Desenvolupament d'un prototip d'e-mail amb encriptació basada en la identitat

Autor: Meritxell Jordana Gavieiro
Director: Josep M. Miret Biosca

Universitat de Lleida
Escola Politècnica Superior
Grau en Enginyeria Informàtica

Treball Final de Grau

Juny de 2016

Índex

Índex de figures

Capítol 1

Introducció

La paraula criptografia prové del grec **cryptos** (ocult) i **grafé** (escriptura), pel que literalment significa **escriptura oculta**. L'única finalitat que se li atribuïa la criptografia clàssica era la d'aconseguir confidencialitat en els missatges, és a dir, amagar el text original per a que no pogués ser entès per tothom. D'aquesta necessitat van sorgir els xifrats més antics que ja utilitzaven les civilitzacions antigues (xina, mesopotàmica, egípcia...). Un dels xifrats documentats més antics conegut és el de Juli Cèsar, que consistia únicament en canviar cada lletra per la tercera lletra posterior a l'abecedari. Però com ja es pot imaginar, aquesta necessitat d'amagar missatges anava creixent cada vegada més en l'àmbit militar, i això feia que la complexitat dels sistemes criptogràfics anés creixent cada vegada més.

L'aparició de noves tecnologies, l'evolució de la informàtica moderna i l'ús massiu que se li està donant (comerç electrònic, correu electrònic o la banca a través de la xarxa, entre altres) han fet que el nombre de problemes de seguretat vagi creixent. Com passa en una comunicació parlada entre dues persones, una transacció informàtica també pot ser escoltada/interceptada per una tercera part no desitjada i això pot suposar un problema. El que s'intenta fer és crear noves tècniques per poder garantir la seguretat d'aquesta informació. Aquesta situació és la que ha fet que, avui en dia, la criptografia sigui vista com la metodologia per proveir la seguretat en la xarxa, incloent la identificació d'entitats i el control d'accés als recursos, la confidencialitat dels missatges transmesos, la integritat dels missatges i la no repudació d'aquestos. Així doncs la criptografia, s'utilitza no solament per protegir la confidencialitat de les dades, sinó per garantir també la seva integritat i

autenticitat.

Les corbes el·líptiques van ser proposades per primer cop per ser utilitzades en aplicacions criptogràfiques l'any 1985 per Miller [28] i Koblitz [29] de forma independent. Les corbes el·líptiques estan presents també en altres temes relacionats directament amb la criptografia i el criptoanàlisis com són els testos de primalitat i els algoritmes de factorització.Els tests de primalitat són una eina imprescindible en la configuració dels criptosistemes de clau pública tant pels de tipus RSA com els de ElGamal. Si bé s'utilitzen habitualment en tests probabilístics per descartar possibles nombres compostos, sembla convenient utilitzar tests deterministes per assegurar i certificar la primalitat d'un nombre.

Lògicament, els avanços en les tècniques i resultats criptoanalítics afavoreixen a que hi hagi una intensa investigació per garantir la seguretat dels protocols criptogràfics actuals, ja sigui descobrint nous mètodes, explorant l'adequació d'altres grups o proposant nous problemes matemàtics computacionalment intractables.

Durant el treball s'intentarà introduir el contepte de corba el·líptica, així com també el de pairing (o aparellament). A més a més, per treballar una mica més amb aquestes corbes i aquests aparellaments, s'ha realitzat la implementació de dues senzilles aplicacions que serviran per veure l'ús dels aparellaments i corbes el·líptiques d'una manera més pràctica. La primera implementació és una compartició de claus a tres bandes, que més enllà de tenir una utilitat pràctica concreta, com passarà en el segon cas, ens serveix per veure com es compleixen les propietats dels aparellaments que es descriuen de la manera teòrica. La segona implementació és una de les aplicacions més conegudes en les que es pot utilitzat la encriptació basada en Identitat, que és l'enviament/recepció de missatges mitjançant un **correu electrònic**. Aquestes dues implementacions ens serveixen per veure com les característiques dels pairings ens poden ajudar a resoldre un ampli ventall de problemes.

Capítol 2

Corbes el·líptiques

En aquest capítol s'intentaran deixar ben definits els principals conceptes sobre les corbes el·líptiques, així com introduir les corbes el·líptiques sobre un cos $\mathbb{F}_q$.

2.1 Introducció a les corbes el·líptiques

Una corba el·líptica sobre un cos $\mathbb{K}$ és una corba algebraica sense punts singulars que ve donada per una equació del tipus

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad a_i \in \mathbb{K} \tag{2.1}$$

denominada equació general de Weierstrass. Si la característica de $\mathbb{K}$ és diferent de 2 i 3, l'equació de la corba es pot expressar com

$$y^2 = x^3 + ax + b, \quad a, b \in \mathbb{K} \tag{2.2}$$

anomenada equació reduïda de Weierstrass. Per a que això es compleixi, es necessita que el discriminant del polinomi cúbic en x no sigui nul, per tant, $\triangle = 4a^3 + 27b^2 \neq 0$. Si es disposa d'aquesta característica, la corba no té singularitats (punts singulars). En la Figura 2.1 es pot veure un exemple de corba el·líptica sobre $\mathbb{R}$ que té com a equació: $y^2 = x^3 - 13x - 12$.

Si $E/\mathbb{K}$ és una corba el·líptica sobre un cos $\mathbb{K}$, denotarem per $E(\mathbb{K})$ el conjunt de punts $P = (x, y) \in \mathbb{K} \times \mathbb{K}$ que satisfan l'equació de la corba juntament amb el punt de l'infinit de la corba($\mathcal{O}$). Expressat de manera formal tindríem

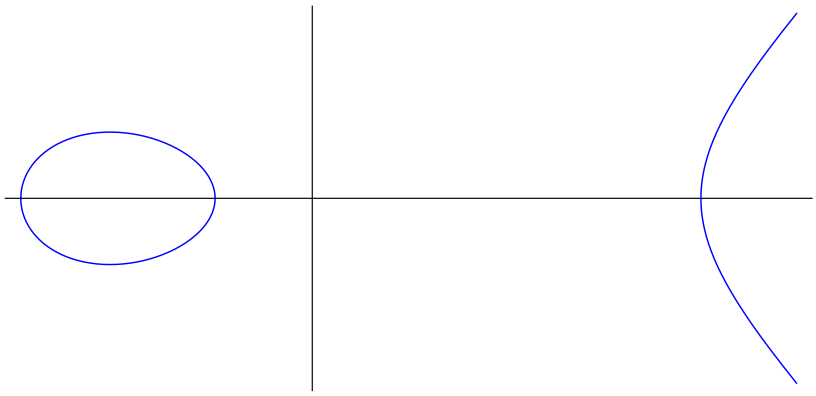

Figura 2.1: Gràfica de la corba $y^2 = x^3 - 13x - 12$ definida sobre $\mathbb{R}$

que :

$$E(\mathbb{K}) = \left\{(x, y) \in \mathbb{K} \times \mathbb{K} \mid y = x^3 + ax + b,\ a, b \in \mathbb{K}\right\} \cup \{\mathcal{O}\} \tag{2.3}$$

Per facilitar els càlculs prenem com a punt base el punt de l'infinit de la corba, $\mathcal{O} = (0 : 1 : 0)$ en coordenades projectives.

