Elliptic Curve Array Ballots for Homomorphic Tallying Elections

M.A. Cerveró[1], V. Mateu[1,2], S. Martínez[1], J.M. Miret[1], and F. Sebé[1]

[1] Dept. of Mathematics. Universitat de Lleida.
Jaume II, 69, 25001, Lleida, Spain.
{mcervero, vmateu, santi, miret, fsebe}@matematica.udl.cat
[2] Scytl Secure Electronic Voting.
Tuset, 20, 08006, Barcelona, Spain.

Abstract. Remote voting systems implementing the homomorphic tallying paradigm have proven to be the best option for elections with a small range of candidates. In this paper, we propose a new homomorphic tallying remote voting system that makes use of elliptic curve cryptography. The proposed system is suitable for multiple choice elections. Detailed security and performance analysis are provided.

Keywords: Electronic Voting · Elliptic Curve Cryptography · Homomorphic Cryptosystem · Security

1 Introduction

Remote voting systems allow the participants to cast their ballots from any place with an available Internet connection. At the end of the voting period, the votes can be tallied automatically, reducing the economic cost while improving the speed and accuracy of the process. Unfortunately, the fact that the voting platform is accessible through the Internet makes it vulnerable to attacks coming from the network. Hence, security is a key aspect to consider. A remote voting system must provide the following security properties:

Authentication: only the votes cast by eligible voters are taken into account.
Unicity: a participant can vote once at most.
Privacy: vote content cannot be related to the identity of its caster.
Fairness: partial results are not revealed.
Verifiability: correctness of the voting process can be checked. A voting system is *universally verifiable* if any external entity can verify that all votes have been properly counted. If voters can only verify that their own vote has been taken into account, the scheme is *individually verifiable*.
Uncoercibility: a voter cannot prove she voted in a particular way.

The aforementioned security requirements are achieved by making use of advanced cryptographic techniques. According to the way in which cryptography

is used, remote voting schemes can be classified into three main paradigms: *mix-type*, *blind signature-based* and *homomorphic tallying*.

The *mix-type* paradigm [16,18,21] resembles paper-based traditional voting using a ballot box. Eligible voters generate their votes, encrypt them using the election public key and sign their encrypted vote (ballot) before sending it to the polling station. In order to authenticate each voter, the polling station verifies the signature of the received ballots. Once the voting period is concluded, the ballots are mixed (shuffled and remasked) in order to break the link between them and the identity of voters who cast them. Finally, they are decrypted and tallied. Ballot mixing has to be performed verifiably. To that end, the polling station computes a zero–knowledge proof to demonstrate that the cleartexts of the received ballots are equal to the cleartexts of the mixed ballots. Such proofs are usually hard to generate and verify in terms of computational effort. Their conceptual complexity is also very high so that it is very difficult for observers to get convinced about the security they enforce.

In *blind signature-based* voting schemes [10,17], a participant composes her vote, encrypts it and then authenticates herself to a trusted authority (the authentication server) who manages the electoral roll. That trusted authority checks whether this is the first time the participant is authenticated. In that case, the authentication server blindly signs [4] the participant's ballot. Next, the participant transmits her blindly signed ballot to the polling station through an anonymous channel. The voting platform will only accept ballots that have been signed by the authentication server. Once the voting period is concluded, votes are decrypted and tallied. The process needs a verifiable anonymous channel if universal verifiability is needed.

The *homomorphic tallying* paradigm, first proposed in [6], is constructed over a homomorphic public key cryptosystem. In this paradigm [3,12,14,15,23], participants cast their votes encrypted under the homomorphic cryptosystem. After collecting all the ballots, the polling station aggregates them using the homomorphic operation. As a result, a single ciphertext is generated. Its decryption will provide the homomorphic addition/multiplication of the cleartext votes. An appropriate coding of votes permits to obtain the global vote tally from this message. In such a system, the participants are required to prove in zero-knowledge that their ballots have been properly composed [8,11]. These proofs can be verified as ballots are being received by the polling station. However, their complexity renders this paradigm to be only applicable to elections in which choices can be coded in a very simple way.

Homomorphic cryptosystems can be additive or multiplicative. The problem when operating with a multiplicative homomorphism is that cleartext multiplication easily overflows the cryptosystem cleartext range. To avoid that, proposals like [19,20] aggregate the ballots in small groups so that cleartext overflowing is avoided. After that, the resulting aggregated ciphertexts are mixed. Although such hybrid systems are more efficient that mix-type schemes, they are not as fast as classic homomorphic tallying schemes.

Helios 2.0 is a remote voting system that belongs to the homomorphic tallying paradigm. The original Helios 1.0 was presented in [1] and has been successfully used for small elections [2]. Our proposal is an improvement to Helios 2.0. It enhances that system by allowing a better perception of blank ballots, especially in elections in which voters can vote for more than one candidate. In addition, the use of elliptic curve cryptography provides a better performance.

