Security Analysis of the Estonian Internet Voting System

J. Alex Halderman[1] Harri Hursti Jason Kitcat[2] Margaret MacAlpine
Travis Finkenauer[1] Drew Springall[1]

[1] University of Michigan, Ann Arbor, MI, U.S.A.
[2] Open Rights Group, U.K.

Technical report – May 2014

For additional materials and contact information, visit estoniaevoting.org.

1. INTRODUCTION

Several countries have experimented with casting votes over the Internet, but today, no nation uses Internet voting for binding political elections to a larger degree than Estonia [38]. When Estonia introduced its online voting system in 2005, it became the first country to offer Internet voting nationally. Since then, it has used the system in local or national elections five times, and during recent elections 20–25% of participating voters cast their ballots online [24].

Nevertheless, the system remains controversial. Many Estonians view Internet voting as a source of national pride, but one major political party has repeatedly called for it to be abandoned [29]. Although Estonia's Internet Voting Committee maintains that the system "is as reliable and secure as voting in [the] traditional way" [18], its security has been questioned by a variety of critics, including voices within the country (e.g. [43, 46]) and abroad (e.g. [52]).

For these reasons, the Estonian Internet voting (I-voting) system represents a unique and important case study in election security. Its strengths and weaknesses can inform other countries considering the adoption of online voting, as well as the design of future systems in research and practice.

In this study, we evaluate the system's security using a combination of observational and experimental techniques. We observed operations during the October 2013 local elections, conducted interviews with the system developers and election officials, assessed the software through source code inspection and reverse engineering, and performed tests on a reproduction of the complete system in our laboratory. Our findings suggest the system has serious procedural and architectural weaknesses that expose Estonia to the risk that attackers could undetectably alter the outcome of an election.

The weakness of the Estonian system stems from its basic design. Most e-voting schemes proposed in recent years use cryptographic techniques to achieve end-to-end (E2E) verifiability [9]. This means that anyone can confirm that the ballots have been counted accurately *without* having to trust that the computers or officials are behaving honestly. In contrast, Estonia's design implicitly trusts the integrity of voters' computers, server components, and the election staff.

Rather than proving integrity through technical means, Estonia relies on a complicated set of procedural controls, but these procedures are inadequate to achieve security or transparency. During our in-person observations and in reviewing official videos of the 2013 process, we noted deviations from procedure and serious lapses in operational security, which leave the system open to the possibility of attacks, fraud, and errors. Transparency measures, such as video recordings and published source code, were incomplete and insufficient to allow outsider observers to establish the integrity of results.

The threats facing national elections have shifted significantly since the Estonian system was designed more than a decade ago. Cyberwarfare, once a largely hypothetical threat, has become a well documented reality [44, 51, 55, 56], and attacks by foreign states are now a credible threat to a national online voting system. Given that Estonia is an EU and NATO member that borders Russia, it should not discount the possibility that a foreign power would interfere in its elections.

To test the feasibility of such attacks, we reproduced the I-voting system in a lab environment and played the role of a sophisticated attacker during a mock election. We were able to develop client-side attacks that silently steal votes on voters' own computers, bypassing safeguards such as the national ID smartcard system and smartphone verification app. We also demonstrate server-side attacks that target the implicitly trusted counting server. By introducing malware into this server, a foreign power or dishonest insider could alter votes between decryption and tabulation, shifting results in favor of the attacker's preferred candidate.

We conclude that there are multiple ways that state-level attackers, sophisticated online criminals, or dishonest insiders could successfully attack the Estonian I-voting system. Such an attacker could plausibly change votes, disrupt elections, or cast doubt on the integrity of results. These problems are difficult to mitigate, because they stem from basic architectural choices and fundamental limitations on the security and transparency that can be provided by procedural controls. For these reasons, we recommend that Estonia discontinue the I-voting system.

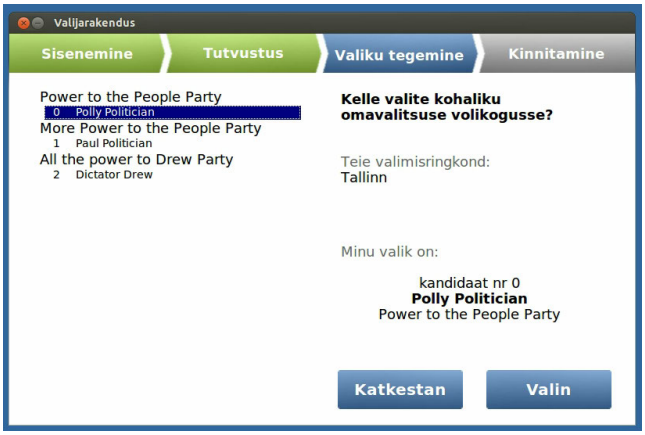


Figure 1: **I-voting client software**

2. BACKGROUND

Our observations and analysis focus on the Estonian I-voting system as it was used for the 2013 municipal elections ("KOV2013") [20]. In these elections, Internet voting was available for seven days, from October 10–16, and the main in-person polling took place on election day, October 20. Results were declared that evening. According to official statistics, 133,808 votes were cast online, corresponding to 21.2% of participating voters [24].

In this section, we review the design and operation of the I-voting system. Figure 2 gives an overview of the interactions between the main system components.

2.1 National ID Cards

An essential building block of the I-voting system is Estonia's national ID infrastructure [13]. Estonian national ID cards are smartcards with the ability to perform cryptographic functions. With the use of card readers and client software, Estonians can authenticate to websites (via TLS client authentication [48]) and make legally binding signatures on documents [14]. The cards are popularly used for online banking and accessing e-government services [19]. In the I-voting system, voters use their ID cards to authenticate to the server and to sign their ballots.

Each card contains two RSA key pairs, one for authentication and one for digital signatures. Certificates binding the public keys to the cardholder's identity are stored on the card and in a public LDAP database [15]. The card does not allow exporting private keys, so all cryptographic operations are performed internally. As an added safeguard, each key is associated with a PIN code, which must be provided to authorize every operation.

2.2 I-Voting Server Infrastructure

The majority of the I-voting server source code is published to a GitHub repository 2–3 weeks prior to the election [28]. The server infrastructure is configured in a public ceremony one week before the election and consists of four machines:

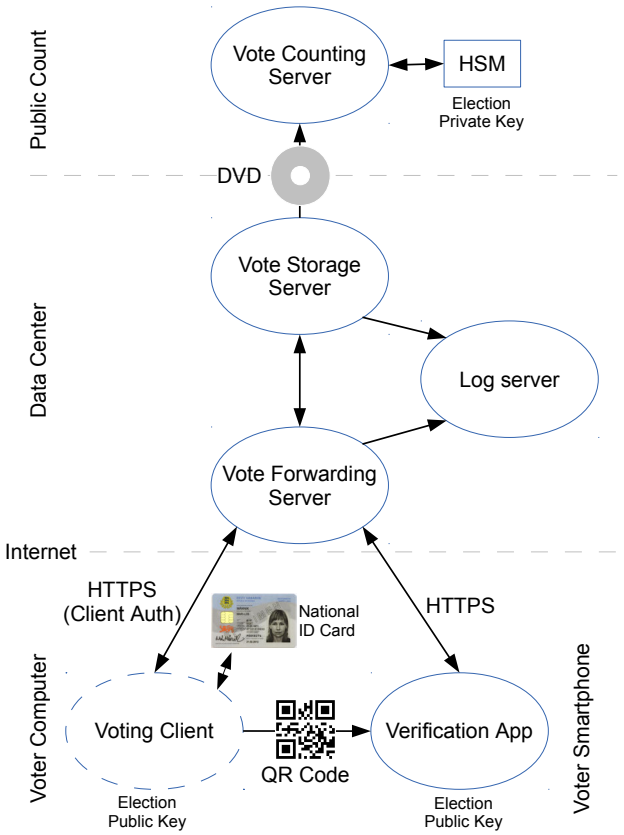


Figure 2: **I-voting system overview** — Major components of the system, and how information flows between them.