2.2 Suma de punts en una corba el·líptica

Al conjunt de punts d'una corba $E/\mathbb{K}$ podem definir-hi una operació suma, que podem expressar de forma gràfica o de forma algebraica, tal i com es mostrarà a continuació.

2.2.1 Mètode de la corda i la tangent

Aquest mètode consisteix en traçar una recta que uneixi dos punts P, Q que són els que volem sumar. Aquesta recta talla un tercer punt de la corba, el qual anomenarem R. Sabem que hi haurà aquest tercer punt de tall perquè si tenim una corba de grau 3 i una recta de grau 1, aleshores hi haurà 3 punts d'intersecció entre ambdues. Si P i Q són el mateix punt, aleshores s'agafaria com a recta la tangent a la corba en el punt.

El punt $P + Q$ (o el punt $2 \cdot P = P + P$) és el punt d'intersecció de la corba amb la recta que passa per R i $\mathcal{O}$, és a dir, la recta que passa per R i es

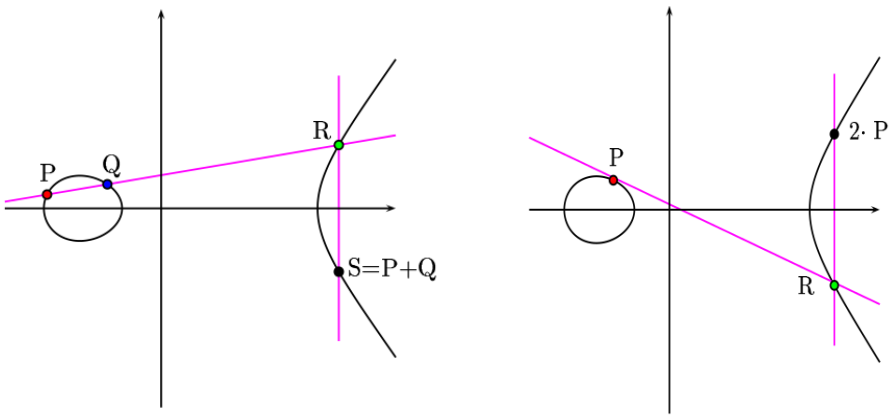

Figura 2.2: P+Q i P+P amb el mètode la corda i la tangent

paral·lela a l'eix d'ordenades. En la Figura 2.2 es pot observar com a partir dels punts P i Q de la corba i el punt R també de la corba, aconseguim generar el punt S que serà el punt suma de $P+Q$. També es pot veure, de manera similar, com funciona el doblatge del punt P també utilitzant el mateix mètode.

Amb aquesta operació, $(E/\mathbb{K}, +)$ té estructura de grup abelià amb el punt de l'infinit $\mathcal{O}$, com a punt neutre.

2.2.2 Càlcul algebraic de la suma de dos punts

Les expressions de les coordenades del punt $P+Q=(x_3,y_3)$, si $P+Q\neq\mathcal{O}$, en termes de $P=(x_1,y_1)$ i $Q=(x_2,y_2)$, venen donades per:

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2, \\ y_3 &= (x_1 - x_3)\lambda - y_1, \end{aligned}$$

on λ és la pendent de la recta tangent en P i el seu valor és:

- $\lambda = \frac{(y_1-y_2)}{(x_1-x_2)}$ si $x_1 \neq x_2$ o
- $\lambda = \frac{(3x_1^2+a)}{(2y_1)}$ si $x_1 = x_2$ i $y_1 \neq 0$

Substituint el valor de λ en les equacions anteriors, obtindrem les coordenades del punt suma.

2.2.3 Múltiples d'un punt

Utilitzant l'operació suma definida anteriorment, es pot definir el producte de $n \in \mathbb{K}$ i un punt $P \in E/\mathbb{K}$ de la següent forma:

$$n \cdot P = \begin{cases} \overbrace{P + P + P + ... + P}^{n \ vegades} & si \ \ n > 0, \\ \overbrace{(-P) + ... + (-P)}^{|n| \ vegades} & si \ \ n < 0, \\ \mathcal{O} & si \ \ n = 0. \end{cases}$$

Quan tenim un enter n molt gran, aquest mètode és molt ineficient, ja que té cost $O(n)$, que és molt més gran que el cost que tindria per exemple un algoritme com el **camperol rus** que tindria un cost de $O(\log_2 n)$.

2.3 Corbes el·líptiques sobre cossos finits

Una corba el·líptica sobre un cos finit $\mathbb{F}_q$, anomenada a partir d'ara $E/\mathbb{F}_q$, on $q = p^m$ i p és un nombre primer, ve definida per una equació de la forma:

$$y^2 = x^3 + ax + b \tag{2.4}$$

on a, b són elements de $\mathbb{F}_q$ i $4a^3 + 27b^2 \not\equiv 0$. Aquestes corbes proporcionen grups finits de gran interès criptogràfic degut a la dificultat que té el problema del logaritme discret plantejat sobre ells. Normalment es consideren com a cos base $\mathbb{F}_p$, amb una p gran, o $\mathbb{F}_{2^m}$, amb m gran.

Totes les propietats vistes anteriorment sobre cossos, també són aplicables a les corbes el·lítiques sobre cossos $\mathbb{F}_q$, a més a més d'una sèrie de propietats i característiques pròpies.