Motivation and Contribution

Although elliptic curve cryptography provides very fast encryption and decryption methods, it is rarely used within the homomorphic tallying remote voting paradigm due to the elevated computational cost of the process required to obtain the election result from the cleartext of the aggregated ciphertext. Furthermore, there exist some zero- knowledge proofs that do not work with elliptic curves while others just do not provide the security of their multiplicative group version. Although the proposal in [3] is completely functional and meets all the security requirements listed before, the proof and the verification of ballots as well as its homomorphic decoding step have an elevated computational cost. In this paper we exploit the additive homomorphic property of the Elliptic Curve ElGamal cryptosystem to create an efficient remote voting protocol suitable for real elections. The proposal allows to aggregate all the ballots into a single ciphertext, so a mixing step is not needed.

The paper is structured as follows. Section 2 recalls some theoretical aspects of elliptic curve cryptography, focusing on the Elliptic Curve ElGamal cryptosystem. Our proposal is described in Section 3. The security of the presented system is discussed in Section 4. Finally, Section 5 is devoted to provide some details about the performance together with some concluding remarks.

2 Elliptic Curves

An elliptic curve E over a prime finite field $\mathbb{F}_p$ is an algebraic curve given by the reduced Weierstraß equation [22],

$$E : y^2 = x^3 + ax + b,$$

with nonzero discriminant $4a^3 + 27b^2 \neq 0$. We denote by $E(\mathbb{F}_p)$ the set of points $(x, y) \in \mathbb{F}_p \times \mathbb{F}_p$ that satisfy the curve equation, along with the point at infinity $\mathcal{O}$. An addition operation can be defined over $E(\mathbb{F}_p)$ using the chord-tangent method. This operation endows the set $E(\mathbb{F}_p)$ with an abelian group structure in which $\mathcal{O}$ is the identity element. Considering this group law, the *Elliptic Curve Discrete Logarithm Problem* (ECDLP) consists in, given two points P and Q, find an integer d that satisfies $Q = dP$. This is a computationally hard problem when the cardinality of $E(\mathbb{F}_p)$ has a large prime factor.

2.1 Elliptic Curve ElGamal Cryptosystem

In this paper, we use an analogue of the ElGamal cryptosystem [9] using elliptic curves, the Elliptic Curve ElGamal (EC-ElGamal). For simplicity, only elliptic curves defined over a prime order finite field are considered.

Setup and Key Creation. The setup of the cryptosystem requires choosing a prime p, which defines a finite field $\mathbb{F}_p$, and two parameters a and b defining an elliptic curve E over $\mathbb{F}_p$. We also need an order m point $P \in E(\mathbb{F}_p)$, such that m is a large prime of the same size as p.

A private key is created by taking a random integer $d \in \{1, \ldots, m-1\}$. The corresponding public key is $Q = dP$.

Encryption and Decryption. A point $V \in E(\mathbb{F}_p)$ is encrypted under public key Q as follows:

$$Enc_Q(V) = C = (A, B) = (rP, V + rQ),$$

where r is a random integer in $\{1, \ldots, m-1\}$.

A ciphertext $C = (A, B)$ can be decrypted when the private key d is known. The cleartext V is recovered by computing:

$$Dec_d(C) = V = B - dA.$$

A ciphertext $C = (A, B)$ can be *verifiably decrypted* by publishing its cleartext V and proving in zero-knowledge that $\log_P Q = \log_A(B - V)$ by means of Chaum-Pedersen's proof [5]. However, there is an alternative way to reveal the cleartext of C. This can be done if the random value r generated for encryption is known. First, we have to check whether rP equals A and next compute:

$$Rev_r(C) = V = B - rQ.$$

Homomorphic Property. The EC-ElGamal cryptosystem has a homomorphic property with respect to the point addition operation. Two ciphertexts $C_1 = (A_1, B_1)$ and $C_2 = (A_2, B_2)$, encrypting V_1 and V_2, can be aggregated as

$$C_3 = C_1 + C_2 = (A_1 + A_2, B_1 + B_2) = ((r_1 + r_2)P,\ \ (V_1 + V_2) + (r_1 + r_2)Q).$$

The decryption of C_3 provides $V_3 = V_1 + V_2$ as a result.

3 E-voting Scheme

In this section, we present a remote voting system implemented using the EC-ElGamal cryptosystem. Our proposal can be extended to accommodate multi-candidate elections, as shown in Section 3.4.

