TrustedVote
Secure E-voting on a Trusted Mobile Device

Diogo Monteiro

INESC-ID / Instituto Superior Técnico
University of Lisbon, Portugal
diogo.p.monteiro@tecnico.ulisboa.pt

Abstract. The wide availability of mobile devices, such as smartphones, enabled mobility in our lifestyles. However, traditional voting systems require physical presence of the voter at a specific place and time, which is incompatible with the concept of mobility. The goal of this paper is to propose an Internet voting system, called *TrustedVote*, that allows voters to cast their vote anywhere. Moreover, Internet voting significantly raises the turnout rate, reduces administrative costs and tallying time. In order to tackle malware and other insecurities in the client mobile platform, the solution is based on smartphones with a Trusted Execution Environment (TEE). *TrustedVote* leverages the isolation properties of TEEs available in Android and iOS smartphones to perform the cryptographic steps of an Internet voting system, such as vote encryption and voter authentication.

Keywords: Electronic Voting; Internet Voting; Security; Cryptography; Trusted Computing;

1 Introduction

Traditional, paper-based, voting systems require the voter to cast his vote during a certain period (the election period) and at a specific place, which is inconsistent with the concept of mobility widely seen today (e.g., with smartphones, tablets, smartwatches, etc.). This lack of mobility compromises the goal of democracy, as citizens may not be available to vote (on the designated places) during the election period.

The most recent example of this problem is related to the portuguese local elections. The Portuguese League of Professional Football scheduled four football matches to the election day, October 1, 2017. According to the portuguese government, they introduce significant traffic on the streets and reduce the citizens' availability to move to a voting place (as the traditional voting method requires). The government went further and there are plans to change the law in order to forbid certain sport events at election days.

Thus, the goal of this work is to design, implement, and evaluate an Internet voting system called *TrustedVote* that allows voters to vote anywhere with high level of security. Special focus is given to the need for a trusted software base on the mobile client. From this goal, the following requirements are considered:

1. **Accuracy** - it is not possible for an invalid vote to be counted in the final tally.
2. **Integrity** - a malicious attacker cannot, arbitrarily or in a deterministic way, modify a vote without detection at the client, communication channels or server.
3. **Democracy** - only authorized voters may cast a vote and an eligible voter may only cast one vote.
4. **Privacy** - no entity besides the voter learns how he cast his vote. He is not able to prove to a third party how he voted.
5. **Verifiability** - any independent entity is able to verify that all votes were counted correctly. Additionally, a voter can verify if his vote was recorded correctly.
6. **Robustness** - the protocol should consider the following robustness requirements:
   (a) **Availability** - the system should be available during the election period.
   (b) **Collusion Resistance** - the protocol should be resistant to collusion of corrupt voting authorities.
   (c) **Malware Resistance** - the insecure platform problem is defined as the insecurity that is found at the vote casting platforms because they are uncontrolled environments vulnerable to attacks. This problem should also be mitigated by tolerating malware in the client voting machine.
7. **Mobility** - the e-voting system should not impose mobility restrictions to the voter. The voter has the freedom to vote anywhere.
8. **Usability** - to successfully cast a vote, it is not required for the user to acquire special or dedicated devices that are not widely available. Moreover, from the point of view of the voter, the voting process imposed by the protocol should be intuitive and easily recognizable.

Thus, we consider the requirements that a non-electronic voting scheme should have, such as accuracy, integrity, democracy, privacy, verifiability, collusion resistance, availability and usability. In addition to non-electronic voting requirements, we consider mobility and malware resistance.

The design and implementation of an 100% secure Internet voting system is obviously difficult. Protocols that achieve the core security properties (accuracy, integrity, democracy, privacy and verifiability) and robustness [8, 10, 6, 13], have serious usability problems. On the other side, usable e-voting schemes that guarantee the core security properties protecting the vote at server-side [14, 15, 17, 15, 10, 6, 13], fail to provide malware resistance. They do not consider malware in the voting client machine. As a matter of fact, malware in the operating system is able to perform arbitrary operations on the vote before being encrypted, compromising the privacy and integrity of the vote without detection.

