a measure.

We have also made two other new recommendations – adding freshness notification to the individual verification protocol, and requiring several independent e-ID tools to submit a valid vote. These recommendations also require further analysis from the coercibility and usability points of view, respectively.

In conclusion – any voting protocol suite is a complex set of mechanisms balancing between conflicting requirements. Improving one component may actually decrease the overall security level of the whole system. Thus, before implementing any of the above-mentioned measures, a holistic study of the whole suite needs to be conducted. This will be general direction of our future research steps.

Acknowledgements. This paper has been supported by the Estonian Personal Research Grant number 920 and European Regional Development Fund through the grant number EU48684.

References

1. Ablon, L., Bogart, A.: Zero Days, Thousands of Nights: The Life and Times of Zero-Day Vulnerabilities and Their Exploits. RAND Corporation (2017)

2. Anderson, R.: Open and closed systems are equivalent (that is, in an ideal world). In: Perspectives on free and open source software. MIT Press, Cambridge, MA (2005), https://www.cl.cam.ac.uk/~rja14/Papers/toulousebook.pdf
3. Ansper, A., Buldas, A., Jürgenson, A., Oruaas, M., Priisalu, J., Raiend, K., Veldre, A., Willemson, J., Virunurm, K.: E-voting concept security: analysis and measures (2010), https://www.valimised.ee/sites/default/files/uploads/eng/E-voting_concept_security_analysis_and_measures_2010.pdf
4. Buldas, A., Kalu, A., Laud, P., Oruaas, M.: Server-Supported RSA Signatures for Mobile Devices. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10492, pp. 315–333. Springer (2017)
5. Gasparis, I., Qian, Z., Song, C., Krishnamurthy, S.V.: Detecting android root exploits by learning from root providers. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 1129–1144. USENIX Association, Vancouver, BC (Aug 2017), https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/gasparis
6. Heiberg, S., Laud, P., Willemson, J.: The Application of I-Voting for Estonian Parliamentary Elections of 2011. In: Kiayias, A., Lipmaa, H. (eds.) E-Voting and Identity - Third International Conference, VoteID 2011, Tallinn, Estonia, September 28-30, 2011, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7187, pp. 208–223. Springer (2011). https://doi.org/10.1007/978-3-642-32747-6_13, https://doi.org/10.1007/978-3-642-32747-6_13
7. Heiberg, S., Martens, T., Vinkel, P., Willemson, J.: Improving the Verifiability of the Estonian Internet Voting Scheme. In: Krimmer, R., Volkamer, M., Barrat, J., Benaloh, J., Goodman, N.J., Ryan, P.Y.A., Teague, V. (eds.) Electronic Voting - First International Joint Conference, E-Vote-ID 2016, Bregenz, Austria, October 18-21, 2016, Proceedings. Lecture Notes in Computer Science, vol. 10141, pp. 92–107. Springer (2016). https://doi.org/10.1007/978-3-319-52240-1_6, https://doi.org/10.1007/978-3-319-52240-1_6
8. Heiberg, S., Parsovs, A., Willemson, J.: Log Analysis of Estonian Internet Voting 2013-2014. In: Haenni, R., Koenig, R.E., Wikström, D. (eds.) E-Voting and Identity - 5th International Conference, VoteID 2015, Bern, Switzerland, September 2-4, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9269, pp. 19–34. Springer (2015). https://doi.org/10.1007/978-3-319-22270-7_2, https://doi.org/10.1007/978-3-319-22270-7_2
9. Heiberg, S., Willemson, J.: Verifiable internet voting in Estonia. In: Krimmer, R., Volkamer, M. (eds.) 6th International Conference on Electronic Voting: Verifying the Vote, EVOTE 2014, Lochau / Bregenz, Austria, October 29-31, 2014. pp. 1–8. IEEE (2014). https://doi.org/10.1109/EVOTE.2014.7001135, https://doi.org/10.1109/EVOTE.2014.7001135
10. Kiniry, J., Zimmerman, D., Wagner, D., Robinson, P., Foltzer, A., Morina, S.: The future of voting: end-to-end verifiable Internet voting (2015), U.S. Vote Foundation, https://www.usvotefoundation.org/E2E-VIV
11. Krips, K., Farzaliyev, V., Willemson, J.: Developing a Personal Voting Machine for the Estonian Internet Voting System (2020), submitted
12. Meng, H., Thing, V.L., Cheng, Y., Dai, Z., Zhang, L.: A survey of Android exploits in the wild. Computers & Security **76**, 71–91 (2018). https://doi.org/https://doi.org/10.1016/j.cose.2018.02.019, http://www.sciencedirect.com/science/article/pii/S0167404818301664

13. Specter, M.A., Koppel, J., Weitzner, D.J.: The Ballot is Busted Before the Blockchain: A Security Analysis of Voatz, the First Internet Voting Application Used in U.S. Federal Elections (2020), https://internetpolicy.mit.edu/wp-content/uploads/2020/02/SecurityAnalysisOfVoatz_Public.pdf
14. Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., MacAlpine, M., Halderman, J.A.: Security Analysis of the Estonian Internet Voting System. In: Ahn, G., Yung, M., Li, N. (eds.) Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014. pp. 703–715. ACM (2014). https://doi.org/10.1145/2660267.2660315, https://doi.org/10.1145/2660267.2660315
15. Zhang, H., She, D., Qian, Z.: Android root and its providers: A double-edged sword. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. p. 1093–1104. CCS '15, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2810103.2813714, https://doi.org/10.1145/2810103.2813714