3.1 Participating Parties

We assume there exists a publicly readable Bulletin Board (BB) on which only some specific authorities can write. Our protocol involves the following parties:

Registrar: it publishes the electoral roll on the BB.
Key Storage Trusted Party (KSTP): it generates and stores the election private key and publishes the election public key on the BB. It will perform a verifiable decryption when required. It may be a distributed entity.
Voters: eligible voters are those appearing in the electoral roll.
Polling Station (PS): it collects the ballots and, after verifying their correct composition and checking the people casting them are listed in the electoral roll, publishes them on the BB so that anyone can check their validity.

The BB is publicly accessible for reading, but only the Registrar, the KSTP and the PS can write on it.

3.2 Voting System

The forthcoming system description assumes a single-choice election (see Section 3.4 for multi-candidate elections). The system description is divided into three phases: *setup*, *vote casting* and *tallying*. Each stage starts when the previous one ends.

Setup Phase. Let n be the amount of eligible voters in the electoral roll. First of all, the Registrar publishes the electoral roll and the list of candidates $L = \{L_1, L_2, \ldots, L_k\}$ on the BB. The electoral roll consists of a list of all the eligible voters $\{v_1, v_2, \ldots, v_n\}$ and their public keys $pk_1, pk_2, \ldots, pk_n$, which could be certified by some certificate authority.

Next, the KSTP chooses a suitable elliptic curve and chooses a secret key d. Next, it publishes an order m point $P \in E(\mathbb{F}_p)$ together with the election public key $Q = dP$ on the BB.

Reliability on the KSTP can be increased by distributing it into a set of entities $\{\mathrm{KSTP}_1, \ldots, \mathrm{KSTP}_t\}$. Thus, each KSTP_i generates its own private key d_i and publishes $Q_i = d_i P$. Then, the election public key is $Q = \sum_i Q_i$. A message encrypted under Q can only be decrypted if all the entities composing the KSTP do collaborate.

Finally, the PS publishes a point V of the same curve. It also precomputes and stores the points $V, 2V, \ldots, \lfloor n/2 \rfloor V$. Each point iV is stored together with the corresponding integer i in a hash table $\mathcal{T}$.

Vote Casting Phase. A voter v_i creates her ballot for a candidate L_j by generating a ciphertext array $(C_{i,1}, \ldots, C_{i,k}, C_{i,k+1})$ in which the cleartext of $C_{i,j}$ is $2V$ while the remaining ciphertexts are an encryption of V. Ciphertext $C_{i,k+1}$ would accommodate and eventual blank vote. Assuming the chosen candidate

was L_k, v_i would randomly select $r_{i,1}, \ldots, r_{i,k}, r_{i,k+1} \in_R \{1, \ldots, m-1\}$ and would generate:

$$C_{i,j} = (r_{i,j}P,\ V + r_{i,j}Q), \quad \text{for } j \neq k,$$
$$C_{i,k} = (r_{i,k}P, 2V + r_{i,k}Q).$$

Next, v_i signs her ballot and creates a zero-knowledge proof proving a proper composition of it. This is done by means of the procedure described in Section 3.3. Finally, she sends $(C_{i,1}, C_{i,2}, \ldots, C_{i,k}, C_{i,k+1})$, her signature and the zero-knowledge proof to the PS.

When the PS receives a ballot, it, firstly, verifies the signature using the voter's public key, available on the BB. Then, it checks that the voter has not voted before and that no other ciphertext with the same ciphertexts has been cast before by some other voter. Finally, it verifies the zero-knowledge proof. If all the verifications are satisfied, it publishes the ballot, its zero-knowledge proof and its signature on the BB. Otherwise, the ballot is discarded.

Vote Tallying Phase. When the vote casting phase has ended, the PS aggregates the received ballots into a ciphertext array $(T_1, \ldots, T_k, T_{k+1})$ by computing,

$$T_j = \sum_{i=1}^{z} C_{i,j} = \left(\sum_{i=1}^{z} A_{i,j}, \sum_{i=1}^{z} B_{i,j} \right), \quad \forall j \in \{1, \ldots, k+1\},$$

where z is the amount of voters that have cast a vote. Then, the PS asks the KSTP to perform a verifiable decryption of each ciphertext T_j, obtaining

$$Dec_d(T_j) = \sum_{i=1}^{z} y_{i,j}V, \quad y_{i,j} \in \{1, 2\}.$$

The value $\sum_{i=1}^{z} y_{i,j}$ ranges between z and $2z$. Therefore, the PS computes $Dec_d(T_j) - zV = x_jV$ and searches for x_jV in the precomputed table $\mathcal{T}$. If candidate L_j has received more than $\lfloor n/2 \rfloor$ votes, then x_jV will not be found in the table. In that case, the PS computes $x'_jV = x_jV - \lfloor n/2 \rfloor V$ and checks x'_jV against $\mathcal{T}$ again. Thus, the number of votes for candidate L_j is $x_j = x'_j + \lfloor n/2 \rfloor$. This operation generates the amount of votes for candidate L_j, *i.e.* x_j, as a result.

