A secure electronic voting protocol with a simple ballot’s encryption function

Wafa Neji*
Laboratory LIP2,
The General Directorate of Technological Studies,
Higher Institute of Technological Studies of Beja,
TN, Tunisia
Email: wafa.neji@gmail.com
*Corresponding author

Kaouther Blibech
Laboratory LIP2,
Higher Institute of the Environment,
Town planning and Building Technologies, Tunisia
University of Carthage,
TN, Tunisia
Email: kaouther.blibech@gmail.com

Narjes Ben Rajeb
Laboratory LIP2,
National Institute of Applied Science and Technology,
University of Carthage,
TN, Tunisia
Email: narjes.benrajeb@gmail.com

Abstract: In this paper, we present a new electronic voting protocol. It is based on the ballot’s encryption function defined by Schoenmakers in 1999. We use this encryption function in a different way such that we reduce time, communication and computational complexity. In addition, compared to Schoenmakers’ protocol, we satisfy the receipt-freeness property. For this, we rely on the protocol defined by Lee and Kim in 2002 and we use a secure hardware engine called SE. This engine re-encrypts ballots through the use of randomisation technique. Our protocol uses a simple encryption function which requires less computational costs than the one used by Lee and Kim. Our protocol becomes then more secure than the protocol of Lee and Kim. Moreover, an extended version with a multi-way election is provided to allow voters to choose between a number of several candidates.

Keywords: electronic voting; receipt-freeness; secret sharing; zero-knowledge proofs.


Biographical notes: Wafa Neji received an Engineering degree and a Master of Research in Software Engineering from the National Institute of Applied Sciences and Technology of Tunis, in 2010 and 2012, respectively. Currently, she is a university teacher at the General Directorate of Technological Studies, Higher Institute of Technological Studies of Beja at Tunisia. She is in the process of preparing her thesis which is focused on applying secret sharing techniques to electronic voting protocols.

Kaouther Blibech holds an Engineering degree in Networks and Telecommunications from the National Institute of Applied Sciences and Technology (INSAT) of Tunis and a PhD in Computer Science from Pau University in France. Currently, she is an Assistant Professor at the Higher Institute of the Environment, Town planning and Building Technologies at Tunisia. She is interested in security protocols, such as timestamping and electronic voting protocols.

Narjes Ben Rajeb holds an Engineering degree in Computer Science from the Faculty of Sciences of Tunis, a PhD in Computer Science from Henri Poincaré University at Nancy, and a Habilitation for conducting research in Computer Science from the Faculty of Sciences of Tunis. Currently, she is a...
1 Introduction

Electronic voting has several advantages compared to traditional voting process. It is more efficient, faster (final result is almost immediate), cheaper and more reliable (difficulty of vote tampering). In this fact, for several years, there have been extensive researches to define secure electronic voting protocols.

Note that there exist different electronic voting systems (Mursi et al., 2013). The first that we can mention is the direct recording electronic (DRE) system. It is a non-remote electronic engine (such as touch screen system) allowing a voter to cast his vote. Votes are stored locally on memory devices (such as memory card or compact disc) which are transferred at the end of the voting process to a centralised location for tallying. There are other systems used in the electronic voting process which are Poll station and Kiosk electronic voting systems. These voting systems allow voters to cast their vote using electronic engines that are connected to a remote network (such as the internet or a private network). Each casted vote will be transferred secretly over the network to a central tallying location. In Poll station electronic voting systems, electronic devices are installed within official polling stations. In this case, the authentication of voters is controlled by electoral authorities through traditional techniques. Contrariwise, in Kiosk voting systems, the electronic devices are installed within secure suitable locations (such as post offices or shopping malls) chosen by electoral authorities. In this case, the authentication of voters is provided by the electronic devices. The last electronic voting systems that we quote are internet voting systems which allow voters to vote anywhere using a computer connected to the internet.

Electronic voting protocols are based on cryptographic primitives that allow them to satisfy security requirements of the voting process. These requirements have been intensively studied in the literature (Qadah and Taha, 2007; Gritzalis, 2002; Mitrou et al., 2002; Mursi et al., 2013; Sampigethaya and Poovendran, 2006). Note that several definitions of them have been proposed for such protocols. Furthermore, some definitions of the same requirement are presented differently by the authors as described by Fouard et al. (2007).

In what follows, we outline the main security requirements that we want to achieve in our electronic voting protocol:

- **Privacy:** All casted ballots must be secret and/or no traceability between the voter and his ballot can be established. An unauthorised party cannot know the value of the vote contained in the casted ballot and cannot link the voter to his ballot.
- **Correctness:** All valid ballots should be counted correctly. Invalid ballots should not be counted in the final voting result.
- **Eligibility** This property ensures that only voters who are allowed to vote will participate in the voting process.
- **Unreusability:** Each voter is unable to vote twice in the same election.
- **Fairness:** Voters must not know any information about either partial or final voting result. This could affect the intentions of the voters who have not yet voted.
- **Robustness:** The voting system has to be robust against a coalition of a partial number of dishonest authorities as well as a presence of a certain number of failing authorities.
- **Receipt-freeness and incoercibility:** A protocol is said receipt-free if a voter is unable to construct a receipt of his vote or to prove to any party that he has voted in a specific manner. This requirement prevents vote-buying and coercion. Incoercibility ensures that an attacker (or a coercer) cannot force a voter to vote in a particular manner. It will be assumed that the voter is not accompanied by the coercer when he casts his ballot and that this latter cannot spy on the voter when he votes. In this case, the property of receipt-freeness is equivalent to the property of incoercibility. Thus, if a protocol satisfies receipt-freeness, he also satisfies incoercibility.
- **Verifiability:** As described in Neumann et al. (2014) and Kiayias et al. (2015), verifiability is composed of three sub-properties:
  - **Cast-as-intended (CAI).** Each voter can check individually if his ballot has been casted correctly, as he intended.
  - **Stored-as-cast (SAC).** Each voter can check individually if his ballot has been stored correctly for tallying, as he casted it.
  - **Tallied-as-stored (TAS).** Any party can check that the final result has been computed correctly from valid stored ballots.

In this paper, we present a new electronic voting protocol that satisfies the defined security requirements. This protocol is based on the ballot’s encryption function defined by Schoenmakers (1999). We show that the time and the computational complexity of our protocol are lower than Schoenmakers’ protocol. This is due to the fact that unlike this latter, the work accomplished by the voter in our protocol is independent of the number of the electoral authorities. Moreover, to satisfy the receipt-freeness property, our protocol is based on the idea presented by Lee and Kim (2002). Thus, we use a secure hardware engine named $SE$ to replace the role of trusted parties or untappable channels (an untappable channel is a physically secure unobservable communication channel (Sako and Kilian, 1995)). The secure engine $SE$ uses a
randomisation technique to re-encrypt ballots casted by voters ensuring the receipt-freeness property of our protocol.

Note that the electronic voting protocol defined by Lee and Kim (2002) is based on the distributed key generation (DKG) protocol of Pedersen (1991) which doesn’t guarantee a uniform random distribution of the generated keys (Gennaro et al., 2007). In our electronic voting protocol, we use the DKG protocol defined by Gennaro et al. (2007). This protocol satisfies the security requirements related to DKG protocols and ensures a uniform distribution of the generated keys. Furthermore, we show that the encryption function that we used in our protocol requires less computational cost than the one used by Lee and Kim (2002).

The first proposed version of our electronic voting protocol is dedicated to a binary vote in which voters have to answer with yes or no. In this case, each vote is a one bit (1 for yes and 0 for no). Then, we extend the previous yes/no voting protocol to a multi-way election in which voters have to choose between several candidates.

We choose to design an electronic voting protocol intended for Poll stations or Kiosk electronic voting systems. Note that internet voting systems fail to ensure privacy and incorcierlity and have several security problems (Estehghari and Desmedt, 2010; Koenig et al., 2013; Weinstein, 2000) and that DRE systems suffer from many vulnerabilities (such as accidental loss of data and difficulties related to maintenance) and attacks (Bannet et al., 2004).

This paper is organised as follows: first, the approaches and cryptographic mechanisms of existing electronic voting protocols are introduced. Second, electronic voting protocols on which we based our researches are detailed. Then, we present a new electronic voting protocol. After that, an extended version of a multi-way election protocol is provided. Next, the security of our electronic voting protocol is studied and finally, a comparison between it and the protocols on which it is based is done.

2 Approaches and cryptographic mechanisms of existing electronic voting protocols

In the first part of this section, we introduce the most important approaches used in existing electronic voting protocols. More precisely, these approaches cover blind signatures, mix-nets, homomorphic encryptions and secret sharing. In the second part, we focus our analysis on secret sharing techniques by providing the different ways to apply them during the voting process. Afterwards, we specify the role of DKG protocols in electronic voting.