Vote forwarding server (VFS/HES) The VFS is the only publicly accessible server. It accepts HTTPS connections from the client software, verifies voter eligibility, and acts as an intermediary to the backend vote storage server, which is not accessible from the Internet.

Vote storage server (VSS/HTS) The VSS is a backend server that stores signed, encrypted votes during the online voting period. Upon receiving a vote from the VFS, it confirms that the vote is formatted correctly and verifies the voter's digital signature with the help of an external OCSP server.

Log server This server is an internal logging and monitoring platform that collects events and statistics from the VFS and VSS. The source code and design have not been published. While this server is not publicly accessible, it can be accessed remotely by election staff.

Vote counting server (VCS) The VCS is never connected to the network and is only used during the final stage of the election. Officials use a DVD to copy encrypted votes (with their signatures removed) from the VSS. The VCS is attached to a hardware security module (HSM) that contains the election private key. It uses the HSM to decrypt the votes, counts them, and outputs the official results.

2.3 Voting Processes

The I-voting system uses public key cryptography to provide a digital analog of the "double envelope" ballots often used for absentee voting [22]. Conceptually, an outer envelope (a digital signature) establishes the voter's identity, while an inner envelope (public key encryption) protects the secrecy of the ballot. Once each voter's eligibility has been established, the signature is stripped off, leaving a set of anonymous encrypted ballots. These are moved to a physically separate machine, which decrypts and counts them.

Voting At the start of each election, the election authority publishes a set of voting client applications for Windows, Linux, and Mac OS, which can be downloaded from https://valimised.ee. The client is customized for each election and includes an election-specific public key.

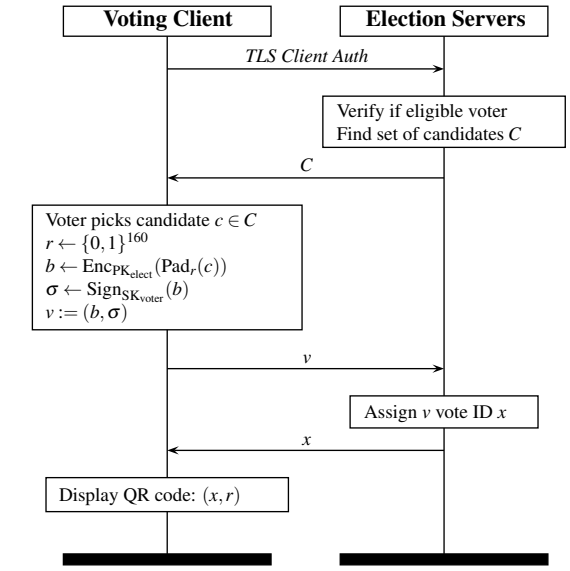
Figure 3a shows the protocol for casting a vote. To begin the process, the voter launches the client application and inserts her ID card. She enters the PIN associated with her authentication key, which is used to establish a client-authenticated TLS connection to the election server. The client verifies the server's identity using a hard-coded certificate. The server confirms the voter's eligibility based on her public key and returns the list of candidates for her district [11].

The voter selects her choice c and enters her signing key PIN. The client pads c using OAEP and randomness r, encrypts it with the election public key, and signs the encrypted vote with the voter's private key. The signed and encrypted vote is transmitted to the server, which associates it with an unguessable unique token x and returns x to the client. The client displays a QR code containing r and x.

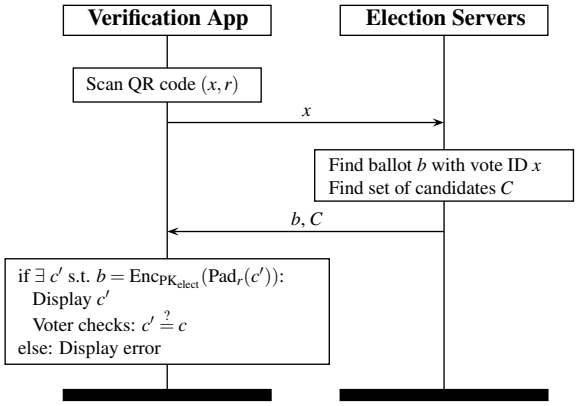
As a defense against coercion, voters are allowed to vote multiple times during the online election period, with only the last vote counted. All earlier votes are revoked but retained on the storage server for logging purposes. While the voting client indicates whether the user has previously voted, it does not display the number of times. The voter can also override her electronic vote by voting in person on election day.

Verification The voter can confirm that her vote was correctly recorded using a smartphone app provided by the election authority [23, 25, 26], as seen in Figure 4. This protocol is shown in Figure 3b. The app scans the QR code displayed by the voting client to obtain r and x. It sends x to the election server, which returns the encrypted vote (but not the signature) as well as a list of possible candidates. The app uses r to encrypt a simulated vote for each possible candidate and compares the result to the encrypted vote received from the server. If there is a match, the app displays the corresponding candidate, which the voter can check against her intended choice. The server allows verification to be performed up to three times and up to 30 minutes after casting.

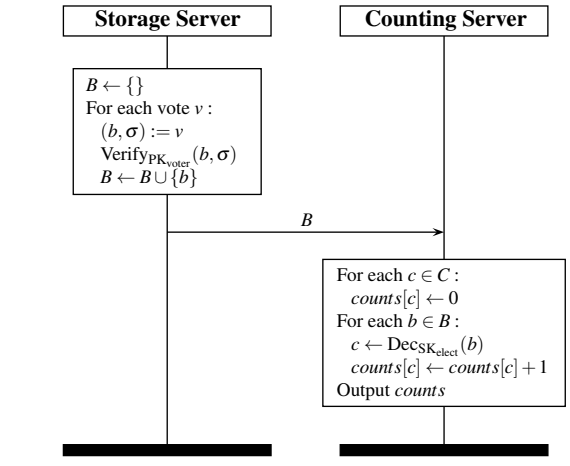
Tabulation Figure 3c shows the sequence that occurs at the conclusion of an election. After online voting has ended, the storage server processes the encrypted votes to verify the



(a) **Vote casting process**



(b) **Vote verification process**



(c) **Vote counting process**

Figure 3: **I-voting system protocols**

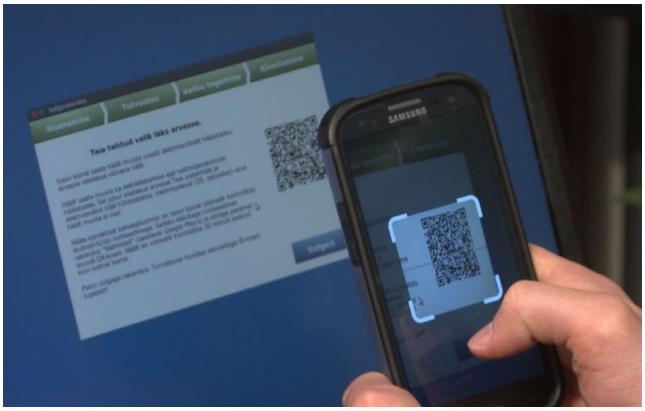


Figure 4: **Verification app** — A smartphone app allows voters to confirm that their votes were correctly recorded. We present two strategies that an attacker can use to bypass it.

signatures and remove any revoked or invalid votes. Officials then export the set of valid votes after stripping off the signatures, leaving only anonymous encrypted votes. These are burned to a DVD to transfer them to the counting server.

The counting server is attached to the HSM that contains the election private key. The server uses the HSM to decrypt each vote and tallies the votes for each candidate. Officials export the totals by burning them to a DVD. These results are combined with the totals from in-person polling stations and published as the overall results of the election.

3. OBSERVATIONS

The first part of our analysis follows an observational methodology. Four of the analysis team (Halderman, Hursti, Kitcat, and MacAlpine) visited Estonia as officially accredited election observers during the October 2013 municipal elections and witnessed the operation of the I-voting servers. During that time, they also met with election officials and the I-voting software developers in Tallinn and Tartu. Later, we closely reviewed published artifacts from the election: the server source code [28], written procedures [16], and nearly 20 hours of official videos that recorded the I-voting configuration, administration, and counting processes [17]. Ultimately, we identified a range of problems related to poor procedural controls, lapses in operational security, and insufficient transparency measures.