3.3 Zero-Knowledge Proof of Correct Ballot Composition

During the vote casting phase, the voter must provide a zero-knowledge proof proving her ballot has been properly composed. To that end, we have adapted a non-interactive proof presented in [8], to operate with EC-ElGamal ciphertexts.

This proof proves in zero-knowledge that a given ciphertext is an encryption of a point in a set $\{V_1, V_2, \ldots, V_\ell\}$. In our system, we prove that each component of a ballot is an encryption of V or $2V$. Moreover, by making use of the reveal operation, presented in Section 2.1, we guarantee that a voter cannot vote for more candidates than she is allowed. This is done by revealing the cleartext of the ciphetext obtained from the aggregation of all the ballot components.

Prover. The prover (voter v_i) has to prove in zero-knowledge that each ciphertext $C_{i,j} = (A_{i,j}, B_{i,j})$ in vector $(C_{i,1}, \ldots, C_{i,k}, C_{i,k+1})$ is an encryption of either V or $2V$. Thus, for each j, if $C_{i,j}$ is an encryption of V, the prover randomly generates the values $w''_j, u''_j, s_j \in_R \{1, \ldots, m-1\}$ and computes

$$\begin{aligned} A'_j &= s_j P, & B'_j &= s_j Q, \\ A''_j &= w''_j P + u''_j A_{i,j}, & B''_j &= w''_j Q + u''_j (B_{i,j} - 2V). \end{aligned}$$

He also computes

$$\begin{aligned} chall_j &= \mathcal{H}(A'_j, A''_j, B'_j, B''_j) \pmod m, \\ u'_j &= chall_j - u''_j \pmod m, \\ w'_j &= s_j - u'_j r_{i,j} \pmod m, \end{aligned}$$

where $\mathcal{H}$ is a cryptographic hash function like SHA-256 [7]. Recall that $r_{i,j}$ is the random integer taken to generate $C_{i,j}$. On the other hand, if $C_{i,j}$ is an encryption of $2V$, the prover will generate A''_j, B''_j as A'_j, B'_j and vice versa, taking into account that the computation of B'_j will involve V instead of $2V$. The generation of u'_j and w'_j will also be swapped with the generation of u''_j and w''_j, respectively.

After that, the prover computes $r_i = \sum_{j=1}^{k+1} r_{i,j} \pmod m$ and sends

$$A'_j, A''_j, B'_j, B''_j, u'_j, u''_j, w'_j, w''_j$$

for each j, $1 \leq j \leq k+1$, together with r_i to the verifier.

Verifier. For each j, $1 \leq j \leq k+1$, the verifier checks that

$$\begin{aligned} A'_j &= w'_j P + u'_j A_{i,j}, \\ A''_j &= w''_j P + u''_j A_{i,j}, \\ B'_j &= w'_j Q + u'_j (B_{i,j} - V), \\ B''_j &= w''_j Q + u''_j (B_{i,j} - 2V), \\ u'_j + u''_j &= \mathcal{H}(A'_j, A''_j, B'_j, B''_j). \end{aligned} \tag{1}$$

Then, in order to prove the voter has voted for only one candidate, the verifier aggregates $C_i = \sum_{j=1}^{k+1} C_{i,j}$ and uses r_i to reveal $C_i = (A_i, B_i)$. Then it checks that,

$$Rev_{r_i}(C_i) = B_i - r_i Q = (k+2)V.$$

All these verifications ensure that v_i has voted for just one candidate.

3.4 Multi-candidate Elections

Some elections allow voters to vote for more than one candidate or to choose up to a number of candidates. Our scheme can accommodate this kind of elections by slightly changing the ballot composition and the proof of correct composition.

Setup Phase. The system is configured and the information is published as described in Section 3.2. Moreover, the PS also publishes max and min, the maximum and minimum amount of candidates to be voted in each ballot.

Vote Casting Phase. A voter v_i generates a ballot in which f ciphertexts, $0 < min \leqslant f \leqslant max < k$, are an encryption of $2V$ (the chosen candidates), while the $k-f$ remaining ones are an encryption of V (non-chosen candidates). A blank vote is always permitted regardless of the value of min. Assuming that the chosen candidates are indexed between $k-f$ and k (with the blank subvote at position $k+1$), v_i generates:

$$\begin{aligned} C_{i,j} &= (r_{i,j}P, V + r_{i,j}Q) && \text{for } j \notin \{k-f, \ldots, k\}, \\ C_{i,l} &= (r_{i,l}P, 2V + r_{i,l}Q) && \text{for } k-f < l \leqslant k, \\ C_{i,k+1} &= (r_{i,k+1}P, r_{i,k+1}Q) && \text{and} \\ C_{i,aux} &= (r_{aux}P, (max-f)V + r_{aux}Q). \end{aligned}$$

If the voter v_i wants to cast a blank ballot, she will need to generate k ciphertexts encrypting V, $C_{i,k+1}$ encrypting $maxV$, and $C_{i,aux}$ encrypting $\mathcal{O}$:

$$\begin{aligned} C_{i,j} &= (r_{i,j}P, V + r_{i,j}Q) && \text{for } 1 \leqslant j \leqslant k, \\ C_{i,k+1} &= (r_{i,k+1}P, maxV + r_{i,k+1}Q) && \text{and} \\ C_{i,aux} &= (r_{aux}P, r_{aux}Q). \end{aligned}$$

After that, the ballot is signed by v_i.