2.1 Approaches to existing electronic voting protocols

Electronic voting protocols must guarantee several security requirements which ensure the validity of the voting process. For this purpose, they use different cryptographic mechanisms. The majority of them are based on blind signatures (Chaum, 1988; Juang et al., 1998; Ohkubo et al., 1999; Fujioka et al., 1992), mix-nets (Aditya et al., 2004; Hirt and Sako, 2000; Hirt, 2001, 2010; Jakobsson, 1998; Lee et al., 2003; Park et al., 1993; Sako and Kilian, 1995; Juels et al., 2005; Clarkson et al., 2008), homomorphic encryptions (Baudron et al., 2001; Benaloh, 1987; Benaloh and Tuinstra, 1994; Cramer et al., 1996, 1997; Hirt, 2001; Rjaskova, 2002; Sako and Kilian, 1995; Chaidos et al., 2016) and secret sharing (Benaloh and Tuinstra, 1994; Cramer et al., 1996, 1997; Rjaskova, 2002; Schoenmakers, 1999; Zou et al., 2014; Chondros et al., 2015).

The use of each approach has several advantages and disadvantages (Lee and Kim, 2002). Electronic voting protocols using blind signatures are unable to ensure the universal verifiability and the receipt-freeness properties at the same time. The blind factor used by the voter can be used to construct a receipt for his ballot.

Electronic voting protocols using homomorphic encryptions have a problem to ensure the receipt-freeness property. Therefore, several researches have been proposed to ensure this property (Acquisti, 2004; Benaloh and Tuinstra, 1994; Hirt, 2001, 2010; Lee and Kim, 2002).

Electronic voting protocols using Mix-nets require important computational and communication costs related to the proofs used to ensure the validity of mixed ballots.

The secret sharing techniques are used in the electronic voting protocols to avoid that a single trusted electoral authority conducts exclusively the election. Furthermore, a single authority can be easily corrupted. In this fact, secret sharing is a technique used to distribute trust between several authorities. Thus, it will be impossible for a single authority to decrypt ballots and to compute the final or partial voting result. Note here that the use of secret sharing techniques can be combined with blind signatures, mix-nets or homomorphic encryptions.

The electronic voting protocol that we will present in this paper is based on the combination of homomorphic encryption and secret sharing approaches. Thus, in what follows we study the secret sharing techniques used through the voting process. Note that we don’t use blind signatures since one of our goals is to define an electronic voting protocol which ensures both receipt-freeness and universal verifiability. We also drop the use of mix-nets because of their significant computational and communication costs.

2.2 Secret sharing techniques

During the voting process, secret sharing can be used in three different ways as follows:

- **Technique 1.** The secret is a single private key shared between authorities. The public key associated with this private key is used by voters to encrypt their ballots and the private key is used to decrypt all ballots. To compute the final tallying of votes, the coalition of authorities is necessary to reconstruct the secret key and to decrypt all ballots. The use of this technique appeared in Cramer et al. (1997) and has been improved in Hirt and Sako (2000), Baudron et al. (2001), Damgard and Jurik (2001), Lee and Kim (2002), Acquisti (2004), Juels et al. (2005), Clarkson et al. (2008), Porkodi et al. (2011), Philip et al. (2011), Chondros et al. (2015) and Chaidos et al. (2016).
The secret is a ballot. Each voter acts as a dealer in the process of a secret sharing scheme and shares his ballot between electoral authorities. The coalition of authorities allows reconstruction of ballots casted by all voters. In the literature, a multitude of electronic voting protocols uses this technique (Benaloh and Tuinstra, 1994; Cramer et al., 1996; Iftene, 2006; Otsuka and Imai, 2010; Zou et al., 2014; Nair et al., 2015).

Technique 3. The secret is a decryption key of a single ballot. Each voter acts as a dealer in the process of a secret sharing scheme and shares the private key related to his ballot between electoral authorities. Then, the coalition of authorities is needed to reconstruct the secret key of each voter and to decrypt ballots. The protocols proposed by Schoenmakers (1999) and Chen et al. (2014) use this technique.

In reviewing the performance of the voting process of each technique, it is clear that computational operations done by each voter in the protocols using the first technique is less consuming than the second and the third ones. Indeed, the complexity and the communication time for voters is independent of the number of authorities. Thus, to vote, each voter will simply send a single encrypted ballot with a single proof that proves the validity of his vote. In protocols using the second and the third technique, each voter must compute and send several shares dependently of the number of electoral authorities and must prove the validity of each sent share.

However, in certain cases, the use of the first technique has a major disadvantage. In fact, the generation of the secret key shared between electoral authorities is usually carried out using secret sharing schemes (Damgard and Jurik, 2001; Porkodi et al., 2011). This process is done by a single party called the trusted dealer who initially holds exclusively the secret key. This latter can decrypt individual ballots and casted by all voters. In the literature, a multitude of verifiable secret sharing (VSS) protocols defined by Pedersen (1991) and Feldman (1987).

In this paper, we use Gennaro’s DKG protocol to design our electronic voting protocol. This DKG protocol is used to share a secret key between electoral authorities. The public key related to this secret key will be used by voters to encrypt their ballots while the secret key will be used by electoral authorities to compute the final voting result. In what follows, we present in detail the process of Gennaro’s DKG protocol.

Gennaro’s DKG protocol. This protocol runs into two steps as follows:

Step 1: Generating sk. In this step, participants jointly generate a secret key sk. For this, each participant Pi acts as a dealer in the VSS protocol of Pedersen (1991) to share a secret random value spi. The secret key sk is defined as the sum of the sp values.

Let p and q denote two large prime numbers, such that q|(p − 1). Let further Gq denote a subgroup of prime order q in Zp. Moreover, g and h denote independently selected generators of Gq. Participant Pi chooses a secret random value spi ∈ Zq and shares it between the other participants.

For this, he chooses two polynomials over Zq of degree t such as f1(x) = ∑tk=0ak,kxk and f2(x) = ∑tk=0bk,kxk where ak,k and bk,k are chosen randomly in Zq and ak,0 = spi. The participant Pi sends privately the shares si,j = f1(j) mod q and si′,j = f2(j) mod q to the participant Pj. He broadcasts also the public values Cij,k = gai,khbj,k mod p for k = 0, ..., t. Each participant Pj can verify the validity of the received shares by checking the following equality:

\[ g^{s_{i,j}h_{i′,j}} \equiv \prod_{k=0}^{t} (C_{i,k})^{x_k} \mod p. \]

If the check fails, Pj broadcasts a complaint against Pi. The complaint management strategy is described in details in Gennaro et al. (1999). At the end of this first step, the set QUAL contains honest participants is built. Each honest participant Pi sets his share of the secret key sk by computing the values Si = ∑j∈QUAL sij mod q and Si′ = ∑j∈QUAL s′,ij mod q.

Step 2: Extracting pk = gsk mod p. In the second step, participants in the set QUAL compute the public key pk related to the secret key sk. For this, they run the Feldman-VSS protocol using the same polynomial f1(x). Each participant Pi just broadcasts the public values Aik,k = gai,k mod p for k = 0, ..., t. The validity of these values can be checked by the other participants using the following equality:

\[ g^{s_{i,j}} \equiv \prod_{k=0}^{t} (A_{i,k})^{x_k} \mod p. \]

Note that if a dishonest participant Pi broadcasts invalid values during this step, the other honest participants can recover his secret value spi by running the reconstruction phase of Pedersen’s VSS protocol.

Gennaro et al. (1999) proved that, unlike DKG Pedersen’s protocol, their protocol satisfies the security requirements
related to DKG protocols and ensures a uniform distribution of the generated keys.

3 Some electronic voting protocols

In this section, we present the protocols on which our electronic voting protocol is based.

3.1 The electronic voting protocol of Schoenmakers

Schoenmakers (1999) proposed a binary voting protocol (yes/no vote) based on a publicly verifiable secret sharing (PVSS) scheme. In this protocol, to vote, each voter shares a secret decryption key related to his ballot between the electoral authorities. For this, a voter acts as a dealer in the PVSS scheme defined by Schoenmakers (1999). Schoenmakers’ electronic voting protocol uses Technique 3 (see Section 2.2). This protocol runs through the following steps:

Step 1: Initialisation. During this step, the initialisation phase of Schoenmakers’ PVSS scheme is performed. Let \( p \) and \( q \) denote two large prime numbers, such that \( q \mid (p - 1) \). Let \( G \) denotes a group of prime order \( q \) in \( Z_p^* \). Moreover, \( g \) and \( G \) denote independently selected generators of \( G \). The players include a set of \( m \) voters \( V_i \) for \( 1 \leq i \leq m \) and \( n \) electoral authorities \( A_j \) for \( 1 \leq j \leq n \). Each electoral authority chooses a private key \( sk_j \in Z_q^* \) and publishes his public key \( pk_j = G^{sk_j} \).

Step 2: Voting. Each voter chooses a vote \( v_i \in \{0, 1\} \) and a random secret value \( s_i \in Z_q \). He encrypts his ballot by computing the public values \( U_i = G^{s_i + v_i} \) and \( C_i = g^{s_i} \). Then, he constructs the proof \( PROOF_i \) (Cramer et al., 1994; Schoenmakers, 1999) showing that \( U_i \) and \( C_i \) encrypt a valid vote \( v_i \in \{0, 1\} \). In addition, the voter shares the value \( s_i \) between the electoral authorities by running the distribution phase of Schoenmakers’ PVSS scheme. The value \( s_i \) is the secret decryption key related to the encrypted ballot. Let \( s_{i,j} \) denotes the share of \( s_i \) sent from the voter \( V_i \) to the electoral authority \( A_j \). The voter \( V_i \) sends the share \( s_{i,j} \) to \( A_j \) in encrypted form such as \( Y_{i,j} = pk_j^{s_{i,j}} \).

