# An Elliptic Curve Based Homomorphic Remote Voting System

M.A. Cerveró V. Mateu J.M. Miret F. Sebé J. Valera
Dept. Matemàtica, Universitat de Lleida. Jaume II, 69, 25001, Lleida, Spain.
{mcervero, vmateu, miret, fsebe, jvalera}@matematica.udl.cat

***Abstract*—A remote voting system allows participants to cast their ballots through the Internet. Remote voting systems based on the use of homomorphic public key cryptography have proven to be a good option for carrying out simple elections with a reduced amount of candidates. In this paper, we present a new system that makes use of the additive homomorphic capabilities of the Elliptic Curve ElGamal (EC-ElGamal) cryptosystem. All the stages of the system are described together with an experimental analysis section which provides an assessment on the type of election our system would be suitable for.**

***Index Terms*—Electronic Voting, Elliptic Curve Cryptography, Knapsack Problem**

## I. Introduction

*Electronic voting* (*e-voting*) refers to the use of advanced technology to election processes. E-voting systems reduce the economic cost of an election and increase the speed and accuracy of vote tallying. An e-voting system allowing voters to cast their ballots through the Internet is called a *remote voting system*. The security provided by such a system should include, at least, the following features:

- *Authentication*: only people in the electoral roll can vote.
- *Unicity:* every participant can vote once at most.
- *Privacy:* votes can not be related to voter identities.
- *Fairness:* no partial results can be revealed before the end of the voting period.
- *Verifiability:* correctness of the process can be checked.
- *Uncoercibility:* nobody can prove that a voter voted in a particular way.

The previous security requirements are obtained by making use of advanced cryptographic techniques. Current remote voting systems can be classified into three main paradigms: *blind signature-based*, *mix-type* and *homomorphic tallying*.

In the *blind signature-based* paradigm [1]–[3], a voter authenticates against a trusted authority which is responsible for checking that the voter appears in the electoral roll and she has not voted before. In that case, voter's ballot (the encrypted vote) is blindly signed by that authority. The *polling station* only accepts ballots that have been properly signed by the authority. When the voting period is concluded, ballots are decrypted and tallied.

In the *mix-type* paradigm [4]–[9] a voter casts her ballot after having signed it. Once the voting period has ended, the polling station shuffles and re-encrypts (mixes) the collected ballots in order to break the relation between each ballot and the identity of the voter who cast it. After that, the mixed ballots are decrypted and tallied.

In *homomorphic tallying* schemes [10]–[15], participants cast their ballots encrypted under some public key cryptosystem having a homomorphic property. The received ballots are homomorphically aggregated by the polling station into a single or a set of ciphertexts whose decryption will show the result of the election. These systems require the votes to be coded in such a way that the final tally can be recovered from the cleartext of the aggregated ballots. Also, each voter has to prove in *zero-knowledge* that her ballot has been composed properly.

It is well known that homomorphic tallying systems do not scale well as the number of candidates increases. Despite their benefits regarding decryption (just one decryption is needed), homomorphic tallying systems need an additional decoding step in order to get the amount of votes for each candidate from an aggregated ballot cleartext. The method employed for coding votes should permit to get the election result at a reasonable processing time. It is also important to be able to manage a large enough amount of candidates and voters.

### A. Contribution and Plan of this Paper

In homomorphic tallying remote voting systems, the ballots are encrypted using some public key cryptosystem. In elections with few candidates, the Elliptic Curve ElGamal (EC-ElGamal) cryptosystem turns out to be more efficient that ElGamal implemented over a multiplicative group. ElGamal requires 1024 bit long keys while EC-ElGamal achieves an equivalent security employing just 160 bits. Hence, EC-ElGamal provides better memory and computational costs. However, in elections with a large amount of candidates, EC-ElGamal becomes not as efficient as ElGamal.

In this paper, we present an e-voting system belonging to the homomorphic tallying paradigm based on the use of the Elliptic Curve ElGamal (EC-ElGamal) cryptosystem. Its vote coding system allows a large number of candidates while offering a good performance at the decoding step.

The paper is structured as follows: Section II presents some basic concepts of elliptic curve cryptography and the EC-ElGamal cryptosystem. Section III provides the description of of the proposed e-voting system, while Section IV emphasizes a special case: the Referendum. Then, Section V is dedicated to prove the security of the system and Sections VI and VII are devoted to experimental results, conclusions and future

ISBN: 978-84-9717-323-0

work. Finally, Annex A presents the elliptic curves used in the experimental part of this work.