When computing the proof of correct ballot composition, v_i now generates a proof that the cleartext of ballot $C_{i,k+1}$ is either $\mathcal{O}$ or $maxV$. Moreover, the ciphertext $C_{i,aux}$ is proven to encrypt a point in the set $\{\mathcal{O}, V, \ldots, (max - min)V\}$ [8]. Finally, she sends the values,

$$(C_{i,1}, \ldots, C_{i,k+1}), C_{i,aux},$$

their signature and the proofs to the PS. The PS will proceed as in a one-candidate election, but the verification of the proof has to take into account that the aggregated components ciphertext is computed as, $C_i = (A_i, B_i) = \sum_{j=1}^{k+1} C_{i,j} + C_{i,aux}$ so that,

$$Rev_{r_i}(C_i) = B_i - r_i Q = (k + max)V,$$

with $r_i = \sum_{j=1}^{k+1} r_{i,j} + r_{i,aux} \pmod m$. The new ranges of $B_{i,k+1}$ and $B_{i,aux}$ have to be also taken into account. Notice that when $min = max$ there is no need to generate $C_{i,aux}$, since it will always correspond to an encryption of $\mathcal{O}$.

Tallying Phase. Although in a multi-candidate election more than one candidate may receive more than $\lfloor n/2 \rfloor$ votes, the tallying phase remains the same except for the blank votes. When the PS computes $T_{k+1} = \sum_{i=1}^{z} C_{i,k+1}$, it will obtain $x_{k+1}V = (max+1)^{-1} Dec_d(T_{k+1})$. Thus, it will search for $x_{k+1}V$ in the table $\mathcal{T}$. Notice that the values $C_{i,aux}$ do not need to be aggregated nor decrypted.

4 Security

In this section, we prove that the presented e-voting system fulfills the security requirements enumerated in Section 1. Our proofs assume a one-candidate election. Their extension to multi-candidate elections is straightforward.

Authenticity. Only people in the electoral roll are able to cast a ballot since each ballot is digitally signed by the participant casting it. The electoral roll, which includes the public key of each voter, and the received ballots are publicly available on the BB. Thus, any one can check ballot signatures.

Unicity. Our system has to ensure that each voter has not voted more than once. As we have pointed out above, each ballot comes signed, so that two ballots cast by the same voter are easily linked through the public key that permits to verify their signature. The system must also ensure that the content of a ballot represents a vote for just one candidate. This is proven by the following two lemmata.

Lemma 1. *Assuming $\mathcal{H}$ a secure cryptographic hash function, there is no ciphertext $C_{i,j}$ able to pass the zero-knowledge proof unless it is an encryption of either V or $2V$.*

Proof. Let us assume $C_{i,j} = (A_{i,j}, B_{i,j}) = (r_{i,j}P, X + r_{i,j}Q)$ with $X \notin \{V, 2V\}$. According to the verifications that will be performed by the verifier (Eq. 1) the attacker has to generate four points A'_j, A''_j, B'_j, B''_j and four integers u'_j, u''_j, w'_j, w''_j satisfying

$$\begin{aligned} A'_j &= w'_j P + u'_j A_{i,j}, \;\; B'_j = w'_j Q + u'_j (B_{i,j} - V), \\ A''_j &= w''_j P + u''_j A_{i,j}, \; B''_j = w''_j Q + u''_j (B_{i,j} - 2V). \end{aligned}$$

One possibility is to generate first u'_j, u''_j, w'_j, w''_j at random and compute A'_j, A''_j, B'_j, B''_j according to the previous formulas. In that case, the verifier performs an additional checking requiring that $\mathcal{H}(A'_j, A''_j, B'_j, B''_j)$ outputs exactly $u'_j + u''_j$, which is unlikely because $\mathcal{H}$ is a secure hash function, so it is preimage resistant.

