

How to fit cryptographic e-voting into smart cards

Sébastien Canard*

France Telecom, Research and Development
42 rue des Coutures, BP 6243
14066 Caen Cedex 4, France
sebastien.canard@orange-ft.com

Hervé Sibert

France Telecom, Research and Development
42 rue des Coutures, BP 6243
14066 Caen Cedex 4, France
herve.sibert@orange-ft.com

Abstract. The complexity of voting procedures make it challenging to design a secure electronic voting system. In many proposals, the security of the system relies mainly on a black box voting machine. Meanwhile, the most advanced proposals base their security arguments on (complicated) cryptographic protocols, e.g. blind signatures or homomorphic schemes.

Canard and Traoré proposed cryptographic primitives dedicated to provide anonymous services using smart cards. Among these primitives, list signatures are specially suitable for e-voting, as they provide specific properties such as multiple vote detection. Moreover, unlike blind signatures, they do not involve a signing authority during the ballot creation process.

The purpose of this paper is to present an e-voting system, whose security relies on a tamper-resistant smart card embedding several cryptographic primitives, including list signatures.

Keywords: Group signatures, e-voting, smart cards

Address for correspondence: France Telecom, Research and Development - 42 rue des Coutures, BP 6243 - 14066 Caen Cedex 4, France

*The authors want to thank Benoît Calmels, Mohamed El Marbough, Vanessa Gondonneau, Richard Musil and Jacques Traoré for their help and anonymous reviewer for their comments.

1. Introduction

An e-voting scheme is a set of protocols allowing voters to securely vote by interacting with a set of authorities who collect the votes and calculate the result of the election. We usually distinguish between two types of e-voting: on-line, e.g. via Internet, and off-line, by using a voting machine or an electronic polling booth.

The main goal of a secure e-voting system is to ensure the privacy of the voters and the accuracy of votes. A secure system shall fulfil (at least) the following requirements [6]:

- **Eligibility:** only votes of legitimate voters shall be taken into account.
- **Unreusability:** each voter shall only be able to cast one vote.
- **Anonymity:** all votes shall be secret.
- **Accuracy:** cast ballot cannot be altered. Moreover, it must not be possible to delete ballots nor to add ballots once the election has been closed.
- **Fairness:** partial tabulation before the end of the election must be impossible.
- **Vote and go:** once a voter has cast his vote, there is no further action he needs to take.
- **Public verifiability:** anyone should be able to readily check the validity of the whole voting process.

E-voting systems translate the traditional vote to a digital context. Several experimentations have already taken place, based either on black box machines or on cryptographic frameworks. The purpose of these systems is to obtain the results immediately after the end of the poll, while (at least) preserving the security of the traditional vote. Cryptography-based frameworks are designed to enhance security while providing some functionalities that remain mainly theoretical in traditional voting because of practical issues.

In this paper, we propose a smart card-based e-voting scheme, designed to ensure the main properties that one can expect from such a scheme. This scheme is designed in a flexible way, which means some parts of it can be slightly modified, or some components may be added, in order to have it adapted to the election laws of most countries.

Our point is to propose a system adapted to the constraints of the smart card environment. Thus, we use only cryptographic primitives that already exist in smart cards, so as to implement a prototype using real smart cards. This prototype is now fully functional.

We first provide an overview of our system, and the cryptographic tools it relies on. Next, we describe the setup of the system, both inside and outside the card. We finally describe the interactions that take place on an election day.

2. Overview of the system

In this section, we first give a brief description of our solution. We introduce the cryptographic tools involved in our system and their smart card implementation.

2.1. The Actors and their roles

Our system is designed for off-line voting. Every voter owns a voting smart card that is used twice: first, in a polling booth, and second, in front of a ballot box, in order to remain close to traditional vote. Our system involves several actors:

- Several **Registration Centers** \mathcal{RC} where citizens register to become voters, after some checks by authorities. These are managed by the **Central Registration Center** \mathcal{CRC} .
- A **Smart Card Creation Center** \mathcal{SCCC} where smart cards are personalized for voters.
- A **Certification Authority** \mathcal{CA} that controls the certification of public signature keys for every voter. Each voter will make an attendance using a digital signature¹ and needs, as a consequence, a certificate.
- Several **Controllers** \mathcal{C} who form a set of trusted entities in charge, for a given voting room, of the election. They generate all required data for the convenient execution of the protocols. Each voting room is designated by an identifier Id_{vr} .
- Several **Recover Authorities** \mathcal{RA}_i that will be called by the Controllers in order to provide a voter that has lost his smart card with a new one.
- Several **Key Authorities** \mathcal{KA}_i in charge, for a given Registration Center \mathcal{RC} , of the generation of a shared private signature key that is used for anonymity purpose (see the used list signature scheme).
- Every **Voter** \mathcal{V} who owns a voter smart card and is registered in a particular voting room. This smart card may authenticate its owner through a PIN code or biometrics. In the following, we consider the PIN code case, which also requires a visual authentication of the voter. Each voter also owns a certificate $Cert$ issued by \mathcal{CA} .
- Several **Tellers** \mathcal{T} who form a set of entities involved in the counting of the votes. They own a pair of keys $epk_{\mathcal{T}}/esk_{\mathcal{T}}$ for an encryption algorithm. In fact, each of them has a private key $esk_{\mathcal{T}_i}$ and the global public key is computed using all these ones (see section 2.2).

Each election is denoted by an identifier Id_{elec} , which may, for practical purpose, be diversified voting room-wise and thus contain the voting room identifier Id_{vr} . There are two major steps in our e-voting system: the system setup, with a sub-setup for every new election, and the running of an election.

When a voter wants to vote at election Id_{elec} , he enters a polling booth that contains a voting machine. This machine enables the voter to create his ballot inside his smart card. Outside the polling booth, the voter casts his ballot on the ballot box machine and makes his attendance, using the smart card again. Figure 1 presents our global architecture.

2.2. Cryptographic tools

Here is a description of the main cryptographic components encountered in our voting system.

¹Our solution is also suitable for a handwritten signature, since some electoral laws do not yet accept digital signatures.

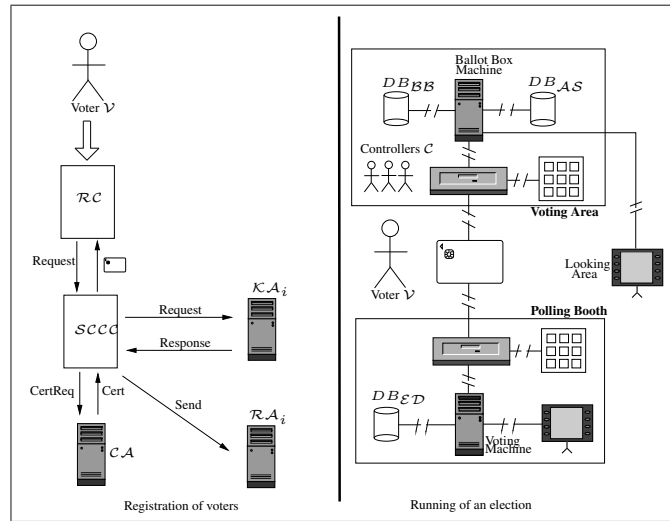


Figure 1. Global Architecture

2.2.1. Signature scheme

Our system includes a classical signature scheme to produce attendances. For this purpose, every voter is provided with a PKI key pair. Every PKI-compatible signature scheme can be used in our system, and as the signature is created inside the card, it is better to use either lightweight signatures such as Schnorr signatures or signatures derived from the GPS scheme in case of a constrained environment, or an RSA signature if the card has a dedicated cryptoprocessor.

2.2.2. Encryption scheme

Our system requires a probabilistic encryption scheme. It is used by each voter to encrypt his ballot, which is decrypted during the counting phase. Instead of using a classical RSA scheme, a discrete logarithm-based threshold encryption scheme such as El Gamal both remains simple and ensures fairness of the vote, as it prevents a single teller to count the votes alone. In this case, there is a unique encryption key, while each teller owns one decryption key, and decryption necessarily involves all tellers. Using the El Gamal encryption scheme for example, we denote by $esk_{\mathcal{T}_i}$ the private key of the teller \mathcal{T}_i . The corresponding public key is consequently $epk_{\mathcal{T}_i} = g^{esk_{\mathcal{T}_i}}$, where g is a generator of the group where all the computations are done. The global encryption key that is used by the smart cards is then $epk_{\mathcal{T}} = \prod epk_{\mathcal{T}_i}$. Another possibility is to use a mix-net [1], which implies more computations for the smart card, but provides the voters with extra anonymity, in case the voting machines would be open to intrusions.

