An electronic voting platform with elliptic curve cryptography

Treball Final de Carrera

Enginyeria Tècnica en Informàtica de Sistemes
Escola Politècnica Superior
Universitat de Lleida

Author:
Sergi Sisó Gòdia

Advisors:
Josep M. Miret Biosca
Francesc Sebé Feixas

July, 2011

Abstract

The future of elections seems to be electronic voting systems due to its advantages over the traditional voting. Nowadays, there are some different paradigms to ensure the security and reliability of e-voting. This is a topic in continuous development.

This document is part of a wider project which presents an e-Voting platform based on elliptic curve cryptography. It uses an hybrid combination of two of the main e-Voting paradigms to guarantee privacy and security in the counting phase, these are precisely, the mixnets and the homomorphic protocols.

This document is focused in the description of the system and the maths and programming needed to solve the homomorphic part of it. In later chapters, there is a comparison between a simple mixing system and our system proposal.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The electoral process is the most important and fundamental activity in democracy because it makes possible to represent every person in the population. For common people this is the only way to participate actively in politics and the process main goal is to determine the government for the next years.

Consequently, it moves lots of interests and many attempts of fraud and manipulation cases have happened during the history. This is why the electoral process must be a open process paying special attention to security and reliability.

The traditional electoral process consists on a system where every person is called to move to a college and deposit its vote there. Then the votes are counted manually by the people who have been supervising the correct deposit of votes.

It is evident that traditional voting has some disadvantages such as the lot of time needed for the manual counting and the error probabilities of it, besides the displacement to the college of every person who participates. Moreover, the time and economic cost are also high.

Nowadays, with the big advances and improvements in the world of the new technologies, added to the fact that practically everybody has access to them, it seems reasonable to develop a new way of voting which uses these technologies. The objective is to make electoral process more comfortable, secure and universal. These new systems are known as Electronic Voting, or simply e-Voting.

E-voting is more comfortable because it allows you to vote remotely. People can use their home computer or even their mobile phone and send the vote through the Internet. It avoids the need to move to the college and improves the accessibility and the availability problems that someone may have.

Another evident advantage is the fact that vote counting is performed by computers. So that the process will be significantly faster and the errors disappear. The same happens with the possible recounts and verifications.

The whole process must be trusted by participants. For this reason, the most modern and advanced cryptography is used. Nowadays, there is a lot of research on how to improve the efficiency and robustness of the e-Voting protocols.

1.1 Properties of an e-voting process

Regarding the topics we are talking about, we must define and accomplish a set of properties to have a correct voting. The system must ensure the following aspects:

- *Authentication*: Every vote has to come from a person who is registered in the electorate. The system must ensure participants vote only once.
- *Integrity*: The content of each vote has to rest immutable through all the electoral process.
- *Privacy*: It must not be possible to link the voter with the content of his vote. This implies the non-coercion possibility.
- *Intermediate privacy*: The partial results must be secret until the voting process is closed.
- *Accuracy*: The result has to be exactly which the votes contain.
- *Verifiability*: Everybody should be able to verify that his vote has been included and counted properly.

Some of these properties can be easily broken in the traditional voting process by human errors. Nevertheless, electronic voting must provide, with verifiable processes, a high rate of accomplishment.

1.2 State of the art

There is an extended documentation about electronic voting, covering a wide range of topics. This is due to the fact that electronic voting can be split in a number of different types of problems focusing in the different properties announced in the previous section. That includes: authentication protocols like digital signature, anonymity in the sending phase, ensure that the voting device does not save information about the vote, and many others.

For a real e-voting platform all these aspects should be analysed and implemented. Our project is mainly focused in the counting phase. Consequently, this document only exposes the protocols that implement privacy in this phase, allowing verifications and ensuring the integrity of votes.

1.2.1 Homomorphic tally protocols

These protocols use the homomorphic property of some cryptosystems (ex. ElGamal, EC-ElGamal, Paillier, ...). This property makes that a certain operation applied on a pair of ciphertexts, coincides with the encryption of the message resulting from the operation on their cleartexts.

There are two types of homomorphism depending the involved operation:

- Additive Homomorphism: $E(v_1) \oplus E(v_2) = E(v_1 + v_2)$
- Multiplicative Homomorphism: $E(v_1) \oplus E(v_2) = E(v_1 \cdot v_2)$

That is useful in the electronic voting because we can join the votes when they are encrypted and at the end of the process just decrypt the final bullet. The most interesting fact of this protocol, apart of decreasing the number of decryptions we have to calculate, is that the decrypted aggregated result can not be directly linked with any voter. That is because the decrypted cleartexts do not come from the encrypted votes that voters have emit.

We can list the advantages of homomorphic protocols:

- The privacy problem is solved, except the coercion part.
- The final counting is fast because it consists of a simple decryption.

But there also exist some disadvantages, that become intractable when the voting is complex.

- In homomorphic tally the voting options have to be prefixed. This does not permit text or numerical input.
- So as to detect possible corrupted votes, the verification has to be done when the votes are still encrypted, before applying the homomorphic operation. This can be done with the zero-knowledge-proofs. The drawback is that such proofs are computationally expensive and require a lot of time.
- Vote format has to allow repetition of the homomorphic operation for a large amount of times. If the voting is big enough decoding the cleartext becomes a challenging problem.

1.2.2 Mixnet protocols

The mixnet is another process to ensure the privacy of voting by breaking the relation between the voter and its vote.

In this case the process is done when all votes are collected but they are not decrypted yet. The functionality is similar to the traditional voting where the votes are in boxes and they are shacked and mixed. In the electronic voting this is done by means of a process that shuffles and re-encrypts the votes in a way that makes hard to infer the path through the mixnet.

Finally, when the correlation is broken,votes can be decrypted.

Its advantages are:

- Corrupted votes are simply discarded.
- The privacy is ensured.

Nevertheless, some disadvantages appear:

- The process requires a big number of re-encryptions whose correctness is hard to compute.
- Vote counting is not as efficient as the homomorphic protocol because the votes must be decrypted one by one.

1.3 About this project

This project is part of an electronic voting platform developed and implemented by the Cryptography and Graphs research group of the University of Lleida in collaboration with Scytl Secure Electronic Voting, SL. This collaboration is funded by the Ministry of Industry under the *Avanza* program. TSI-020100-2010-185.

During the course 2010/2011 I have been working, collectively with Núria Busom and Oriol Carro, as this project grant-holder.

Our objective is to test and see the viability of an electronic voting system with the following characteristics:

- It uses a cryptosystem based on elliptic curves, more precisely, the EC-ElGamal cryptosystem. It has been selected due to its fulfilment of the additive homomorphism property and because the number of bits required for a good security level is smaller than other cryptosystems not based on Elliptic Curve Cryptography (ECC).
- It implements a hybrid solution involving the two actual paradigms of electronic voting, attempting to take advantage of both and minimizing their negative characteristics. The key is to find the best possible mix to make the voting process as computationally efficient as possible.

