

Mobile implementation and formal verification of an e-voting system

Stefano Campanelli, Alessandro Falleni, Fabio Martinelli, Marinella Petrocchi, Anna Vaccarelli
IIT-CNR, Via G. Moruzzi 1, 56124 Pisa, Italy

Abstract

We propose a mobile implementation of an e-voting protocol. We also provide a formal analysis to validate a security property of our system.

1. Introduction

Technological revolutions in computer and communication are enabling the deployment of mobile communication, based on hand-held computing devices and wireless networking. Connection capabilities are manifold, and performances of new generation machinery become better and better in terms of computing power and memory size. Their software is able to offer elaborate and complex services, and mobile systems may be exploited for novel applications spread out in a variety of directions.

In this paper, we consider e-voting protocols and their implementation, see, *e.g.* [5, 7, 11, 14, 16]. They offer an interesting field of research, given the challenges arising in order to make them popular to large communities for, *e.g.* national elections, that are rapidly diffusing. As an example, the possibility to electronically vote has been offered to Estonian citizens in the 2005 national elections. Given the criticality of the application, proofs of the overall system security should be provided, together with the capability of a friendly management and usability.

Surveys on the goals of e-voting protocols have been proposed in the past, see, *e.g.* [11]. Here, we briefly cite some of them. First, any e-voting system should assure voter privacy while avoiding opportunities for fraud. Also, possible errors in the final tally should be detected (*soundness*), nobody must vote twice (*unreusability*), and valid votes must not be removed from the final tally. On the other hand, invalid votes must not be added (*completeness*). For preventing vote buying and extortion, no voter can prove to have voted in a particular way, *uncoercibility*, [3, 14].

Here, we focus on the so called *mobility* property [7]. We show how a complex system like one implementing an architecture for polling and voting can be made available for

mobile devices, enlarging the number of potential voters. Indeed, a mobile e-voting system will allow the maximum freedom with respect to the location from which a user can cast its vote.

In particular, we show the portability of an e-voting protocol, based on the Sensus protocol, [7], and already defined for polling from a fixed location [2], on mobile devices. We call our mobile voting system M-SEAS, *i.e.* the Mobile Secure E-voting Applet System. We will give details of our mobile implementation. Also, we perform a formal verification of the protocol.

Indeed, the architecture of M-SEAS has been thought to fix a well-known vulnerability of Sensus, that basically allows one of the entities involved in the election process to cast votes of eligible users that, although registered, abstain to vote. These illegitimate votes would fall into the final tally. In order to prove the correctness of M-SEAS with respect to this vulnerability, we will formally analyze it. Indeed, formal methods have been extensively used for the specification and analysis of cryptographic protocols. In particular, in the area of e-voting protocols, recent research has proved their effectiveness, and some security properties have been analyzed, [8, 15]. Here, we will exploit our analysis approach, based on the use on a CCS-like process algebra to model the system, and a model checker for the analysis, based on partial model checking techniques, [1].

The paper is structured as follows. After this introduction, we describe the M-SEAS protocol in Section 2. In Section 3 we then present our prototype implementation, and we give hints of its performances, after which we formally analyse a security property of the protocol in Section 4. Finally, Section 5 contains our conclusions.

2. Protocol Description

Before presenting M-SEAS, we briefly explain some notation. We assume the reader to be familiar with the basics of cryptographic primitives.

Sending and reception of a message msg from A to B is

$$\begin{aligned}
0a) \quad P \longrightarrow T & : \{(h(pk_P^1, ID^1))_{blind}, ID^2\}_{pk_P^2-1} \\
0b) \quad T \longrightarrow P & : \{(h(pk_P^1, ID^1))_{blind}\}_{pk_T^{-1}} \\
0c) \quad P \longrightarrow T & : \{h(pk_P^1, ID^1)\}_{pk_T^{-1}}, pk_P^1, ID^1
\end{aligned}$$