Another possibility is to generate $w'_j, u'_j, s_j, s'_j \in_R \{1, \ldots, m-1\}$ at random and compute

$$\begin{aligned} A'_j &= w'_j P + u'_j A_{i,j}, \; B'_j = w'_j Q + u'_j (B_{i,j} - V), \\ A''_j &= s_j P, \qquad\qquad\;\; B''_j = s'_j Q. \end{aligned}$$

Next, compute $chall_j = \mathcal{H}(A'_j, A''_j, B'_j, B''_j) \pmod m$ and $u''_j = chall_j - u'_j \pmod m$, and find a value w''_j such that:

$$A''_j = w''_j P + u''_j A_{i,j}, \; B''_j = w''_j Q + u''_j (B_{i,j} - 2V).$$

Let $X = 2V + tQ$. From the equality involving A''_k we get

$$w''_j = s_j - u''_j r_{i,k} \pmod m.$$

By substituting w''_k in the equality involving B''_k we get

$$s'_j = s_j + u''_j t.$$

If $s_j = s'_j$, the previous equality is satisfied if $t = 0$, in which case, $C_{i,j}$ would be an encryption of $2V$ or $u''_j = 0$ in which case, the output of $\mathcal{H}(A'_j, A''_j, B'_j, B''_j)$ should be exactly u'_j which is very unlikely. If $s_j \neq s'_j$, then it is required that $u''_j = (s'_j - s_j)t^{-1} \pmod m$ in which case, the output of $\mathcal{H}(A'_j, A''_j, B'_j, B''_j)$ should be exactly $(s'_j - s_j)t^{-1} + u'_j$ which is very unlikely.

First generating $w''_j, u''_j, s_j, s'_j \in_R \{1, \ldots, m-1\}$ and next computing

$$\begin{array}{ll} A'_j = s_j P, & B'_j = s'_j Q, \\ A''_j = w''_j P + u''_j A_{i,j}, & B''_j = w''_j Q + u''_j (B_{i,j} - 2V), \end{array}$$

leads to an equivalent situation. Hence, the claim follows.

Lemma 2. *Assuming Lemma 1 is true, a voter can only vote for one candidate.*

Proof. From Lemma 1, each $C_{i,j}$ is an encryption of either V or $2V$. Hence, $C_i = \sum_{j=1}^{k} C_{i,j}$ is an encryption of a point in the set $\{kV, (k+1)V, \ldots, 2kV\}$. Revealing C_i outputs $(k+1)V$ if, and only if, all the ciphertexts $C_{i,j}$ encrypts V except one of them, whose plaintext is $2V$.

Both lemmata remain true when changing the points V and $2V$ for any other point pair. Furthermore, they are both true if the amount of possible points increases.

Privacy. This property requires that the candidate chosen by each voter must remain secret. In our system the vote is encrypted under the EC-ElGamal cryptosystem, which is assumed to be a semantically secure cipher. Therefore, no information can be obtained from an encrypted ballot. Assuming the KSTP only deciphers the aggregated ballot, an attacker could only obtain information from the proof of correct ballot composition. That proof, given in Section 3.3, is composed of two parts. The first one proves that the cleartext in each of the ciphertexts $C_{i,j}$ is either V or $2V$. The second one is a proof to validate that just one candidate is voted in the ballot.

First, we will prove that the proof corresponding to the first part is zero-knowledge. This is formalized by showing that there exists a simulator that can produce a transcript that looks like a proper interaction in its interactive version. That transcript is easy to generate if the challenge was known in advance.

Lemma 3. *The proof correct ballot composition is zero-knowledge.*

Proof. Given a value $chall'_j$ a simulator can generate a proof for any ciphertext $C_{i,j} = (A_{i,j}, B_{i,j})$ by generating $u'_j, w'_j, w''_j \in_R \{1, \ldots, m-1\}$ at random and next computing $u''_j = chall'_j - u'_j$. After that, it is easy to generate the points:

$$\begin{array}{ll} A'_j = w'_j P + u'_j A_{i,j}, & B'_j = w'_j Q + u'_j (B_{i,j} - V), \\ A''_j = w''_j P + u''_j A_{i,j}, & B''_j = w''_j Q + u''_j (B_{i,j} - 2V). \end{array}$$

These values would satisfy the conditions given in Eq. 1, regardless of the cleartext of $C_{i,j}$. Hence, the claim follows.

In the second part of the proof, the prover reveals an integer r_i, computed as $r_i = \sum_{j=1}^{k+1} r_{i,j} \pmod{m}$. The value r_i reveals information about some $r_{i,l}$ if, and only if, all the other values $r_{i,j}$ with $j \neq l$ are known. Hence, no information about any $r_{i,l}$ is leaked and no ciphertext $C_{i,l}$ can be revealed.

Fairness. Assuming the KSTP behaves correctly, our system provides fairness because no vote is decrypted until the vote casting phase has ended.