## II. Preliminaries

An elliptic curve $E$ defined over a finite field $\mathbb{F}_p$ is an equation of the form

$$E : y^2 = x^3 + ax + b, \tag{1}$$

with $4a^3 + 27b^2 \neq 0$. The set of points of the curve, denoted $E(\mathbb{F}_p)$, is composed of the points $(x, y) \in \mathbb{F}_p \times \mathbb{F}_p$ satisfying Equation (1) together with the point at infinity, $\mathcal{O}$. The set $E(\mathbb{F}_p)$ can be endowed with an abelian group structure, having $\mathcal{O}$ as the identity element, by means of the chord-tangent method [16]. This method provides an operation for adding curve points. Given two curve points $P$ and $Q$, the *elliptic curve discrete logarithm problem* (ECDLP) consists of finding an integer $d$ satisfying $Q = d \cdot P$. The ECDLP is computationally hard when the cardinality of $E(\mathbb{F}_p)$ has a large prime factor. The assumed intractability of the ECDLP has led to the design of public key cryptosystems constructed over the group of points of an elliptic curve [17].

### A. The Elliptic Curve ElGamal Cryptosystem

The *Elliptic Curve ElGamal* (EC-ElGamal) cryptosystem is composed of the following procedures.

1) *Set up:* A finite field $\mathbb{F}_p$ is first selected. After that, two integers $a$ and $b$ defining an elliptic curve $E$ over $\mathbb{F}_p$ (see Eq. 1) are chosen so that the cardinality of $E(\mathbb{F}_p)$ has a large prime factor $q$. Finally, a point $P$ of order $q$ is taken as a generator of the order $q$ cyclic subgroup of $E(\mathbb{F}_p)$. The values $(p, E, P, q)$ are made public.
2) *Key generation:* Given the set up parameters, a private key is generated by randomly choosing an integer $d$ in the range $[1, q-1]$. Next, its related public key $Q$ is computed as $Q = d \cdot P$.
3) *Encryption:* A plaintext $M$ consisting of a point of $E(\mathbb{F}_p)$ is encrypted under public key $Q$ by computing

$$E_Q(M) = C = (A, B) = (r \cdot P,\ M + r \cdot Q)\,, \tag{2}$$

where $r$ is an integer selected randomly in the range $[1, q-1]$. Each encryption makes use of a different random $r$, whose value must be kept secret.
4) *Decryption:* If the private key $d$ is known, a ciphertext $C$ can be decrypted by computing

$$D_d(C) = B - d \cdot A.$$

The cleartext $M$ is obtained as a result.

The EC-ElGamal cryptosystem has an homomorphic property. Let $C_1 = (A_1, B_1)$ and $C_2 = (A_2, B_2)$ be two ciphertexts encrypting $M_1$ and $M_2$, respectively. They are aggregated by computing,

$$C = C_1 + C_2 = (A_1 + A_2,\ B_1 + B_2).$$

The decryption of $C$ will provide $M_1 + M_2$ as a result.

## III. Our Proposal

The presented remote voting system is composed of the following parties:

- *Polling Station:* It coordinates the system. It is responsible for collecting the votes and publishing the final election result. When all the ballots have been collected and aggregated, it asks the *key storage trusted party* to decrypt the aggregated ballots. The received ballots are published on some publicly accessible bulletin board for verifiability purposes.
- *Participants:* They are voters able to emit a vote.
- *Key Storage Trusted Party (KSTP):* It is responsible for generating and storing the election private key and publishing the election public key material. When required, it will decrypt the ciphertexts containing the aggregation of cast ballots. The election public key should be certified to ensure its authenticity.

Next, the different stages of an election are explained in detail.

### A. Set Up

Let us consider an election with $m$ candidates. The collected ballots will be aggregated into packages containing $n$ votes each.

Let $t$ be an integer so that $t \leq m$ (for the sake of simplicity, we are assuming that $t \mid m$). The KSTP generates an elliptic curve $E$ defined over a finite field $\mathbb{F}_p$ so that the cardinality of $E(\mathbb{F}_p)$ has $t$ large prime factors. That is,

$$\#E(\mathbb{F}_p) = h \prod_{i=1}^{t} q_i,$$

with $h$ being a small integer and each $q_i$ being a large prime (at least 160 bits long). Next, an order $q = \prod_{i=1}^{t} q_i$ point $P$ is chosen.