3.1 Inadequate Procedural Controls

While the Internet Voting Committee has published extensive written procedures covering many steps in the election process [16], we observed that some procedures were not consistently followed and others were dangerously incomplete.

For example, tamper-evident seals are used on the server racks in the data center[1]. Election staff were unsure when asked what would happen if the seals were found to be compromised. Procedures for handling anomalous conditions that could imply an attack appear to be inadequately specified or do not exist.

Anomalous situations that did occur during the 2013 election were handled in an ad hoc manner, sometimes at the discretion of a single individual. On multiple occasions, we observed as data center staff restarted server processes to resolve technical glitches, and repeated failed commands rather than troubleshooting the root cause of the issues. Similar issues were observed during tabulation during poll workers' attempt to boot the vote storage server. The machine reported errors stating that the drive configuration had changed—a possible indication of tampering. Instead of investigating the cause of the alert, staff bypassed the message.

Some procedures appeared to change several times over the observation period. For example, observers were initially allowed to film and photograph inside the server room, but were prohibited the next day because of the unsubstantiated claim of "possible electronic interference." In a similarly abrupt change in procedure, observers were required to leave their mobile phones outside the data center after multiple days where this was not the policy. Rewriting the rules on the fly suggests that the procedures had not been adequately thought out or were insufficiently defined for staff to implement them consistently.

Even when procedural safeguards were clear, they were not always followed. For example, procedure dictates that two operators will be present when performing updates and backups [11]. Second-operator procedures like this are commonplace in situations, such as voting, where the outcome must be robust to a single point of failure. On October 14 we observed that a lone staff member performed these tasks. The same staff member arrived with update disks and left with backup disks. Without a second operator present, the security of the system relies on the integrity of a single staff member.

3.2 Lax Operational Security

Since the I-voting system treats parts of the server infrastructure as implicitly trusted, the processes used to install and configure those servers determine the security of the election. We witnessed numerous serious lapses in operational security both during our on-site observations and in the official videos released by the Internet Voting Committee.

Pre-election setup Several problems can be seen in the official videos of the pre-election setup process, which takes place in the National Electoral Committee's offices in Tallinn.

The videos show election workers downloading software for use in the setup process from a public website over an unsecured HTTP connection (Figure 6). They also explicitly disabled certificate checking while downloading a program, via HTTPS, whose purpose was to verify the integrity of the election application source code. A network-based man-in-the-middle attacker could have compromised either of these applications.

[1]We note that tamper-evident seals like those used in Estonia are easily compromised [34–37], and their use in elections has been called into question in other countries [4].

In other instances workers unintentionally typed passwords and PINs in view of the camera. These included personal national ID card PINs and server root passwords (Figures 7 and 8). Similar problems were present during daily maintenance operations in the data center. Physical keys to the server room and rack were revealed to observers; these keys could potentially be duplicated using known techniques [41].

The most alarming operational security weakness during pre-election setup was workers using an "unclean" personal computer to prepare election client software for distribution to the public. As seen in Figure 5, the desktop has shortcuts for an online gambling site and a BitTorrent client, suggesting that this was not a specially secured official machine. If the computer used to prepare the client was infected with malware, malicious code could have spread to voters' PCs.

Daily maintenance We observed other operational security weaknesses during daily maintenance operations that took place during the voting period. The I-voting servers are hosted at a government data center in Tallinn, and workers go there to perform operations directly from the server consoles. While there were security video cameras at the data center, there appeared to be no 24/7 security personnel nor any definitive information on who monitored the cameras.

Standard practice during the daily maintenance operations is for workers to log in to the election servers under the `root` account and perform operations at the shell. Logging in as `root` is contrary to security best practice, as it simplifies many attacks, eliminates user-based privilege separation for operator functions, and increases the risk of human error.

Unencrypted daily backups were casually transported in workers' personal backpacks. DVDs holding updated voter lists from the population register were handled in a similarly casual way after having been created, we were told, by a member of staff at their own computer. We did not observe any audit trail or checks on the provenance of these DVDs, which were used daily at the heart of the I-voting system.

Tabulation The tabulation process at the end of the election was also concerning. After the votes were decrypted on the counting server, an unknown technical glitch prevented workers from writing the official counts and log files to DVD. Instead, they elected to use a worker's personal USB stick to transfer the files to an Internet-connected Windows laptop. This USB stick had been previously used and contained other files, as shown in Figure 10. This occurred despite protest from an audience member and deviated significantly from the written procedure, adding multiple potential attack vectors. Malware present on the laptop could potentially have altered the unsigned ballots, or malware on the USB stick could have been transferred to the trusted counting server.

These instances illustrate a pattern of operational security lapses on the part of the workers who operate the I-voting system. This is particularly alarming given the high degree of trust the I-voting system design requires of the election servers, client software, and the election workers themselves.

3.3 Insufficient Transparency

While the I-voting system involves many trusted components, we have been impressed by the initial steps towards transparency implemented by election officials, including allowing in-person public observation, publishing videos of operator tasks, and releasing large parts of the system source code. However, these measures are incomplete and insufficient to fully establish the integrity of election results.

One limitation is that these measures cannot show whether malicious actions were performed on the servers or hard disks before recording and observation commenced. To illustrate these limitations, we conducted an experimental server-side attack on a reproduction of the system, as detailed in Section 5.2. We have published a series of videos[2] matching the procedures in the official videos step-by-step, but where the result of the election is dishonest because of malware surreptitiously introduced before the start of the pre-election setup process.

The official video records often did not capture everything going on in the facilities. On many occasions, there were multiple machines or screens in use simultaneously but only a single camera. On Monday, October 14, we were physically present in the server room when one of the servers produced abnormal output, which appeared to be a failure of the update operation. The official video recording was following the other display, and the operator, upon seeing the error message, quickly flushed it from the screen.

Although we attempted to follow these simultaneous operations during our in-person observations, on occasion the operators appeared to be deliberately evading us. In another instance when an error appeared on the server console, an election worker quickly cleared the display and then asked us to rotate out of the room and let other observers in, allowing him a block of observation-free time.

An auditor from a major international consulting firm had been hired by the Internet Voting Committee, and his report also documents procedural and operational shortcomings [49]. However, the auditor's role was chiefly to observe, and he was not provided with the access needed to confirm secure operation of the system.

Election officials have made large parts of the server software open source [28]. This is a positive measure as it allows independent review and assessment — indeed, our study made extensive use of the code. However, security-critical pieces of code are missing from the published sources, including the entire client application and code that is executed on every server machine (see Section 4.1).

Officials told us that the client source is not released (and furthermore, the client binary is obfuscated) because they are concerned that attackers might modify it and distribute a trojan lookalike. Creating client-side malware is entirely feasible without the source, as we show in Section 5.1. At the same time, keeping source files secret prevents members

[2] See https://estoniaevoting.org/videos/.

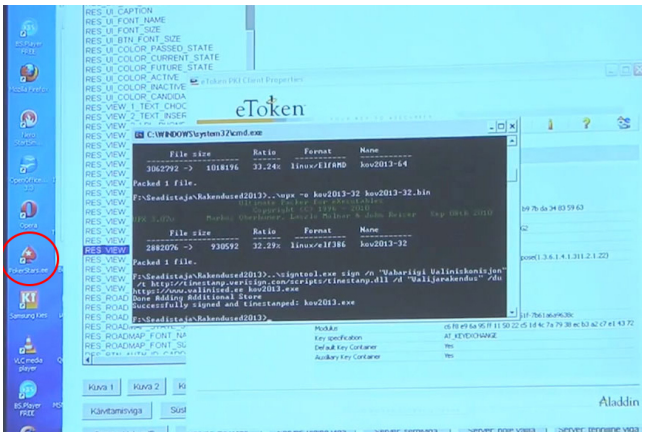


Figure 5: **Unsecured build systems** — Operators used a PC containing other software, including PokerStars.ee, to build the official voting client for distribution. This risks infecting the client with malware spread from the unsecured PC.