Step 3: Tallying. Suppose that \( m \) voters provide valid ballots. Each electoral authority \( A_j \) computes the product of all valid encrypted shares \( Y_{i,j} \) such as \( Y^*_j = \prod_{i=1}^{m} Y_{i,j} = pk_j^{\sum_{i=1}^{m} s_{i,j}} \). Then, \( A_j \) decrypts \( Y^*_j \) using his private key \( sk_j \) and computes \( (Y^*_j)^{1/sk_j} = G^{\sum_{i=1}^{m} s_{i,j}} \). The electoral authorities run the reconstruction phase of Schoenmakers’ PVSS scheme to compute the value \( G^{\sum_{i=1}^{m} v_i} \). Finally, the voting result \( R \) is computed as follows:

\[
R = \frac{\prod_{i=1}^{m} U_i}{G^{\sum_{i=1}^{m} s_{i} + v_i}} = G^{\sum_{i=1}^{m} v_i} = G^{\sum_{i=1}^{m} v_i}.
\]

The value \( T = \sum_{i=1}^{m} v_i \) can be computed efficiently because \( 0 \leq T \leq m \).

Schoenmakers’ electronic voting protocol has some drawbacks. First, the protocol does not satisfy the receipt-freeness property. Thus, the voter \( V_i \) can easily construct a receipt of his ballot using the secret \( s_i \) and proves to any party he has voted using the public values \( U_i = G^{s_i + v_i} \) and \( C_i = g^{s_i} \). Second, the protocol is not appropriate for large-scale elections because of the complexity of the computational operations done by voters and electoral authorities. This is due to the use of Technique 3 (see Section 2.2).

3.2 The electronic voting protocol of Lee and Kim

Lee and Kim (2002) proposed a multi-way election (1-out-of-L) protocol in which voters have to choose between \( L \) candidates. They also proposed an extended version for a multi-choices election (K-out-of-L) in which voter has to choose \( K \) candidates between \( L \) candidates. The electronic voting protocol defined by Lee and Kim uses Technique 1 (see Section 2.2). This protocol uses Pedersen’s DKG protocol as building block to share a secret key \( sk \) between electoral authorities without a trusted party. The public key \( pk \) related to the secret key \( sk \) is used to encrypt ballots. To compute an encrypted ballot, threshold ElGamal encryption is used. The protocol of Lee and Kim ensures the receipt-freeness property thanks to the use of the Tamper-Resistant Randomiser (TRR). TRR is an individual secure hardware given to each voter which replaces the role of third-party randomiser and untappable channels used to achieve receipt-freeness.

The multi-way election protocol runs through the following steps:

Step 1: Initialisation. Let \( p \) and \( q \) denote two large prime numbers, such that \( q \mid (p - 1) \). Let \( G \) denotes a group of prime order \( q \) in \( Z_p^* \) and \( g \) a generator of \( G \). Let \( m \) be the maximum number of voters and \( n \) the number of electoral authorities. Moreover, let \( L \) denote the total number of candidates. During this step, the electoral authorities jointly execute the generation phase of Pedersen’s DKG protocol to jointly generate the secret key \( sk \). At the end of this phase, the public key \( pk = g^{sk} \) related to the secret key is published by electoral authorities. Furthermore, each electoral authority holds a shared \( sk_i \) of the secret key \( sk \).

Step 2: Registration. Each eligible voter may register with an administrator \( A \) (which is a registration manager) to take his certificate \( Cert_i \) and a specific TRR with a certificate \( CertTRR_i \). The values of \( Cert \) and \( CertTRR \) related to the voter \( V_i \) are made public by \( A \).

Step 3: Voting. To vote, each voter chooses a \( j \)th candidate among the \( L \) candidates. Then, he casted his initial ballot using ElGamal encryption such as \( (x, y) = (g^\alpha, pk^\alpha g^{\alpha(m-1)}) \) where \( \alpha \) is a random value chosen by the voter. He sends his encrypted ballot to his \( TRR \). This latter chooses a random value \( \beta \) and re-encrypts the initial ballot by computing \( (x_f, y_f) = (x, g^{\beta} \cdot y \cdot pk^\beta) \). Then, voter’s \( TRR \) proves the validity of the re-encrypted ballot by providing a designated-verifier re-encryption proof (Jakobsson et al., 1996) to the voter. After that, the voter and his \( TRR \) jointly provide the validity proof related to the final ballot using the divertible proof of validity protocol (Cramer et al., 1994; Lee and Kim, 2002). The re-encrypted ballot \( (x_f, y_f) \) and its validity proof are published by the voter on the bulletin board. The bulletin board is a public broadcasted channel in which anyone can read and verify the
We begin by explaining the technique used to ensure the receipt-freeness property. This is done through the use of the secure engine \( SE \). Our electronic voting protocol includes a registration authority \( RA \), a set of \( m \) voters \( V_1, V_2, \ldots, V_m \) and \( n \) electoral authorities \( A_1, A_2, \ldots, A_n \). We follow the communication model defined by Benaloh (1987), which uses a public broadcast channel called the bulletin board. However, in our protocol, the secure engine \( SE \) will post encrypted ballots (with proofs) to the bulletin board instead of voters. Thus, each valid casted ballot will immediately be displayed in encrypted form with its validity proof on the bulletin board. Any party can read the values posted on the bulletin board and nobody can erase any information from it. It will be assumed that an adversary cannot drop or inject messages in the network channel between \( SE \) and the bulletin board. Note that we assume that all exchanged and published values are provided with signatures of voters and \( SE \).

### 4.4 Communication model

Our electronic voting protocol includes a registration authority noted \( RA \), a set of \( m \) voters \( V_1, V_2, \ldots, V_m \) and \( n \) electoral authorities \( A_1, A_2, \ldots, A_n \). We follow the communication model defined by Benaloh (1987), which uses a public broadcast channel called the bulletin board. However, in our protocol, the secure engine \( SE \) will post encrypted ballots (with proofs) to the bulletin board instead of voters. Thus, each valid casted ballot will immediately be displayed in encrypted form with its validity proof on the bulletin board. Any party can read the values posted on the bulletin board and nobody can erase any information from it. It will be assumed that an adversary cannot drop or inject messages in the network channel between \( SE \) and the bulletin board. Note that we assume that all exchanged and published values are provided with signatures of voters and \( SE \).

### 4.3 Overview of our electronic voting protocol

In this section, we give an overview of our electronic voting protocol. Our protocol runs as follows: first electoral authorities run the generating and extracting phases of DKG protocol defined by Gennaro et al. (2007) and jointly generate a public key noted \( pk \). Then, each voter chooses his vote and...
encrypts his initial ballot noted \( E_i \) using the public key \( pk \). We use the encryption function defined by Schoenmakers (1999) but in a different way. Thus, the voter computes an additional value noted \( C_i \). This value is used to generate a validity proof \( \text{Proof}_{E_i} \), provided by the voter to prove the validity of his encrypted ballot. We use a proof that an encrypted message lies in a given set of messages (Cramer et al., 1994; Schoenmakers, 1999). Each voter gives the encrypted ballot \( E_i \) and \( C_i \) with its validity proof \( \text{Proof}_{E_i} \) to \( SE \). This latter generates the re-encrypted ballot \( E_{F_i} \), with the value \( C_{F_i} \), and proves its validity to the voter using the designated-verifier re-encryption proof (DVRP) (Jakobsson et al., 1996). Finally, the voter and \( SE \) cooperate to generate a validity proof of the final ballot noted \( \text{Proof}_{E_{F_i}} \) (Cramer et al., 1994; Lee and Kim, 2002). \( SE \) posts the final ballot \( E_{F_i} \), the value \( C_{F_i} \), and the proof \( \text{Proof}_{E_{F_i}} \) in the bulletin board. To compute the final voting result, electoral authorities use their secret shares related to \( sk \) and cooperate to decrypt the product of all valid ballots posted on the bulletin board. At the end of tallying process, electoral authorities publish the final result \( R \) with a proof noted \( \text{Proof}_R \) of its validity. We use the proof of equality of two discrete logarithms defined by Chaum and Perdersen (1992). All zero-knowledge proofs are detailed in Appendix A. Figure 1 shows an overview of our electronic voting protocol.

### 4.4 Main protocol

Our electronic voting protocol runs through the following steps:

**Step 1: Registration.** To vote, eligible voters may register before the election. Each voter \( V_i \) requests an identifier from the registration authority \( RA \). This latter uses information from the identity card of the voter \( V_i \) to generate a unique identifier \( ID_{V_i} \). In addition, each voter \( V_i \) must choose a secret password that will be used to authenticate to \( SE \). In this fact, the voter, each voter must connect to \( SE \) using his \( ID_{V_i} \) and his password.

Note that we can consider the use of biometric token (Ahmed and Aborizka, 2011; Jain et al., 2004) or fingerprint control (Almun and Bilgin, 2011; Kumar and Begum, 2011) to ensure an increased security during the registration and authentication phases.

**Step 2: Configuration.** Let \( p \) and \( q \) denote two large prime numbers, such that \( q | (p - 1) \). Let further \( G_q \) denotes a subgroup of prime order \( q \) in \( Z_p^* \), such that computing discrete logarithms in this group is infeasible and \( g \) a generator of \( G_q \). In this protocol, we perform all the computations in \( Z_q^* \).