Since $\#E(\mathbb{F}_p)$ has $t$ large prime factors, the group of points $E(\mathbb{F}_p)$ has $t$ large cyclic subgroups. Next, a generator $P_i$ for each subgroup is generated by computing,

$$P_i = \Big( \prod_{\substack{1 \leq j \leq t \\ j \neq i}} q_j \Big) \cdot P, \tag{3}$$

so that $\mathrm{ord}(P_i) = q_i$.

Then, the KSTP creates the election private key $d$ by choosing its value randomly in $[1, q-1]$, computes the election public key $Q = d \cdot P$ and publishes all the previous parameters (the private key is stored privately in a safe place).

After that, the KSTP generates $m$ points $\{M_1, \ldots, M_m\}$, each one representing a different candidate. Being $s = m/t$, the points are computed as shown in Table I. Note that points $M_1, \ldots, M_s$ are in the subgroup of $E(\mathbb{F}_p)$ generated by $P_1$, points are $M_{s+1}, \ldots, M_{2s}$ in the subgroup generated by $P_2$, and so on.

Table I
GENERATION OF A CURVE POINT FOR EACH CANDIDATE.

| **Base Point $\mathbf{P_1}$** | **Base Point $\mathbf{P_2}$** | … | **Base Point $\mathbf{P_t}$** |
|---|---|---|---|
| $M_1 = P_1$ | $M_{s+1} = P_2$ | | $M_{(t-1)s+1} = P_t$ |
| $M_2 = (n+1) \cdot P_1$ | $M_{s+2} = (n+1) \cdot P_2$ | … | $M_{(t-1)s+2} = (n+1) \cdot P_t$ |
| … | … | | … |
| $M_s = (n+1)^{s-1} \cdot P_1$ | $M_{2s} = (n+1)^{s-1} \cdot P_2$ | | $M_{ts} = (n+1)^{s-1} \cdot P_t$ |
| Candidates from 1 to $s$ | Candidates from $s+1$ to $2s$ | | Candidates from $(t-1)s+1$ to $ts = m$ |

## B. Voting

The voting process starts when a participant $\mathcal{P}$ wants to vote by electing a candidate. It is composed of four steps: *candidate choice*, *electoral roll checking*, *vote coding verification* and *vote packing*.

*1) Candidate Choice:* Let $M_{c_\mathcal{P}}$ be the curve point representing the choice of participant $\mathcal{P}$ who emits her vote by performing the following steps:

- Encrypt $M_{c_\mathcal{P}}$ under the public key $Q$. The encrypted vote $C_\mathcal{P} = E_Q(M_{c_\mathcal{P}})$ is generated as shown in Eq. (2).
- Compute the signature of the encrypted vote.
- Send $C_\mathcal{P}$ together with its signature to the polling station.

*2) Electoral Roll Checking:* When the polling station receives a ballot, it first checks the validity of its digital signature. Next, it checks that the voter who has cast it appears in the electoral roll and that she has not voted before. In that case, the voter is asked to prove that her ballot was correctly generated. This is done as described next.

*3) Vote Coding Verification:* The participant has to demonstrate that the cleartext of the ballot she is casting corresponds to a point representing one of the candidates. This demonstration is performed by means of a *zero knowledge proof*. It consists of a data exchange between the *Prover* (participant $\mathcal{P}$) and the *Verifier* (*polling station*). This proof does not leak any information about the actual choice of the voter. The following proof is an adaptation of the proof presented in [18] for its use on elliptic curve cryptography.

The *Prover* has to prove that her vote $C_\mathcal{P} = (A_\mathcal{P}, B_\mathcal{P}) = (r_\mathcal{P} \cdot P, M_{c_\mathcal{P}} + r_\mathcal{P} \cdot Q)$ is an encryption of one of the points in the set $\mathcal{M} = \{M_1, \ldots, M_m\}$ (the points of $E(\mathbb{F}_p)$ representing each of the candidates). In order to do that, the *Prover* generates the points $A_k, B_k$, for $1 \leqslant k \leqslant m$:

$$\begin{array}{ll} A_k = w_k \cdot P + u_k \cdot A_\mathcal{P}, & \forall k \neq c_\mathcal{P}; \\ A_{c_\mathcal{P}} = s \cdot P, & \\ B_k = w_k \cdot Q + u_k \cdot (B_\mathcal{P} - M_k), & \forall k \neq c_\mathcal{P}; \\ B_{c_\mathcal{P}} = s \cdot Q, & \end{array}$$

where $w_k, u_k, s \in [1, q-1]$, are random values. Next, the *Prover* computes

$$\begin{array}{l} chall = \mathcal{H}(A_1, A_2, \ldots, A_m, B_1, B_2, \ldots, B_m), \\ u_{c_\mathcal{P}} = chall - \sum_{k \neq c_\mathcal{P}} u_k, \\ w_{c_\mathcal{P}} = s - u_{c_\mathcal{P}} r_\mathcal{P}, \end{array}$$

where $\mathcal{H}$ is some cryptographic hash function like SHA256 [19]. Finally, the *Prover* sends $A_k, B_k, u_k, w_k$ for $1 \leq k \leq m$ to the *Verifier*.

The *Verifier* checks that

$$\begin{array}{ll} A_k = w_k \cdot P + u_k \cdot A_\mathcal{P}, & \forall k \in [1, m], \\ B_k = w_k \cdot Q + u_k \cdot (B_\mathcal{P} - M_k), & \forall k \in [1, m], \\ chall = \sum_{k=1}^m u_k, & \end{array}$$

with *chall* computed as $\mathcal{H}(A_1, A_2, \ldots, A_m, B_1, B_2, \ldots, B_m)$. This verification ensures that the voter has voted for a point in set $\mathcal{M}$. If all the checkings are satisfied, the signed ballot and the data required to verify it was properly generated are published on the bulletin board so that any external entity can check its correctness.

*4) Vote Packing:* The verified votes are homomorphically aggregated into packages. Each package is an aggregation of up to $n$ ballots.

Let us consider a set of ballots $\mathcal{C} = \{C_{\mathcal{P}_1}, \ldots, C_{\mathcal{P}_n}\}$ that will be aggregated into package $S_\ell$. Package $S_\ell$ is generated as:

$$S_\ell = \sum_{j=1}^{n} C_{\mathcal{P}_j}, \tag{4}$$

where $C_{\mathcal{P}_j} = \left( r_{\mathcal{P}_j} \cdot P,\ M_{c_{\mathcal{P}_j}} + r_{\mathcal{P}_j} \cdot Q \right)$. Hence, package $S_\ell$ is of the form,

$$S_\ell = \left( \sum_{j=1}^{n} r_{\mathcal{P}_j} \cdot P,\ \sum_{k=1}^{m} x_k \cdot M_k + \sum_{j=1}^{n} r_{\mathcal{P}_j} \cdot Q \right), \tag{5}$$

where $x_k$ is the number of votes for the candidate $k$ in this package and $\sum_{k=1}^m x_k = n$ is the capacity of the package.

## C. Vote Opening

Once the election has finished, it is time to decrypt the ballots and tally the votes. This process can be divided into four steps: *decryption*, *unpacking*, *scrutiny* and *publication*.

*1) Decryption:* When the election is finished, the polling station has a set of aggregated packages that have to be decrypted. The KSTP is asked to decrypt them.

*2) Unpacking:* After decryption, the polling station has to obtain the amount of votes for each candidate from the cleartext of each package. A decrypted package is of the form $\widehat{S}_\ell = \sum_{k=1}^m x_k \cdot M_k$, or equivalently,

$$\widehat{S}_\ell = \sum_{i=1}^{t} \sum_{k=1}^{s} x_{k+(i-1)s} (n+1)^{k-1} \cdot P_i, \tag{6}$$

where $\sum_{k=1}^m x_k = n$ and $ts = m$.

$\widehat{S}_\ell$ is decoded as follows:

1) For each base point $P_i$, compute $z_i = \prod_{1 \leq j \leq t, j \neq i} q_j$:

$$\widehat{S}_{\ell,i} = z_i \cdot \widehat{S}_\ell = \sum_{k=1}^{s} x_{k+(i-1)s} (n+1)^{k-1} \cdot (z_i \cdot P_i). \tag{7}$$

2) For each $i$, compute the values $x_{k+(i-1)s}$ for $1 \leq k \leq s$ which satisfy equation 7. This bounded discrete logarithm can be solved as a knapsack problem using the *Meet in the Middle* (MITM) algorithm as described next. In a preprocessing phase, compute $\sum_{k=s/2+1}^{s} x_k (n+1)^{k-1} \cdot (z_i \cdot P_i)$ for all the feasible combinations of $x_k$

values (those whose addition is not greater than $n$), and store each resulting point, together with the related $x_k$ values in a hash table.