2.3.1 Algoritmes per calcular el cardinal

El cardinal d'una corba el·líptica $E/\mathbb{F}_q$, que denotarem per $\#E(\mathbb{F}_q)$ és el nombre de punts que conté la corba amb coordenades a $\mathbb{F}_q$, més el punt a l'infinit $\mathcal{O}$. Per tant, com a mínim el nombre de punts de la corba és $N \geq 1$. Prenem ara, $x \in \mathbb{F}_q$ (x pot prendre per valor q diferents valors), si $\exists y \in \mathbb{F}_q$ tal que $y^2 = x^3 + ax + b$, llavors $-y$ també compleix l'equació i, per tant, podem dir que $N \leq 1 + 2q$.

Per acotar més els càlculs del raonament anterior, s'utilitza el teorema de Hasse:

Teorema 1. *(de Hasse). Sigui E una corba el·líptica definida sobre un cos finit $\mathbb{F}_q$ i sigui $N = \#E(\mathbb{F}_q)$, llavors:*

$$q + 1 - 2\sqrt{q} \leq N \leq q + 1 + 2\sqrt{q}$$

és a dir, $N = q + 1 - t$ amb $|t| < 2\sqrt{q}$. A l'enter t se l'anomena traça de l'endomorfisme de Frobenius de $E/\mathbb{F}_q$.
Es diu que $E/\mathbb{F}_q$ és supersingular si $t \equiv 0 \pmod{q}$. En cas contrari es diu que és ordinària.

Teorema 2. *(de Cassels). Sigui E una corba el·líptica sobre $\mathbb{F}_q$.*

$$E(\mathbb{F}_q) \text{ és isomorf a } \begin{cases} \mathbb{Z}_m & \text{on } m = \#E(\mathbb{F}_q) \\ & \text{o} \\ \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} & \text{on } \begin{cases} m_1 \cdot m_2 = m = \#E(\mathbb{F}_q) \\ m_2 | m_1 \\ m_2 | (q-1) \end{cases} \end{cases}$$

El càlcul del cardinal d'una corba el·lípica és una tasca computacionalment molt costosa, és per això que el mètode del recompte exhaustiu no s'utilitza, sinó que s'utilitzen d'altres algorismes com el SEA(Schoof–Elkies–Atkin) de

Schoof [11] o el de Shanks-Mestre(o *baby-step giant-step)* [17]. A continuació es mostra el pseudocodi de l'algoritme de Schoof que és el que s'utilitza:

Algorisme 2.1 Algorisme de Schoof

INPUT: The prime power q and $E|\mathbb{F}_q$.
OUTPUT: The integer $t = q + 1 - \#E(\mathbb{F}_q)$

1. $l$max$\leftarrow$min$\{p \in \mathbb{P} : \prod_{l\in\mathbb{P}, l\leq \mathrm{p}} l > 4\sqrt{q}\}$.
2. If $2 \mid q$ then do:
   If $j(E) = 0$ then $t_2 \leftarrow 1$, else $t_2 \leftarrow 0$,
3. else do:
   If $\#E(\mathbb{F}_q)[2] = 1$ then $t_2 \leftarrow 1$ else $t_2 \leftarrow 0$.
4. For all $l \in \mathbb{P}$, $3 \leq l \leq lmax$, do:
   Take a random point $P \in E[l]\backslash\{\mathcal{O}\}$
   Compute $\varphi_q^2(P) + q_l P$ with $0 \leq q_1 < l$, $q_1 \equiv q \ (mod\ l)$
   For $\tau = 0$ to l do:
      Compute $\tau\varphi_q(P)$.
      If $\tau\varphi_q(P) = \varphi_q^2(P) + q_1 P$ then do:
         $t_l \leftarrow \tau$
         Go to to the next prime in Step 4
5. Use the Chinese Remainder Theorem to determine t with $|t| \leq 2\sqrt{q}$ and $t \equiv t_l \ (mod\ l)$ for all $l \in \mathbb{P}$, $2 \leq l \leq lmax$
6. Return t

L'algorisme de Shanks-Mestre té complexitat $O(q^{\frac{1}{4}+\varepsilon})$, on ε és una constant positiva que es pot fer arbitràriament petita. En aplicacions criptogràfiques s'utilitza un mètode amb complexitat $O(log^8 q)$. Aquesta millora es deu al treball de Schoof. La idea és computar l'ordre del grup mòdul primers petits i aleshores utlitzar el *Chinese Remainder Theorem (*Teorema xinès del resi-

du) per obtenir l'ordre exacte. Per computar l'ordre mòdul primers petits $l > 2$ apliquem l'endomorfisme de Frobenius de la corba.

En [20] es pot trobar una àmplia explicació sobre el funcionament de l'algoritme així com les demostracions necessàries per veure que l'algoritme funciona correctament. Aquí no s'entrarà en més detalls ja que no és la veritable finalitat del treball.

Capítol 3

Criptografia amb pairings

Cada vegada està creixent més l'interès en la criptografia, i cada vegada hi ha més esquemes i signatures basats en pairings. En aquest capítol explicarem què són els pairings, quines característiques tenen i una petita pinzellada de com els utilitzem en criptografia. Aquest és el capítol principal del treball, doncs tota la part d'implementació es basarà en la teoria que hi ha darrere d'aquest capítol.

3.1 Pairings

Definició. Considerem dos grups additius $(G_1, +)$ i $(G_2, +)$ amb neutre 0 i un grup multiplicatiu $(G_T, \cdot)$ amb neutre 1. Un pairing o aparellament és una aplicació

$$e : G_1 \times G_2 \to G_T$$

que satisfà les següents propietats:

1. (Bilinealitat).
   $\forall S_1, S_2 \in G_1, \forall T \in G_2, \forall a_1, a_2 \in \mathbb{Z}\,, e(a_1S_1+a_2S_2, T) = e(S_1, T)^{a_1} e(S_2, T)^{a_2}$
   $\forall S \in G_1, \forall T_1, T_2 \in G_2, \forall b_1, b_2 \in \mathbb{Z}\,, e(S, b_1T_1+b_2T_2) = e(S, T_1)^{b_1} e(S, T_2)^{b_2}$

2. (No degeneració). $\forall S \in G_1 \;\; S \neq 0$, existeix $T \in G_2$ tal que $e(S, T) \neq 1$.
   De la mateixa forma també tenim que:
   $\forall T \in G_2 \;\; T \neq 0$, existeix $S \in G_1$ tal que $e(S, T) \neq 1$.
   A més per a que es pugui utilitzar en criptografia, $e(S, T)$ ha de ser eficientment computable $\forall (S, T) \in G_1 \times G_2$.

Si $G_1 = G_2$ aleshores diem que l'aparellament és simètric. Altrament diem que l'aparellament és asimètric.