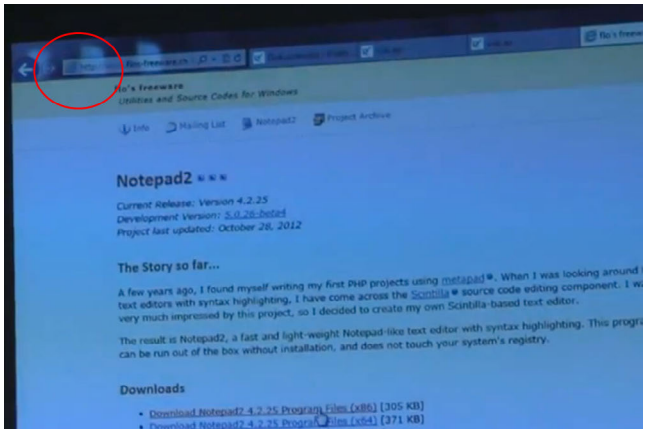


Figure 6: **Insecure software downloads** — Operators downloaded software over insecure connections for use in pre-election setup. An attacker who injected malicious code into these downloads might be able to compromise the process.

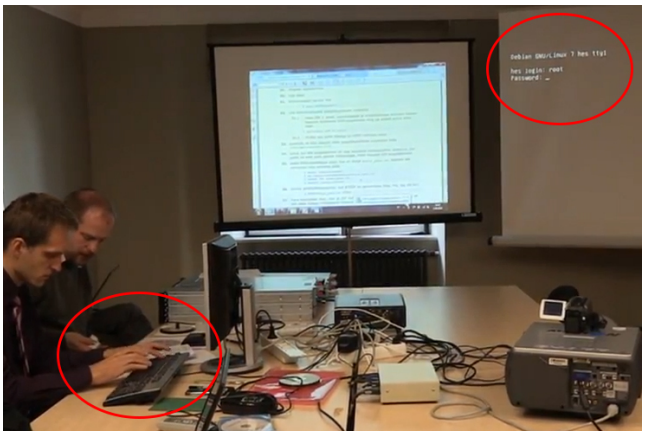


Figure 7: **Keystrokes reveal critical passwords** — Videos posted by officials during the election show operators typing, inadvertently revealing critical system passwords.

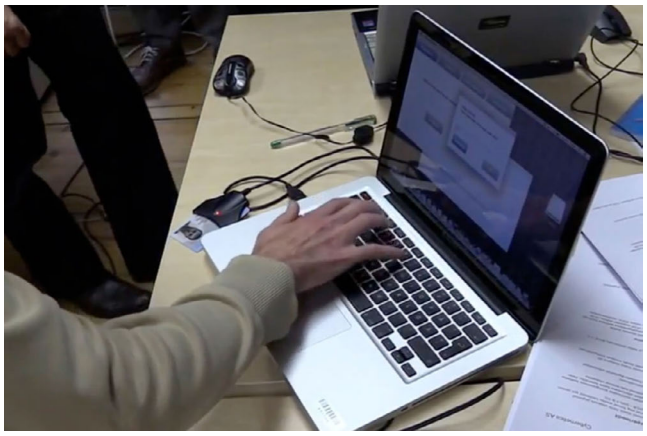


Figure 8: **Video shows national ID PINs** — During pre-election setup, someone types the secret PINs for their national ID card in full view of the official video camera.

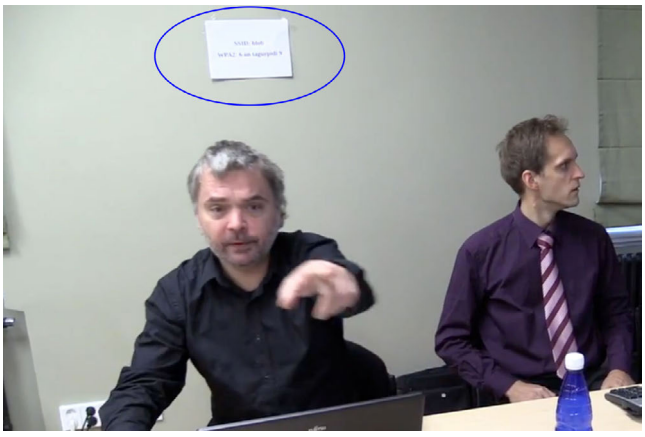


Figure 9: **Posted Wi-Fi credentials** — The official video of the server setup process reveals Wi-Fi credentials, which have been posted on the wall.

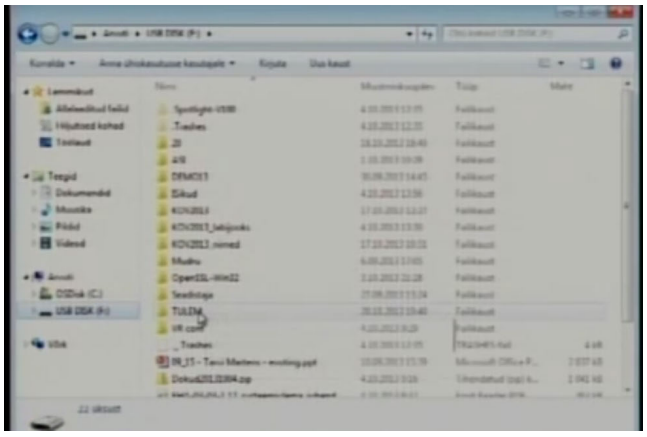


Figure 10: **Personal USB stick** — Against procedures, an official used a USB stick, containing personal files, when moving the official election results off of the counting server.

of the public from understanding what they are being asked to run on their computers, and it increases the risk that any centrally introduced malicious changes to the official client will go undetected.

As these transparency measures are practiced today, the videos, public observation, and open source components risk providing a false sense of security. It is plausible for the systems to be corrupted prior to videotaping, for media used to update systems to be maliciously modified, or for unpublished pieces of software to contain errors or malicious code — all invisible to the public under current transparency measures.

3.4 Vulnerabilities in Published Code

The published portions of the I-voting server software [28] contain 17,000 lines of code, with 61% in Python, 37% in C++, and the remainder in shell scripts. The codebase is quite complex, with a large number of external dependencies, and exhaustively searching for vulnerabilities would be a challenging task well beyond the scope of this project. We understand that volunteers from the Estonian security community have already audited it — a testament to the virtues of publishing code.

Nonetheless, we uncovered some additional vulnerabilities in the process of conducting our other experiments. We have disclosed these issues to the Internet Voting Committee and will not release details publicly until after the 2014 election. We do not consider these specific vulnerabilities to be severe problems, but they are a reminder that open source cannot guarantee the absence of vulnerabilities [40].

4. EXPERIMENTAL METHODOLOGY

In order to further investigate the security of the I-voting system, we set up a copy of the system in our lab, reproducing the software and configuration that was used for the October 2013 elections. Compromising a real election would have involved numerous legal and ethical violations, but our laboratory setup allowed us to play the role of attacker in our own mock election without hazard.

4.1 Mock Election Setup

To reproduce the I-voting servers, we used the source code published on GitHub by the election authority [28]. We set up the servers by following published configuration documents [16] and matching step-for-step the actions performed by election workers in the official videos [17]. As part of this process, we generated our own public keys for the web server's TLS certificate as well as for the election's encryption key.

Some components were missing from the published server code, but we attempted to recreate them as faithfully as possible. First, the software for the log server, the `ivote-monitor` package, was not made available; we operated a standard `rsyslog` [2] server instead. Additionally, there was no source provided for the `evote_post.sh` script, which runs on every server during installation of the packages. We attempted to reproduce its functionality based on output shown during server configuration in the official videos.

In real election, Estonia uses a hardware security module (HSM) in order to handle the election private key and decrypt votes. Since we did not have compatible hardware available, we emulated the HSM in software using OpenSSL and Python. Since this is a deviation from the fielded setup, we ensured that none of our attacks depend on vulnerabilities in the HSM itself.