Particularly, this thesis is focused on how to apply the homomorphic property in elliptic curves finding a way to code and decode vote packages of a given size. The harder problem is the decoding phase since it involves a discrete logarithm problem over the group of points of an elliptic curve or a Knapsack problem, which are very hard to compute.

Chapter 2

Mathematical Background

2.1 Modular Arithmetic

The modular arithmetic is the base of the modern cryptography and public key cryptography in particular. The initial idea is quite simple: we fix a positive integer N, called modulo. Then, given two integers, we say a and b are congruent modulo N and we write $a \equiv b \pmod{N}$ if N divides $a-b$.

We define a *group* as a set with a binary operation, which has an identity element, is associative, and every element has an inverse. If the operation is commutative the group is said to be *Abelian*.

A group G is called *cyclic* if it has a generator g. A *generator* is an element from which every other group element can be obtained. Depending on the operation used to obtain the elements we can classify the cyclic groups in:

- Multiplicative group $(G, \cdot)$: $g \cdot \overset{n)}{\ldots} \cdot g = g^n$
- Additive group $(G, +)$: $g + \overset{n)}{\ldots} + g = n \cdot g$

A field is a set with two operations $(\mathbb{K}, +, \cdot)$ where:

- $(\mathbb{K}, +)$ is an Abelian group with identity denoted by 0.
- $(\mathbb{K} - \{0\}, \cdot)$ is a group.
- $(\mathbb{K}, +, \cdot)$ satisfies the distributive law.

There is a fact in modular arithmetic which has a lot of interest in cryptography. Not all the elements in the ring $\mathbb{Z}_N$ of integers modulo N have inverse. The modular inverse of an element only exists when this element e and N are co-prime, $gcd(e, N) = 1$. Then if N is a prime p all non-zero elements have a unique inverse, and $\mathbb{Z}_p$ is a field, often denoted by $\mathbb{F}_p$. For that reason, it is interesting to work in $\mathbb{F}_p$, the finite prime field of characteristic p.

2.2 Elliptic Curves

In this project we use protocols based on elliptic curves because they offer an improved efficiency and bandwidth in contrast to other public key systems. Furthermore, when defined over elliptic curves, the discrete logarithm problem, which is the base of the project cryptosystem, is harder to solve.

An elliptic curve E over a field $\mathbb{K}$ is given by the Weierstrass equation:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

where $a_i \in \mathbb{K}$ and $\Delta \neq 0$, being Δ the discriminant which is defined as:

$$\Delta = -b_2^2b_8 - 8b_4^3 - 27b_6^2 + 9b_2b_4b_6,$$
$$b_2 = a_1^2 + 4a_2, b_4 = 2a_4 + a_1a_3, b_6 = a_3^2 + 4a_6$$
$$b_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2$$

The $\mathbb{K}$-rational points in the affine plane are the points of $(x, y) \in \mathbb{K}^2$ which satisfy the equation of the curve E. The set of these points together with the point at infinity $\mathcal{O}$ is denoted $E(\mathbb{K})$.

If the characteristic of $\mathbb{K}$ is different from 2 and 3, the equation of the curve E can be simplified and expressed with the reduced Weierstrass form:

$$E : y^2 = x^3 + ax + b,\ a, b \in \mathbb{K}$$

with discriminant: $\Delta = -16(4a^3 + 27b^2) \neq 0$.

In the figure 2.1 we show an example of an elliptic curve over the real field. All figures in this section will represent elliptic curves over real numbers for their better understanding, but in cryptography elliptic curves are defined over finite fields. Particularly we will work over $\mathbb{F}_p$ (however it's also possible

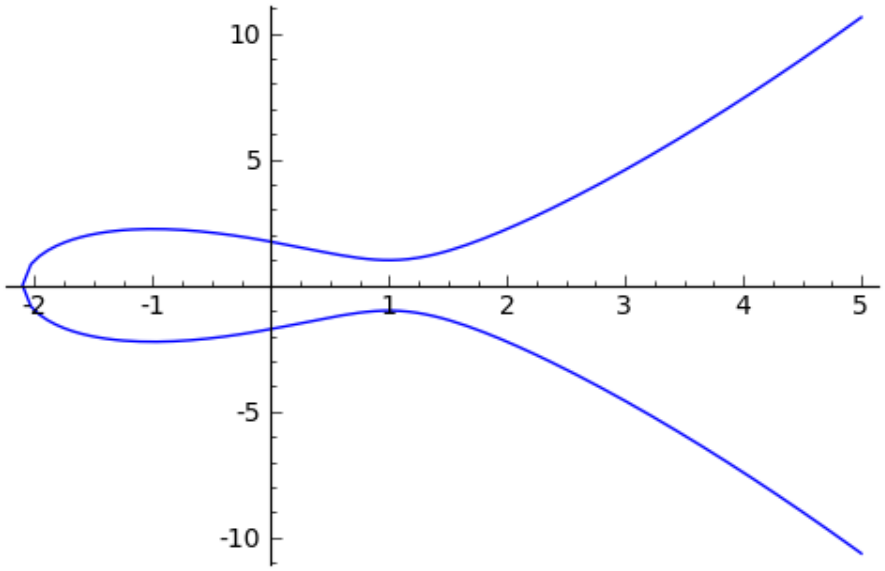

Figure 2.1: An elliptic curve over $\mathbb{R}$

to work over binary fields $\mathbb{F}_{2^m}$ or even $\mathbb{F}_q, q = p^m$). So, we can define the domain set where we work as:

$$E(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p \mid y^2 = x^3 + ax + b\} \cup \mathcal{O}.$$

2.2.1 Point Addition

The addition operation over $E(\mathbb{K})$ can be defined with the chord-tangent method. Together with this addition operation, the set of points $E(\mathbb{K})$ forms an Abelian group with $\mathcal{O}$ serving as its identity.

The best way to see this addition rule is geometrically. For the addition of two points we have to trace the line over them and select the opposite of the third point of the curve that the line cuts. In the case we want to do a point doubling $(P + P)$ we will trace the first line as the curve tangent at the point P.

We can see this procedure in the figure 2.2.