Classificació de pairings. Donat un aparellament $e : G_1 \times G_2 \to G_T$, es diu que és de:

- Tipus 1: Si $G_1 = G_2$, és a dir, si és un aparellament simètric.
- Tipus 2: Si $G_1 \neq G_2$ i hi ha un morfisme $\phi : G_2 \to G_1$ computable eficientment però no hi ha morfisme computable de G_1 cap a G_2.
- Tipus 3: Si $G_1 \neq G_2$ i no hi ha morfismes computables entre G_1 i G_2 .

En tots els casos existeix un morfisme entre G_1 i G_2 (són grups cíclics del mateix ordre) però computar aquests morfismes, aparentment és tant complex com computar logaritmes discrets en el grup.

Per a que un pairing sigui útil en criptografia cal que sigui computacionalment eficient i que el problema bilineal de Diffie-Hellman sigui computacionalment difícil. Aquest problema es pot enunciar com:

Donats un aparellament $e : G_1 \times G_2 \to G_T$, i els punts $P, aP, bP, cP \in G_1$ tals que $e(P, P) \neq 1$, computar $e(P, P)^{abc} \in G_2$.

Els pairings utilitzats actualment en criptografia són els basats en els pairings de Weil i Tate en corbes el·líptiques sobre cossos finits. Aquests pairings són aplicacions bilineals des d'un grup d'una corba el·líptica $E(\mathbb{F}_q)$ cap a un grup multiplicatiu $\mathbb{F}^*_{q^k}$. El paràmetre k és el que s'anomena grau d'immersió de la corba el·líptica.

Definició. (Grau d'immersió): Sigui $E/\mathbb{F}_q$ una corba el·líptica i sigui l un divisor de $N = \#E(\mathbb{F}_q)$ (habitualment l primer). S'anomena grau d'immersió de $E/\mathbb{F}_q$ respecte a l, al mínim enter positiu k verificant les condicions equivalents:

- $l \mid (q^k - 1)$
- $\mathbb{F}^*_{q^k}$ conté un subgrup cíclic d'ordre l .

Si l és el major divisor primer de N, k es denomina simplement grau d'immersió de $E/\mathbb{F}_q$. És per això que és molt important escollir de forma apropiada la corba per a que aquesta sigui prou forta als atacs i les operacions siguin computacionalment eficients.

En particular, per a la criptografia basada en la Identitat (la clau pública d'un usuari és el seu propi nom o qualsevol altre atribut lligat a ell mateix) requereix corbes amb un grau d'immersió petit. En particular, les corbes supersingulars (aquelles per les que $p \mid t$) són idònies per a aquest propòsit, ja que aquestes corbes tenen $k \leq 6$. Per contra, les corbes el·líptiques ordinàries amb grau d'immersió petit són una minoria, complicades de trobar i amb una caracterització complexa. En l'apartat 3.3, es farà un estudi més detallat d'aquest dos tipus de corbes.

El pairing es considera segur si, agafant logaritmes discrets en els grups $E(\mathbb{F}_q)$ i $\mathbb{F}^*_{q^k}$, són ambdós computacionalment no factibles. Per a un òptim comportament, els paràmetres q i k haurien de ser elegits de tal manera que els dos problemes de logaritmes discrets fossin aproximadament d'igual dificutat quan s'utilitzen els millors algoritmes coneguts, i l'ordre del grup $\#E(\mathbb{F}_q)$ hauria de tenir un factor primer r gran.

Aquests aparellaments de Weil i de Tate són aplicacions bilineals que assignen a un parell de punts (P, Q) de la corba E sobre $\mathbb{F}_q$, una arrel d'ordre l d'una extensió $\mathbb{F}_{q^k}$, on k és el grau d'immersió de la corba i l és un nombre primer i divisor del cardinal de la corba $N = \#E(\mathbb{F}_q)$. Agafant punts de l-torsió com a entrada i com a sortida elements d'un cos finit, els pairings es defineixen utilitzant funcions racionals.

Un càlcul efectiu dels aparellaments, es pot realitar utilitant l'algoritme de Miller [32].

3.2 Three Party Key Distribution

En aquest apartat s'intentarà explicar com funciona la mecànica de l'intercanvi a tres parts utilitzant una generalització de l'algoritme de Diffie-Hellman [27]. Més endavant (en el Capítol 5) es mostrarà el codi que s'ha utilitzat per veure com, utilitzant la llibreria jPBC, es pot fer una petita demostració

de la creació de claus, que és el més important d'aquest algoritme juntament amb la compartició de claus públiques per al càlcul de la clau compartida. En aquest apartat només es mostraran els conceptes teòrics que s'han aplicat de cara a la implementació.

3.2.1 Intercanvi de claus de Diffie-Hellman

La idea fonamental en la que es basa el protocol creat per Whitfield Diffie i Martin Hellman l'any 1976 [27], va ser la que va originar poc després els criptosistemes que coneixem com criptosistemes de clau pública, en els que no és necessari compartir cap informació secreta. La seguretat es basa en alguns problemes matemàtics subjacents difícils de computar, com pot ser el problema del logaritme discret.

Generació de claus

Per a l'intercanvi de claus de Diffie-Hellman entre dos usuaris A i B, hem de considerar un grup $(G, \cdot)$ d'ordre n i un generador g. Si A i B volen tenir una clau compartida en comú, han d'elegir dos enters a i b respectivament i, aleshores, A farà públic l'element $g^a \in G$ i B farà públic l'element $g^b \in G$. Ara A podrà fer el càlcul de $(g^b)^a$ i B podrà calcular $(g^a)^b$, i d'aquesta manera és com creen la clau compartida.

Obtenir $(g)^{ab}$ (coneixent g, g^a, g^b) és difícil en un grup G en el que el problema del logaritme és computacionalment difícil i, per tant, no podrem obtenir les claus privades a i b d'una manera senzilla i per conseqüent tampoc la clau compartida.

3.2.2 Objectiu del problema

Aquest problema involucrarà tres participants, que he anomenat per distingir-los Alice (A) , Bob (B) i Charlie (C). El que volen fer aquests tres participants és comunicar-se entre ells en un canal en el que, només ells tres tinguin la clau compartida per encriptar i desencriptar els missatges. Per fer això del que bàsicament ens hem d'aprofitar és de la propietat bilineal que disposen els aparellaments.