Also in a preprocessing phase, compute $\sum_{k=1}^{s/2} x_k(n+1)^{k-1} \cdot (z_i \cdot P_i)$ for each feasible combination of $x_k$ values and store each result, together with the related $x_k$ values, in an array.

At decoding, each point $R$ in the array is taken and subtracted from $\widehat{S}_{\ell,i}$,

$$\widehat{S}'_{\ell,i} = \widehat{S}_{\ell,i} - R.$$

If $\widehat{S}'_{\ell,i}$ is in the hash table, we are done. In that case, the values $x_{k+(i-1)s}$ for $1 \leq k \leq s$ are obtained from the values stored together with $R$ (in the array) and $\widehat{S}'_{\ell,i}$ (in the hash table). Notice that if $s = 1$, the algorithm can directly cast $\widehat{S}_{\ell,i}$ against the hash table.

The explained MITM algorithm achieves a good balance between computational cost and memory consumption. If no precomputed data were used, the required computing time would be too large. On the other hand, precomputing and storing all the feasible combinations would be unaffordable in terms of memory storage requirements.

*3) Scrutiny:* When each package $\widehat{S}_\ell$ has been decoded, the polling station adds all the votes to finally scrutiny the election result. The total amount of votes for candidate $k$ is $\sum_{\ell=1}^{\mathcal{L}} x_k^\ell$, where $\mathcal{L}$ is the total number of packages and $\{x_1^\ell, \ldots, x_m^\ell\}$ are the values obtained from package $\widehat{S}_\ell$.

*4) Publication:* Finally, the election result is published on the bulletin board. At the end of the election, the bulletin board contains all the information needed to verify the correctness of the whole process. That is,

1) The result of the election (amount of votes for each candidate).
2) The electoral roll (name and public key of each participant).
3) The received ballots together with their digital signature and proof of correct composition.
4) Each aggregated package, $S_\ell$, together with its cleartext $\widehat{S}_\ell$, and the amount of votes it contains for each candidate.

## IV. Specific Case - Referendum

A *Referendum* is an election in which the voters can vote for *yes*, *no*, or *blank*. Next, we will show that such an election can be implemented very efficiently.

*1) Set Up:* We propose to choose an elliptic curve $E$ over a finite field $\mathbb{F}_p$ whose group order has $t = 3$ large prime factors $q_1, q_2, q_3$. Hence, $\#E(\mathbb{F}_p) = h \cdot q_1 \cdot q_2 \cdot q_3$, with $q = q_1 \cdot q_2 \cdot q_3$ being a 480 bits long integer. Finally, we take an order $q$ point $P$.

Since there are three possible options (candidates), we generate the following points: $P_1 = q_2 \cdot q_3 \cdot P$, $P_2 = q_1 \cdot q_3 \cdot P$, $P_3 = q_1 \cdot q_2 \cdot P$, satisfying that $ord(P_i) = q_i$. This way, we can code each option in a different base point, so that $s = 1$. Table II shows the three options represented by those base points.

Table II
The points representing the options in a *Referendum*.

| Option ***Yes*** | Option ***No*** | Option ***Blank*** |
|---|---|---|
| $M_1 = P_1$ | $M_2 = P_2$ | $M_3 = P_3$ |

*2) Unpacking:* Since $s = 1$, the unpacking process can be carried out in a very fast way. In the preprocessing phase of the MITM algorithm, we store all the possible values for each option $\{0 \cdot P_i, 1 \cdot P_i, \ldots, n \cdot P_i\}$ in the hash table so that the unpacking operation for each choice can be solved through a single hash lookup.

## V. Security Analysis

In this section we show how the proposed system achieves the security requirements enumerated in Section I.

*1) Authentication:* Each ballot is digitally signed by the participant who casts it. Hence, the polling station can authenticate the voter and check that she appears in the electoral roll. Moreover, the electoral roll and the received ballots are made publicly available on the bulletin board so that any entity can check that all the ballots have been cast by an authenticated participant.