2.2.3. Anonymous signature scheme

An anonymous signature scheme is a mechanism that enables a user to authenticate himself to another without revealing his complete identity: he only proves that he owns some right. The best-known anonymous signature are blind signatures [4] and group signatures. The concept of group signature was intro-

duced in [5]. It enables a member of a group to anonymously sign a document on behalf of the group. Whenever needed, a designated group manager can reveal the identity of the signer.

In our system, we use a variant of group signatures, called list signatures [2]. A list signature scheme does not permit anonymity revocation, but everyone is able to link two signatures produced by the same group member during a particular time period.

More precisely, we use the list signature scheme introduced at Cardis 2004 [3]. This scheme is very efficient since it can be built upon a classical signature scheme and a pseudo-random number generator (*prng*). The *prng* is designed by using e.g. a secret key encryption scheme such as AES. In [3], every smart card owns the same private signature key ssk_{vr} and has a proper secret key $k_{\mathcal{V}}$ that is used in the *prng* (see section 2.3 for details). Thus, these smart cards shall be tamper resistant.

In our system, a smart card produces a list signature S of the choice v , then encrypts $v||S$. In order to improve the security of the list signature scheme, the shared private key is divided into several parts. Each part is owned by a distinct authority, called a Key Authority. With this mechanism, nobody knows the global shared private key, except the smart cards.

2.3. Design of the voting smart card

The central component of our solution is the smart card. Indeed, unlike several other smart card based voting systems, the system described therein relies on advanced cryptographic algorithms implemented inside the card, further than usual RSA signature and encryption. The smart card we use handles a PIN code protection the way banking cards do. We detail more advanced cryptographic capabilities of the card.

- **Sign** is a classical signature algorithm, such as RSA or Schnorr signature. It takes on input a message m and a private key sk and outputs a signature S .
- **Encrypt** is a classical encryption algorithm that takes on input a message m and the public key $epk_{\mathcal{T}}$ of the tellers, and outputs a ciphertext C .
- **Decrypt** is the decryption algorithm corresponding to **Encrypt**. It takes on input an encrypted message C and the private key $esk_{\mathcal{V}}$ of the voter, and outputs the corresponding plaintext message m . The corresponding public encryption key is denoted by $epk_{\mathcal{V}}$.
- **CreateSecretKey** enables a smart card to create its own secret key, which is involved as the symmetric key in the PRNG procedure during the creation of anonymous signatures (see section 3.1 for details).
- **PRNG** is a pseudo random number generator, required by the list signature scheme to reproduce the same number for the same input. This procedure is called using a secret key and a seed. The algorithm used can be a block cipher algorithm like AES-CBC.
- **Concat** is the concatenation of all inputs.
- **LSign** is the list signature algorithm used during the creation of the ballot

- Input²: *term* Message m
card Shared private signature key ssk_{vr}
card Secret key $k_{\mathcal{V}}$
term Linkability Identifier $Id_{\mathcal{L}}$ (64 bits)
- Output: *card* Anonymous signature S_a
- Steps:
 1. $R = \text{PRNG}(k_{\mathcal{V}}, Id_{\mathcal{L}})$
 2. $M = \text{Concat}(R, m)$
 3. $s = \text{Sign}(M, ssk_{vr})$
 4. $S_a = \text{Concat}(s, R)$
 5. Output S_a .

We now detail the computational procedures implemented in the card, which involve the cryptographic functions introduced above.

- **CreateBallot.** This step consists in creating the ballot inside the card, as follows:

- Input: *term* Choice v , Public key $epk_{\mathcal{T}}$
term Election identifier Id_{elec}
card Shared private signature key ssk_{vr}
card Secret key $k_{\mathcal{V}}$
- Output: *card* Ballot B
- Steps:
 1. $S = \text{LSign}(v, ssk_{vr}, k_{\mathcal{V}}, Id_{elec})$
 2. $m = \text{Concat}(v, S)$
 3. $B = \text{Encrypt}(m, epk_{\mathcal{T}})$
 4. Output B .