Figure 1. M-SEAS registration phase

represented as:

$$i \quad A \longrightarrow B : msg$$

where i is the i -th communication channel, on which the exchange takes place. Notation throughout the paper is as follows:

$$\begin{aligned}
\{\dots\}_{pk_i} & := \text{message encrypted} \\
& \quad \text{by public key of party } i \\
\{\dots\}_{ek} & := \text{message encrypted} \\
& \quad \text{by encryption key } ek \\
(\dots)_{blind} & := \text{blinded message} \\
h(m) & := \text{digest of message } m
\end{aligned}$$

Along with standard cryptographic mechanisms, e-voting protocols make often use of blind signatures, introduced in [4], that, basically, makes it possible to sign a message without being aware of its content. Upon applying a blind signature, making message m not understandable, party i can apply a signature, obtaining $\{(m)_{blind}\}_{pk_i^{-1}}$. Then, who has originally applied the blinding can also remove it (notation \leftarrow), though maintaining the signature. The following *qualitative* equation holds: $\{m\}_{pk_i^{-1}} \leftarrow \{(m)_{blind}\}_{pk_i^{-1}}$.

There are three entities involved in the protocol. The pollster P , representing the set of hardware/software modules through which a voter can cast its ballot; the validator V , a server that first checks the eligibility of the pollster P and the uniqueness of its submission, and then it validates the submitted vote; the tallier T , a server that counts all the validated votes. The overall architecture is shown in Fig. 3(a).

M-SEAS consists of two phases, the first is the registration, described in Fig. 1:

1. P registers a first public key pk_P^1 and an associated identifier ID^1 with T . To do this, P blinds the digest of the pair pk_P^1, ID^1 , it adds a second identifier ID^2 , it signs everything and it sends the result to T .
2. The tallier knows pk_P^2 and ID^2 , they are public values contained in the list $l2$ of eligible voters. It verifies the signature by using pk_P^2 . Then, it signs $(h(pk_P^1, ID^1))_{blind}$ and sends it to P . To trace the electors whose blind pairs have been signed, T updates $l2$.

$$\begin{aligned}
1) \quad P \longrightarrow V & : \{(h(\{B\}_{ek}))_{blind}\}_{pk_P^2-1}, ID^2 \\
2) \quad V \longrightarrow P & : \{(h(\{B\}_{ek}))_{blind}\}_{pk_V^{-1}} \\
3) \quad P \longrightarrow T & : \{h(\{B\}_{ek})\}_{pk_V^{-1}}, \{B\}_{ek}, ID^1 \\
4) \quad T \longrightarrow P & : rec\#, \{B\}_{ek}\}_{pk_T^{-1}} \\
5) \quad P \longrightarrow T & : rec\#, dk
\end{aligned}$$

Figure 2. Second phase of M-SEAS

3. The pollster removes the blinding, obtaining a message digest digitally signed by the tallier, and sends it back to T , along with the pair as a plaintext. The tallier verifies the validity of its own signature. If the verification succeeds, then T records the pair (pk_P^1, ID^1) in the special list $l1$, a list designed to contain pairs (pk_P^1, ID^1) .

The introduction of the registration phase is necessary to prevent illegitimate ballots from being included in the final tally, as it will be clear in the following.