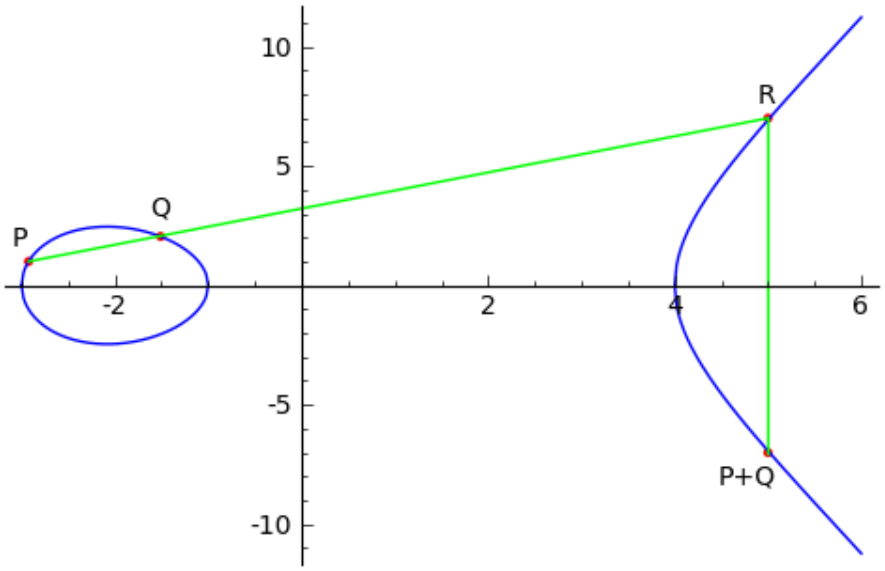

Figure 2.2: Elliptic curve point addition operation

Analytically, the addition operation can be defined as follows:

Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, be points of $E(\mathbb{K})$. Then the addition point $P + Q = (x_3, y_3)$ is given by:

$$P + Q = (\lambda^2 - x_1 - x_2, (x_1 - x_3)\lambda - y_1),$$

where

$$\lambda = \begin{cases} (y_1 - y_2)/(x_1 - x_2), & \text{if } x_1 \neq x_2, \\ (3x_2^2 + a)/2y_1, & \text{if } x_1 = x_2 \text{ and } y_1 \neq -y_2. \end{cases}$$

The algorithm we adopted to implement this operation in our project is described in the algorithm 2.2.1.

It is interesting to highlight that the opposite of a point $P = (x, y)$ is quickly obtained as $-P = (x, -y)$.

Algorithm 2.2.1 Elliptic Curve Point Addition

Input: $P = (x_1, y_1), Q = (x_2, y_2) \mid P, Q \in E(\mathbb{F}_p)$
Output: $S = P + Q$

```
if P = O then
  return Q
end if
if Q = O then
  return P
end if
if x1 = x2 then
  if y1 = −y2 then
    return O
  else
    λ = (3 · x2^2 + a)/(2 · y1)
  end if
else
  λ = (y1 − y2)/(x1 − x2)
end if
X ← λ^2 − x1 − x2
Y ← λ · (x1 − x3) − y1
return Point(X, Y)
```

2.2.2 Point Multiplication

The point multiplication operation over elliptic curves is the $k \cdot P$ operation where $k \in \mathbb{Z}$ and $P \in E(\mathbb{F}_p)$, which is defined as $k \cdot P = P \cdot \overset{k)}{\ldots} \cdot P$. The best way to compute it is using the binary method, based on doubling and adding points as many times as needed. For example, $21 \cdot P$ can be calculated using following expression which reduces the amount of point additions:

$$21 \cdot P = 2 \cdot (2 \cdot (2 \cdot (2 \cdot P) + P)) + P.$$

Hence, we can use the algorithm 2.2.2 which uses the binary representation of the operand to produce the described results.

An improvement to this multiplication method is achieved by reducing the number of 1's in the operand representation. This can be done by the NAF (Non-Adjacent Form) representation. For further information about NAF methods we refer the reader to [5].

Algorithm 2.2.2 Binary Method for Point Multiplication on an elliptic curve

Input: $k = (k_{t-1}, ..., k_1, k_0)_2, P \in E(\mathbb{F}_p)$
Output: $k \cdot P$
  $Q \leftarrow \mathcal{O}$
  for $i = 0 \rightarrow t - 1$ **do**
    if $k_i = 1$ **then**
      $Q \leftarrow Q + P$
    end if
    $P \leftarrow 2 \cdot P$
  end for
  return Q

2.3 ElGamal cryptosystem

The cryptographic systems which best fit in Electronic-Voting environments are the *Public-Key Cryptosystems* [9] [12] since they do not require a secure initial exchange of the secret keys between the sender and receiver. It would be an unachievable work to distribute a secret key for every voter.

The ElGamal cryptosystem is a public-key encryption scheme defined over a cyclic group G where the *Discrete Logarithm Problem* (DLP)[9] is intractable. In the basic ElGamal case explained in this section, we will assume it is defined over $\mathbb{F}_p^*$.

So, we define the parameters of the cryptosystem by selecting:

- A subgroup G of order a prime n of $\mathbb{F}_p^*$.
- g a generator of G.

Key generation:

- Choose a random integer $a \in [2, n-1]$.
- Calculate $h = g^a \pmod{p}$.
- The public key will be (p, g, h) and the secret key will be a.

Encryption:

$$m \xrightarrow{encryption} c$$

- Obtain p, g and h form the public key.
- Represent the message as an integer m in the range $\{0, 1, ..., p-1\}$.
- Select a random integer r, $1 \leq r \leq n-1$.
- Compute $\gamma = g^r \pmod{p}$ and $\delta = m \cdot h^r \pmod{p}$.
- Send the cipher-text: $c = (\gamma, \delta)$.

Decryption:

$$m \xleftarrow{decryption} c$$

- Use the private key a to compute $\gamma^{-a} \pmod{p}$.
- Recover m by computing $(\gamma^{-a}) \cdot \delta \pmod{p}$.

It is easy to see that the decryption operation returns the cleartext:

$$\frac{\delta}{\gamma^a} = \frac{m \cdot h^r}{(g^r)^a} = \frac{m \cdot h^r}{(g^a)^r} = \frac{m \cdot h^r}{h^r} = m.$$

2.4 EC-ElGamal cryptosystem

ElGamal has the multiplicative homomorphic property we need for our project but the recommended key length for having enough security is 1024 or 2048 bits.

Therefore, we decided to use the EC-ElGamal cryptosystem, an adaptation of ElGamal over elliptic curves [8] [10]. It keeps the homomorphic property we are looking for and the Discrete Logarithm Problem is harder over the group of points of an elliptic curves field than in the modular integers ring.

The EC-ElGamal cryptosystem is defined with the following set of parameters, which have to be shared among the users of the cryptosystem:

- Define a finite field, in our case $\mathbb{F}_p$, being p a large prime.
- Choose an elliptic curve E over $\mathbb{F}_p$ defined by the coefficients a and b of the Weierstrass equation.
- Determine a cyclic subgroup G of $E(\mathbb{F}_p)$ and a generator P of G.
- n will be the order of the group G and h the cofactor.

$$\text{EC-ElGamal parameters} = (p, a, b, P, n, h)$$

Key generation:

- Choose a random integer $x \in [2, n-1]$.
- Calculate $Q = x \cdot P$.
- The public key will be the point Q and the secret key will be x.

Encryption:

$$m \xrightarrow{encryption} c$$

- Obtain the cryptosystem parameters and the public key Q.
- Represent the message as a point M of the elliptic curve $E(\mathbb{F}_p)$.
- Select a random integer r, $1 \leq r \leq n-1$.
- Compute $R = r \cdot P$ and $S = M + r \cdot Q$.
- Send the cipher-text: $c = (R, S)$.