- **CreateAttendance.** This step corresponds to the attendance signature by the voter, proving that he/she has participated to the current vote and works as follows:

- Input: *card* Private key $ssk_{\mathcal{V}}$
term Challenge $m = Id_{elec} || \text{timestamp}$
- Output: *term* Signature S
- Steps:
 1. $S = \text{Sign}(m, ssk_{\mathcal{V}})$
 2. Output S .

²In the following, we mention for each input of an algorithm executed by the smart card whether it comes from the terminal *term* or from the card *card*.

- **CheckVoting**: the smart card checks whether it has already voted for the current election. For this purpose, the card contains a file $List_{elec}$ with append-only rights. This file contains the identifiers of all the elections the owner of the card has participated in. When invoked by the terminal with input Id_{elec} , this procedure checks that Id_{elec} is not already in $List_{elec}$, otherwise it outputs an error.
- **ValidateVoting**: the smart card registers the fact that it has participated to the current vote. This last procedure completes the participation to an election. The smart card will not be able to vote again for this election. When invoked by the terminal with input Id_{elec} , this procedure appends Id_{elec} to the file $List_{elec}$.

At last, the smart card sends various data to each voting machine during the vote. For this purpose, three procedures are implemented inside the card. The `SendVotingRoomId`, `SendCertificate` and `SendBallot` procedures respectively send Id_{vr} , $Cert$ and the ballot created by `CreateBallot`.

3. Setup of our e-voting system

In this section, we describe the setup of our system. We divide it into three parts, namely the personalization of voting cards, the registration of the voters, and the specific setup that takes place before every new election.

3.1. Smart card personalization

The personalization of the smart card consists in incorporating some data that depend on the voter into the smart card.

1) Embed the PIN that corresponds to that card. This PIN is independently sent to the card owner. We do not discuss the PIN in this paper since it is relatively standard.

2) Insert the identifier of the voting room Id_{vr} of the voter into the smart card.

3) Generation of signature keys: for the attendance sheet, it is required that each voter signs a particular message. This is done using a classical signature scheme and a certificate. This personalization step consists (i) in requesting the smart card to create its pair of signature keys $spk_{\mathcal{V}}/ssk_{\mathcal{V}}$ and (ii) in asking \mathcal{CA} to certify the public one. The smart card finally imports its certificate $Cert$.

4) Generation of a secret key `CreateSecretKey`: for the anonymous signature scheme that this e-voting system relies on, it is required that each smart card owns a secret key used by a block cipher algorithm (see the `LSign` algorithm). The secret key generation process uses the public key of the Recover Authorities but these are not necessary on-line during the creation of the card. In our context, this algorithm takes on input $pk_{\mathcal{RA}_1}, \dots, pk_{\mathcal{RA}_K}$, and outputs the secret key k and K encrypted values $(c_{\mathcal{RA}_1}, \dots, c_{\mathcal{RA}_K})$, one for each \mathcal{RA}_i . This will enable the recover authorities to create a new smart card for the voter in case he has lost his (see section 4). The `CreateSecretKey` procedure works as follows:

- Input: $term$ Size of the secret key l
 $term$ K encryption keys epk_1, \dots, epk_K
- Output: $term$ Encryption data c_1, c_2, \dots, c_K
 $card$ Secret key k

– Steps:

1. $k = \text{RNG}^3(l)$
2. for i from 1 to $K - 1$, $m_i = \text{RNG}(l)$
3. $m_K = k \oplus \left(\bigoplus_{i=1}^{K-1} m_i \right)$
4. for i from 1 to K , $c_i = \text{Encrypt}(m_i, \text{epk}_i)$
5. Store k
6. Output (c_1, \dots, c_K)

When the Smart Card Creation Center has created enough smart cards, it can send to each Recover Authority \mathcal{RA}_i the corresponding encrypted secret key $c_{\mathcal{RA}_i}$. After that, each \mathcal{RA}_i updates its database containing all created secret keys by adding $(\text{Identity}, c_{\mathcal{RA}_i})$.