Thus, there is the need to tolerate malware in the voter's computer while maintaining usability. For that purpose, a Trusted Execution Environment - TEE (a special area of the main processor that executes in isolation from the rest of the hardware) is a fundamental aspect to consider as part of the solution. With a TEE, it is possible to leverage secure storage and the isolation feature to perform the cryptographic steps of an e-voting algorithm, guaranteeing privacy of the vote even to the operating system. The solution is to split the client of the voting protocol that executes in the voter's machines in two components: 1) a trusted component that executes sensitive operations in isolation from the operating system and, 2) an untrusted component that implements the steps of the voting protocol.

The remaining of this paper is organized as follows: Section 2 overviews the state of the art regarding electronic voting and TEEs. In Section 3, we present the architecture of *TrustedVote*. The implementation details are discussed in Section 4, and the evaluation is discussed in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

This Section outlines the related work and overviews the state of the art of e-voting and TEE technologies.

Electronic voting protocols use secure channels to perform network communications (e.g. SSL/TLS [4, 5]). They assume the existence of Certification Authorities that certificate asymmetric public keys of the protocol entities [3].

REVS [11] is an e-voting protocol proposed by Joaquim *et. al* in 2003 that uses blind signatures [7] to separate the authentication of users from the authentication of ballots, removing the link between the voter and his ballot.

REVS focuses on server-side fault tolerance, as the design allows for replication of all election servers. However, it is assumed that the machine used by the voter must be trusted and follow the protocol. If this assumption does not hold, then the integrity and privacy of the vote are compromised.

The online voting system proposed by Mohammadpourfard *et. al* [12] takes advantage of the Java Card 3 technology to remove trust from the client device. The Java Card technology facilitates the development of Java applications to smart cards. A smart card is a device with very limited amount of memory and processing power. However, the smart card is tamper resistant. It provides an environment where the processor instructions and contents of the memory cannot be eavesdropped or tampered. The smart card uses blind signatures [7], to authenticate the user and the ballot.

The system assumes that all citizens have access to a Java smart card reader, which affects usability. However, the Java smart card reader can be attached to a mobile phone, achieving mobility. EVIV [9] is an end-to-end verifiable internet voting system that uses homomorphic encryption [10] and takes into consideration that the client platform may be insecure and controlled by a malicious attacker. EVIV assumes that the voter has a component called Voter Security Token (VST), that is responsible for the encryption of the vote and authentication of the voter using digital signatures. In EVIV, the VST is implemented using a tamper-proof Java smart card containing the private key of the voter.

Prior to the election, the voter inserts the VST into his computer, in order to generate a code card to the upcoming election. The code card associates a random vote code (string) for each candidate. During the voting period, the voter uses the code card to insert the vote code associated with the candidate. The code card mechanism creates a secure channel between the voter and the VST.

EVIV ensures the security properties defined in Section 1, but fails to provide usability, as it is assumed that the voter has access to a smart card reader.

2.1 Trusted Execution Environments

There is a clear trade off between malware resistance and usability in e-voting protocols. Protocols that tolerate malware in the client platforms have low usability [8, 9, 12], and protocols with high usability do not tolerate malware [17, 16, 11, 6]. In the context of e-voting protocols, TEEs are able to provide malware resistance. They can protect against privacy and integrity attacks executed by malicious software in the operating system, without compromising usability.

A TEE is a dedicated area of the main processor that executes in isolation from the remaining of the hardware. It offers a secure environment for applications to execute. This section delineates the main concepts of ARM TrustZone [1], a TEE technology that is used in *TrustedVote*.

ARM TrustZone is a security extension architecture available in ARM processors that allows execution of code and services isolated from the operating system [1]. The hardware layer of ARM TrustZone provides two worlds of execution: 1) the normal world, where the rich operating system kernel runs, and 2) the secure world, where the secure operating system runs.

At a given time, the processor is only executing instructions in one world. Therefore, the hardware is equipped with the Secure Monitor Call (SMC) system call to switch between the two worlds of execution. When a TrustZone-enabled application running in the normal world wants to perform a sensitive operation in the secure world, it issues the SMC instruction that switches worlds.

Each execution mode has its independent memory space. Code executing in normal world can only access normal world memory space, while code running in the secure world can only access secure world memory space.

As ARM processors are widely available in Android and iOS phones today, the ARM TrustZone design allows the development of secure applications for mobile environments.

3 Architecture

This Section describes the design of *TrustedVote*. The *TrustedVote* protocol is an end-to-end verifiable Internet voting system that tolerates malware in the client computers, while keeping high levels of usability, i.e., without the need to use a device that is not widely available to the public.