Decryption:

$$m \xleftarrow{decryption} c$$

- Use the private key x to compute $x \cdot R$.
- Recover M by subtracting $x \cdot R$ to S. $M = S - x \cdot R$.

It is easy to see that the decryption operation returns the cleartext:

$$\begin{aligned} S - (x \cdot R) = (M + r \cdot Q) - x(r \cdot P) &= (M + r \cdot Q) - r(x \cdot P) \\ &= (M + r \cdot Q) - r \cdot Q = M. \end{aligned}$$

It is possible to perform re-encryptions to ciphertexts:

- Select a random integer r', $1 \leq r' \leq n - 1$.
- Modify the cipher-text with $R = R + r' \cdot P$ and $S = S + r' \cdot Q$.

We can see that this process modifies the ciphertext, but preserves the cleartext:

$$\begin{aligned} ReEncryption(Encryption(M)) &= (r \cdot P + r' \cdot P, M + r \cdot Q + r' \cdot Q) \\ &= ((r + r') \cdot P, M + (r + r') \cdot Q). \end{aligned}$$

In a similar way we can see the additive homomorphism property of this cryptosystem:

$$Encryption(M) = (r \cdot P, M + r \cdot Q),$$

$$Encryption(M') = (r' \cdot P, M' + r' \cdot Q),$$

$$Enc(M) + Enc(M') = ((r + r') \cdot P, M + M' + (r + r') \cdot Q) \xrightarrow{decryption} M + M'.$$

2.5 Security

The two cryptosystems presented in previous sections have a good security level. Both of them base their security in the discrete logarithm problem (DLP), which results NP-complete over finite fields.

The *Discrete Logarithm Problem (DLP)* is defined as follows: given a cyclic group $(G, \cdot)$, with order n and g being the generator of G. Given the element $x \in G$, find an integer a such that $g^a = g \cdot \overset{a)}{\ldots} \cdot g = x$.

Its homonymous on elliptic curves is the *Elliptic Curve Discrete Logarithm Problem (ECDLP)*[6] which is enunciated in the following way: given an elliptic curve E over a finite field $\mathbb{F}_p$, a generator point P of a group

G and order n of $E(\mathbb{F}_p)$ and a point $Q \in G$. Find the integer k such that $Q = k \cdot P$, also expressed as $k = \log_P Q$.

In the next table 2.1, their security is compared. From this table we can clearly see how the EC-ElGamal achieve the same security of ElGamal with shorter keys.

Security Bits (Based on symmetric encryption)	**ElGamal cryptosystem**	**EC-ElGamal cryptosystem**
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Table 2.1: Bit security comparison of ElGamal and EC-ElGamal cryptosystems

Chapter 3

Electronic voting platform

In order to introduce our electronic voting platform, the first thing to say, is that it was conceived as a testing platform for voting protocols using elliptic curve cryptography. It is not intended to be a real application platform, since other fundamental aspects have not been considered so as to simplify the implementation. A representative example is the non-existence of an authentication process.

With this idea in mind, we proceed to describe the voting platform in terms of their functional parts. Then we describe it from the point of view of the cryptographic operations involved in the process.

3.1 Functional scheme

The system has three participants: the voters, a college (or polling station) and a server.

The functional scheme follows these steps:

1. The server creates the voting process. It includes:
    - Initialization of the cryptosystem (keys generation).
    - Definition of the list of integers that represent each candidate.
2. The server shares the public key and the parameters of the cryptosystem in a public XML.

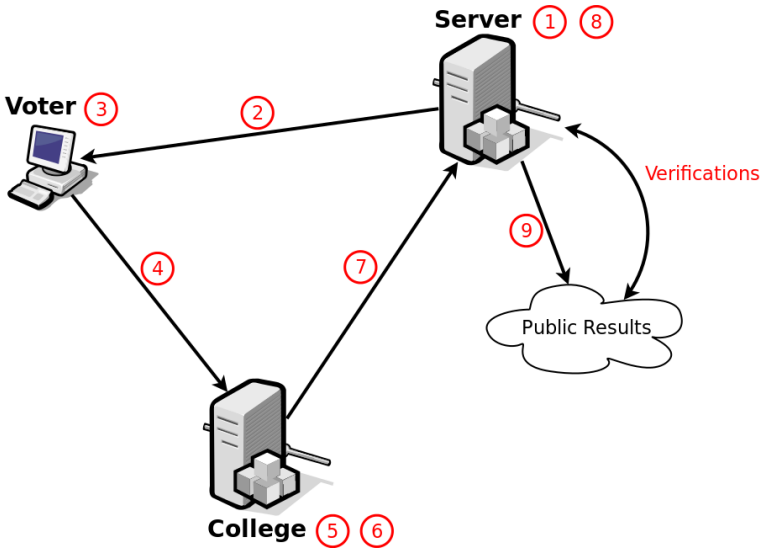

Figure 3.1: Functional Scheme

3. The voter selects a candidate. To encrypt the vote, it multiplies the candidate representation by the generator point and encrypts the result.
4. Then the voter sends the vote to the college.
5. When the college receives a vote:
   - Performs input verifications.
   - Collects them in n-size packages applying the homomorphic operation.
6. When the voting time has finished, the college:
   - Starts the mixing process of all the packages of grouped votes.
   - Generates the verification proofs of correct mixing.
7. The college sends the mixed packages to the server.
8. The server decrypts and decodes the packages and counts the results.
9. The results are published.

Finally, and optionally, some mixing and other zero-knowledge verifications can be performed.

3.2 Hybrid system between Mixnets and Homomorphic tally

The main characteristic of our platform is the combination of the homomorphic cryptography paradigm with the mixnet nodes. This combination tries to exploit the benefits of both and minimize their efficiency weaknesses.

To implement this hybrid system we have created the package object. It is a collector of a determined capacity of votes through the homomorphic operation. Then the mixnet nodes work with packages and not with single votes.

The figure 3.2 exposes the operation made when we track the path that votes follow through the hybrid system.