*2) Unicity:* Unicity is composed of two requirements:

- The system must ensure that every voter votes only once.
- The system must ensure that each ballot contains only one vote. That is, a ballot can only encrypt a single point of list $\mathcal{M}$.

The first item is addressed by keeping a register of the voters that have already voted. If any participant tried to cast two or more ballots, the system would only accept the first one. The second item is ensured by means of the zero knowledge proof of ballot correct composition (see Section III-B3).

*3) Privacy:* Privacy of the choice made by a participant holds on the following facts:

- All the votes are encrypted using the EC-ElGamal cryptosystem, so that no information can be obtained from an encrypted ballot.
- All the encrypted votes are homomorphically packed, and only the resulting packages are decrypted. As a result, the voter and her choice are decoupled.
- Only the aggregated packages are decrypted. This is achieved by considering the KSTP is a trusted party which acts honestly.
- The proof needed to ensure that a ballot was correctly composed is *zero knowledge*. Hence, no information leaks from it.

*4) Fairness:* Assuming a correct praxis of the KSTP, no vote is decrypted before the opening stage, which takes place after the ending of the voting period.

*5) Verifiability:* The verifiability of our system is based in four points:

- The electoral roll is public and all the received (signed) ballots are also made public. Hence, any entity can check all the ballots come from an authenticated participant.
- Each *zero knowledge proof* of correct ballot composition is published on the bulletin board so that it can be checked by any entity which will get convinced that each ballot is coding a single vote.
- The homomorphic packing operation can be performed by any entity and next check that the obtained packages correspond to those published on the bulletin board.
- The decryption carried out by the KSTP can be performed verifiably [20].

Our proposal offers end-to-end verifiability: the correctness of the whole process can be verified by everyone.

*6) Uncoercibility:* Uncoercibility can be provided by applying any coercion-resistance solution like [21].

## VI. Experimental Results

The most time consuming part of the proposed system is given by the unpacking step. Hence, we have developed a test program to check the time and memory consumption of the MITM algorithm proposed for solving that step. The program has been implemented in *C++* using the library *Crypto++* and has been executed in a PC with an *Intel Core i5 650 3.2GHz* CPU with 6GB of RAM running *Debian 8.0 Jessie* as OS. Table III shows the data extracted during the tests using elliptic curves with 160, 320 and 480 bits long cardinalities (the used elliptic curves are shown in Annex A). The columns *Preprocess time* and *Memory* concern to the time and memory consumption of an unpacking operation.

Table III
Time and memory consumption using 160, 320 and 480 bits elliptic curves with packages of 200 votes.

| EC (bits) | #Base Pnts. | #Cands. Base Pnt. | Preprocess time (s) | Unpacking time (s) | Memory (MB) |
|---|---|---|---|---|---|
| 160 | 1 | 4 | 4.474 | 0.089 | 4.337 |
| | | 5 | 225.768 | 0.081 | 148.895 |
| 320 | 2 | 4 | 81.920 | 0.196 | 8.674 |
| | | 5 | 4 254.340 | 0.175 | 297.791 |
| 480 | 3 | 4 | 243.798 | 0.286 | 13.010 |
| | | 5 | 12 571.200 | 0.251 | 446.686 |

As we can see, the most time consuming part corresponds to the generation of the preprocessed data, which can be performed some days before the election takes place. This preprocessed data permits to unpack packages at a very reduced time. This last operation has to be fast because it determines the delay between the end of the voting period and the publication of the results. Table III shows that the proposed system is able to unpack packages with 200 votes and 15 candidates very efficiently (see the last row, corresponding to 5 candidates per base point). Furthermore, the memory needed to store the data generated during the preprocessing has a reasonable size which can be perfectly stored by any commodity PC.

We have also analyzed the time and memory consumption in the Referendum case, which requires the use of an elliptic curve with a 480 bits long cardinality and 3 cyclic subgroups. Table IV highlights the efficiency of our system to resolve elections with 3 candidates, needing a little more than 3 milliseconds to unpack a 200 votes package. The memory requirements and preprocessing time are negligible.

Table IV
Time and memory consumption for the *Referendum* case with packages of 200 votes.

| Preprocess time (s) | Unpacking time (s) | Memory (MB) |
|---|---|---|
| 0.599 | 0.003 | 0.064 |

Furthermore, our system can deal with larger packets. Table V shows the time and memory consumption when working with packets aggregating 1000000 votes. Although the preprocessing time and the memory requirements increase, the time required for unpacking keeps constant for any package size.