3.2.3 Paràmetres del sistema

Durant tot el problema, l'element bàsic sobre el qual treballarem és un aparellament bilineal $e : G_1 \times G_1 \to G_2$ i recolzant-nos en un generador P de $(G_1, +)$. Com es pot veure, l'aparellament que utilitzarem és un aparellament simètric, i més concretament, en la implementació s'ha utilitzat, de tots els que s'han descrit en l'apartat 5.1, l'aparellament amb una corba el·líptica de tipus A, perquè és el que més s'adequa a les necessitats que tenim aquí.

3.2.4 Generació de les claus privades i públiques

Per buscar el generador P, cal tenir clar que, en un grup cíclic com G_1 si l'ordre del grup és primer, cada element diferent del neutre pot ser un generador. El tipus A de jPBC ens assegura que l'ordre serà primer, per tant podem prendre com a generador un element aleatori del grup.

Després de buscar un generador P de $(G_1, +)$, necessitarem deixar clara la nomenclatura de les claus que calcularem dels usuaris:

- **Alice**: Una clau privada a i una clau pública $aP \in G_1$.
- **Bob**: Una clau privada b i una clau pública $bP \in G_1$.
- **Charlie**: Una clau privada c i una clau pública $cP \in G_1$.

De la mateixa manera que hem fet amb el generador, amb les claus privades a, b i c podem fer el mateix i treure un element aleatori de G_1. I pel que fa a les claus públiques, que seran les que s'enviaran a la resta de participants perque tothom pugui fer el posterior càlcul de la clau compartida, únicament hem de fer el càlcul indicat, que en aquest cas és el producte de l'enter corresponent amb el generador.

3.2.5 Generació de la clau compartida

El càlcul que haurà de fer cada participant per obtenir la clau compartida, haurà de realitzar-se després de rebre la clau pública de cada un dels altres dos participants. Així doncs els càlculs que es faran seran els següents:

- **Alice**: $e(bP, cP)^a$

- **Bob:** $e(aP, cP)^b$
- **Charlie:** $e(aP, bP)^c$

Amb aquests càlculs, cada un dels participants genera una clau compartida igual per a tots tres, doncs es compleix com hem dit abans, la bilinealitat del pairing:

$$e(bP, cP)^a = e(aP, cP)^b = e(aP, bP)^c = e(P, P)^{abc}.$$

En aquest cas, per garantir la seguretat del protocol hem de suposar que el problema bilineal de Diffie-Hellman per a l'aparellament e és computacionalment difícil, és a dir, donats P, aP, bP, cP és difícil determinar $e(P, P)^{abc}$.

3.3 Corbes el·líptiques pairing-friendly

Les corbes el·líptiques amb graus d'immersió petits i subgrups d'ordre primer grans són els ingredients clau per a les implementacions de criptosistemes basats en pairings. En aquest apartat es descriuran àmpliament algunes famílies de corbes el·liptiques que tenen graus d'immersió petits, que han anat sortint durant definicions prèvies, sense aprofundir del tot en el seu significat ni en les seves característiques. Aquestes corbes són les que anomenem *pairing-friendly curves*.

3.3.1 Corbes supersingulars

Una corba el·líptica $E/\mathbb{F}_{p^m}$ es diu que és supersingular si satisfà que

$$p \mid (p^m + 1 - \#E(\mathbb{F}_{p^m})).$$

Totes les corbes el·líptiques supersingulars tenen grau d'immersió $k \leq 6$ i, per tant, són *pairing-friendly.* Per qualsevol corba supersingular, els pairings de Tate o Weil representen un pairing criptogràfic sobre $E[n]$, on n és un divisor de $\#E(\mathbb{F}_q^*)$. A més a més, un pairing $\hat{e}$ de tipus 1 sobre la corba E pot obtenir-se utilitzant l'aplicació

$$\hat{e}(P, Q) = e(P, \Psi(Q)),$$

on e és el pairing de Weil o Tate usual i $\Psi : E(\mathbb{F}_{p^m}) \to E(\mathbb{F}_{p^m})$ és una aplicació de distorsió. El corresponent pairing $\hat{e}$ és conegut com el pairing de Weil (o Tate) modificat.

3.3.2 Corbes ordinàries

Algunes aplicacions com signatures curtes requereixen *pairing-friendly elliptic curves* amb grau d'immersió més gran que 6. A continuació es descriuran mètodes capaços de produir corbes el·líptiques amb un grau d'immersió superior a 6. El primer, el mètode Cocks-Pinch [30] produeix corbes el·líptiques ordinàries (aquelles que no són supersingulars) amb graus d'immersió arbitraris. La segona construcció, la de Barreto-Naehrig [13], produeix corbes el·líptiques amb grau d'immersió $k = 12$ i d'ordre primer.

Mètode Cocks-Pinch

El mètode de Cocks-Pinch produeix corbes el·líptiques ordinàries que tenen un grau d'immersió arbitrari. És considerat el mètode més flexible per construir corbes ordinàries *pairing-friendly*.

Suposem que volem construir una corba E sobre $\mathbb{F}_q$ de grau d'immersió k i que el seu cardinal factoritza en un primer gran l. Aquest mètode es basa en què si n és el cardinal de la corba, aleshores $l \mid n = (p + 1 - t)$ i $l \mid (p^k - 1)$.

Corbes Barreto-Naehrig

La família de corbes el·líptiques de Barreto-Naehrig [13] tenen un grau d'immersió $k = 12$ i a més a més tenen ordre primer.
Per a generar-les, es consideren els polinomis:

$$\begin{aligned} N(x) &= 36x^4 + 36x^3 + 18x^2 + 6x + 1 \\ P(x) &= 36x^4 + 36x^3 + 24x^2 + 6x + 1 \end{aligned}$$

i s'agafa un valor de x_0 per al que tant $n = N(x_0)$ i $p = P(x_0)$ siguin primers. (Un exemple podria ser $x_0 = 82$ per al que tindríem $n = 1647609109$ i $p = 1647649453$, ambdós primers).

Les corbes que es construeixen amb aquest mètode, tenen equació $y^2 = x^3 + b$.

Així, busquem ara un valor $b \in \mathbb{F}_p$ per al que $b+1$ sigui un residu quadràtic en $\mathbb{F}_p$, i el punt $Q = (1, \sqrt{b+1})$ en la corba el·líptica $E : y^2 = x^3 + b$ satisfaci que $n \cdot Q = \mathcal{O}$. El procediment de cerca podria ser tant simple com començar per $b = 1$ i incrementar b gradualment fins que el valor adequat es trobi. Per a tal valor b, la corba $E/\mathbb{F}_p$ que ve donada per l'equació $y^2 = x^3 + b$ té n punts, grau d'immersió $k = 12$ i el punt $Q = (1, \sqrt{b+1})$ pot ser agafat com a punt base.