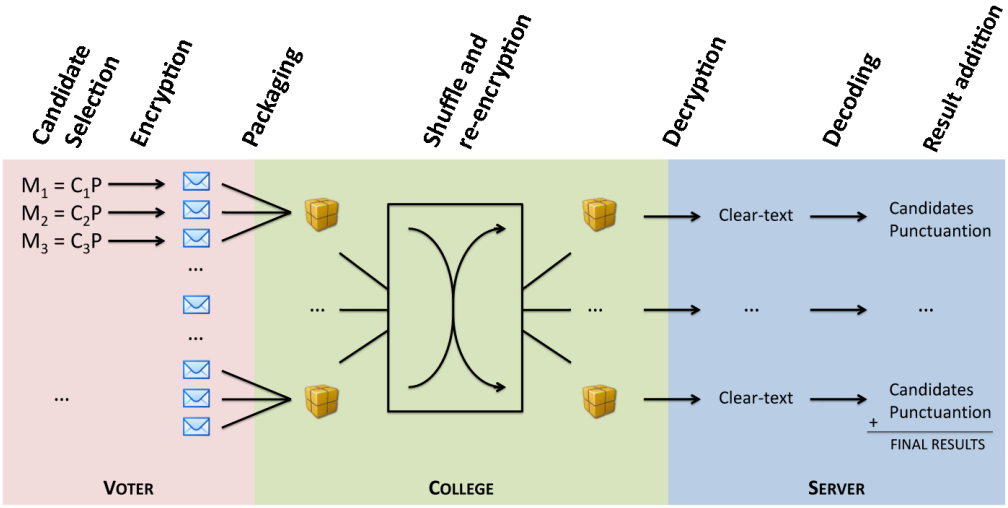

Figure 3.2: Hybrid paradigm process

The process is:

1. **Candidate Selection**. The point P is the generator of a cyclic group of the group of points of an elliptic curve used by the cryptosystem. The set $\{c_0, c_1, c_2, ..., c_m\}$ are the integers which represent each candidate. In next chapter it is explained how they are selected for coding purposes. The elliptic curve message for the selected candidate i is going to be:

$$M_i = c_i \cdot P.$$

2. **Encryption** of the votes using EC-ElGamal cryptosystem:

$$e_i = encryption(M_i) = (r_i \cdot P, M_i + r_i \cdot Q),$$

where Q is the public key and r_i is a random integer.

3. **Packaging**. We aggregate the votes employing the homomorphic operation of the cryptosystem:

$$\begin{aligned} Package_j &= ... + e_{i-1} + e_i + e_{i+1} + ... \\ &= (r \cdot P, (... + M_{i-1} + M_i + M_{i+1} + ...) + r \cdot Q), \end{aligned}$$

where $r = ... + r_{i-1} + r_i + r_{i+1} + ...$.

4. **Mixing**. Here the packages are shuffled and re-encrypted. This process is explained with more detail in my partner Oriol's thesis [3]:

$$(Package_1, Package_2, Package_3, ...)$$

$$\downarrow\swarrow\searrow$$

$$(Package_{k_1}, Package_{k_2}, Package_{k_3}, ...).$$

5. **Decryption** of the packages using the private key of EC-ElGamal cryptosystem:

$$D = decryption(Package_{k_j}) = ... + M_{i-1} + M_i + M_{i+1} + ...$$

6. **Decoding**. For decoding we must efficiently solve a knapsack problem as explained in the next chapter:

$$\text{Find } (x_1, x_2, ..., x_m) \text{ where } D = (x_1 \cdot c_1 + x_2 \cdot c_2 + ... + x_m \cdot c_m)P.$$

7. **Result counting**. The sum of the votes for each candidate:

$$\sum_{\text{every decrypted package}} (x_1, x_2, ..., x_m).$$

The benefits of this process are:

- The privacy is highly ensured.
- The final count is faster than the one in mixing protocols because it has to decrypt less ciphertexts. In our hybrid case, it has to decrypt as many as the number of packages. By contrast, the simple mixing protocol has to decrypt as many ciphertexts as votes.
- The mixing times are significantly decreased. It happens again because it is performed over the packages and not over the single votes.

In conclusion, we achieve a computationally efficient protocol without overlooking the privacy security.

3.3 Implementation details

To implement the system described above we have used the Java language. It is not an efficient environment but there exist two influential reasons for which we have chosen this language.

First of all, it is an object-oriented paradigm language with a clear C like syntax and a lot of tools and API's that makes programming easy. In a second place, previously the Cryptography and Graphs research group (UdL) has worked on a similar voting system project which was programmed with Java. Hence, we have reused some of the cryptographic classes already implemented.

On the elliptic curve point representation, we have used the standard Java class: java.security.spec.ECPoint. Even though, we have encapsulated it in a new class adding some more functionalities to the ones it offers.

The curves selected for the system test and debug were those recommended by Certicom Researchers in their article *Recommended Elliptic Curve Domain Parameters* [4].

Chapter 4

Vote Packager Block

This module is responsible of the implementation of the homomorphic paradigm in the electronic voting system. The idea is not to do the homomorphic operation to join all the votes in one single bullet, but join them in packages of a determined capacity of votes.

In this way, we can work with acceptable sized packages to decrypt them in terms of efficiency, reducing significantly the set of bullets we have to introduce in the mix process.

To implement this, we can split the process in two differenced tasks:

- Firstly, find a package codification suitable for putting all votes in a single package, ensuring that the repetition of the homomorphic operation will not alter the meaning of the final package.
- Secondly, choose a decoding algorithm that minimizes the time to find the quantification of every candidate votes.

4.1 Package Coding

The coding is quite simple. Suppose n is the package capacity, that is, the maximum number of votes we can put together in a package. And suppose m is the number of candidates. Finally, we must define a generator P of the group of points of the elliptic curve E defined over $\mathbb{F}_p$ we have chosen.

The first candidate C_0 will be represented as P if he receives one vote, $2 \cdot P$ if he receives two votes, until $n \cdot P$ if he receives all the possible votes of the package. Then, the second candidate, C_1, will be represented

with $(n+1) \cdot P$ if he has one vote, $2 \cdot (n+1) \cdot P$ if he has two votes, until $n \cdot (n+1) \cdot P$ if he has all votes.

According to this method we complete table 4.1.

	1 Vote	2 Votes	3 Votes	...	n Votes
C_0	P	$2 \cdot P$	$3 \cdot P$	...	$n \cdot P$
C_1	$(n+1) \cdot P$	$2(n+1) \cdot P$	$3(n+1) \cdot P$	...	$n(n+1) \cdot P$
C_2	$(n+1)^2 \cdot P$	$2(n+1)^2 \cdot P$	$3(n+1)^2 \cdot P$	...	$n(n+1)^2 \cdot P$
...	...	...	...	...	...
C_m	$(n+1)^m \cdot P$	$2(n+1)^m \cdot P$	$3(n+1)^m \cdot P$	...	$n(n+1)^m \cdot P$

Table 4.1: Decoding process

From this table we can deduce that an empty package will be initialized to $\mathcal{O}$. Then we will add $(n+1)^{candidate} \cdot P$, where *candidate* is the number of the selected candidate. It is important to highlight that the number of added votes in the package is never higher than its capacity, which in the table is represented by n.

4.2 Package Decoding

The decoding process is the main and most challenging task in homomorphic protocols. In the group of points of an elliptic curve, it becomes a knapsack problem which represents a NP-complete problem. This is the reason why our packages have a size limitation: to have tractable instances of the problem.

The definition of the decoding is simple: we have to find the factor which the generator is multiplied by, to become the value of the package.