5) Recovery of the shared signature private key: the anonymous signature we use requires that a signature private key is shared by all smart cards attached to the same voting room. During this phase, it is necessary that the smart card, by way of *SCCC*, connects to the key authorities \mathcal{KA} that manage the shared private key. The interactions between the Smart Card Creation Center *SCCC* and a \mathcal{KA}_i are depicted in figure 2. The aim of this phase is to embed the shared private key while ensuring that only the smart cards can retrieve the global shared private key ssk_{vr} . After receiving all $c_{\mathcal{KA}_i}$, *SCCC* sends

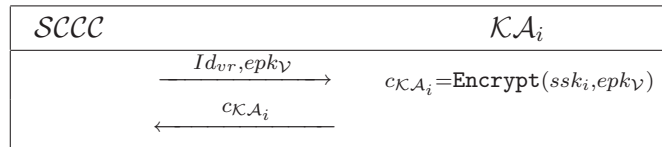


Figure 2. Generation of the Shared Private Key

the request `StoreSharedKey` to the smart card with $c_{\mathcal{KA}_1}, \dots, c_{\mathcal{KA}_P}$ on input:

- Input: *term* Encrypted keys $c_{\mathcal{KA}_1}, \dots, c_{\mathcal{KA}_P}$
card Private key esk_γ
- Output: *card* Shared key ssk
- Steps:
 1. for $i = 1 \dots P$, $ssk_i = \text{Decrypt}(c_{\mathcal{KA}_i}, \text{esk}_\gamma)$
 2. $ssk = f(ssk_1, \dots, ssk_P)$
 3. Store ssk

As there is a global set of Key Authorities for all voting rooms, each Key Authority \mathcal{KA}_i has to request its database $DB_{\mathcal{KA}_i}$ with the entry Id_{vr} to retrieve the corresponding part of the key ssk_i .

³RNG is smart card hardware specific random number generator.

3.2. Registration of voters

When a citizen with identification data Id_{γ} wants to register as a voter, he goes to a Registration Center \mathcal{RC} that first verifies his right to vote, and then links the current voter to a voting room Id_{vr} using some predefined criteria, e.g. the address of the voter.

The Registration Center \mathcal{RC} requests the Smart Card Creation Center \mathcal{SCCC} for the creation and the personalization of a new card for Id_{γ} that belongs to the voting room Id_{vr} . It consequently sends to \mathcal{SCCC} a new entry with the identity Id_{γ} of the voter, and his voting room Id_{vr} . Then, \mathcal{SCCC} launches the smart card personalization procedure (see section 3.1).

At the end of this procedure, \mathcal{RC} retrieves the new smart card and updates its database by adding the new voter. The PIN code of this smart card is directly sent to the new voter. The voter then must retrieve his smart card at the Registration Center.

3.3. Setup of a new election

The system must be prepared for a newly scheduled election. Part of the required actions are to be undertaken before the election day, while others are done on the election day. In this section, we introduce some mechanisms and we detail the necessary updates of the databases prior to the election.

3.3.1. Revocation of voters (cards)

It is sometimes required to revoke the right to vote of a particular voter Id_{γ} . This may be because this voter has moved, or because he lost his right to vote. It can also be useful to setup a mechanism to revoke a voter smart card, e.g. when it has been lost.

In both cases, \mathcal{RC} has to request the Certification Authority \mathcal{CA} for revocation by sending Id_{γ} . Then, the authority \mathcal{CA} searches its database for the certificate $Cert$ of this voter and adds $Cert$ to the revocation list. For user revocation, the Registration Center also has to update its database by deleting the entry Id_{γ} .

3.3.2. Creation of the voting room

An election is created at the level of a voting room by the Controllers \mathcal{C} of this voting room Id_{vr} .

First of all, the Controllers \mathcal{C} create the list of N candidates Cd_1, \dots, Cd_N for the election Id_{elec} (previously created by \mathcal{RC}) and the voting room Id_{vr} .

They have then to create three databases needed throughout the voting process. The first one, denoted by $DB_{\mathcal{ED}}$, consists of the electoral data. It contains the following data the identifier of the election Id_{elec} (that includes the corresponding voting room identifier Id_{vr} , as explained in section 2.1), the number N of candidates, the names of the candidates Cd_1, \dots, Cd_N , and the public key of the Tellers $epk_{\mathcal{T}}$. It is possible to make every possible request to this database. In the following, we will use the request denoted as $\text{Request}(Id_{vr})$ that takes on input an voting room identifier Id_{vr} and that outputs all corresponding data $(Id_{elec}, epk_{\mathcal{T}}, N, Cd_1, \dots, Cd_N)$.