The protocol continues as specified in Fig. 2. Messages 1 and 2 are devoted to validate ballot B , *i.e.* the validator checks the credentials of the aspirant pollster. It first verifies the signature sent in message 1, by applying the public key paired with pk_P^2 . After verifying the signature, it checks if i) ID^2 belongs to the list $l2$ of eligible and registered voters; ii) it is the first time that the voter identified by ID^2 sends a vote. If all these checks succeed, V applies its digital signature to the digest and it returns the signature to P . The pollster sends its vote to the tallier (message 3). In particular, the digest of the encrypted ballot is signed by the validator and by the pollster too, through its second private key, pk_P^1 . To complete message 3) the pollster adds its second identifier ID^1 . Upon the reception of message 3), the tallier must verify that ID^1 belongs to $l1$ and that it is the first vote issued by the voter identified by ID^1 . It verifies the outer signature through the public key associated to ID^1 $l1$. Note that this signature does not allow the tallier to know the identity of the voter, since ID^1 was first registered through blind signatures (0a, 0b, 0c). Then, it verifies the validator signature. In message 4, T verifies V 's signature, computes an own digest on $\{B\}_{ek}$ and verifies its equality with what received. Then, it inserts $\{B\}_{ek}$ into $l2$. Finally, T signs the encrypted ballot and sends it back to P with a receipt number, to pair the ballot decryption key with the ballot itself. T also updates $l2$ by inserting the receipt number. In message 5, P verifies the T 's signature of message 4, and it sends the ballot decryption key dk to the tallier. The tallier uses the key to decrypt the ballot, adds the vote to the final tally and it pairs the decryption key with the correspondent entry in $l2$. At the end of the voting session, the

tallier publishes l_2 and the final tally.

All the communications are assumed to be encrypted with the public key of the receiver. Consequently, all the receivers have at first to retrieve the plaintexts by applying their own private keys to the received messages. In fact, dk does not travel as a clear text. Step 0c, 3, and 5 assume the use of an anonymous communication channel. This is required to avoid that the tallier establishes a link between the pollster and its sensitive information by a simple traffic analysis.

M-SEAS message exchange is partly based on Sensus. However, it does not fall in the well-known vulnerability of Sensus, that allowed the validator to cast its own ballot B_V , in place of who abstained. In fact, this ballot will be signed with a private key, say $pk^{x^{-1}}$. T will accept it if the correspondent public key, say pk^x , has been previously registered. The identity of who supplied that public key can be verified through the tallier signature, see step 0b. Hence, the only way for the validator to have its vote accepted is to be a registered user. If so, it cannot cast its vote twice.

3. Implementation

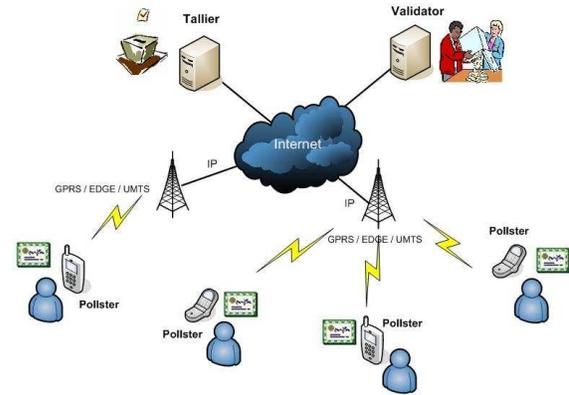
To build up a portable system not dependent on the underlying platform, we choose Java as the implementation language. Further, given the huge diffusion of mobile devices supporting the Java language (Java Micro Edition [12]), we realize the pollster module, *i.e.* the module running at the elector's side, as a Java MIDlet. The MIDlet can be executed over whatever platform supporting a Java Virtual Machine.

The MIDlet is composed by two main modules. The first module lets the user cast its vote. It implements all the communications with T and V , together with the cryptographic operations, on behalf of the user. For the cryptographic operations we use "SignIT Mobile", a JME cryptographic library developed by our team, which gave us the possibility to use advanced mechanisms like blind signature.

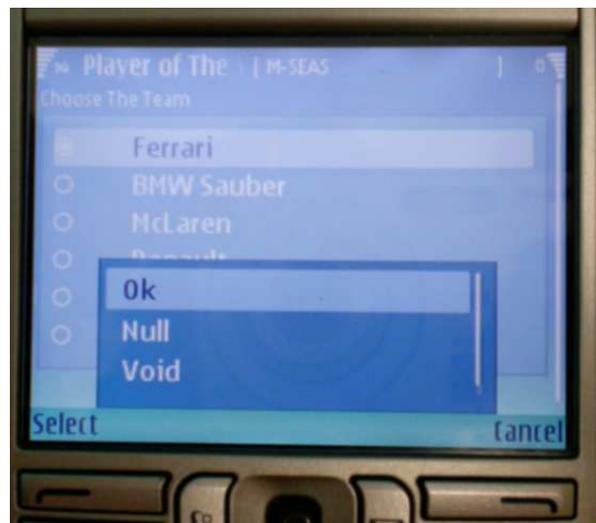
The second module is the friendly user interface which also contains an XML parser, to interpret the ballot, whose structure is indeed defined by a XML document. Parsing an XML file needs the Java Web Services API (JSR 172) which provides some basic parsing features.