To achieve that, *TrustedVote* combines the EVIV network and its cryptography protocol [9] with a new architecture of the client application that executes

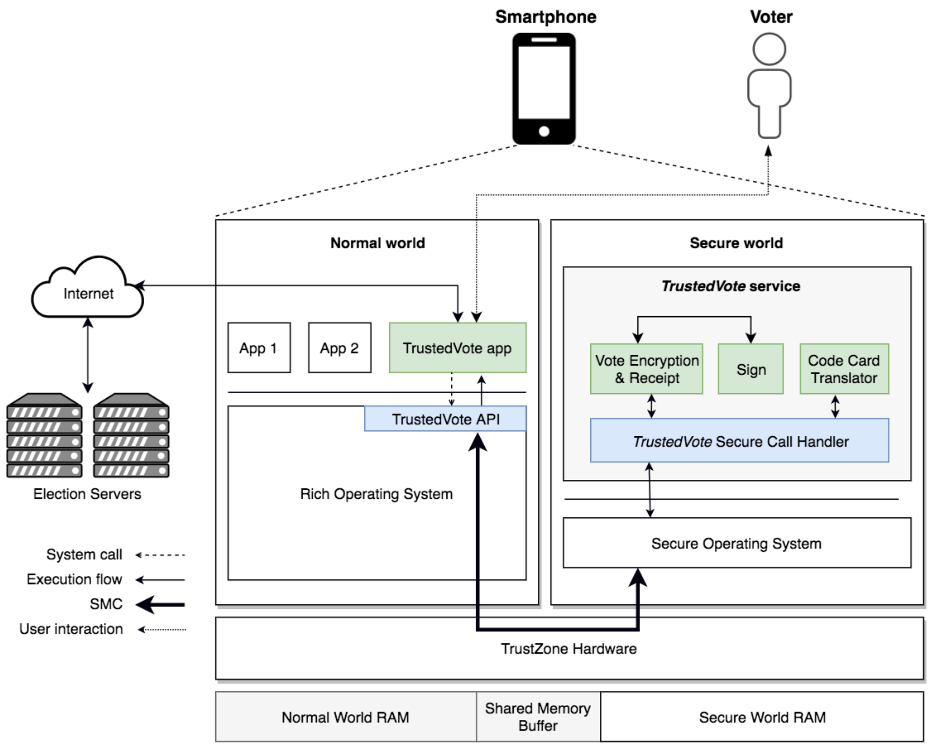

Fig. 1. *TrustedVote* client application architecture.

in the voter's mobile device. We remove the necessity of a separate tamper-proof smart card by using the isolation features provided by ARM TrustZone TEE (Section 2.1).

We assume that the adversary is able to control the operating system and execute arbitrary operations in the normal world of ARM TrustZone. However, we assume that the ARM TrustZone hardware behaves correctly, and we do not consider side channels or other physical attacks to the hardware.

3.1 Client architecture

The architecture of the *TrustedVote* client is illustrated in Figure 1. In this architecture, there are two distinct software stacks running over the ARM processor: the software running in the normal world and the software running in the secure world of ARM TrustZone. The component of the voting client that executes in the normal world is the *TrustedVote* app, while the *TrustedVote* service executes in the secure world. The *TrustedVote* service contains the sensitive components: Vote Encryption & Receipt, Code Card Translator and Sign.

The reason because Vote Encryption & Receipt, Code Card Translator and Sign must be executed in isolation in the secure world is that the goal is to protect the vote from unwanted manipulation or eavesdropping. These are the components that manipulate have access to the code card (know the candidate from a particular vote code), the vote and the voter's private key. Without isolation of these components, the vote could be leaked or modified.

During the process of voting, it is necessary that the vote created in the secure world by the Vote Encryption & Receipt component reaches the normal world so that it can be sent to the Election Servers. Therefore, in the architecture of *TrustedVote* there is a Shared Memory Buffer that is accessible by both worlds of execution. The *TrustedVote* Secure Call Handler and the *TrustedVote* API (represented in blue in Figure 1) are auxiliary components that manage the Shared Memory Buffer and perform reads/writes on it. The *TrustedVote* API provides the interface that the *TrustedVote* app must invoke in order to execute Vote Encryption & Receipt, Code Card Translator and Sign. The *TrustedVote* Secure Call Handler is a component that reads/writes data from the Shared Memory Buffer and contains the driver code for the *TrustedVote* secure calls that allow the creation of the vote inside the secure world.