During the configuration process, electoral authorities cooperate to generate the public key \( pk = g^{sk} \) associated to the shared secret key \( sk \). For this, they run the generating and extracting phases of DKG protocol defined by Gennaro et al. (2007). The public key \( pk \) will be used by voters to encrypt their ballots. The shares related to the secret key \( sk \) will be used in the tallying phase to decrypt all ballots and to compute the final voting result.

**Step 3: Voting**

**Step 3.1: Ballot’s encryption.** First, each voter \( V_i \) chooses his vote \( v_i \in \{0, 1\} \) and computes his encrypted ballot \( E_i \) using the public key \( pk \) and a random value \( \beta_i \in R Z_q \) such as:

\[
E_i = pk^{(v_i + \beta_i)}.
\]

Then, each voter \( V_i \) must prove the validity of his encrypted ballot \( E_i \) by computing the value \( C_i = g^{\beta_i} \) and using the proof \( \text{Proof}_{E_i} \). This proof is based on the technique of Cramer et al. (1994). Thus, \( V_i \) provides the proof that \( E_i \) encrypts a valid value \( \in \{0, 1\} \) without revealing the content of his vote. For this purpose, we use a proof that an encrypted message lies in a given set of messages: \( \text{Proof}_{E_i} \) (see Appendix A). The voter gives his encrypted ballot \( E_i \) and \( C_i \) with the proof \( \text{Proof}_{E_i} \) to \( SE \).

**Step 3.2: Ballot’s re-encryption.** The secure engine \( SE \) re-encrypts the ballot of each voter without changing its content. Let \( E_i = pk^{(v_i + \beta_i)} \) and \( C_i = g^{\beta_i} \) denote the initial encrypted ballot of the voter \( V_i \). \( SE \) chooses a random value \( \varepsilon_i \in R Z_q \) and computes the re-encrypted ballot such that:

\[
E_{F_i} = E_i pk^{\varepsilon_i},
\]

\[
C_{F_i} = C_i G^{\varepsilon_i}.
\]

In addition to the new re-encrypted ballot, \( SE \) proves that the new values \( E_{F_i} \) and \( C_{F_i} \) encrypt the correct vote of the voter \( V_i \). This proof should not be transferable to avoid providing to the voter a receipt proving the content of his vote. For this purpose, we use a DVRP Proof of Jakobsson et al. (1996). This proof is convincing only the voter and it is completely useless when it is transferred to any other party (see Appendix A).

Finally, the voter \( V_i \) and \( SE \) cooperate to generate jointly a validity proof of the final re-encrypted ballot noted \( \text{Proof}_{E_{F_i}} \). To provide this proof, we use the same technique defined by Lee and Kim (2002) (see Appendix A). \( SE \) posts the values \( E_{F_i}, C_{F_i}, \) and \( \text{Proof}_{E_{F_i}} \) on the bulletin board.

**Step 4: Tallying.** To compute the final voting result, at least \( t \) honest electoral authorities cooperate to decrypt ballots. Suppose that \( m \) voters cast valid ballots. Each electoral authority \( A_j \) uses his secret share \( S_j \) (see Section 2.3) related to \( sk \) and computes \( (\prod_{i=1}^m C_{F_i})^{S_j} \). Then, electoral authorities use the Lagrange interpolation to compute the value \( (\prod_{i=1}^m C_{F_i})^{S_j} \). Finally, they compute \( R \) as such:

\[
R = \frac{\prod_{i=1}^m E_{F_i}}{(\prod_{i=1}^m C_{F_i})^{S_j}}.
\]

Note that \( R = pk^{v_1 + v_2 + \ldots + v_m} \) and that \( T = v_1 + v_2 + \ldots + v_m \) is the final voting result. Computing the value of \( T \) involves solving the discrete logarithm problem. This is possible for a reasonable size of \( m \) (Cramer et al., 1997). Thus, the final result \( T \) can be computed efficiently knowing that the value of \( T \) is between \( 0 \) and \( m \).
Therefore, if the voting protocol is used for large-scale elections, we can group the valid ballots in several subgroups of reasonable size \( m \). Thus, electoral authorities can decrypt the partial products and easily find the final result.

To prove the validity of the final result, the authorities show that \( R \) is valid by proving that \( \prod_{i=1}^{m} C_i \) and \( \prod_{i=1}^{m} E_i / R \) have the same discrete logarithm for bases \( g \) and \( pk \) respectively. We use non-interactive zero-knowledge proof (Proof\( _{R}\)) of Chaum and Perdersen (1992) in the random oracle model to prove the equality of two discrete logarithms (see Appendix A).

5 Extension to a multi-way election

In our electronic voting protocol, each voter has to choose a vote \( \in \{0, 1\} \). In this case, it is a question of 1-out-of-2 voting. However, it’s often required that a voter has to choose between a number of several options instead of two. This type of vote is named 1-out-of-\( L \) voting. Thus, each voter has \( L \) possibilities and may choose one of them. To achieve this type of vote, we propose In this section, an extension to a multi-way election protocol to allow voters to choose between a number of several candidates.

Let \( m \) be the maximum number of voters and \( L \) the total number of candidates. The voter \( V_i \) can vote for the \( k \)th candidate by choosing \( k \) such that \( 1 \leq k \leq L \). Then, he computes his encrypted ballot \( E_i \) using the public key \( pk \) and a random value \( \beta_i \in R \ Z_q \) such as:

\[
E_i = pk^{(m^k - 1 + \beta_i)}.
\]

The voter \( V_i \) proves the validity of his encrypted ballot \( E_i \) to \( SE \) by computing the value \( C_i = g^{\beta_i} \) and using the Proof\( _{E_i} \) for a multi-way election (see Appendix B). Then, \( SE \) generates the re-encrypted ballot using a random value \( \varepsilon_i \in R \ Z_q \) such that \( E_i = E_i pk^\varepsilon_i \) and \( C_i = C_i g^\varepsilon_i \). Moreover, \( SE \) proves the validity of the re-encrypted ballot to the voter using the same DVRP proof defined in the initial version of our protocol (see Appendix A). Finally the voter and \( SE \) cooperate to generate a proof of validity of the final ballot noted Proof\( _{E_{Fi}} \) for a multi-way election (see Appendix B).

During the Tallying phase, electoral authorities compute \( R = pk_{t_1}m^1 + pk_{t_2}m^2 + \ldots + pk_{t_L}m^L \) where \( (t_1, t_2, \ldots, t_L) \) are the final voting result. Computing the value of \( (t_1, t_2, \ldots, t_L) \) involves to solve the discrete logarithm problem. This is still possible for a reasonable size of \( m \) and \( L \) (Cramer et al., 1997). Validity proof of the final voting result is the same as defined in the initial version of our protocol (see Appendix A).

6 Security analysis

The proposed electronic voting protocol satisfies the following security requirements:

- **Privacy**: In our protocol, an unauthorised party cannot determine values of votes from encrypted ballots published on the bulletin board (see Theorem 4 in Appendix C). Moreover, the validity proofs of initial and final ballots are zero-knowledge. Thus, any information related to the value of votes can be known from these proofs. In order to dispel the link between the voter and his ballot, we assume that \( SE \) posted ballots into bulletin board without voters’ identifiers. Thus, no one is able to link a voter to his ballot.

- **Correctness**: During the voting phase, all ballots are posted by \( SE \) to the bulletin board with proofs of their validity. Note that all invalid ballots will be rejected by \( SE \) during Step 3.1 and cannot be posted to the bulletin
board. This is ensured through the use of Proof$_{E_i}$. All values published into bulletin board are public. Thus, anyone can check the validity of each ballot, the correctness of the sum of all ballots and the validity of the final result published by electoral authorities.

- **Eligibility.** In our protocol, during the Step 1 of Registration, only legitimate voters can register to the registration authority. Moreover, only eligible voters who have a valid identifier delivered by RA can authenticate to SE. Thus, only authorised voters are allowed to vote.

- **Robustness.** The DKG protocol that we use to generate the secret key $sk$ can tolerate maximally the failure of $t - 1$ authorities. At most, we have $t - 1$ dishonest and/or failing authorities.

- **Unreusability.** In our protocol, we assume that a voter cannot authenticate again to SE after he voted the first time. Thus, a voter cannot vote twice in the same election.

- **Fairness.** In our protocol, all ballots are secret and all proofs are zero-knowledge. To compute a partial tally, an attacker must decrypt posted ballots. This is possible only if he holds the private key $sk$. Our protocol protects the secrecy of the secret key $sk$ (see Theorem 1 in Appendix C). Note that at least $t$ honest authorities must cooperate to reconstruct the secret key $sk$. In addition, under the computational Diffie-Hellman assumption, it is impossible to break the encryption of ballots (see Theorems 3 and 4 in Appendix C). In this fact, no participant can gain any knowledge about the tally or the partial tally and thus nothing can affect the voting process.