Capítol 4

Criptografia basada en la Identitat

Per evitar problemes amb l'autentificació de les claus públiques (certificats i autoritats de certificació) que planteja la criptografia de clau pública clàssica de Shamir, ell mateix l'any 1984 va proposar un nou paradigma: la criptografia basada en la Identitat [9]; en la qual la clau pública d'un usuari és el seu propi nom o qualsevol altre atribut lligat a ell mateix. Exemples d'aquests atributs poden ser les direccions de correu electrònic, números de telèfon, IP's o noms de dominis entre altres. Aquests atributs, a part de ser únics (no es poden repetir o duplicar), cada un pertany a una entitat en concret, que serà la que estarà en una banda del canal de comunicació.

Aquest esquema el que permet és comunicar dos usuaris de manera segura i verificar cada un la signatura de l'altre sense haver de fer intercanvis de claus (públiques o privades), sense haver de mantenir directoris de claus i sense haver d'utilitzar el servei d'una tercera part. El que sí que assumeix l'esquema, és l'existència de centres de generació de claus, de confiança, la finalitat dels quals és donar a cada usuari una "targeta intel·ligent personalitzada" quan s'uneix per primera vegada al sistema. La informació que conté aquesta targeta permet a l'usuari signar i encriptar els missatges que envia i desencriptar i verificar els missatges que rep d'una manera independent, sense haver de tenir en compte la identitat de l'altra part.

L'esquema és ideal per grups tancats d'usuaris com poden ser executius d'una multinacional o les sucursals d'un gran banc, ja que l'oficina central de

la companyia pot servir com a generador central de claus en el que tothom confia. Aquest esquema pot ser la base per a un nou tipus de targeta d'identificació personal amb la qual, cadascú, electrònicament pot firmar xecs, rebuts bancaris, documents legals i correus electrònics.

4.1 Esquema bàsic de funcionament

A partir d'ara anomenarem al generador de claus privades **PKG** (Private Key Generator). Abans de que les operacions puguin començar, el PKG ha de generar un parell de claus: una pública i una altra privada, que s'anomenen sk_{PKG} i pk_{PKG} respectivament i un enter que anomenarem a partir d'ara s. Aquestes claus són el que s'anomenen clau pública mestra i clau privada mestra. Per operar, el PKG el primer que fa és publicar sk_{PKG} als usuaris dels seus serveis, per a que així tots puguin calcular la clau pública corresponent a una entitat combinant la clau pública mestra amb la cadena identificativa de l'entitat. Per obtenir la corresponent clau privada, la part autoritzada a utilitzar-la, contacta amb el PKG per a que utilitzi s per a generar la clau privada per a l'entitat ID.

4.1.1 Encriptació basada en la Identitat (IBE)

A continuació s'exposa un exemple teòric en el que es pot veure la finalitat de l'encriptació basada en la Identitat i com treballa a nivell teòric:

1. L'Alice vol enviar un missatge M al Bob. Ella utilitza la identitat del Bob que anomenarem ID_{BOB} i la clau pública mestra pk_{PKG} per encriptar M, obtenint un missatge encriptat C. L'Alice li envia al Bob el missatge C. Es veu com, abans de l'encriptació, l'Alice ja coneixia ID_{BOB} i sk_{PKG} , així doncs, no ha estat necessària l'aportació d'informació/preparació d'en Bob.

2. En Bob rep el missatge de l'Alice. En algunes implementacions s'assumeix que C s'envia amb instruccions per contactar amb el PKG per agafar la clau privada requerida per desencriptar. En Bob s'autentifica amb el PKG, bàsicament enviant la suficient informació/prova de que a ID_{BOB} li pertany a ell. Un cop fet, el PKG li transmet al Bob la seva clau privada pk_{PKG} per un canal segur.

3. En Bob desencripta C utilitzant la seva clau privada $pk_{ID_{BOB}}$ per recuperar el missatge original M.

Una variació podria ser que el PKG desencriptés C per a en Bob i ja li transmitís a ell després d'autentificar-se, per a que el sistema encara fós més transparent.

En la figura 4.1 es pot observar de manera senzilla, sense entrar en detalls, com funciona el procés en cas de voler enviar un correu electrònic que és el mateix cas que s'ha suposat en la implementació. En aquest cas l'Alice és la que vol enviar el missatge al Bob i utilitza la seva identitat (el seu correu electrònic).

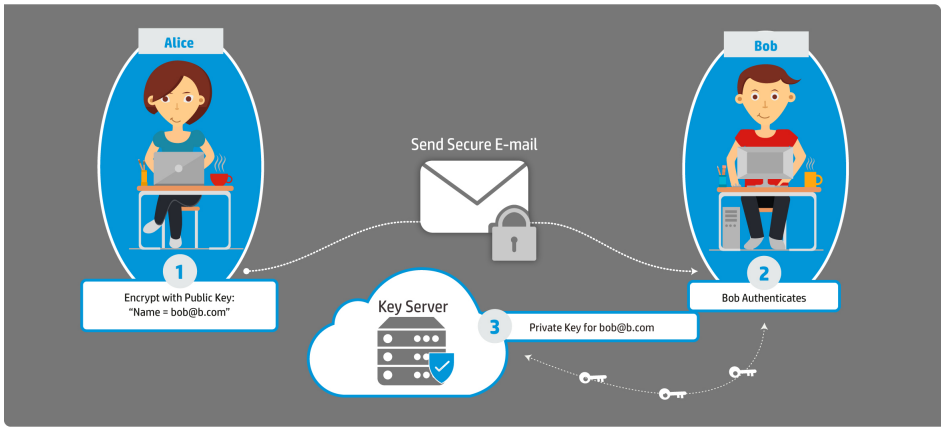

Figura 4.1: Procés de l'enviament d'un missatge utilitzant IBE

4.1.2 Pros i contres de l'Encriptació Basada en la Identitat (IBE)

- **PROS**
  - No es necessiten certificats. A partir de la identitat d'una persona (com pot ser el seu mail, el seu nom o alguna altra dada que l'identifiqui de forma única), es pot obtenir la seva clau pública.
  - No es necessita una presa de contacte prèvia abans de la recepció/enviament de missatges per part d'ambdues parts.