We will denote:
Q : Package Point
P : Generator of the group of points of the elliptic curve.
x : The factor, being $x_0, x_1, x_2, ..., x_m$ the votes for each candidate, that is $x = x_0 + x_1(n+1) + x_2(n+1)^2 + ... + x_m(n+1)^m$, such that,

$$Q = x \cdot P.$$

The naïve algorithm to find x is also simple. It is an exhaustive search comparing Q with the domain of the package $\{P, 2 \cdot P, 3 \cdot P, ..., ((n+1)^{m+1}-1) \cdot P\}$. The cost of this search is exponential. For this reason in the next sections we try to achieve a better solution, which in our case is efficient due to the size of n.

Below we have proposed two options to solve the decoding problem: the Baby-Step Giant-Step solution and the Knapsack solution.

4.2.1 Baby-Step Giant-Step solution

One possibility is to deal with the problem in two parts. The first step, is to find x taking it directly as the discrete logarithm problem over the group of points of an elliptic curve (ECDLP) with a good solver for tractable instances. Once we have got the integer x, we will decompose it in $\{x_0, x_1, x_2, ..., x_m\}$.

$$Q = x \cdot P \rightarrow x = log_P Q \rightarrow \text{decompose } x \text{ in terms of } \{x_0, x_1, x_2, ..., x_m\}$$

There is a wide range of studies and algorithms about solving the ECDLP. We have choose the *Baby-Step Giant-Step algorithm* [9] (Algorithm 4.2.1) because it has a good efficiency (although it has an exponential cost) and it is relatively easy to implement. However, we can find more complex and efficient algorithms like Pollard's rho algorithm.

Algorithm 4.2.1 Baby-Step Giant-Step algorithm over Elliptic Curves

Input: An elliptic curve E over $\mathbb{F}_p$, of group order a prime q, a generator P for $E(\mathbb{F}_p)$ and Q a point of $E(\mathbb{F}_p)$.

Output: The integer x, where $x = log_P Q$

1: $l \leftarrow \sqrt{q}$
2: Construct a table with the key $(Q - j \cdot P)$ and value j , for each $j \in \{0, 1, ..., l\}$
3: **for** $0 \leq i \leq l$ **do**
4: $key \leftarrow i \cdot l \cdot P$
5: **if** key is in the table **then**
6: **return** $(l \cdot i) + j_{key}$
7: **end if**
8: **end for**

Once we have obtained the integer x such that $Q = x \cdot P$, we only have to decompose this integer. We do it dividing it by the weight of the candidate with the biggest representation. The result is the votes for this candidate. Then we proceed by dividing the quotient for the weight of the next candidate in the representation other. We repeat the process until we decode all the votes.

4.2.2 Knapsack solution

A second approximation to the problem is taking it as a *Knapsack Problem* (sometimes called the Subset Sum Problem)[7] where we have to find the values of $\{x_0, x_1, x_2, ..., x_m\}$ so as to solve the equation:

$$Q = x_0 \cdot P + x_1(n+1) \cdot P + ... + x_m(n+1)^m \cdot P.$$

Managing the problem as described above has some advantages over solving the ECDLP problem directly:

- We can filter the situations we already know that never happen. For example $x_0 + x_1 + ... + x_m$ can not be larger than n.
- We can start searching the most probable candidate combinations. This improvement has not been implemented in this project, even though, it is more extensively described in the *Future Work* section.

The Knapsack solver algorithm we have choose has again a relatively easy implementation with a reasonably good efficiency. It is the *Meet in the Middle Algorithm* (Algotithm 4.2.2), one of the best in solving the Knapsack in general cases with a cost of $O(n2^{n/2})$. Moreover, it has an easy and direct adaptation to work over elliptic curves.

Algorithm 4.2.2 Meet in the Middle over EC

Input: An elliptic curve E over $\mathbb{F}_p$, a generator P for $E(\mathbb{F}_p)$, Q a point of $E(\mathbb{F}_p)$, the capacity n of the package and the number m of candidates.

Output: The set $\{x_0, x_1, ..., x_m\}$, such that $\sum_{i=0}^{m} x_i \cdot (n+1)^i P = Q$

1: Set $t \leftarrow n/2$
2: Create a Map
3: **for all** $\{x_0, x_1, ..., x_t\}$ combinations where $0 \leq x_i \leq n$ **do**
4: $key \leftarrow Q - \sum_{i=0}^{t} x_i \cdot (n+1)^i P$
5: **if** $key = 0$ **then**
6: **return** $\{x_0, ..., x_t, 0, ...^{n/2}..., 0\}$
7: **else**
8: Insert in the map and entry with key key and value $\{x_0, ..., x_t\}$
9: **end if**
10: **end for**
11: **for all** $\{x_t, x_{t+1}..., x_n\}$ combinations where $0 \leq x_i \leq n$ **do**
12: $key \leftarrow \sum_{i=t}^{n} x_i \cdot (n+1)^i P$
13: **if** key is in the map **then**
14: $\{x_0, x_1, ..., x_t\} \leftarrow map_{key}$ value
15: **return** $\{x_0, x_1, ..., x_t\} + \{x_t, x_{t+1}..., x_n\}$
16: **end if**
17: **end for**
18: **return** Null Solution

This Knapsack algorithm and the previous BSGS algorithm have a similar computational cost. However, it is not necessary to go over the combinations in any strict order. This is valid for both the table construction and the table search processes. The only requirement is to pass over all possible combinations.

This means we can alter the order introducing the improvements described above regarding the non-use of impossible combinations and starting the search for the most probable ones.

Consequently, the function that decides which element should be the next to be evaluated is crucial for a good efficiency. The one we have implemented in this project is described in Algorithm 4.2.2.

Algorithm 4.2.3 Next Combination

Input: The last $\{x_0, x_1, ..., x_{n/2}\}$ combination
Output: The following combination or and advise if it is the last one

```
1: repeat
2:    i ← 0
3:    while combination not changed do
4:       if x_i < n then
5:          x_i ← x_i + 1
6:          combination ← changed
7:       else
8:          if  i = n/2  then
9:             return  Last Combination
10:         else
11:            x_i ← 0
12:            i ← i + 1
13:         end if
14:      end if
15:   end while
16:   suma = x_0 + x_1 + ... + x_{n/2}
17: until suma >  pakage capacity
```

4.3 Comparisons and Chosen Proposal

To decide which algorithm is the best, we have implemented and compared both of them: Figure 4.1 compares the required time to unpack a single pack in a 6 candidates voting simulation. The test is done for different package sizes.