- **Receipt-freeness.** In our protocol, the engine SE re-encrypts the ballots by adding the random value $\varepsilon_i$. The voter cannot obtain any information of $\varepsilon_i$ chosen by SE. Therefore, he cannot construct a receipt to prove the content of his vote from the final re-encrypted ballot. In addition, the DVRP proof provided by SE is completely useless when it is transferred to any other party and cannot be used to construct a receipt. Note that the validity proof of the final ballot Proof$_{E_i}$ is also randomised by SE and cannot be used by the voter to prove the content of his vote (see Theorem’5 in Appendix C).

- **Verifiability.** Our e-voting protocol ensures the three sub-properties of verifiability:
  - **Cast-as-intended.** After casting his initial ballot $(E_i, C_i)$, the voter receives from SE a DVRP proof proving that SE re-encrypts his ballot without changing its content. This proof shows that the new values $(E_F, C_F)$ encrypt the correct vote chosen initially by the voter under $(E_i, C_i)$. This ensures CAI verifiability.
  - **Stored-as-cast.** Due to the fact that encrypted ballots are publicly readable on the bulletin board, SAC verifiability is ensured. Each voter can verify that his re-encrypted Ballot $(E_F, C_F)$ is stored as he intended by SE simply by consulting the public bulletin board.
  - **Tallied-as-stored.** Any party can verify the validity of each encrypted ballot $(E_F, C_F)$ because it is posted on the public bulletin board with a public proof Proof$_{E_F}$ of its validity. Moreover, electoral authorities provide the public proof Proof$_{E_F}$ of the validity of the final result dependently on public valid encrypted ballots. This ensures TAS verifiability.

### 7 Comparison

In this section, we compare at first our electronic voting protocol with Schoenmakers’ protocol. Second, the comparison is done with Lee and Kim’s protocol. Finally, we compare our protocol with other existing ones which are based on different approaches and cryptographic mechanisms ensuring receipt-freeness.

#### 7.1 Comparison with Schoenmakers’ protocol

Our new electronic voting protocol is simpler than the protocol defined by Schoenmakers (1999). Compared to this latter, our protocol requires less computational operations during the voting process.

In Schoenmakers’ protocol, each voter acts as a dealer and shares his ballot between electoral authorities using a PVSS scheme. In this fact, he must provide a validity proof of each sent share in addition to the validity proof of the casted ballot. Each electoral authority must verify the validity of all shares received from each voter. Note as well that the complexity of computation made by voters depends on the number of electoral authorities.

In our protocol, each voter uses a public key $pk$ to encrypt his ballot and provides only a single validity proof of his encrypted ballot. Thus, compared to Schoenmakers (1999), the complexity of computation is minimal both for the voters and for the electoral authorities. This is due to the use of a different technique of secret sharing during the voting processes. In this fact, Schoenmakers’ protocol is more appropriate for elections on a smaller scale, while our protocol can be used for large-scale elections.

Note also that Schoenmakers’ protocol does not ensure the receipt-freeness property. This is due to the random values chosen by the voter to share his ballot using the PVSS scheme during the voting process. Thus, a voter can construct a receipt which can prove the content of his vote by revealing the random values that he used during the dealing phase of PVSS scheme (see Section 3). Compared to Schoenmakers’ protocol, our electronic voting protocol ensures the receipt-freeness property thanks to the use of the secret engine SE which re-encrypts ballots through randomisation technique.
7.2 Comparison with Lee and Kim’s protocol

The protocol defined by Lee and Kim (2002) uses Exponential ElGamal encryption to encrypt ballots. In our protocol, the encryption of ballots is secure under the CDH assumption.

Note that compared to Lee and Kim’s protocol, we reduce the computational cost of ballot’s encryption function. Indeed, in the protocol of Lee and Kim, to cast a ballot using Exponential ElGamal encryption, a voter \( V_i \) chooses a random value \( \beta_i \) and computes \( (X_i, Y_i) = (g^{\beta_i}, pk^{\beta_i} \cdot g^{v_i}) \). In our protocol, to vote, a voter \( V_i \) computes \( E_i = pk^{v_i} \cdot g^{\beta_i} \) and \( C_i = g^{\beta_i} \). Exponential ElGamal encryption takes a single exponentiation \( (X_i) \) and a multi-exponentiation \( (Y_i) \) while our simple encryption function involves 2 single exponentiations \( (E_i) \) and \( (C_i) \). To compare computation cost of our encryption function with Exponential ElGamal encryption, we rely on the results presented by Kurosawa (2014). Kurosawa (2014) prove that the cost of a multi-exponentiation with two elements can be counted as approximately 20% less than the cost of a single exponentiation (thanks to the use of specific algorithms for computing multi-exponentiations). Therefore, our ballot’s encryption function requires less computational costs than the one used by Lee and Kim.

In our protocol, we add moreover a validity proof of the initial ballot. This proof ensures that a voter casted a valid value into the encrypted ballot. Furthermore, we use a secure DKG protocol defined by Gennaro et al. (2007) to share the secret key \( sk \) without a trusted party. Note that Lee and Kim use Pedersen’s DKG protocol which does not guarantee a uniformly random distribution of the generated keys (Gennaro et al., 2007).

Note that Lee and Kim designed their protocol for internet voting. However, receipt-freeness is achieved only if voters will not transfer or sell their TRR(s) to an adversary. Since there are no ways to prevent voters from selling their TRR(s), receipt-freeness cannot be ensured in this case. In contrast, our protocol is intended for Poll stations or Kiosk electronic voting systems. To ensure receipt-freeness, we only need a secure hardware \( SE \). Thus, we replace the use of individual TRR(s) for voters by multiple \( SE \) installed in different secure locations. \( SE \) cannot be physically moved or transferred to an adversary.

7.3 Comparison with other electronic voting protocols

Unlike our protocol which is based on a secure DKG protocol, several electronic voting protocols based on Technique 1, use as building block the insecure Pedersen’s DKG protocol to generate the secret key shared between electoral authorities (Cramer et al., 1997; Hirt and Sako, 2000; Baudron et al., 2001; Lee and Kim, 2002; Philip et al., 2011). Other protocols based on Technique 1 (Acquisti, 2004; Porkodi et al., 2011) use Shamir Secret Sharing (Shamir SS) scheme for the generation of the secret key. This has several drawbacks. On the one hand, the secret key is held by a single trusted party. In the other hand, it’s not possible to verify the validity of distributed shares related to the secret key. Note that we recommend the use of PVSS schemes which provide solutions to ensure the validity of the distributed shares. It is even better to use secure DKG protocols based on PVSS schemes to share the secret key without allowing any trusted party to learn it (Neji et al., 2016).

Chaidos et al. (2016) proposed an electronic voting protocol named BeleniosRF. It is also based on Technique 1. This protocol ensures receipt-freeness via a re-randomisation server in a non-interactive way. BeleniosRF protocol is characterised by the fact that it uses signatures on randomisable ciphertexts (Blazy et al., 2011) to ensure receipt-freeness. However, this protocol suffers from a single point of failure because it assumes that there is a single trusted server which re-randomises ballots. In addition, if the registration authority and the re-randomisation server cooperate (or if an adversary controls both of them), voters’ choices could be modified easily. Voters cannot detect this and cannot verify that their final ballots were randomised correctly without changing their content. In this fact, BeleniosRF ensures CAI verifiability only assuming that either the re-randomisation server or the registration authority is honest.

Compared to BeleniosRF, in our protocol, ballots are randomised correctly without requiring any specific assumption related to the behaviour of participants. \( SE \) re-encrypts ballots without changing voters’ choices and provides a DVRP proof proving the validity of this. In addition, unlike BeleniosRF (in which voters verify the validity of randomised ballots after publishing them into the bulletin board), randomised ballots in our protocol are published only if voters approve the validity of re-encrypted ballots generated by \( SE \). This guarantee that ballots published on the bulletin board contain the real choices casted by voters.

Note that protocols defined by Hirt and Sako (2000) and Philip et al. (2011) achieve receipt-freeness by involving multiple authorities who re-randomise ballots. This is equivalent to the use of verifiable (or publicly verifiable) re-encryption mix-nets in which each authority plays the role of a single mix-net server. This approach distributes the trust between several parties and eliminates the assumption which requires voters to trust a single server (Chaidos et al., 2016) or a specific hardware engine (like \( SE \) in our protocol or TRR in Lee and Kim (2002)). However, the use of multiple parties (or mix-nets) to re-randomise each ballot increases the complexity of computation and communication costs and makes the voting process not efficient for large scale elections.

Recall that we design our protocol for Poll stations or Kiosk electronic voting systems. Thus, in our protocol, we don’t require all voters to trust a single \( SE \) and we assume that there are diverse \( SE \)s provided by several organisations. In this fact, voters can choose a specific \( SE \) provided by a trusted organisation of their choice.

Kiayias et al. (2015) define a protocol named Demos which is based on code voting schemes (Chaum, 2001). Compared to our protocol, Demos ensures receipt-freeness and verifiability in the standard model and without any setup assumption. However, the receipt-freeness is ensured only for honest voters who will act correctly during the voting phase and who will not forward their vote-codes to the adversary. In contrast, in our protocol, we ensure receipt-freeness even if there are dishonest voters who act maliciously during the voting phase.
(by revealing the initial ballot \((E_i, C_i)\), the re-encrypted ballot \((E_{F_i}, C_{F_i})\), or the proofs used to show their validity).