Table V
Time and memory consumption for the *Referendum* with packages of 1000000 votes.

| Preprocess time (s) | Unpacking time (s) | Memory (MB) |
|---|---|---|
| 3 395.550 | 0.003 | 320.435 |

By comparing our referendum system with that presented by Peng et al. [15], we can see that our unpacking algorithm is much more efficient than that in [15] (also implemented in *C++* using *Crypto++* and executed in the same PC). The proposal in [15] is implemented using the multiplicative homomorphic property of ElGamal cryptosystem over a multiplicative group. When using a 1024 bits public key, it can only manage packages of up to 440 votes, while our proposal can manage much bigger packages, as it can be seen in Table V. Moreover, the system in [15] requires 0.022 seconds to decode a 440 votes package while our system can perform an equivalent operation employing only 0.003 seconds.

## VII. Conclusion and Future Work

A new e-voting system that makes use of the EC-ElGamal cryptosystem has been proposed. The new proposal makes use of a MITM algorithm to unpack aggregated ballots at a high speed. Our system can be used in an election with a large amount of candidates. The experiments carried out have shown that our proposal is faster than the multiplicative homomorphic ElGamal cryptosystem proposed by Peng et al. [15], in the case of referendum type elections.

As future research, we will investigate techniques to further reduce the time devoted to ballot unpacking.

## Acknowledgments

Research of the authors was supported in part by grants MTM2010-21580-C02-01 (Spanish Ministerio de Ciencia e

Innovación), 2014SGR-1666 (Generalitat de Catalunya) and IPT-2012-0603-430000 (Spanish Ministerio de Economía y Competitividad).

## Appendix

In this appendix we show the elliptic curves used in our experiments. Finding elliptic curves with a given cardinality is a hot topic of research. Several algorithms have been proposed to this end. The most widely known was proposed by Atkin and Morain [22] but there exist other approaches like that proposed by Agashe et al. [23], or that by Bröker et al. [24], [25]. In particular, the curves used in this paper have been generated using the algorithm described in [25].

### *Elliptic Curve with a 160 bits Cardinality*

**Prime number** $p$**:** 1461501637330902918203684832716283019655932542983
**Coefficient** $a$**:** 1268133167195989090596625406312984755854486256116
**Coefficient** $b$**:** 3867369402698276552141188528065965276028925737344
$\#E(\mathbb{F}_p)$**:** 1461501637330902918203684149283858612734394057783

### *Elliptic Curve with a 320 bits Cardinality*

**Prime number** $p$**:** 53399675898022752059875542654238802865067613060\\
3927027765660916426535451401046499195937174770261 7
**Coefficient** $a$**:** 20881059596806233258424772504350452848300278627 85\\
870211433492825096738057013555851670055268682213
**Coefficient** $b$**:** 11637556704410285543025997645537897168467055804 67\\
529854789623514071878399983353288619261210338191
$\#E(\mathbb{F}_p)$**:** $2 \cdot$ 1461501637330902918203684832716283019655932542983 $\cdot$
$\cdot$ 1826877046663628647754606040895353774569915678761

### *Elliptic Curve with a 480 bits Cardinality*

**Prime number** $p$**:** 99505735041322252375288411699657175995736565 8\\
0696044268676583193632704892755795210303651532931 5\\
9676515285332084476809492997893652227166881856911 3
**Coefficient** $a$**:** 27070699584125069026028244478136940279906881839 5\\
622388832076793348602062306022242786736095545104\\
4640503732397723488780961683652364955407331149336
**Coefficient** $b$**:** 45364335873072114323231922743383295435922802127 8\\
359649203504732666066364663535035456892962202950\\
7484145388525397790533006611631792360854156270441
$\#E(\mathbb{F}_p)$**:** $2 \cdot$ 1461501637330902918203684832716283019655932542983 $\cdot$
$\cdot$ 1552845489664084350591415134761050708384428327041 $\cdot$
$\cdot$ 2192252455996354377305527249074424529483898814481

## References

[1] D. Chaum, "Security without identification: transaction systems to make big brother obsolete," *Communications of the ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.