The second database, denoted by DB_{AS} , corresponds to the Attendance Sheet. It contains, for each valid voter, the identity Id_{γ} of the voter, the corresponding certificate $Cert$, he voting room Id_{vr} of the voter Id_{γ} and an empty field Att ready to contain the attendance of the voter. In the following, we will use two distinct requests. The first one, $\text{Request}(Cert)$ takes as input a certificate $Cert$ and

outputs *OK* if the targetted certificate is present on the database and *Revoked* otherwise. the second one, $\text{Request}(Identity)$ takes as input an voter identity *Identity* and outputs *OK* if the field *Att* is empty and *AlreadyVoted* otherwise.

The third database is the ballot box, denoted by DB_{BB} . This database is empty for now and will contain the ballot *B* and the belonging voting room Id_{vr} .

3.3.3. Between the controllers and the tellers

The tellers have to create their cryptographic keys. These keys are only valid for this election and will enable the final counting of the result of this election for the voting room to which they belong.

The creation process of all these keys is described in section 2.2. At the end of the process, each teller T_i owns a private key esk_{T_i} , and together they can compute the corresponding public encryption key $epk_{\mathcal{T}}$. This key is sent to the Controllers that enter it into their database $DB_{\mathcal{ED}}$. The Controllers can then certify the public key $epk_{\mathcal{T}}$ using their signature key $ssk_{\mathcal{C}}$. This certificate expires at the end of the election day, after the counting phase.

4. Running of an election

During the election day, voters can come to the voting room. The process in the voting room is divided into three steps detailed in this section. We also describe a mechanism used if a voter has lost his card, as well as a possibility for anybody to watch the election process.

Everybody must be able to verify attendance and/or number of cast ballots at every moment. For this purpose, the voting room includes a screen which is linked with ballot box database and displays required information. This step is no more detailed in this paper.

When entering the voting room, each voter presents his voting card, so that the controllers can verify the validity of the voter by checking visually the voter and the card. After this verification is done, the voter can enter the polling booth. If someone has lost his smart card, it is possible to create a new one without compromising the security of our system.

4.1. Creating a new voting smart card

In case the registered voter has lost his voting card, it is possible to set up a mechanism that permits the Controllers \mathcal{C} to create (personalize), on-line, a new smart card for this voter using the identity Id_V of the voter and the following procedures:

- Generation of the signature keys for the attendance.
- Generation of a secret key for the anonymous signature.

We consider that the generation of the shared private signature key has already been done during the creation of the smart card, using the same mechanism as explained in section 3.1.

The generation of the signature keys is standard and is not developed anymore.

The recovery of the secret key that is used in the anonymous signature is an interactive protocol between the new smart card \mathcal{SC} and all the Recover Authorities $\mathcal{RA}_1, \dots, \mathcal{RA}_N$, each \mathcal{RA}_i sending back the known part of the key.

After that, the smart card can recover the global secret key by computing $k = \bigoplus_{i=1}^N k_i$. The Recover procedure for \mathcal{RA}_i consists in requesting its database $DB_{\mathcal{RA}_i}$ with the entry $Id_{\mathcal{V}}$ so as to recover the k_i part of the key.

4.2. In the polling booth

In the polling booth, the protocol consists of interactions between the voter \mathcal{V} , his smart card \mathcal{SC} , the voting machine \mathcal{VM} and the Electoral Data database $DB_{\mathcal{ED}}$. All these interactions are described in figure 3.

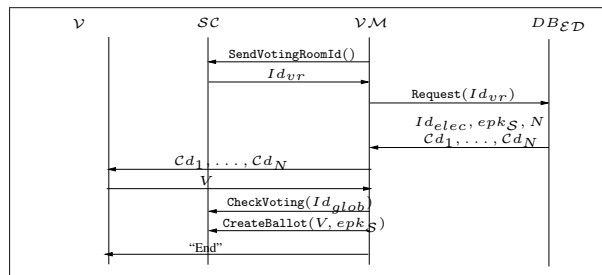


Figure 3. In the Polling Booth

4.3. In front of the ballot box

In front of the ballot box, the protocol consists of interactions between the voter \mathcal{V} , his smart card \mathcal{SC} , the ballot box machine \mathcal{BBM} , the Ballot Box database $DB_{\mathcal{BB}}$ and the Attendance Sheet $DB_{\mathcal{AS}}$. All these interactions are described in figure 4.