Moreover, Demos protocol suffers from a single point of failure because it involves a single trusted electoral authority during all steps of the voting process. Chondros et al. (2015) proposed a distributed version of Demos named D-Demos. This protocol is based on Technique 1 and uses Pedersen-VSS scheme to distribute the trust between tallying authorities. However, in this distributed version, the authors always assume that there is a single trusted authority who initialises the voting process and forwards the vote-codes to eligible voters. Note that the privacy of voters could be violated if the trusted authority collaborates with tallying authorities by keeping track and revealing the values of vote-codes sent to each voter. This is a recurrent drawback in electronic voting protocols based on code voting schemes (Joaquim et al., 2009; Kiayias et al., 2015; Chondros et al., 2015).

Other electronic voting protocols combine the use of Technique 1 with anonymous credential and mix-nets to ensure receipt-freeness and coercion-resistance (Acquisti, 2004; Juels et al., 2005; Clarkson et al., 2008). Compared to our protocol, they ensure coercion-resistance even in the presence of a coercer who forces voters to cast a particular vote. However, a weakness of the receipt-free voting protocol of Acquisti (2004) was described by Araújo and Traoré (2013). In addition, protocols defined in Juels et al. (2005); Clarkson et al. (2008) cannot be used for large-scale elections due to the complexity of the tallying process which involves quadratic time complexity of \(O(N^2)\) independently on the number of all casted ballots (Araújo and Traoré, 2010; Spycher et al., 2011).

Note that some recent researches such proposed by Zou et al. (2014); Chen et al. (2014); Nair et al. (2015) define electronic voting protocols based on Technique 2 or Technique 3. In the one hand, this makes them inefficient for large scale elections. In the other hand, these protocols fail to satisfy the property of Receipt-Freeness. This is due to the random values chosen by the voter to share his vote (Technique 2), or to share the secret key related to his encrypted vote (Technique 3). Thus, a voter can construct a receipt which can prove the content of his vote by revealing the random values that he used during the dealing phase of secret sharing. In general cases, during the voting process, when the voter chooses random values to encrypt or compute his vote, he can easily use it to construct a receipt (Hirt and Sako, 2000).

Compared to the protocols defined by Zou et al. (2014); Chen et al. (2014); Nair et al. (2015), our protocol ensures receipt-freeness and becomes suitable to use for large scale elections. This is due to the fact that we based the definition of our protocol on Technique 1 combined with re-encryption approach.

The protocol defined (Zou et al., 2014) is based on Technique 3. This protocol uses an anonymous submission which allows voters to cast their votes in a voting vector without allowing anyone knowing the location of a specific vote within it. Indeed, ballots are represented by a voting vector which can be visualised as a table with \(m\) rows and \(L\) columns (where \(m\) is the number of voters and \(L\) the number of candidates). Each candidate is represented by a single column and each voter owns a single secret row. To obtain a unique secret position within the voting vector, voters run a location anonymisation scheme (Zou et al., 2014). However, this scheme requires multiple rounds between all voters. This may considerably slow down the execution of the election. In addition, to ensure the privacy of the vote, the authors use an additively homomorphic \((m, m)\) secret sharing scheme that divides secret data into random values (which can be published and summed to compute the final voting result). The major disadvantage of the use of \((m, m)\) secret sharing scheme is the big amount of generated random values which grows to \(O(m^2)\). Thus, the protocol defined by Zou et al. (2014) can be applied only for elections with a small number of voters. Moreover, the protocol defined by Zou et al. (2014) does not satisfy receipt-freeness property. The value of the voting vector is public and ballots are visible to all participants. Thus, a dishonest voter can construct a receipt of his vote by revealing his position in the voting vector and the related public values used to divide his ballot. The use of mix-nets or re-encryption may provide a solution to this drawback. Compared to the protocol of Zou et al. (2014), our protocol satisfies the receipt-freeness property and can be used for large-scale elections.

Despite these drawbacks, this protocol reminds original from the point of view of involving tallying authorities who have conflicting interests. Note that electronic voting protocol intended for elections with conflicts of interest parties can be designed based on Technique 1 with DKG protocols which resolve effectively conflicts between involved authorities. Neji et al. (2016) proposed a DKG protocol with an efficient complaint management strategy that clearly identifies dishonest authorities. This DKG protocol can be used as a central component to design new electronic voting protocols.

Finally, it’s important to note that protocols defined in (Cramer et al., 1997; Hirt and Sako, 2000; Baudron et al., 2001; Lee and Kim, 2002; Acquisti, 2004; Clarkson et al., 2008; Porkodi et al., 2011; Philip et al., 2011; Chaidos et al., 2016; Zou et al., 2014; Chen et al., 2014; Nair et al., 2015) are designed to be used for both internet voting systems and Poll stations or Kiosk electronic voting systems, while our protocol is intended to be used only for Poll stations or Kiosk electronic voting systems.

Table 1 summarises the comparison done between our protocol and the main existing ones. The comparison is done with respect to the used Technique (see Section 2.2) and the DKG protocol or the Secret Sharing (SS) scheme on which each protocol is based. For each protocol, we specify if receipt-freeness and verifiability properties are satisfied. Note that these two security properties seem to be incompatible and are hard to be ensured simultaneously (Chevallier-Mames et al., 2010). Moreover, if the receipt-freeness property is satisfied, we indicate the technique used to achieve it. The receipt-freeness property is considered conditionally satisfied for protocols assuming that voters follow honestly the voting process and interact with the adversary only after the elections.

Finally, the comparison is done in terms of scalability. We consider a protocol as scalable if the complexity of the protocol used in the voting process is dedicated to large-scale elections. Note that an efficient voting protocol has to be scalable.
a secure electronic voting protocol with a simple ballot's encryption function

Table 1 Comparison of our protocol with other e-voting protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Technique</th>
<th>SS Scheme</th>
<th>Receipt-Freeness</th>
<th>Technique to ensure Receipt-Freeness</th>
<th>Verifiability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cramer et al. (1997)</td>
<td>1</td>
<td>Pedersen DKG</td>
<td>X</td>
<td>–</td>
<td>✓</td>
</tr>
<tr>
<td>Schoenmakers (1999)</td>
<td>3</td>
<td>Schoenmakers PVSS</td>
<td>X</td>
<td>–</td>
<td>✓</td>
</tr>
<tr>
<td>Hirt and Sako (2000)</td>
<td>1</td>
<td>Pedersen DKG</td>
<td>✓</td>
<td>Multiple randomisation authorities</td>
<td>✓</td>
</tr>
<tr>
<td>Lee and Kim (2002)</td>
<td>1</td>
<td>Pedersen DKG</td>
<td>✓</td>
<td>Trusted TRR for each voter</td>
<td>✓</td>
</tr>
<tr>
<td>Acquisti (2004)</td>
<td>1</td>
<td>(k,n) Shamir SS</td>
<td>AF</td>
<td>Anonymous credentials</td>
<td>✓</td>
</tr>
<tr>
<td>Civitas/JCJ (2008)</td>
<td>1</td>
<td>Not specified</td>
<td>✓</td>
<td>Anonymous credentials</td>
<td>✓</td>
</tr>
<tr>
<td>Philip et al. (2011)</td>
<td>1</td>
<td>Pedersen DKG</td>
<td>✓</td>
<td>Multiple randomisation authorities</td>
<td>✓</td>
</tr>
<tr>
<td>Zou et al. (2014)</td>
<td>2</td>
<td>(m,m) SS</td>
<td>X</td>
<td>–</td>
<td>✓</td>
</tr>
<tr>
<td>Nair et al. (2015)</td>
<td>3</td>
<td>(k,n) Shamir SS</td>
<td>X</td>
<td>–</td>
<td>✓</td>
</tr>
<tr>
<td>BeleniosRF (2016)</td>
<td>1</td>
<td>Not specified</td>
<td>✓</td>
<td>Single trusted re-randomisation server</td>
<td>✓</td>
</tr>
<tr>
<td>Our protocol (2017)</td>
<td>1</td>
<td>Gennaro DKG</td>
<td>✓</td>
<td>Trusted SE(s) for several voters</td>
<td>✓</td>
</tr>
</tbody>
</table>

✓: Satisfied, X: Not satisfied, C: Conditionally satisfied, AF: Attack found.

according to the cost of computation and communication complexity, dependently on the number of voters and electoral authorities.

8 Conclusion

In this paper, we have proposed a secure electronic voting protocol which ensures the principal security requirements of the voting process, especially receipt-freeness and verifiability. By using the encryption ballot function of Schoenmakers in a different way, our protocol requires less communication and computational costs than the protocol defined by Schoenmakers (1999), and it is more secure than the one proposed by Lee and Kim (2002). This paper also explains the role of secret sharing techniques in electronic voting process.

In future works, we will first investigate other secret sharing techniques used in the electronic voting protocols. Thus, we will propose a classification depending on the used secret sharing technique combining with other cryptographic mechanisms. In addition, we will study the achieved security requirements obtained through combinations of specific cryptographic mechanisms and other approaches related to the electronic voting protocols.

Second, as future research, we also intend to study homomorphic deniable encryption schemes with the intention of making our election voting incoercible (without assuming that the coercer cannot spy on the voter when he votes). It is an interesting open problem whether we can construct a fully coercion-resistant electronic voting protocol.