  - Les claus caduquen, pel que no necessiten ser anul·lades. En un sistema de clau pública tradicional, les claus hauran de ser anul·lades si són compromeses.
  - Hi ha baixa vulnerabilitat per spam.
  - Permet l'expiració automàtica, fent que els missatges no siguin "llegibles" passat un cert temps.

- **CONTRES**

  - Requereix un servidor centralitzat. L'enfoc centralitzat implica que algunes claus hauran de ser creades i mantingudes "sota custodia" i tenen, per tant, un risc més alt de divulgació.
  - Requereix un canal segur entre emisor/receptor del missatge i el servidor IBE per poder transmetre de manera segura la clau privada, requerida pel receptor en el moment de desencriptar. En cas de voler delegar feina (de l'usuari cap al servidor central) podem també fer que sigui el servidor central el que desencripti el missatge abans de guardar-lo en la base de dades i que es demani una validació per part de l'usuari receptor per a saber que el missatge original anava destinat cap a ell.

4.1.3 Ús d'aparellaments en IBE

A continuació s'enumeraran les parts principals de l'esquema d'encriptació basada en la identitat [16, 22] proposat per Boneh-Franklin, però pensant ara ja, amb una implementació en la que es requeriran aparellamanets bilineals per a que tot funcioni correctament:

- Paràmetres del sistema:

  - Pairing $e : G_1 \times G_1 \to G_T$
  - G_1 un grup additiu.
  - G_T és un grup multiplicatiu

- El **PKG** (Private Key Generator) disposarà de :

  - Un generador P_k de G_1.

  - Una clau secreta global s (és un enter), una clau pública mestra $S_k = s \cdot P_k$

- Es disposarà en tot moment d'una funció hash $h : G_T \to \{0,1\}^n$.
- El PKG generarà **per a cada usuari** les següents claus:
  - Un usuari A té una clau pública $Q_A \in G_1$
  - La clau privada de l'usuari A és $D_A = s \cdot Q_A$
- Si volem, per exemple, **encriptar** un missatge m per enviar-li a l'usuari A seguirem els següents passos:
  - Escollir un enter aleatori que anomenarem r.
  - Calcular la tupla $(U, V) = (r \cdot P_k, m \oplus h(e(Q_A, s \cdot P_k)^r))$.
  - Enviar la tupla (U, V) a l'usuari A.
- En el moment que A vol desencriptar el missatge que se li ha enviat, usant la seva clau privada D_A, farà:
  - Recupera el missatge m calculant: $m = V \oplus h(e(D_A, U))$.
    - Cal notar que és compleix la següent propietat:

$$e(D_A, U) = e(s \cdot Q_A, r \cdot P_k) = e(Q_A, s \cdot P_k)^r = e(Q_A, S_k)^r$$

4.2 Signatura basada en la Identitat (IBS)

En aquesta secció es descriurà el procediment que es duu a terme en la signatura basada en la identitat que va proposar Shamir per permetre que els usuaris verifiquessin una signatura utlitzant informació pública com pot ser l'identificador d'un usuari.

1. L'Alice s'autentifica amb el PKG i rep la seva clau privada $pk_{ID_{ALICE}}$.
2. Utilitzant la seva clau privada $pk_{ID_{ALICE}}$, l'Alice genera una signatura σ per a un missatge M i li transmet a en Bob.

3. Després de rebre M i σ de l'Alice, en Bob comprova que σ sigui autèntica en M utilitzant la identitat de l'Alice ID_{ALICE} i la clau pública mestra sk_{PKG}. Si és autèntica envia "Accept". En cas contrari envia "Reject". Cal veure que en Bob no necessita tenir cap tipus de certificat de l'Alice.

Aquesta va ser la forma que va proposar Shamir d'utilitzar l'algoritme RSA per a la firma electrònica, però es va haver d'esperar fins al 2001 per a que Boneh i Franklin proposessin sistemes per aconseguir sistemes de xifrat basat en la identitat o IBE (Identity-Based Encryption).

4.2.1 Signatura de Boneh, Lynn i Shacham

En aquesta secció es descriuran les principals característiques de la signatura descrita per Boneh, Lynn i Shacham [34], que permet que un usuari verifiqui si una signatura és autèntica. Aquest esquema té signatures amb 170 bits de longitud envers els 320 bits que tenen les signatures DSA.

Paràmetres:

- Tenim un pairing $e : G_1 \times G_1 \to G_2$
- G_1 i G_2 són grups d'ordre primer r
- P és un generador de G_1
- h és la funció de codificació i $h : \{0,1\}^n \to G_1$

Algoritme de generació de signatura

1. INPUT: Paràmetres anteriors (G_1, G_2, e, P, r), la clau privada d i el missatge M
2. OUTPUT: El missatge M amb la signatura σ
3. Procediment:
   (a) Calcular el hash del missatge: $H = h(M) \in G_1$.
   (b) Calcular $\sigma = dH \in G_1$
   (c) Retornar M i σ

Algoritme de verificació de la signatura

1. INPUT: Els paràmetres (G_1, G_2, e, P, r), la clau pública Q i M amb σ
2. OUTPUT: Acceptació o rebuig de la signatura
3. Procediment:
   (a) Calcular el hash del missatge : $H = h(M) \in G_1$
   (b) Si $e(\sigma, P) = e(H, Q)$ aleshores retorna "signatura acceptada"

Cal tenir present que $e(\sigma, P) = e(dH, P) = e(H, dP) = e(H, Q)$

Capítol 5

Implementació d'una aplicació d'e-mail

En aquest capítol s'intentaran introduïr els mètodes bàsics que s'han utilitzat durant la implementació per a treballar amb pairings amb la llibreria jPBC (*Java Pairing Based Cryptography)*[14, 15], no sense abans fer una petita pinzellada als tipus de corbes que ens dóna la llibreria i quines característiques principals presenta cada una. Així mateix, es farà una explicació extensa de les dues implementacions que s'han dut a terme durant el treball: intercanvi de claus a tres bandes i la implementació d'un prototip de correu electrònic utilitzant encriptació basada en la identitat. Cal comentar abans de poder avançar que s'ha treballat en tot moment amb l'entorn de desenvolupament Netbeans i que s'ha escollit Java com a llenguatge de desenvolupament, d'una banda per la facilitat de programació i d'altra banda perquè la llibreria jPBC (sobre Java) presentava una manera molt senzilla de treballar amb pairings.