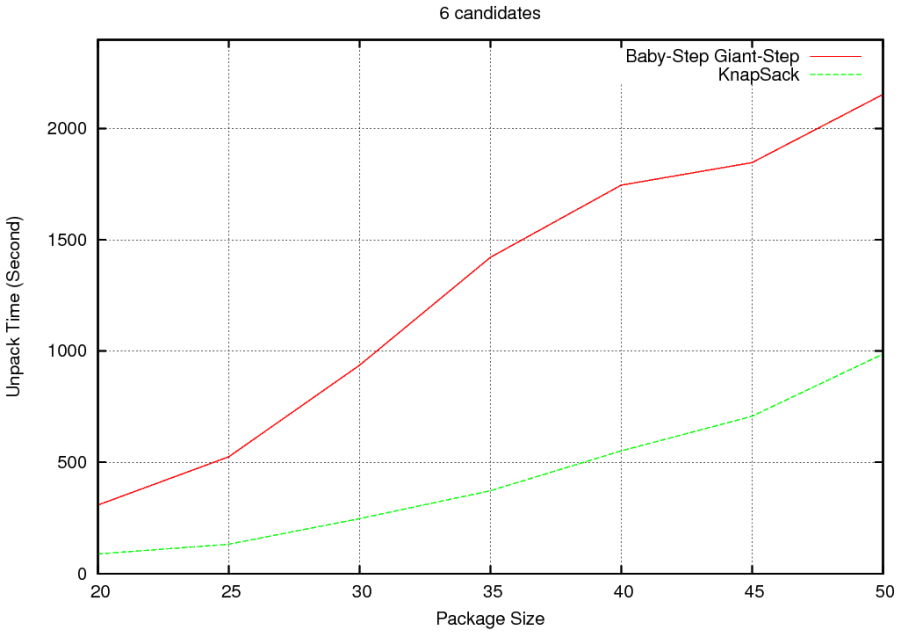

Figure 4.1: Unpack algorithms comparison

We can appreciate that the Knapsack problem used to be faster than the BSGS Algorithm. We can also note, their behaviour tent to be the same when increasing the package sizes. Therefore, for its better results, plus the highest flexibility provided in the selection of the searching path, the chosen proposal was the Knapsack one.

Chapter 5

Analysis

In this chapter we present the execution times of the implementation of our system. We also analyse its behaviour with respect to some of the parameters which may vary in a voting process.

5.1 Analysis of our hybrid system proposal

The first thing we see is that the times vary linearly with the increment of the census. For this reason, we conclude that the census is not a vital parameter. The figure 5.1 displays this fact.

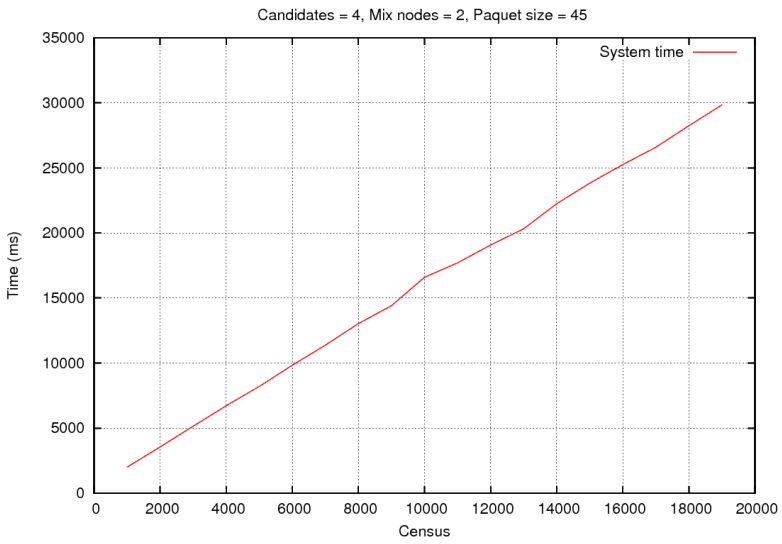

Figure 5.1: Census behaviour

After that, we have separated the times of the system in the following representative parts: candidate selection, mixing process (and mixing verification) and decoding and decryption phase (with its verifications).

The candidate selection time can be ignored, because it is obtained in a random selection process. However, in a real voting process it takes as much time as the voters need to decide their vote.

To observe the times of the Mixing process and the Decryption and Decoding phase, a key parameter is the package size. This is because it determines the input elements the mixing has and also the size of search domain which the decoding must face.

Then we can observe that mixing times decrease exponentially when the package size grows. By contrast, the decoding time increases exponentially. So, as we can appreciate in the figure 5.2, there is an optimum package size that minimizes the overall time.

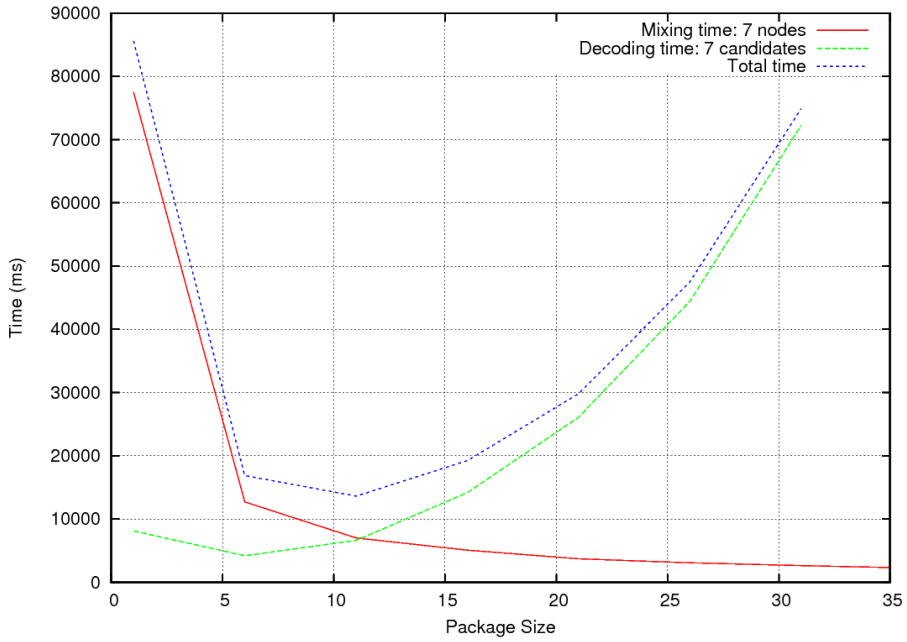

Figure 5.2: Optimum package size

Moreover, as the mixing is repeated as many times as the number of nodes the mixing process has, the process varies linearly with the number of nodes. See figure 5.3.