Similarly, we set up our own certificate authority and OCSP responder as a stand-in for the ID card PKI. We assumed for purposes of this study that those pieces of the infrastructure are secure.

For the client software, we started with the official voting client from the 2013 election, which we downloaded from the election website [21] in October. For convenience, we focused on the Linux version of the client. Since the election public key and server certificate are hard-coded into the client, we needed to patch it in order to replace these with the keys of our mock election and server. Similarly, we used the official source code for the Android-based verification app [27] and modified it to communicate with our server.

Because we did not have access to actual Estonian ID cards for testing, we had to virtualize them. By setting up our own local certificate authority, we were able to generate identities for voters and infrastructure for the PKI. We replaced the ID card on the client with a software-based emulator that speaks the protocol expected by the voting client application. Once again, we ensured that the success of our attacks does not rely on the changes we made; we treated the emulated ID cards as secure black boxes for purposes of our analysis.

4.2 Threat Model

After setting up the mock election, we attempted to compromise it, allowing ourselves the resources and capabilities of a sophisticated but realistic attacker. This attacker could be a foreign state, a well-funded criminal organization, or a dishonest election insider. These kinds of attackers are difficult to defend against, but they represent a serious and realistic threat to modern elections given the enormous financial and policy consequences at stake.

Since the time the Estonian system was introduced, cyberwarfare has become a well documented reality. Chinese espionage against U.S. companies [44], U.S. sabotage of Iran's nuclear enrichment program [51], and attacks by the U.K. against European telecommunications firms [56] are just a few examples. Estonia itself suffered widespread denial-of-service attacks in 2007 that have been linked to the Russia [55]. An increasing number of nations possess offensive computer security capabilities [47], and investment in these capabilities is reported to be growing at a significant rate [30].

Such an attacker would have powerful capabilities. We assume they could obtain a detailed knowledge of the I-voting system's operation, which can be gleaned from published sources and reverse engineering (as we did), from insider

knowledge, or by compromising systems used by the software developers and election officials. We assume that if reverse engineering is required, the attacker has sufficient human and technical resources to accomplish this on a short timescale. Such capabilities are assumed to be within the grasp of a state intelligence agency.

For client-side attacks, we assume that the attacker has the ability to deliver malware to voters' home computers. This could be done externally to the voting system, either by purchasing pre-existing criminal resources, such as a botnet, or by buying or discovering zero-day vulnerabilities in popular software. Another route would be to compromise the voting client before its delivery to voters, either by a dishonest insider who can alter the software or by other attackers who can compromise the computers used to build it. The operational security problems documented in Section 3.2 suggest that this is a practical mode of attack.

Figure 11: **Malware records secret PINs** — Estonians use their national ID smartcards to sign and cast their ballots. We developed demonstration client-side malware that captures the smartcard PINs and silently replaces the user's vote.

5. ATTACKS

We used our reproduction of the I-voting system to experiment with a range of attacks. The I-voting system places significant trust in client and server components, making these highly attractive targets for an attacker.

While certain server operations are protected by cryptography (e.g., cast votes cannot be decrypted on the front-end web server, since it lacks the requisite private key), in other instances the servers are completely trusted to perform honestly and correctly when handling votes. Similarly, while the smartphone verification app gives voters some ability to check that the client software is behaving honestly, there are major limitations to this safeguard that can be exploited to hide malicious client behavior.

We experimentally verified that these trusted components are vulnerable by conducting two sets of demonstration attacks against them in our mock election setting. The first type are attacks on the client within reach of a financially capable attacker, in which an attacker can change votes in a retail manner for large numbers of individual voters. The second kind are server-side attacks within the reach of a well-resourced state-level attacker or dishonest insider, in which an attacker could change the wholesale results of the entire election through an exploit on the vote counting server.

5.1 Client-side Attacks

The voter's client machine is a trusted component of the I-voting system, so malware on the client could potentially tamper with the voter's ballot. There are several ways that an attacker might try to infect a sufficient number of Estonian clients to alter election results in a close race. One is to rent out bots from pre-existing botnets. Botnet operators frequently rent them on the black market, and these can be targeted to a specific country or region [12]. A second way would be to discover or purchase a zero-day exploit against popular software used in Estonia. While this would be expensive, it would not be out of reach for a state-level attacker—several companies specialize in selling zero-day exploits to governments [31]. A third strategy would be to infect the official I-voting client before it is delivered to voters.

If the attacker's goal is merely to disrupt the election, far fewer infected system would be required. Estonian law allows election officials to cancel online balloting if a problem is detected [32]. In that case, Internet voters would have their electronic votes canceled and be required to go to the polls on election day. This is a useful emergency measure, but it requires election officials to both detect the attack and make a snap decision about its severity. Suppose an attacker infected a small number of clients with vote-stealing malware, allowed some of them to be detected, and designed the attack such that it was difficult to quickly determine the size of the infected population. Under these circumstances, officials might be compelled to cancel the online portion of the election, yet the attacker would need relatively few resources.

In order to investigate how an attacker could modify votes through the client application, we implemented two experimental attacks. Both assume that the attacker has used one of the techniques noted above to initially infect the client machine. Each attack uses a different mechanism to defeat the smartphone verification app (see Section 2.3), which is the only tool available to voters to detect whether their ballot choices have been manipulated by client-side malware.

Each attack involves a hidden malicious process that runs alongside the I-voting client application and tampers with its execution. In order to develop them, we first needed to reverse engineer parts of the client software used in 2013. The client is closed source, and the developers took measures to complicate reverse engineering. The executable is obfuscated using the UPX packing tool. Strings, public keys, and other resources are hidden by XORing them with the output of a linear congruential generator. These measures did not significantly complicate the construction of our attacks. We used the UPX application with the `-d` switch to unpack the

binary and used the IDA Pro disassembler and Hex-Rays decompiler to reverse the portions necessary for our attacks.

Ghost Click Attack In the first attack, we use malware on the client machine to silently replace the user's vote with a vote in favor of an attacker-selected candidate. At a high level, the malware silently sniffs the victim's PIN during their original voting session. The real vote is cast, and everything appears normal, including the verification smartphone app if the voter uses it. Then, the malware waits until it is too late to verify again — either until the 30 minute time limit has passed or until after the user closes the client software and the QR code can no longer be scanned.

At that point, the malware checks whether the voter's ID card is still present in the computer. If so, it opens a copy of the I-voting client in a hidden session and, through keystroke simulation, submits a replacement vote. In the case that the ID card has already been removed, the malware remains dormant until the card is inserted again. Since Estonian ID cards are utilized for a variety of applications, many voters are likely to use their cards again within the one-week online voting period.

In our implementation, the malware attaches to the voting client process and captures PINs by setting breakpoints using `ptrace` [1]. Upon reaching a breakpoint, it reads the PIN from the client's memory and stores it for future use. Although our implementation runs in userspace, a kernel-level rootkit could be used to make it even more difficult to detect [33]. The malware could be extended to sniff PINs opportunistically any time voters use their ID cards, such as when logging into a bank.

Bad Verify Attack The Ghost Click attack defeats the verification app, but applying it on a large scale would lead to a suspiciously high number of replacement votes. We also experimented with a stealthier but more complicated style of attack that targets the verification app directly.

The verification app is premised on the notion that the smartphone is an independent device that would be unlikely to be compromised at the same time as the client PC. Therefore, it should be impossible for malware on the PC to change a ballot and hide that fact during verification.

However, smartphones today are not well isolated from users' PCs, as there is typically regular communication between the two devices. Users frequently plug their phones into their PCs to charge them or to transfer files. User content is regularly synchronized between devices through Google Drive, Dropbox, and other cloud services. Android even allows users to remotely install applications on their phones from their PCs through the Google Play Store web interface, and there are similar measures for Apple and Microsoft platforms [45].