Exploiting XML for the ballot structure makes the overall system very flexible. Indeed, elections with multiple and nested choices may be easily implemented. If one consider a survey on, *e.g.* favorite athletes, the system could first present a first level of choice, asking for the selection of the favorite sports activity. Then, the system could reach deeper levels of details. For example, the voter may select the particular team, Fig. 3(b) and finally the favorite athlete.

Also, we developed another MIDlet to transfer in a secure way a digital certificate in PKCS#12 format from a pc



(a)



(b)

Figure 3. (a) Architecture of M-SEAS - (b) Running MIDlet at pollster's side

to a mobile device using Bluetooth channels. Credentials of PKCS#12 are stored into the device, as a digital certificate plus the associated private key. They are encrypted by following guidelines of PKCS#8. PKCS#i are “de facto” standards, issued by RSA Labs and made public and modifiable, for the deployment of public key cryptography.

3.1. Performances

The Java MIDlet representing the pollster module has been tested over a NOKIA E61, firmware version n.3.0633.09.04. The application results to be stable w.r.t. all its functionalities.

In particular, we consider, as an important element for the prototype evaluation, the application usability. Having in mind mobile solutions, it is important not only to have a practical, friendly, user interface, but also to complete the whole process, *i.e.* registering and casting a vote, as soon as possible, say within few seconds. The most critical phase during execution takes place after the client identification. In fact, the MIDlet must generate two pairs of asymmetric keys, pk_P^1, pk_P^{1-1} and pk_P^2, pk_P^{2-1} . This takes about 20 seconds. In our first experimentations, this operation brought to an extraneousness of the user, due to long waits. To some extent, this has been solved by lay this calculation phase on the interactive phase during which the elector gives its preferences, by assigning a different thread to each operation.

Parsing the XML document and visualizing the ballot takes no appreciable time.

4. Formal Verification

This section presents the formal verification of the M-SEAS protocol, with respect to a validator trying to insert its own votes replacing the votes of who abstained.

For our analysis, we adopt the approach of [18]. This is based on the observation that a security protocol under analysis can be described as an open system: a system in which some component has an unspecified behaviour (not fixed in advance). Subsequently one assumes that, regardless of the unspecified behaviour, the system works properly (*i.e.* satisfies a certain property). In case of an e-voting protocol, one can imagine the presence of a hostile adversary trying to interfere with the normal execution of it, in order to achieve some kind of advantage. The adversary could be also one among the honest participants, that decide to maliciously act by not following the protocol guidelines. Such an adversary is added to the specification of the e-voting protocol, as a component with a behaviour that is defined only implicitly by the semantics of the specification language.

We assume the adversary to act in Dolev-Yao fashion [9] by using a set of message manipulating rules that model

cryptographic functions like encryption and decryption. Encryption is opaque, *i.e.* a message encrypted with the public key of one of the participants cannot be decrypted by anyone but the person who knows the corresponding private key (unless the decryption key is compromised of course). As is common in this branch of computer security, we adopt a black-box view of cryptography by assuming all cryptographic primitives involved in the network protocol to be perfect. Like the honest participants, the adversary is able to send and receive messages to other participants. However, it can also intercept and forge messages and, to a certain degree, derive new messages from the set of messages that it has come to know. This set consists of all messages the adversary knows from the beginning (its initial knowledge) united with the messages it can derive from the ones intercepted during a run of the protocol. To analyze whether a system works properly, at a certain point in the run the adversary’s knowledge is checked against a security property. If the adversary has come to know information it was not supposed to know, then the analysis has thus revealed an attack w.r.t. that particular property, *i.e.* a sequence of actions performed by the adversary that invalidates the property.