References


A secure electronic voting protocol with a simple ballot’s encryption function


Appendix A: Zero-knowledge proofs of our electronic voting protocol

In this appendix, we present the zero-knowledge proofs used through the different steps of our electronic voting protocol (see Section 4). Note that we present a non-interactive variant of the used proofs. These latter are zero-knowledge in the random oracle model.

Proof that an encrypted message lies in a given set of messages – Proof_{E_i}.

To prove the validity of his encrypted ballot $E_i$, the voter $V_i$ must show to a verifier (or any party) that $E_i = pk^{(v_i+b_i)}$ and $C_i = g^{b_i}$ encrypt a valid value $v_i \in \{0,1\}$. This proof is based on the technique of Cramer et al. (1994) and can be obtained by running the following steps:

- The voter $V_i$:
  
  **First case:** $v_i = 0$ ($E_i = pk^{b_i}, C_i = g^{b_i}$)
  
  - Chooses randomly $w, r_1, d_1 \in R Z_q$.
  
  - Computes $a_0 = g^w, b_0 = pk^w, a_1 = g^{r_1}C_i^{d_1}$ and $b_1 = pk^{b_1}(E_i/pk)^{d_1}$.
  
  - Computes $c = H(a_0, b_0, a_1, b_1)$.
  
  - Computes $d_0 = c - d_1$ and $r_0 = w - \beta_3d_0$.  

Second case: \( v_i = 1 (E_i = pk^{1+\beta_i}, C_i = g^{\beta_i}) \)
- Chooses randomly \( w, r_0, d_0 \in R \mathbb{Z}_q \).
- Computes \( a_0 = g^{\alpha_i}C_i^{d_0}, b_0 = pk^{\alpha_i}(E_i)^{d_0}, a_1 = g^w \) and \( b_1 = pk^w \).
- Computes \( c = H(a_0, b_0, a_1, b_1) \).
- Computes \( d_1 = c - d_0 \) and \( r_1 = w - \beta d_1 \).
- The voter \( V_i \) sends \( (d_0, r_0, d_1, r_1) \) to the verifier.
- The verifier checks if:
  \[
  d_0 + d_1 \equiv H(g^{\alpha_i}C_i^{d_0}, pk^{\alpha_i}(E_i)^{d_0}, g^c, pk^{\alpha_i}(E_i/pk)^{d_1}).
  \]

Designated-verifier re-encryption proof – DV RP. Let \( E_i = pk^{(v_i+\beta_i)} \) and \( C_i = g^{\beta_i} \) denote the initial encrypted ballot of the voter \( V_i \), and \( E_{F_i} = E_i pk^{\alpha_i} \) and \( C_{F_i} = C_i g^{\alpha_i} \) denote the re-encrypted ballot computed by \( SE \). Let further \( pk_{v_i} = pk^{k_{v_i}} \) denotes the public key of the voter \( V_i \) related to his secret key \( s_{k_{v_i}} \). The secret engine \( SE \) must prove the validity of the re-encrypted ballot of the voter \( V_i \) and the proof have to be completely useless when it is transferred to any other party.

\( SE \) proves that \( E_{F_i}/E_i \) and \( C_{F_i}/C_i \) have the same discrete logarithm for bases \( pk \) and \( g \). This proof is based on the technique defined by Jakobsson et al. (1996) and consists of the following steps:
- \( SE \) chooses randomly \( k, r, t \in R \mathbb{Z}_q \) and computes \( a = pk^k, b = g^k \) and \( d = pk^{\alpha_i}pk^{t_{v_i}} \).
- \( SE \) computes \( c = H(a, b, d, E_{F_i}, C_{F_i}) \) and \( u = k - \epsilon_i(c + r) \). It sends \( (c, r, t, u) \) to the voter \( V_i \).
- The voter \( V_i \) checks if:
  \[
  c \equiv H(pk^k(E_{F_i}/E_i), g^u(C_{F_i}/C_i), pk^{\alpha_i}, E_{F_i}, C_{F_i}).
  \]

Note that the value of \( d = pk^{\alpha_i}pk^{t_{v_i}} \) is a trapdoor commitment which can be used to compute \( r \) and \( t \). Indeed, the voter \( V_i \) can compute \( d \) using his private key \( s_{k_{v_i}} \) and random values \( r' \) and \( t' \) such that \( r' + s_{k_{v_i}}' t' = r + s_{k_{v_i}}t \). He can generate a re-encryption proof for any values \( E_{F_i} \) and \( C_{F_i} \). Thus, \( V_i \) chooses randomly \( (\alpha, \beta, u') \) and computes \( c' \), such as:
\[
  c' = H(pk^k(E_{F_i}/E_i), g^{u'(C_{F_i}/C_i)}, pk^{\beta_i}, E_{F_i}, C_{F_i}).
\]

He computes also \( r' = \alpha - c' \) and \( t' = (\beta - r')/s_{k_{v_i}} \). Then, \( (c', r', t', u') \) is an acceptable proof for any encrypted ballot of the voter \( V_i \). In this fact, the proof used in this protocol cannot be transferred to any party.

Validity proof of the final ballot – Proof \( EF_i \). Let \( E_i = pk^{(v_i+\beta_i)} \) and \( C_i = g^{\beta_i} \) denote the initial encrypted ballot of the voter \( V_i \), and \( E_{F_i} = E_i pk^{\alpha_i} \) and \( C_{F_i} = C_i g^{\alpha_i} \) denote the re-encrypted ballot computed by \( SE \). The voter \( V_i \) and \( SE \) will cooperate to provide a non-interactive validity proof of the final ballot. This validity proof is based on the divertible proof of validity defined by Lee and Kim (2002). It consists of the following steps:
- The voter \( V_i \):
  - First case: \( v_i = 0 (E_i = pk^{\beta_i}, C_i = g^{\beta_i}) \), \( E_{F_i} = pk^{\beta_i+\epsilon_i}, C_{F_i} = g^{\beta_i+\epsilon_i} \)
    - Chooses randomly \( w, r_1', d_1' \in R \mathbb{Z}_q \).
    - Computes \( a_0' = g^w, b_0' = pk^w, a_1' = g^{r_1'}C_{F_i}^{d_1'}, b_1' = pk^{r_1'}(E_i/pk)^{d_1'} \).
  - Second case: \( v_i = 1 (E_i = pk^{1+\beta_i}, C_i = g^{\beta_i}) \), \( E_{F_i} = pk^{1+\beta_i+\epsilon_i}, C_{F_i} = g^{\beta_i+\epsilon_i} \)
    - Chooses randomly \( w, r_0', d_0' \in R \mathbb{Z}_q \).
    - Computes \( a_0' = g^w, b_0' = pk^w, a_1' = g^{r_0'}C_{F_i}^{d_0'}, b_1' = pk^{r_0'}(E_i/pk)^{d_0'} \).
- The voter \( V_i \) sends \( (a_0', b_0', a_1', b_1') \) to \( SE \).
- The Secure Engine \( SE \):
  - Chooses \( r_0''', r_1'', d_0'', d_1'' \in R \mathbb{Z}_q \). Note that the equality \( d_0'' + d_1'' = 0 \) must hold.
  - Computes \( a_0 = a_0' g^{r_0''}C_{F_i}^{d_0''}, b_0 = b_0' pk^{r_0''}(E_i)^{d_0''}, a_1 = a_1' g^{r_1''}C_{F_i}^{d_1''} \), \( b_1 = b_1' pk^{r_1''}(E_i/pk)^{d_1''} \).
  - Sends \( a_0, b_0, a_1, b_1 \) to \( V_i \).
- The voter \( V_i \) computes \( c = H(a_0, b_0, a_1, b_1) \).
- The voter \( V_i \):
  - First case: \( v_i = 0 \)
    - Computes \( d_0' = c - d_1' \) and \( r_0'' = w - \beta d_0' \).
  - Second case: \( v_i = 1 \)
    - Computes \( d_1' = c - d_0'' \) and \( r_1'' = w - \beta d_1' \).
- The voter \( V_i \) sends \( (d_0', r_0'', d_1', r_1'') \) to \( SE \).
- The Secure Engine \( SE \):
  - Computes \( d_0 = d_0' + d_0'', d_1 = d_1' + d_1'', r_1 = r_1' + r_1'' - d_0 \epsilon_i \) and \( r_0 = r_0'' + r_1'' - d_0 \epsilon_i \).
  - Sends \( d_0, r_0, d_1, r_1 \) to the voter \( V_i \).
- The voter \( V_i \) (or any party) checks if:
  \[
  d_0 + d_1 \equiv H(g^{\alpha_i}(C_{F_i}), pk^{\alpha_i}(E_{F_i})^{d_0}, g^{c}(C_{F_i})^{d_1}, pk^{\alpha_i}(E_{F_i}/pk)^{d_1}).
  \]

Thus \( SE \) and \( V_i \) produce valid proof \( (a_0, b_0, a_1, b_1, d_0, d_1, r_0, r_1) \) of the final ballot.