As a result of this convergence, there are abundant means by which malware on the PC could attempt to infect the user's phone. If the attacker can do this, they can create a dishonest verification app that colludes with malware on the PC to fool the voter.

To experiment with such an attack, we implemented tandem PC and smartphone malware. The PC malware determines which candidate the voter actually selected and modifies the QR code so that it encodes the chosen candidate in the padding randomness. The malicious app behaves just like the real verification app, except that it displays whatever candidate is embedded in the QR code, rather than the candidate for whom the vote was actually cast. This lets the PC malware arbitrarily change the vote that gets submitted without being detected by verification or causing a suspicious number of replacement votes.

This form of attack adds complexity, due to the need to compromise both devices. If used on a large scale, it carries an elevated possibility of detection, since some users may attempt verification with devices owned by others. However, it illustrates that as PCs and smartphones continue to converge, it will become increasingly unsafe to treat them as independent devices.

5.2 Server-side Attacks

The integrity of the count depends on the correct operation of the counting server and its HSM, which are the only components with the ability to decrypt votes. Similarly, ballot secrecy depends on the counting server to not leak information about the correspondence between encrypted and decrypted votes. An attacker who injects malicious software into the counting server can violate these critical security properties.

Although the counting server is not connected to a network, there are a number of other means by which it might be attacked. State-level attackers are known to employ firmware-based malware [5], for instance, which could be used to infect the BIOS or hard disk before delivery to election officials. Sophisticated attackers are known to target component supply chains and distribution infrastructure [39].

We were informed during the observation mission that the Internet Voting Committee has a bid process every election to rent the servers that they use. We are concerned that attackers could credibly target the machines through this process.

Another route would be to compromise the server software before it is installed at the beginning of the election. We pursued this strategy in our experiments.

Injecting malware Despite procedural safeguards [16], an attacker who strikes early enough can introduce malicious code into the counting server by using a chain of infections that parallels the configuration process. During pre-election setup, workers use a development machine, which is configured before setup begins, to burn Debian Linux installation ISOs to DVD. These DVDs are later used to configure all election servers. If the machine used to burn them is compromised — say, by a dishonest insider, an APT-style attack on the development facility, or espionage — the attacker can leverage this access to compromise election results.

We experimented with a form of this attack to successfully change results in our mock election setup. We first created a modified Debian ISO containing vote-stealing malware

intended to execute on the counting server. The tainted ISO is repackaged with padding to ensure that it is identical in size to the original. In a real attack, this malicious ISO could be delivered by malware running on the DVD burning computer, by poisoning the mirror it is retrieved from, or by a network-based man-in-the-middle.

Defeating integrity checking During the setup process, election workers check the SHA-256 hash of the ISO file against the SHA256SUMS file downloaded via anonymous FTP from debian.org. Since FTP does not provide cryptographic integrity checking, a network-based man-in-the-middle could substitute a hash that matched the malicious ISO. However, this hash would be publicly visible in videos of the setup process and might later arouse suspicion.

An attacker who had compromised the DVD burner computer could achieve greater stealth. To demonstrate this, we implemented a custom rootkit that defeats the hash verification. Our rootkit is a kernel module that hooks system calls in order to cause the hash verification to succeed and the original ISO's hash to be printed. Hash checks applied in this way are only a minor speedbump under our threat model.

Vote-stealing payload After passing the hash check, the tainted ISOs are used to install the OS on all election servers, spreading the infection. When the new OS boots, the malware checks whether the machine is configured as the counting server, in which case it launches a vote-stealing payload.

During the counting process, this payload acts as wrapper around the process responsible for using the HSM to decrypt votes. As real votes are decrypted, it inspects them on the fly and replaces them with votes for the attacker's preferred candidate. (In our demonstration, we change 100% of the votes, but it would be straightforward to implement a more subtle algorithm that manipulates an arbitrary fraction.) The altered votes are then counted and released as the official results. Such as attack would be unlikely to be detected, as there are no audit mechanisms to check the accuracy of the decryption.

It took less than two man-weeks to devise, implement, and test this server-side vote-stealing attack. The majority of this time was spent finding an appropriate and sufficiently silent way to insert the trojan into the OS installation procedures, due to unfamiliarity with the Debian package system and installation process.

We note that this is far from the only means by which an attacker could inject a malicious payload into the servers. Several other pieces of closed-source software of unknown or untrusted provenance could be vehicles for attacks. These include the `evote_post.sh` script, missing from the server source code repository, which runs on all servers, as well as the driver software for the HSM, which is a closed-source application manually installed to the counting server from a DVD. These programs all touch critical, trusted portions of the I-voting system, yet they are not reviewable by the public and not integrity checked through any visible procedures.

In addition, although the official videos show that election workers build the I-voting server software packages from the published code, they first install prebuilt copies of these packages from an unknown source in order to retrieve necessary dependencies. While the prebuilt packages are later removed, if they are malicious, then merely installing them could trigger execution of attacker code on the supposedly "clean" build machine.

The key lesson from this attack is that it is extremely difficult to ensure the integrity of code running on a critical system, particularly when faced with the possibility of sophisticated attacks or dishonest insiders. If any element in the lineage of devices that handle the software installed on the counting server is compromised, this could jeopardize the integrity of election results [54].

6. DISCUSSION

Though we have spent the majority of this report discussing weaknesses and risks, we would be remiss if we failed to acknowledge the great lengths that the I-voting system developers, security staff, and officials go to in their efforts to protect the election system.

One core strength of the I-voting system is Estonia's national ID card infrastructure and the cryptographic facilities it provides. While the ID cards cannot prevent every important attack, they do make some kinds of attacks significantly harder. The cards also provide an elegant solution for remote voter authentication, something few countries do well.

The Internet Voting Committee's willingness to release source code is a very positive step for transparency. This shows confidence in the software's developers and demonstrates officials' desire to work with the security community at large. Providing access to the source allows many parties to analyze it — not only international researchers like us but also the domestic security community, who have an even greater interest in the system's secure operation. For these reasons, we urge the committee to go further and release the source code to the I-voting client and the missing portions of the server code discussed in Section 4.1.

Finally, we commend the Internet Voting Committee for their dedication to improving the election system. Since its inception in 2005, the system has undergone significant changes. From the switch to a standalone client, to the deployment of the log server that enhances forensic and monitoring capabilities, to the addition of the verification app, the I-voting system has not stood still. Yet as we have argued, even these and an array of other useful safeguards are not enough to secure Estonia's online elections in the face of a determined and well-resourced modern attacker.

Opportunities for Innovation While the risks of Internet voting are clear, the benefits are uncertain. Many Estonians support I-voting because they believe there is widespread fraud in the country's paper-based system. Whether or not these concerns are founded, the I-voting system can do little to help, since nearly 80% of votes are still cast on paper.

Fortunately, there are safe and effective ways to apply new technology to secure paper-based voting.

In recent years, researchers have developed methods that can dramatically increase the security of paper ballots. Statistical risk-limiting audits [7, 8, 42, 53] can minimize the risk of error or fraud during tabulation. Cryptographic techniques that achieve end-to-end verifiability [6, 10, 50] enable individual voters to verify that every vote has been counted accurately. Estonia has an opportunity to be the first country in the world to adopt these technologies on a national scale.

7. CONCLUSIONS

Compared to other online services like banking and e-commerce, voting is an exceedingly difficult problem, due to the need to ensure accurate outcomes while simultaneously providing a strongly secret ballot. When Estonia's I-voting system was conceived in the early 2000s, it was an innovative approach to this challenge. However, the designers accepted certain tradeoffs, including the need to trust the central servers, concluding that although they could take steps to reduce these risks through procedural controls, "the fundamental problem remains to be solved" [3]. More than a decade later, the problem remains unsolved, and those risks are greatly magnified due to the rapid proliferation of state-sponsored attacks.

As we have observed, the procedures Estonia has in place to guard against attack and ensure transparency offer insufficient protection. Based on our tests, we conclude that a state-level attacker, sophisticated criminal, or dishonest insider could defeat both the technological and procedural controls in order to manipulate election outcomes. Short of this, there are abundant ways that such an attacker could disrupt the voting process or cast doubt on the legitimacy of results. Given the current geopolitical situation, we cannot discount state-level attacks targeting the system in future elections.