We use model checking to perform the analysis. Model checking is an automatic technique to verify whether or not a system design satisfies its specifications and certain desired properties [6].

The specification language that we use is Crypto-CCS, [10], a CCS-like process algebra with cryptographic primitives, and the analysis is performed with the model checker PaMoChSA v1.0 [13]. PaMoChSA requires the following input:

- a file with the protocol specification in Crypto-CCS;
- a logic formula expressing the property to be verified;
- the adversary’s initial knowledge.

We considered an adversary X and set its initial knowledge to the set of public messages that it knows at the start of the protocol, *i.e.* the public keys of the pollster P , its own public and private key denoted by pk_X and pk_X^{-1} , its ballot B_X , its encryption key ek_X , and cryptographic material to perform hash functions. Note that, in order to check if the validator is able to insert its own votes in the final tally, the pair of public/private keys known by the adversary may coincide with pk_V/pk_V^{-1} , as well as B may coincide with B_V , and ek_X with ek_V . Indeed, the not specified component in the considered open system is V .

The input and result of the analysis we performed are as follows:

- Specification file: M-SEAS.exp
- Logic formula: *special*

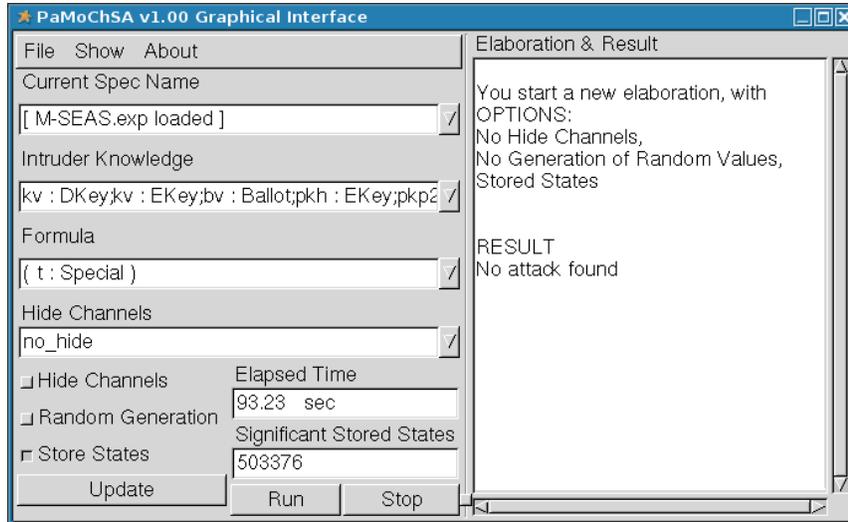


Figure 4. Screenshot of PaMoChSA's interface.

- Initial knowledge: $\{pk_X, pk_X^{-1}, pk_P^1, pk_P^2, B_X, ek_X, public\ hash\}$
- Result: **No attack found**

Figure 4 shows the graphical interface of the tool, with the loaded experiment and the result.

To verify whether or not V is able to insert its own votes in place of who abstained, we add in the Crypto-CCS specification `M-SEAS.exp` a special action: `send(public, special)`. This represents sending the message *special* on a public channel. Given the adversary model discussed at the beginning of this section, once this action is performed, the adversary automatically has in its knowledge message *special*. This special action is inserted in the specification immediately after that the tallier received a voted ballot casted by the adversary. Given sequentiality of actions in a specification, if *special* falls in the adversary's knowledge, this means also that the adversary was been able to cast its vote.

To verify the logic formula specified above, the tool set out to find a run of the protocol with the following characteristic: at the end of the run, the adversary knows message *special*. Such a run does not exist. Hence M-SEAS is correct w.r.t. the analyzed security property, *i.e.* it does not fall to an insertion of a not legitimate vote in the final tally.