The Secure Operating System is a microkernel that is significantly smaller than the Rich Operating System, as it only implements the required functionality to support the execution of *TrustedVote* service in the secure world, i.e., dynamic memory management, basic scheduling and abstractions to write on persistent memory.

4 Implementation

The prototype of the client was implemented in the *iMX53 Quick Start Board* from Freescale. Currently, the bootstrap code in the majority of the boards equipped with ARM processors switch to normal world. As this code is written in ROM (and executed when the processor starts), it is impossible to load an operating system in the secure world before the world switch happens. The *iMX53 Quick Start Board* is one of the few boards that does not contain this world switch written in its bootstrap code. Therefore, it is possible to load a microkernel that manages the secure world.

In order to build the microkernel, we chose Genode framework [2]. The Genode framework is a collection of building blocks, such as file systems, protocol stacks and libraries, that can be used to build a micro kernel that runs in the secure world.

We use the Genode *base-hw* microkernel (shipped with Genode, with only 21 KLOC) in secure world and Linux 2.6 in the normal world. This way, the *TrustedVote* service is implemented on top of the Genode *base-hw* microkernel. In the current implementation, the *TrustedVote* service depends on *OpenSSL* library to manipulate integers with arbitrary number of bits and *libc++* to reuse its implementation of vectors. We modified the Linux kernel to include the *TrustedVote* system call that executes the SMC instruction when a secure call to the *TrustedVote* service is issued.

Table 1. *TrustedVote* client execution time required to cast a valid vote in an election with 10 candidates.

	Secure World	**Normal World**
TrustedVote app	1 ms	1 ms
TrustedVote service secure calls	3544 ms	3068 ms
Total	3545 ms	3069 ms

The configuration of *base-hw* includes an application running on top of the microkernel that: 1) is able to manage a rich operating system running in the normal world and 2) contains a Secure Monitor implementation that handles world switches. This application is called TrustZone Virtual Machine Monitor (TZ VMM), because it is able to manage a rich operating system as a Virtual Machine Monitor (VMM) also does. We modified TZ VMM to include the functionality of the *TrustedVote* Secure Call Handler

TZ VMM does not solve the problem of sharing memory across worlds, i.e., does not implement the Shared Memory Buffer. It is possible to share data across worlds in the CPU registers. But, each CPU register can only store 4 bytes, which is not enough for transferring a *TrustedVote* ballot, for example. To solve this problem, *TrustedVote* system call allocates a DMA shared memory buffer (in normal world) and writes the address and size of the buffer in CPU registers. TZ VMM can access the CPU registers, and read the address and size of the DMA shared memory buffer. The *TrustedVote* service also depends on the *rapidjson* library for object serialization to the Shared Memory Buffer.

5 Evaluation

The evaluation of TrustedVote is focused on the client architecture, as the EVIV network protocol and its cryptography protocol satisfy accuracy, integrity, democracy, privacy, verifiability, availability and collusion resistance.

The *TrustedVote* service has a narrow attack surface. This reduces the amount of vulnerabilities that can be exploited in order to compromise the privacy and integrity of the vote. Moreover, the code size of *TrustedVote* service is very small. The current prototype implementation only has 48.3 KLOC (see Table 2). This also reduces the amount of vulnerabilities that could be exploited if, for instance, a full blown Linux 2.6 kernel was used in the secure world (12300 KLOC).

We also measure the secure world impact on the performance of the client. We setup two tests with an election with 10 candidates: one where *TrustedVote* service secure calls execute in normal world, and the other in secure world. The execution times correspond to the time required for a valid vote cast. However, the they do not include network communications and time required for a voter to insert commands.

Table 1 shows the results. The total client execution time can be divided in two main parts: the execution time of *TrustedVote* app (running in normal

Table 2. Code size of the current implementation

Component	**Code size**
base-hw micro kernel	20 KLOC
TrustedVote service	1.8 KLOC
OpenSSL BN	2.5 KLOC
OpenSSL RSA	5 KLOC
libc++ vector	7.8 KLOC
rapidjson	10.2 KLOC
Total	48.3 KLOC

world) and the execution time of *TrustedVote* service operations. In the latter, we include the world-switch time. We define the world-switch time as the sum of the time required to perform the context-switch (from user mode to kernel mode in normal world), to execute the *TrustedVote* system call and the *SMC* instruction.