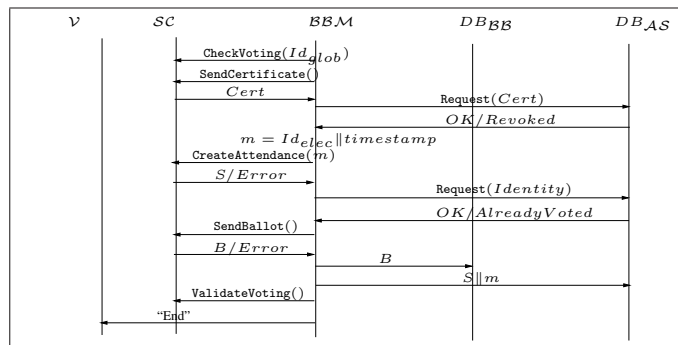


Figure 4. In Front of the Ballot Box

4.4. Counting stage

At the end of the election day, the counting phase first implies that the tellers retrieve all ballots (in DB_{BB}) and all attendances (in DB_{AS}) from machines managed by the Controllers. Then, the tellers can proceed the results by first verifying that there are as many attendances as there are ballots. If there are more attendances than ballots or if one or more attendances are incorrect, it implies a problem either at the network layer level (lost packets), or a fraud of the Controllers. The good choice between the two can easily be made using logs if all transcripts are signed by all protagonists (since a signature provides non-repudiation and integrity). Then, for each ballot B , the tellers have first to decrypt the ballot to obtain $m = v + S = \text{Decrypt}(B, esk_T)$. The next step consists in verifying the list signature produced by the voter. Again, if one list signature is incorrect, this implies either that the Controllers have cheated or that a smart card has been broken.

5. Conclusion and further work

A prototype of this e-voting scheme has been implemented in Java. All authorities were installed on an IBM XSeries345 platform with 2.4 GHz Intel Xeon processor and 1 GB of RAM. The smart card has an 8-bit CPU with 224 kbytes ROM, 6 kbytes RAM and 66 kbytes EEPROM and uses RSA-1024 and 3DES as cryptographic schemes. With this configuration, the overall running time of the `CreateBallot` procedure is 900 ms, and that of the `CreateAttendance` procedure is 800 ms. The counting phase requires less than 1 minute for 1000 ballots. A second prototype needs the voter to use his/her mobile phone to vote. Using RSA-1024 and a mix-net of two mix servers with RSA-1024, the overall running time of the `CreateBallot` procedure for a SIM Javacard is then 800 ms. In order to address security concerns, further works include thorough testing of the components of the system, and the integration of a more complex list signature scheme into the cards. This last step will result in an e-voting scheme whose main interest will be to ensure a stronger level of security, while making the voter more confident in the scheme, as he himself will own the main tools that participate in the security of the system.

References

- [1] Abe, M.: Universally Verifiable Mix-Net with Verification Work Independent of the Number of Mix-Servers, *Advances in Cryptology - Eurocrypt '98* (K. Nyberg, Ed.), 1403, Springer-Verlag, 1998.
- [2] Canard, S., Traoré, J.: List Signature Schemes and Application to Electronic Voting, *Proceedings of Workshop on Coding and Cryptography (WCC'03)*, 2003, 81–90.
- [3] Canard, S., Traoré, J.: Anonymous Services using Smart Card and Cryptography, *Smart Card Research and Advanced Applications VI - Cardis 2004* (J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. E. Kalam, Eds.), Kluwer, 2004.
- [4] Chaum, D.: Blind Signatures for Untraceable Payments, *Advances in Cryptology, Crypto '82* (R. L. R. D. Chaum, A. T. Sherman, Eds.), Lecture Notes in Computer Science, Springer-Verlag, 1983.
- [5] Chaum, D., van Heyst, E.: Group Signatures., *Advances in Cryptology - Eurocrypt '91* (D. W. Davies, Ed.), 547, Springer-Verlag, 1991.
- [6] Cranor, L., Cytron, R.: Design and implementation of a practical security-conscious electronic polling system, *Technical Report WUCS-96-02, Washington University*, 1996.