To complete the analysis, also the Sensus protocol has been specified into Crypto-CCS, and the specification has been given as input to the tool. As expected, an attack has been found.

The insertion of special actions into the formal specification of a protocol does not affect the modeling and analysis. Special actions are just inserted for analysis purposes, and

it is a common practice in security analysis, see, *e.g.* , [17], for what concerns authentication properties.

5. Concluding Remarks

In this paper, we presented M-SEAS, a mobile e-voting protocol that enables people to cast their votes from the majority of mobile devices, obviously including cellular phones, at the only, basic condition that Java technology is enabled.

The implementation of M-SEAS exploits mechanisms for mobile communication security. This contributes to the development of mobile applications exploitable in everyday life. Actually, our prototype system allows users to easily participate into opinion polls and statistical surveys, having the process ultimated within a handful of seconds. The prototype has been developed having in mind surveys/polls for relatively small communities, whose members may range up to dozens of thousands. However, a successful outcome in these specific areas could increase the user confidence in such technologies, leading to a major trust towards the application of e-voting systems, also to large scale elections.

Our implementation strictly fulfils one of the most important goal of an e-voting protocol, *i.e.* *mobility*. Indeed, such systems should reduce as much as possible limitations about the location from which the elector can cast its vote.

Finally, formal methods and tools are popular means for the analysis of security aspects of computer protocols. Here, we have exploited our analysis approach to strengthen our confidence in the M-SEAS security architecture, by having analyzed the protocol and tested the absence of the Sensus vulnerability.

Acknowledgment

This research is partly supported by the EU project SEN-SORIA (IST-2005-016004).

References

- [1] H. R. Andersen. Partial model checking. In *LICS: IEEE Symposium on Logic in Computer Science*, 1995.
- [2] Baiardi et al. Seas, a secure e-voting protocol. *Computers & Security*, 24(8):642–652, 2005.
- [3] J. Benaloh and D. Tuinstra. Receipt-Free Secret-Ballot Election. In *Proc. of ACM STOC'94*, pages 544–553, 1994.
- [4] D. Chaum. Blind Signatures for Untraceable Payments. In *Proc. of Crypto'82*, pages 199–203. Plenum, NY, 1983.
- [5] D. Chaum, P. Y. A. Ryan, and S. A. Schneider. A practical voter-verifiable election scheme. In *ESORICS*, pages 118–139, 2005.
- [6] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.
- [7] L. Cranor and R. K. Cytron. Sensus: A Security-Conscious Electronic Polling System for the Internet. In *Proc. of HICSS'97*, 1997.
- [8] S. Delaune, S. Kremer, and M. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *CSFW*, pages 28–42, 2006.
- [9] D. Dolev and A. C.-C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.
- [10] R. Focardi and F. Martinelli. A uniform approach for the definition of security properties. In *Proc. FM'99*, volume 1708 of *LNCS*, pages 794–813. Springer, 1999.
- [11] A. Fujioka, T. Okamoto, and K. Ohta. A Practical Secret Voting Scheme for Large Scale Election. In *Proc. of Auscrypt'92*, volume LNCS 718, pages 244–260, 1992.
- [12] <http://java.sun.com/javame/index.jsp>. SUN - Java Micro Edition.
- [13] <http://www.iit.cnr.it/staff/fabio.martinelli/pamochsa.htm>. Partial Model Checking Security Analyzer PaMoChSA v1.0.
- [14] A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *WPES*, pages 61–70, 2005.
- [15] S. Kremer and M. Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *ESOP*, pages 186–200, 2005.
- [16] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *ICISC*, pages 245–258, 2003.
- [17] G. Lowe. A hierarchy of authentication specification. In *CSFW*, pages 31–44, 1997.
- [18] F. Martinelli. Analysis of security protocols as open systems. *TCS*, 290(1):1057–1106, 2003.