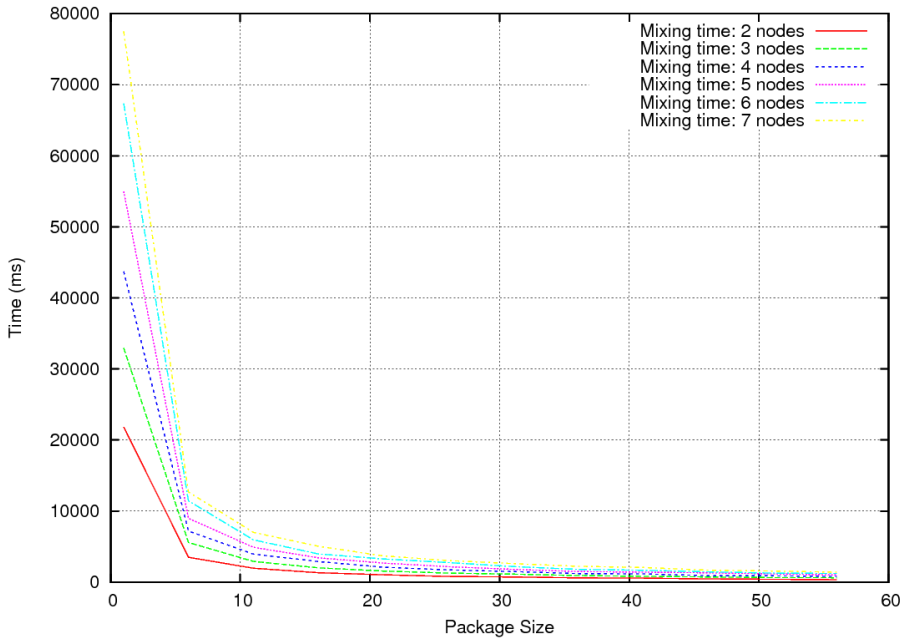

Figure 5.3: Mixing times

A more interesting fact is that when we increment the number of eligible candidates the decoding phase increases its exponential degree. See figure 5.4.

Summarizing, considering the amount of mix nodes and the number of candidates, we have a different optimum package size that minimizes the time.

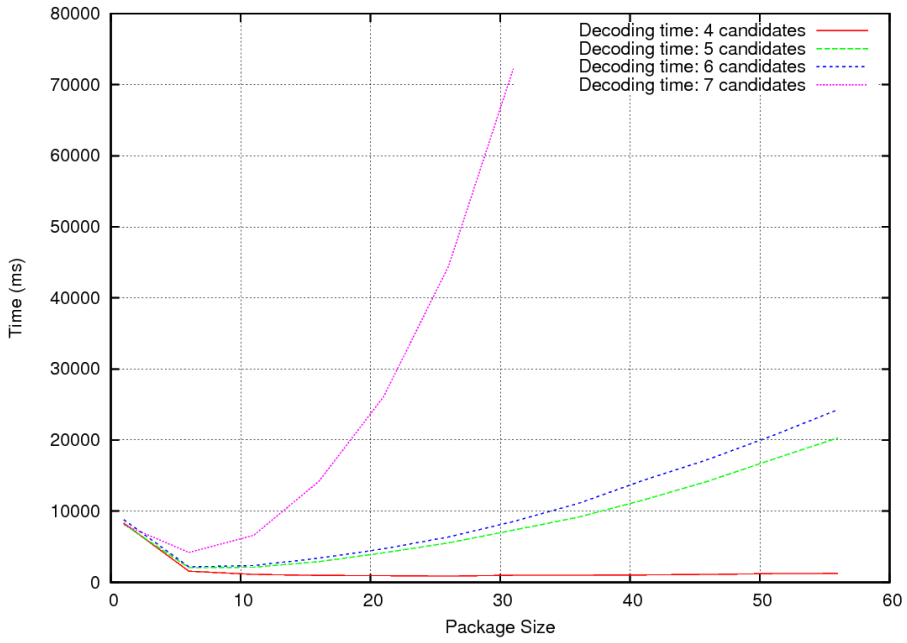

Figure 5.4: Decoding times

Chapter 6

Conclusions

During the project we have studied, compared and implemented different techniques to acquire the initial objectives.

The first objective, which was to use elliptic curve cryptography compatible with an homomorphic operation and a re-encryption operation, was easy to accomplish. It has been quite straightforward by following some bibliography and implementing the EC-ElGamal cryptosystem.

The development of the platform was not more difficult but as the project was advancing, we noticed some unexpected problems and reached some improvements. Consequently, it needed some rethinking and redesign tasks.

The correct merge between the mixing paradigm and the homomorphic tally was the most challenging part. The graphics presented in the analysis section show the achievement of a system that improves the single mixing and homomorphic tally compute times fairly well.

However, it is not a finished job. In fact, it can always be improved depending on the type and the characteristics of the voting. In the following *Future Work* section some improvements are described.

6.1 Future Work

- An interesting approach to improve the knapsack search path is to study the candidates priority that an election may have. Then, we can use this information to find the solution faster. It is possible thanks to the flexibility that Knapsack algorithms provide.
- Following the previous point, it is also possible to consider an adaptive search path that modifies its track according to the results of the decoding of the previous packages.
- Better knapsack solving algorithms have been studied currently. Probably, they could be used in this project to decrease the decoding times.
- As a testing system, it could be useful to implement other proofs, cryptosystems and cryptographic paradigms to enhance the comparisons and the analysis.
- If the application has to become a real voting platform, it needs the implementation of other security aspects not considered on this project. Some of them are: the authentication protocol, a visual interface, among others.

Bibliography

[1] Boneh D. and Golle P. *Almost Entirely Correct Mixing with application to voting*. 9'th ACM conference on Computer and Communications Security (CCS), 2002.

[2] Busom N. *Disseny d'una plataforma de votació electrònica: Mòdul criptogràfic*. Treball final de Carrera. Universitat de Lleida, 2010.

[3] Carro O. *Mescla de vots emprant Random Group Full Checking sobre corbes el·líptiques*. Treball final de Carrera. Universitat de Lleida, 2011.

[4] Certicom Research. *SEC2: Recommended Elliptic Curve Domain Parameters*. Certicom Corp, 2000.

[5] Fan R. *On The Efficiency Analysis of wNAF and wMOF*. Technische Universität Darmstadt, 2005.

[6] Hankerson D., Menezes A.J. and Vanstone S.A. *Guide to Elliptic Curve Cryptography*. Springer, 2003.

[7] Kellerer H., Pferschy U. and Pisinger D. *Knapsack Problems*. Springer, 2004.

[8] Koblitz. *elliptic curve cryptosystems*. Mathematicals of computation, 48 (177):203-209, 1987.

[9] Menezes A.J., van Oorschot P. and Vanstone S.A. *The handbook of Applied Cryptography*. CRC Press, 1997.

[10] Miller V. *Use of Elliptic Curves in Cryptography*. Springer-Verlag New York, Inc., 2001.

[11] Peng K. *A Hybrid e-Voting Scheme*. Institute for Infocomm research, Singapore, Lecture Notes in Computer Science, 2009.

[12] Smart N. *Cryptography: An introduction*. Third Edition, Mcgraw-Hill College, 2004.

[13] Puiggalí J. and Guasch S. *Universally Verifiable Efficient Re-encryption Mixnet*. Scytl Secure electronic Voting. 4th International Conference on Electronic Voting (EVOTE' 10), July 2010.

[14] Sebé F., Miret J.M., Pujolàs J. and Puiggalí J. *Simple and efficient hash-based verifiable mixing for remote electronic voting*. Computer communications, vol 33, num 6, 2010.