Verifiability. Our proposal offers universal verifiability because any entity can check that all the ballots are cast by people who appear in the electoral roll by verifying its digital signature. Additionally, the proof of correct vote composition can also be verified by any entity who can also aggregate the received ballots by itself and check that the PS performed this operation properly. Finally, the aggregated ciphertexts are decrypted verifiably by the KSTP.

Coercion-Resistance. Our proposal is compatible with several coercion-resistance solutions which are able to send a fake vote without the coercer noticing it. Solutions like [13] can be adapted and included in our protocol.

5 Performance Analysis and Conclusion

Remote voting systems should provide both security and efficiency at the same time. The cost of some parts of the system are more critical than the cost of others. For example, in our proposal, during the vote casting phase, the voters are required to prove in zero-knowledge a proper composition of their ballots. These proofs are validated by the PS. It is desirable that the generation and verification of these proofs are as fast as possible but, since the vote casting phase usually takes a long time, and the proofs can be validated while the ballots are being received, these validations are not a problem as long as the PS can handle them.

In contrast, the best efficiency is required for the tallying phase. This is because a long tallying phase would cause a delay in the publication of the election result.

In our performance analysis, we have first compared the encryption and decryption costs of ElGamal and EC-ElGamal cryptosystems. Table 1 shows the results for different security levels. It can be seen that EC-ElGamal is more efficient than ElGamal. The difference in cost between the two cryptosystems increases with the security level. From the obtained results, we can state that by using EC-ElGamal we can achieve a 50% efficiency gain, at least.

Next, we have compared the current proposal with an implementation of a similar remote voting scheme presented in [3]. The system in [3], which will be called *redundant scheme* from now on, is also implemented over elliptic curves.

Table 1. Comparison between encryption and decryption times (ms) of ElGamal and EC-ElGamal.

Bits	**ElGamal**		**EC-ElGamal**	
	Encryption	**Decryption**	**Encryption**	**Decryption**
1024 — 160	1.589	0.879	0.897	0.484
2048 — 224	11.822	6.201	1.311	0.662
3072 — 256	61.294	32.533	1.650	0.843

It includes a redundancy system in order to decrease the time required for obtaining the election result from the aggregated ciphertext. In the simulation of that scheme, we have used homomorphic packages of 500, 1000 and 100000 aggregated ballots, depending on the number of voters (5000, 10000 and 1000000, respectively). Our current proposal is able to homomorphically aggregate all the ballots into a single ciphertext with just a slight increase in the preprocessing time. Hence, the presented simulations create packages with 5000, 10000 and 1000000 aggregated ballots.

Although the redundant scheme is more efficient at ballot creation, Table 2 shows it is twice as fast as the current one, in most phases it requires more time than the new proposal. In the redundant scheme, a ballot is composed of $2 \cdot \lceil k/4 \rceil$ ciphertexts while the current proposal contains k, being k the number of candidates. However, this fact is not enough to consider the redundant scheme is better. Table 2 shows that the zero-knowledge proof validation is twice slower in the redundant scheme than in the current proposal. Proof generation requires the computation of $47 \cdot \lceil k/4 \rceil + 18 \simeq 12 \cdot k$ multiplications in the redundant scheme while the current proposal only needs $6 \cdot k$. On the other hand, the verification requires $56 \cdot \lceil k/4 \rceil + 20 \simeq 14 \cdot k$ multiplications in the redundant scheme, while the current scheme only needs $8 \cdot k$. The time required for ballot validation is more critical. This is because each voter has to create just one ballot while the PS has to validate all of them.

Table 2. Time (ms) spent to compose and prove a ballot (voter) and verify it (PS).

k	**Current scheme**			**Redundant scheme**		
	Compose	**Prove**	**Verify**	**Compose**	**Prove**	**Verify**
4	3.428	10.572	14.481	1.710	29.911	33.099
16	13.668	42.385	56.630	6.789	95.844	106.350
64	54.582	169.357	224.867	26.967	359.720	400.342

Finally, we have also compared the time required for decryption and tally. Table 3 shows the current proposal is faster in both cases. In the new proposal, decryption only requires the computation of k multiplications. In contrast, in the redundant scheme, it involves $\frac{z}{t}\lceil k/4 \rceil$ multiplications, where z is the number of ballots and t is the capacity of the homomorphic packages (500, 1000 and

100000, as mentioned above). On the other hand, obtaining the result from the aggregated cleartext, in the new method, involves a point multiplication, a point subtraction and a query to the hash table $\mathcal{T}$. In contrast, the redundant scheme involves n subtractions and $(n+1)$ hash queries (see [3] for a detailed description).