[2] A. Fujioka, T. Okamoto, and K. Ohta, "A practical secret voting scheme for large scale elections," in *Advances in Cryptology – AUSCRYPT '92*, ser. LNCS, vol. 718, 1993, pp. 244–251.

[3] M. Ohkubo, F. Miura, M. Abe, A. Fujioka, and T. Okamoto, "An improvement on a practical secret voting scheme," in *ISW '99*, ser. LNCS, vol. 1729, 1999, pp. 225–234.

[4] K. Sako and J. Kilian, "Receipt-free mix-type voting scheme: A practical solution to the implementation of a voting booth," in *Advances in Cryptology – EUROCRYPT '95*, ser. LNCS, vol. 921, 1995, pp. 393–403.

[5] M. Jakobsson, "A practical mix," in *Advances in Cryptology – EUROCRYPT '98*, ser. LNCS, vol. 1403, 1998, pp. 448–461.

[6] F. Sebé, J. M. Miret, J. Pujolàs, and J. Puiggalí, "Simple and efficient hash-based verifiable mixing for remote electronic voting," *Computer Communications*, vol. 33, no. 6, pp. 667–675, 2010.

[7] J. Puiggalí and S. Guasch, "Eficiencia y privacidad en una mixnet universalmente verificable," in *XI Reunión Española sobre Criptología y Seguridad de la Información (RECSI)*, 2010, pp. 159–164.

[8] K. Peng, "An efficient shuffling based eVoting scheme," *J. Syst. Softw.*, vol. 84, no. 6, pp. 906–922, 2011.

[9] V. Mateu, J. M. Miret, and F. Sebé, "Verifiable encrypted redundancy for mix-type remote electronic voting," in *EGOVIS 2011*, ser. LNCS, vol. 6866, 2011, pp. 370–385.

[10] J. D. Cohen and M. J. Fischer, "A robust and verifiable cryptographically secure election scheme," in *26th Annual Symposium on Foundations of Computer Science*, 1985, pp. 372–382.

[11] K. Sako and J. Kilian, "Secure voting using partially compatible homomorphisms," in *Advances in Cryptology – CRYPTO '94*, ser. LNCS, vol. 839, 1994, pp. 411–424.

[12] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," in *Advances in Cryptology – EUROCRYPT '97*, ser. LNCS, vol. 1233, 1997, pp. 103–118.

[13] M. Hirt and K. Sako, "Efficient receipt-free voting based on homomorphic encryption," in *Advances in Cryptology – EUROCRYPT 2000*, ser. LNCS, vol. 1807, 2000, pp. 539–556.

[14] K. Peng, R. Aditya, C. Boyd, E. Dawson, and B. Lee, "Multiplicative homomorphic e-voting," in *INDOCRYPT 2004*, ser. LNCS, vol. 3348, 2004, pp. 61–72.

[15] K. Peng and F. Bao, "Efficient multiplicative homomorphic e-voting," in *ISC 2010*, ser. LNCS, vol. 6531, 2011, pp. 381–393.

[16] J. H. Silverman, *The Arithmetic of Elliptic Curves*. Springer-Verlag, 1986.

[17] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Springer-Verlag, 2004.

[18] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *Advances in Cryptology – CRYPTO '94*, ser. LNCS, vol. 839, 1994, pp. 174–187.

[19] "FIPS 180-2: Secure Hash Standard," NIST, 2002.

[20] D. Chaum and T. P. Pedersen, "Wallet databases with observers," in *Advances in Cryptology – CRYPTO '92*, ser. LNCS, vol. 740, 1993, pp. 89–105.

[21] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *WPES '05*, 2005, pp. 61–70.

[22] A. O. L. Atkin and F. Morain, "Elliptic curves and primality proving," *Math. Comput.*, vol. 61, no. 203, pp. 29–68, 1993.

[23] A. Agashe, K. Lauter, and R. Venkatesan, "Constructing elliptic curves with a given number of points over a finite field," Cryptology ePrint Archive, Report 2001/096.

[24] R. Bröker and P. Stevenhagen, "Elliptic curves with a given number of points," in *ANTS-VI*, ser. LNCS, vol. 3076, 2004, pp. 117–131.

[25] R. Bröker and P. Stevenhagen, "Efficient CM-constructions of elliptic curves over finite fields," *Math. Comput.*, vol. 76, no. 260, pp. 2161–2179, 2007.