**Proof of the Equality of Two Discrete Logarithms - ProofR.**

Electoral authorities want to convince the public whether the two elements \( \prod_{i=1}^{m} C_{F_i} \) and \( \prod_{i=1}^{m} E_{F_i}/R \) have the same discrete logarithm for bases \( g \) and \( pk \), i.e., they want to prove that \( \log_g(\prod_{i=1}^{m} C_{F_i}) = \log_{pk}(\prod_{i=1}^{m} E_{F_i}/R) \). Note that \( (g, pk, \prod_{i=1}^{m} C_{F_i}, \prod_{i=1}^{m} E_{F_i}/R) \) are given as common input. We use the proof of Chaum and Perdersen (1992) to prove the equality of two discrete logarithms. The proof consists of the following steps:
Appendix B: Zero-knowledge proofs of our extended version

In this appendix, we provide the modified zero-knowledge proofs according to the extended version with a multi-way election (see Section 5).

\[ c = H(g^{m \sum_{i=1}^{m} C_{i}}, pk) \]

\[ c \times \prod_{i=1}^{m} E_{F_{i}} = H(a, b, u). \]

\[ \text{Authorities compute } u = c - \sum_{i=1}^{m} (\beta_{i} + \varepsilon_{i}) \text{ and publish } (a, b, u). \]

\[ \text{Voters (or any party) compute } c = H(g^{m \sum_{i=1}^{m} C_{i}}, pk) \prod_{i=1}^{m} E_{F_{i}} = H(a, b, u) \text{ and verify } \]

\[ \sum_{i=1}^{m} E_{F_{i}} = H(a, b, u). \]

\[ A \text{ secure electronic voting protocol with a simple ballot’s encryption function } \]

\[ \text{The voter } V_{i} \text{ chooses } w \in R_{Z_{q}} \text{ and computes } a_{w} = g^{w} \text{ and } b_{w} = pk^{w}. \]

\[ \text{For } j = 1, \ldots, k - 1, k + 1, \ldots, L, \text{ the voter } V_{i} \text{ chooses } r_{j}, d_{j} \in R_{Z_{q}} \text{ and computes } a_{j} = g^{r_{j}} C_{i}^{d_{j}} \text{ and } \]

\[ b_{j} = pk^{r_{j}} (E_{i}/pk^{m-1})^{d_{j}}. \]

\[ \text{Then he sends } (A', B') = (a_{0}, b_{0}, \ldots, a_{L}, b_{L}) \text{ to } SE. \]

\[ \text{For } j = 1, \ldots, L \text{ SE chooses } r_{j}' \text{, } d_{j}' \in R_{Z_{q}} \text{ for } v \in \{0, 1\} \text{ and computes } a_{j}' = g^{r_{j}'} C_{i}^{d_{j}'} \text{ and } \]

\[ b_{j}' = pk^{r_{j}'} (E_{i}/pk^{(m-1)}d_{j}'). \]

\[ \text{Note that the following equality } \sum_{j} d_{j}' = 0 \text{ must be hold. } SE \text{ sends } (A, B) = (a_{1}, b_{1}, \ldots, a_{L}, b_{L}) \text{ to } V_{i}. \]

\[ \text{The voter } V_{i} \text{ computes } c = H(a_{1}, b_{1}, \ldots, a_{L}, b_{L}). \]

\[ \text{The voter } V_{i} \text{ computes } d_{j}' = c - \sum_{j \neq k} d_{j} \text{ and } \]

\[ v_{j} = w - \beta_{i} d_{j} \text{. Voter sends } \]

\[ (D', R') = (d_{1}', r_{1}', \ldots, d_{L}', r_{L}') \text{ to } SE. \]

\[ \text{For } j = 1, \ldots, L \text{ SE computes } d_{j} = d_{j}' + d_{j}'' \text{ and } \]

\[ r_{j} = r_{j}' + r_{j}'' - d_{j} \beta_{i}. \text{ SE sends } \]

\[ (D, R) = (d_{1}, r_{1}, \ldots, d_{L}, r_{L}) \text{ to the voter } V_{i}. \]

\[ \text{The voter } V_{i} \text{ (or any party) verifies :} \]

\[ d_{1} + \ldots + d_{L} \neq H(g^{m \sum_{i=1}^{m} C_{i}}, pk^{m-1} (E_{i}/pk^{(m-1)}d_{1}')) \]

\[ \text{Thus, } SE \text{ and } V_{i} \text{ produce validity proof (} a_{1}, b_{1}, \ldots, a_{L}, b_{L}, d_{1}, \ldots, d_{L}, r_{L} \text{) of the final ballot.} \]

Appendix C: Security proofs

In what follows, we first demonstrate that the secret key \( sk \) satisfies the secrecy requirement and so cannot be known by a dishonest party. Afterward, we prove the security of the ballot’s encryption function used to encrypt the initial vote (\( E_{i} \) and \( C_{i} \)) and the re-encrypted vote (\( E_{F_{i}} \) and \( C_{F_{i}} \)). Finally, we prove that the voter cannot construct a receipt of his vote from the values of the initial encrypted ballot and the final encrypted ballot.

Theorem 1 Under the discrete-log assumption, our protocol protects the secrecy of the secret key \( sk \). Thus, no information on \( sk \) can be learned by a dishonest party except for what is implied by the value \( pk = g^{sk} \).

Proof: To break the encryption of the secret key \( sk \), a dishonest party must compute \( sk \) from \( pk = g^{sk} \). This implies solving the Discrete Logarithm Problem. Given that computing the discrete log in \( G_{q} \) is infeasible, the dishonest party is not able to compute \( sk \) from \( g^{sk} \).

Moreover, we rely on the following Lemma of the DKG protocol defined by Gennaro et al. (2007) and used to generate \( sk \):

Lemma 2 Under the Discrete-Log Assumption, the DKG protocol satisfies the secrecy requirement of the secret key \( sk \) with threshold \( t \), for any \( t < n/2 \).

The secrecy proof of \( sk \) is provided by Gennaro et al. (2007). Thus, the secret key \( sk \) is a confidential information which cannot be discovered by a dishonest party.
Theorem 3 Under the computational Diffie-Hellman assumption, it is impossible to break the encryption of the initial ballot.

Proof: For a dishonest party knowing the public key $pk = g^{sk}$ and the values $E_i = pk^{(v_i + \beta_i)} = g^{sk(v_i + \beta_i)}$ and $C_i = g^{\beta_i}$ exchanged between the voter $V_i$ and SE, breaking the encryption of the ballot implies computing $g^{sk\beta_i}$. To be able to do that, he has to compute $g^{sk(v_i)} = E_i / g^{sk\beta_i}$ from the values $E_i$, $pk = g^{sk}$ and $C_i = g^{\beta_i}$. This implies computing $g^{sk\beta_i}$ from the values $g^{sk}$ and $g^{\beta_i}$. Recall that the Computational Diffie-Hellman assumption states that it is infeasible to compute $g^{sk(v_i)}$ given $g^{sk}$ and $g^{\beta_i}$. Thus the dishonest party cannot break the ballot’s encryption.

Furthermore, in our electronic voting protocol, ballots are encrypted with the secret key $sk$. Only the party possessing $sk$ can extract the value of vote from the initial encrypted ballot $E_i$. Note that at least $t$ honest electoral authorities must cooperate to reconstruct the secret key $sk$. Thus, to break the encryption of the ballot, the dishonest party should be able to compute $sk$. We rely on the Theorem 1, which states that it is infeasible to compute $sk$.

Theorem 4 Under the computational Diffie-Hellman assumption, it is impossible to break the re-encryption of the ballot.

Proof: For a dishonest party knowing the public values $E_{F_i} =.pk^{(v_i + \beta_i + \epsilon_i)} = g^{sk(v_i + \beta_i + \epsilon_i)}$, $C_{F_i} = g^{\beta_i + \epsilon_i}$ and $pk = g^{sk}$, breaking the re-encryption of the ballot implies computing $g^{sk(\beta_i + \epsilon_i)}$. For this, the dishonest party should be able to compute $g^{sk(v_i)} = E_{F_i} / g^{sk(\beta_i + \epsilon_i)}$. This implies computing $g^{sk(\beta_i + \epsilon_i)}$ from the values $g^{sk}$ and $g^{\beta_i + \epsilon_i}$. This is infeasible under the Computational Diffie-Hellman assumption. Furthermore, to break the encryption of the re-encrypted ballot, the dishonest party should be able to compute $sk$. This is infeasible under the Theorem 1.

Theorem 5 Under the Discrete-Log Assumption, it is impossible for a voter to construct a receipt of his vote or to prove to any party that he has voted in a specific manner.

Proof: Suppose that a dishonest voter wants to construct a receipt of his vote from values $E_i = pk^{(v_i + \beta_i)} = g^{sk(v_i + \beta_i)}$, $C_i = g^{\beta_i}$, $E_{F_i} = pk^{(v_i + \beta_i + \epsilon_i)} = g^{sk(v_i + \beta_i + \epsilon_i)}$, $C_{F_i} = g^{\beta_i + \epsilon_i}$ and $pk = g^{sk}$. Additional information that this voter can compute are $E_{F_i} / E_i = pk^{\epsilon_i}$ and $C_{F_i} / C_i = g^{\epsilon_i}$. To construct a receipt of his vote, the voter must compute $\epsilon_i$ from $pk^{\epsilon_i}$ or from $g^{\epsilon_i}$. This implies solving the Discrete Logarithm Problem. Given that computing the discrete log in $G_q$ is infeasible, the dishonest voter is not able to compute $\epsilon_i$ from $pk^{\epsilon_i}$ or from $g^{\epsilon_i}$. □