The secure world execution finishes in 3545 ms, while normal world execution in 3069 ms. We notice an overhead of 474 ms (which represents an overhead of 16 %) in the case where the *TrustedVote* service executes in the secure world. However, the 476 ms of overhead when *TrustedVote* service executes in the secure world are not noticeable to the voter.

6 Conclusion

The goal of this work is to propose a fully mobile Internet voting system named *TrustedVote*. With this system, the voters can vote anywhere with an Internet connection and be certain that neither the vote nor the election is compromised. In order to achieve this goal, ARM TrustZone is used to provide an implementation of EVIV's VST functionality. This is possible because ARM TrustZone allows the isolated execution of applications in mobile devices. Unlike previous systems, *TrustedVote* keeps a high level of usability as the voter does not have to acquire special hardware devices to successfully cast his vote.

The security critical code of *TrustedVote* is very small, with only 48.3 KLOC under the current prototype implementation, which reduces the number of vulnerabilities that can be exploited.

Acknowledgements

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

References

1. ARM. ARM Security Technology - Building a Secure System using TrustZone Technology. ARM Technical White Paper. 2009.

http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf, accessed: 2016-11-28
2. Genode - Genode Operating System Framework. https://genode.org/, accessed: 2017-03-05
3. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. https://tools.ietf.org/html/rfc2459, accessed: 2017-06-28
4. The Secure Sockets Layer (SSL) Protocol Version 3.0. https://tools.ietf.org/html/rfc6101, accessed: 2017-06-28
5. The Transport Layer Security (TLS) Protocol Version 1.2. https://tools.ietf.org/html/rfc5246, accessed: 2017-06-28
6. Alrodhan, W., Alturbaq, A., Aldahlawi, S.: A mobile biometric-based e-voting scheme. 2014 World Symposium on Computer Applications and Research, WSCAR 2014 (2014)
7. Chaum, D.: Blind signatures for untraceable payments. In: Advances in cryptology. pp. 199–203. Springer (1983)
8. Grewal, G.S., Ryan, M.D., Chen, L., Clarkson, M.R.: Du-Vote: Remote Electronic Voting with Untrusted Computers. Proceedings of the Computer Security Foundations Workshop 2015-September, 155–169 (2015)
9. Joaquim, R., Ferreira, P., Ribeiro, C.: EVIV: An end-to-end verifiable Internet voting system. Computers & Security 32, 170–191 (2013)
10. Joaquim, R., Ribeiro, C.: An Efficient and Highly Sound Voter Verification Technique and Its Implementation, pp. 104–121. Springer Berlin Heidelberg (2012)
11. Joaquim, R., Zúquete, A., Ferreira, P.: REVS – A Robust Electronic Voting System. IADIS International Journal of WWW/Internet 1(i), 47–63 (2003)
12. Mohammadpourfard, M., Doostari, M.A., Ghaznavi Ghoushchi, M.B., Shakiba, N.: A new secure Internet voting protocol using Java Card 3 technology and Java information flow concept. Security and Communication Networks 8(2), 261–283 (2015)
13. Petcu, D., Stoichescu, D.A.: A hybrid mobile biometric-based e-voting system. In: 2015 9th International Symposium on Advanced Topics in Electrical Engineering (ATEE). pp. 37–42 (May 2015)
14. Ryan, P.Y.A., Bismark, D., Heather, J., Schneider, S., Xia, Z.: Prêt À Voter: a Voter-Verifiable Voting System. IEEE Transactions on Information Forensics and Security 4(4), 662–673 (Dec 2009)
15. Ryan, P.Y.: A variant of the Chaum voter-verifiable scheme. In: Proceedings of the 2005 Workshop on Issues in the Theory of Security. pp. 81–88. ACM (2005)
16. Ryan, P.Y.: Prêt À Voter with Paillier encryption. Mathematical and Computer Modelling 48(9), 1646–1662 (2008)
17. Ryan, P.Y., Schneider, S.A.: Prêt À Voter with re-encryption mixes. In: European Symposium on Research in Computer Security. pp. 313–326. Springer (2006)