Due to these risks, we recommend that Estonia discontinue use of the I-voting system. Certainly, additional protections could be added in order to mitigate specific attacks, but attempting to stop every credible mode of attack would add an unmanageable degree of complexity. Someday, if there are fundamental advances in computer security, the risk profile may be more favorable for Internet voting, but we do not believe that the I-voting system can be made safe today.

Acknowledgments

We thank everyone in Estonia who shared their insights on the e-voting system during our visit, including Sven Heiberg, Andres Hiie, Priit Kutser, Helger Lipmaa, Tarvi Martens, Arnis Parsovs, Jan Willemson, and many others. Of course, the views expressed here (and any errors) are ours alone. We also thank Brian Krebs, David Jefferson, and Barbara Simons for helpful advice. We are particularly grateful to Rop Gonggrijp and Zakir Durumeric for numerous suggestions and contributions.

We have not accepted any financial support from within Estonia, except for travel and accommodations for the international observers during the Oct. 2013 voting period, which were paid for by Tallinn City Council. The only requirement for that arrangement was that we observe the election.

This material is based upon work supported by the U.S. National Science Foundation under Grant No. CNS-1255153 and by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1256260. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

About the Authors

J. Alex Halderman is an assistant professor of computer science and engineering at the University of Michigan. His research focuses on computer security and privacy. A noted expert on electronic voting security, Prof. Halderman helped demonstrate the first voting machine virus, took part in California's landmark "top-to-bottom" e-voting security audit, and helped lead the first independent review of election technology in India, the world's largest democracy. In 2010, he led a team that hacked into Washington D.C.'s Internet voting system as part of a public security test. He is the instructor of Securing Digital Democracy, a massive online course about e-voting's risks and promise. He is also well known for developing the "cold boot" attack against disk encryption, which altered widespread security assumptions about the behavior of RAM, influenced computer forensics practice, and inspired the creation of a new subfield of theoretical cryptography. His work has won numerous distinctions, including three best paper awards and the Election Verification Network's John Gideon Memorial Award. He received his Ph.D. in computer science from Princeton University.

Harri Hursti is a Finnish independent security researcher. He is well known for taking part in a series of e-voting security studies in the U.S. and around the world. A memory card hack he demonstrated against Diebold voting machines is popularly known as "the Hursti Hack" and appeared in the HBO documentary, Hacking Democracy. Among other studies, he participated in the EVEREST report commissioned by the Ohio Secretary of State and a Princeton University study ordered by the New Jersey Superior Court. Hursti is a recipient of the EFF's Pioneer Award for his work on electronic voting. He has lived in the United States since 2009.

Jason Kitcat is the Leader of Brighton & Hove City Council, the UK's first Green-led principal authority. Kitcat sits on the Leading members group for the Key Cities association of 23 mid-size cities and also has a number of roles at the Local Government Association. He has been a Green city councillor in Brighton for six years. Before becoming Leader in 2012 he was Lead member for Finance. His professional background is in technology and online businesses such as Netmums and the Open Knowledge Foundation. Kitcat is a

prominent digital rights campaigner, particularly with regards to electronic voting and counting. He wrote GNU.FREE, the Free Software Foundation's e-voting software before concluding that Internet voting was too risky to be used for elections of any importance. Kitcat helped found the Open Rights Group and served as its e-voting coordinator. He led 25 election observers in monitoring e-voting and e-counting across the 2007 English & Scottish elections. The resulting report helped set the agenda for electoral modernization with e-voting no longer a government priority. He holds a BSc(Hons) from the University of Warwick in Computer Science & Management Science and MSc Technology & Innovation Management from the University of Sussex.

Margaret MacAlpine is an independent advisor on post-election audits in the United States. Her most notable advisory role was the California Post-Election Risk-Limiting Audit Pilot Program of 2011–2012. There she advised the office of the Secretary of State on the topic of the logistics and management of high-speed scanners in the processing and handling of ballots. MacAlpine is recognized as one of the foremost experts in the country on the scanning and handling of paper ballots for election auditing. Before the California pilot, she audited elections for several counties in Florida and Connecticut.

Drew Springall is a Ph.D. student at the University of Michigan, where his research focuses on security and privacy. He earned his bachelor's in computer science from the University of Alabama after serving in the U.S. Marine Corps. He is a recipient of the National Science Foundation's prestigious graduate research fellowship.

Travis Finkenauer is a Ph.D. student at the University of Michigan, where his research focuses on security and privacy. Before entering the Ph.D. program, Finkenauer completed his undergraduate degrees in computer science and mathematics at the University of Maryland.

8. REFERENCES

[1] ptrace(2): process trace. Linux Programmer's Manual.

[2] rsyslog: The rocket-fast system for log processing, April 2014. http://www.rsyslog.com/.

[3] Arne Ansper, Ahto Buldas, Mart Oruaas, Jaan Priisalu, Anto Veldre, Jan Willemson, and Kaur Virunurm. E-voting concept security: analysis and measures. Technical Report EH-02-01, Estonian National Electoral Committee, 2003.

[4] Andrew W. Appel. Security seals on voting machines: A case study. *ACM Trans. Inf. Syst. Secur.*, 14(2):18:1–18:29, September 2011.

[5] Jacob Applebaum, Judith Horchet, and Christian Stöcker. Shopping for spy gear: Catalog advertises NSA toolbox. Der Spiegel, December 2013. http://www.spiegel.de/international/world/catalog-reveals-nsa-has-back-doors-for-numerous-devices-a-940994.html.

[6] Josh Benaloh, Mike Byrne, Philip T. Kortum, Neal McBurnett, Olivier Pereira, Philip B. Stark, and Dan S. Wallach. STAR-Vote: A secure, transparent, auditable, and reliable voting system. *CoRR*, abs/1211.1904, 2012.

[7] Jennie Bretschneider, Sean Flaherty, Susannah Goodman, Mark Halvorson, Roger Johnston, Mark Lindeman, Ronald L. Rivest, Pam Smith, and Philip B. Stark. Risk-limiting post-election audits: Why and how, October 2012. http://www.stat.berkeley.edu/~stark/Preprints/RLAwhitepaper12.pdf.

[8] Joseph A. Calandrino, J. Alex Halderman, and Edward W. Felten. Machine-assisted election auditing. In *EVT'07: Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology*, pages 9–9, Berkeley, CA, USA, 2007. USENIX Association.

[9] D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *Security Privacy, IEEE*, 2(1):38–47, Jan 2004.

[10] D. Chaum, R.T. Carback, J. Clark, A. Essex, Stefan Popoveniuc, R.L. Rivest, P. Y A Ryan, E. Shen, A.T. Sherman, and P.L. Vora. Scantegrity II: End-to-end verifiability by voters of optical scan elections through confirmation codes. *Information Forensics and Security, IEEE Transactions on*, 4(4):611–627, December 2009.

[11] Cybernetica AS. Internet voting solution, 2013. Accessed: May 13, 2014, http://cyber.ee/uploads/2013/03/cyber_ivoting_NEW2_A4_web.pdf.

[12] Dancho Danchev. Study finds the average price for renting a botnet. ZDNet, May 2010. http://www.zdnet.com/blog/security/study-finds-the-average-price-for-renting-a-botnet/6528.

[13] Estonian Certification Authority. Avaleht. In Estonian. http://www.id.ee/.

[14] Estonian Certification Authority. Mis on ID-tarkvara? In Estonian. https://installer.id.ee/.

[15] Estonian Information System's Authority. Public key infrastructure PKI, May 2012. https://www.ria.ee/public-key-infrastructure/.

[16] Estonian Internet Voting Committee. Dokumendid, 2013. In Estonian. Accessed: March 2014, http://vvk.ee/valijale/e-haaletamine/e-dokumendid/.