Table 3. Time (ms) spent to decrypt and tally the aggregated ballots.

k	Voters	Current scheme		Redundant scheme	
		Decryption	Tally	Decryption	Tally
4	**5000**	1.760	0.273	8.649	60.494
	10000		0.273		120.721
	1000000		0.424		12 084.549
16	**5000**	6.856	1.079	34.602	60.139
	10000		1.092		124.119
	1000000		1.610		12 218.945
64	**5000**	27.434	4.273	137.509	61.470
	10000		4.260		121.982
	1000000		6.284		12 245.186

As a conclusion, we state that the remote voting protocol presented in this paper is efficient and, therefore, suitable for its use in real remote voting processes. All the simulations have been implemented in *C++*, using the *Crypto++* library and have been run on a PC with an *Intel Core i5 650 3.2GHz* CPU and 4GB of RAM, running a *Debian 7.8 Wheezy* OS. Additionally, the times shown in Tables 2 and 3 have been obtained using 160 bit elliptic curves.

Acknowledgement. Research of the authors was supported in part by grants MTM2013-46949-P (Spanish Ministerio de Ciencia e Innovación), 2014SGR-1666 (Generalitat de Catalunya) and IPT-2012-0603-430000 (Spanish Ministerio de Economía y Competitividad).

References

1. Adida, B.: Helios: Web-based open-audit voting. In: USENIX Security Symposium. vol. 17, pp. 335–348 (2008)
2. Adida, B., Pereira, O., Marneffe, O.D., Quisquater, J.J.: Electing a university president using open-audit voting: Analysis of real-world use of helios. In: Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE) (2009)
3. Cerveró, M., Mateu, V., Miret, J., Sebé, F., Valera, J.: An efficient homomorphic e-voting system over elliptic curves. In: EGOVIS, LNCS, vol. 8650, pp. 41–53 (2014)
4. Chaum, D.: Security without identification: transaction systems to make big brother obsolete. Communications of the ACM 28(10), 1030–1044 (1985)
5. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Advances in Cryptology – CRYPTO, LNCS, vol. 740, pp. 89–105 (1993)

6. Cohen, J.D., Fischer, M.J.: A robust and verifiable cryptographically secure election scheme. In: 26th Annual Symposium on Foundations of Computer Science (FOCS). pp. 372–382 (1985)
7. US Department of Commerce, N.I.o.S., Technology: Secure hash standard. Federal Information Processing Standard# 180-2 56, 57–71 (1994)
8. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Advances in Cryptology – CRYPTO. LNCS, vol. 839, pp. 174–187 (1994)
9. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Proceedings of CRYPTO 84 on Advances in Cryptology. pp. 10–18. Springer-Verlag New York, Inc. (1985)
10. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Advances in cryptology – AUSCRYPT, LNCS, vol. 718, pp. 244–251 (1993)
11. Groth, J.: Non-interactive zero-knowledge arguments for voting. In: Applied Cryptography and Network Security. pp. 467–482. Springer (2005)
12. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Advances in Cryptology – EUROCRYPT. LNCS, vol. 1807, pp. 539–556 (2000)
13. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Proceedings of the 2005 ACM workshop on Privacy in the electronic society. pp. 61–70. ACM (2005)
14. Katz, J., Myers, S., Ostrovsky, R.: Cryptographic counters and applications to electronic voting. In: Advances in Cryptology – EUROCRYPT, LNCS, vol. 2045, pp. 78–92 (2001)
15. Kiayias, A., Yung, M.: Self-tallying elections and perfect ballot secrecy. In: Proceedings of Public Key Cryptography. pp. 141–158. Springer (2002)
16. Mateu, V., Miret, J.M., Sebé, F.: Verifiable encrypted redundancy for mix-type remote electronic voting. In: EGOVIS 2011. LNCS, vol. 6866, pp. 370–385 (2011)
17. Ohkubo, M., Miura, F., Abe, M., Fujioka, A., Okamoto, T.: An improvement on a practical secret voting scheme. In: Information Security Workshop - ISW'99, LNCS, vol. 1729, pp. 225–234 (1999)
18. Peng, K.: An efficient shuffling based evoting scheme. Journal of Systems and Software 84(6), 906–922 (2011)
19. Peng, K., Aditya, R., Boyd, C., Dawson, E., Lee, B.: Multiplicative homomorphic e-voting. In: Progress in Cryptology-INDOCRYPT, LNCS, vol. 3348, pp. 61–72 (2004)
20. Peng, K., Bao, F.: Efficient multiplicative homomorphic e-voting. In: Information Security Conference - ISC 2010, LNCS, vol. 6531, pp. 381–393 (2011)
21. Sebé, F., Miret, J.M., Pujolàs, J., Puiggali, J.: Simple and efficient hash-based verifiable mixing for remote electronic voting. Computer Communications 33(6), 667–675 (2010)
22. Silverman, J.H.: The Arithmetic of Elliptic Curves. Springer-Verlag, 2nd edition edn. (2009)
23. Yi, X., Okamoto, E.: Practical internet voting system. Journal of Network and Computer Applications 36(1), 378–387 (2013)