[17] Estonian Internet Voting Committee. Ehk videos, 2013. In Estonian. Accessed: March 2014, https://www.youtube.com/channel/UCTv2y5BPOo-ZSVdTg0CDIbQ/videos.

[18] Estonian Internet Voting Committee. Using ID-card and mobil-ID, May 2014. https://www.valimised.ee/eng/kkk.

[19] Estonian Ministry of Foreign Affairs. Estonia today, 2012. http://www.euc.illinois.edu/estonia/documents/E-Estonia.pdf.

[20] Estonian National Electoral Committee. Kohaliku omavalitsuse volikogu valimised 2013. In Estonian. http://www.vvk.ee/kohalikud-valimised-2013/.

[21] Estonian National Electoral Committee. Vabariigi valimiskomisjon. In Estonian. Accessed: October 2013, http://www.vvk.ee/.

[22] Estonian National Electoral Committee. Elektroonilise hääletamise süsteemi üldkirjeldus, 2013. In Estonian. http://vvk.ee/public/dok/elektroonilise-haaletamise-systeemi-yldkirjeldus-EH-03-03-1_2013.pdf.

[23] Estonian National Electoral Committee. Valimised: Android Apps on Google Play, October 2013. In Estonian. Accessed: May 13, 2014, https://play.google.com/store/apps/details?id=ee.vvk.ivotingverification.

[24] Estonian National Electoral Committee. Statistics about Internet voting in Estonia, May 2014. http://www.vvk.ee/voting-methods-in-estonia/engindex/statistics.

[25] Estonian National Electoral Committee. Valimised on the App Store on iTunes, April 2014. In Estonian. Accessed: May 15, 2014, https://itunes.apple.com/ee/app/valimised/id871129256.

[26] Estonian National Electoral Committee. Valimised: Windows Phone'i rakenduste+mängude pood (Eesti), April 2014. In Estonian. Accessed: May 15, 2014, https://www.windowsphone.com/et-ee/store/app/valimised/11c10268-343f-461a-9c73-630940d8234b.

[27] Estonian National Electoral Committee, Estonian Internet Voting Committee, and Cybernetica AS. Android based vote verification application for Estonian i-voting system, September 2013. https://github.com/vvk-ehk/ivotingverification.

[28] Estonian National Electoral Committee, Estonian Internet Voting Committee, and Cybernetica AS. e-hääletamise tarkvara, September 2013. Accessed: March 2014, https://github.com/vvk-ehk/evalimine.

[29] Estonian Public Broadcasting. Center Party petitions European human rights court over e-voting, September 2013. Accessed: May 14, 2014, http://news.err.ee/v/politics/4ee0c8a2-b9c2-4d28-8ae4-061e7d9386a4.

[30] Jeremy Fleming. EU nations developing cyber 'capabilities' to infiltrate government, private targets. EurActiv, December 2013. http://www.euractiv.com/infosociety/eu-nations-lack-common-approach-news-532294.

[31] Andy Greenberg. Shopping for zero-days: A price list for hackers' secret software exploits. Forbes, March 2012. http://www.forbes.com/sites/andygreenberg/2012/03/23/shopping-for-zero-days-an-price-list-for-hackers-secret-software-exploits/.

[32] Sven Heiberg, Peeter Laud, and Jan Willemson. The application of i-voting for Estonian parliamentary elections of 2011. In *VOTE-ID*, pages 208–223, 2011.

[33] Greg Hoglund and James Butler. *Rootkits: Subverting the Windows Kernel*. Addison-Wesley, 2005.

[34] R.G. Johnston. The real deal on seals: Improving tamper detection. *Security Management*, 41:93–100, September 1997.

[35] R.G. Johnston. Some comments on choosing seals & on PSA label seals. In *Proceedings of the 7th Security Seals Symposium*, 2006. http://www.ne.anl.gov/capabilities/vat/pdfs/choosing-seals-and-using-PSA-seals-2006.pdf.

[36] R.G. Johnston. Insecurity of New Jersey's seal protocols for voting machines, October 2010. http://www.cs.princeton.edu/~appel/voting/Johnston-AnalysisOfNJSeals.pdf.

[37] R.G. Johnston and Anthony R.E. Garcia. Vulnerability assessment of security seals. *Journal of Security Administration*, 20:15–27, 1997.

[38] Douglas W. Jones and Barbara Simons. *Broken Ballots: Will Your Vote Count?* Stanford University Center for the Study of Language and Information, 2012.

[39] Erik Kain. Report: NSA intercepting laptops ordered online, installing spyware. Forbes, December 2013. Accessed: May 14, 2014, http://www.forbes.com/sites/erikkain/2013/12/29/report-nsa-intercepting-laptops-ordered-online-installing-spyware/.

[40] Jason Kitcat. Source availability and e-voting: An advocate recants. *Commun. ACM*, 47(10):65–67, October 2004.

[41] Benjamin Laxton, Kai Wang, and Stefan Savage. Reconsidering physical key secrecy: Teleduplication via optical decoding. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pages 469–478, New York, NY, USA, 2008. ACM.

[42] Mark Lindeman and Philip B. Stark. A gentle introduction to risk-limiting audits. *IEEE Security & Privacy*, 10(5):42–49, 2012.

[43] Helger Lipmaa. Paper-voted (and why I did so). Blog post, March 2011. http://helger.wordpress.com/2011/03/05/paper-voted-and-why-i-did-so/.

[44] Mandiant. APT1: Exposing one of China's cyber espionage units, February 2013. http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf.

[45] Nick Mediati. How to remotely install apps on your smartphone. TechHive, November 2013. http://www.techhive.com/article/2067005/how-to-remotely-install-apps-on-your-smartphone.html.

[46] Urmas Oja. Paavo Pihelgas: Elektroonilise hääletamise vaatlemine on lihtsalt võimatu, March 2011. In Estonian. http://forte.delfi.ee/news/digi/paavo-pihelgas-elektroonilise-haaletamise-vaatlemine-on-lihtsalt-voimatu.d?id=41933409.

[47] Francois Paget. Hacking summit names nations with cyberwarfare capabilities. McAfee Blog Central, October 2013. http://blogs.mcafee.com/mcafee-labs/hacking-summit-names-nations-with-cyberwarfare-capabilities.

[48] Arnis Parsovs. Practical issues with TLS client certificate authentication, February 2014. https://

www.internetsociety.org/sites/default/files/12_4_1.pdf.

[49] Teet Raidma and Janno Kase. Kohaliku omavalitsuse volikogu valimiste e-hääletamise protseduuride hindamise lõpparuanne, January 2014. In Estonian. http://vvk.ee/public/KOV13/lopparuanne_2013.ddoc.

[50] Peter Y. A. Ryan, David Bismark, James Heather, Steve Schneider, and Zhe Xia. Prêt à voter: A voter-verifiable voting system. *Trans. Info. For. Sec.*, 4(4):662–673, December 2009.

[51] David E. Sanger. Obama order sped up wave of cyberattacks against Iran. The New York Times, June 2012. http://www.nytimes.com/2012/06/01/world/middleeast/obama-ordered-wave-of-cyberattacks-against-iran.html.

[52] Barbara Simons. Report on the Estonian Internet voting system. Verified Voting Blog, September 2011. https://www.verifiedvoting.org/report-on-the-estonian-internet-voting-system-2/.

[53] Philip B. Stark. Super-simple simultaneous single-ballot risk-limiting audits. In *Proceedings of the 2010 International Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'10, pages 1–16, Berkeley, CA, USA, 2010. USENIX Association.

[54] Ken Thompson. Reflections on trusting trust. *Commun. ACM*, 27(8):761–763, August 1984.

[55] Ian Traynor. Russia accused of unleashing cyberwar to disable Estonia. The Guardian, May 2007. http://www.theguardian.com/world/2007/may/17/topstories3.russia.

[56] Ian Traynor. GCHQ: EU surveillance hearing is told of huge cyber-attack on Belgian firm. The Guardian, October 2013. http://www.theguardian.com/uk-news/2013/oct/03/gchq-eu-surveillance-cyber-attack-belgian.