5.1 Tipus de corbes suportades per la jPBC

La llibreria jPBC suporta els següents tipus de corbes:

1. Tipus A: Són corbes supersingulars d'equació $y^2 = x^3 + ax + b,\ \forall a$ sobre $\mathbb{F}_q$ amb grau d'immersió $k = 2$.

2. Tipus A1: Són les mateixes corbes utilizades que pel tipus A. S'utilitzen quan el criptosistema necessita que el cardinal de la corba sigui un

nombre específic, per exemple $N = p \cdot q$ amb p i q nombres primers grans, fent que aquest N sigui difícil de factoritzar.

3. Tipus D: Són corbes amb grau d'immersió $k = 6$. Es poden construir amb el mètode CM.

4. Tipus E: Són corbes amb grau d'immersió $k = 1$ construides utilitzant el mètode CM. Tots els còmputs necessaris per fer el pairing es poden realitzar sobre $\mathbb{F}_q$.

5. Tipus F: Són una família de corbes amb grau d'immersió $k = 12$. Van ser descobertes per Barreto i Naehrig [13] utilitzant també el mètode CM i considerant els polinomis ciclotòmics.

6. Tipus G: Són corbes amb un grau d'immersió $k = 10$. Van ser descobertes per Freeman [33].

5.2 Operacions de la jPBC bàsiques per a pairings

En aquesta secció desciurem els mètodes bàsiques de la jPBC per a treballar amb pairings:

1. **Utilitzar les funcions implementades en C quan sigui possible**. La llibreria jPBC, es basa en la llibreria PBC que va desenvolupar Ben Lynn [31] utilitzant el llenguatge C. La llibreria jPBC importa tot el codi de la llibreria PBC i, a vegades es dóna el cas de que la jPBC crida a la llibreria PBC degut a que C és més òptim que Java en algunes operacions. Degut a que Java és un llenguatge "lent" respecte d'altres com per exemple C, s'ha donat la possibilitat a la llibreria jPBC de tenir operacions de pre-processament de les operacions d'exponenciació i de pairings, molt útils quan aquestes operacions s'han de repetir al llarg del codi. Així doncs per utilitzar les funcions que provenen de la llibreria PBC s'ha d'activar de la següent forma:

   ```
   PairingFactory.getInstance().setUsePBCWhenPossible(true);
   ```

2. **Generació aleatòria** d'una corba de tipus A, que és el tipus que s'ha utilitzat en ambdues implementacions:

```
PairingParametersGenerator ppg = new
    TypeACurveGenerator(ThreeParty.rBits,ThreeParty.qBits);
Pairing e = PairingFactory.getPairing(ppg.generate(), null);
```

3. **Càrrega de fitxer amb els paràmetres de l'aparellament**

   Partim de l'existència d'un fitxer de configuració com podria ser el següent, amb la configuració rBits=128 i qBits=256, que és l'utilitzat en les proves de l'aplicació de correu:

   ```
   type a
   q 78910235960302648236085819294375443237752251445803596769379
   72552555413508927
   r 170141183460469231731687303715917660161
   h 463792682966947442862892955073819492 48
   exp1 25
   exp2 127
   sign0 1
   sign1 1
   ```

   que s'ha generat volcant en un fitxer el resultat de ppg.generate() del codi del punt anterior.
   L'avantatge que ens dóna això és el poder disposar en tot moment del mateix pairing, sense que vagi canviant en cada execució del programa. Per carregar-lo al programa el que fem és el següent:

   ```
   Pairing e = PairingFactory.getPairing("params.params");
   ```

   on el paràmetre <params.params> és el fitxer amb la forma descrita anteriorment.

4. **Obtenir els camps** (estructures algebraiques) que intervenen en un pairing:

   ```
   Field Zr = e.getZr();
   Field G1 = e.getG1();
   Field G2 = e.getG2();
   Field H = e.getGT();
   ```

5. **Obenir un element aleatori d'un camp**. En el codi següent es mostra com obtenir un element aleatori de G_1:

```
Element el = G1.newRandomElement();
```

6. **Fer que un element sigui immutable**, és a dir, que el seu valor no varïi:

   ```
   el = el.getImmutable();
   ```

7. **Multiplicació d'un element per un enter**

   ```
   int integer = 4;
   Element p = e.getG2().newRandomElement().getImmutable();
   p = p.mul(new BigInteger(String.valueOf(integer)))
   ```

8. **Obtenir el pairing de dos elements**

   La interfície Pairing ens dóna mètodes per aplicar la funció pairing. Donats dos elements Element in1, in2, cada un pertanyent a l'estructura algebraica adequada, el seu pairing pot ser computat invocant el mètode pairing amb in1, in2 com a imputs de la següent forma:

   ```
   Element out = pairing.pairing(in1, in2);
   ```

9. **Potència d'un element a z,** on z és un element de $\mathbb{Z}_r$:

   ```
   Element e = G1.newRandomElement();
   Element z = pairing.getZr().newRandomElement();
   e.powZn(z);
   ```

10. **Obtenir la representació en bytes d'un element:**

    ```
    byte[] bytes = e.toBytes();
    ```

11. **Recuperació d'un element a partir dels bytes**:

    ```
    Element U = e.getG1().newElementFromBytes(bytes)
    ```

Per a veure com treballar d'una manera senzilla amb algunes d'aquestes operacions, en la propera secció s'explicarà el que s'ha implementat en el projecte adjunt anomenat TFGCryptography, en el fitxer ThreeParty. El codi es pot trobar allà, aquí només es donarà l'explicació de cada un dels components de la implementació.

5.3 Three-Party Key Distribution

En aquesta secció explicarem breument les decisions que hem pres a l'hora d'implementar la distribució de claus a tres bandes. Aquest problema ens servirà per donar una versió senzilla de la utilització de la llibreria, amb alguns dels mètodes propis més importants i que més utilitzarem sobretot en la implementació amb més pes que és l'aplicació de correu electrònic. Com ja hem explicat a la secció 3.2, cada usuari té una clau pública, una clau privada i calcula la clau compartida amb la resta de participants, que és la finalitat bàsica d'aquest problema. L'aplicació es basa principalment en quatre mètodes, cada un dels quals realitza les tasques següents:

- **setUp()** : Aquest mètode és el que s'encarrega de, a partir d'un Pairing e, trobar un generador que, en aquest cas com que l'ordre del grup és primer, tots els elements llevat del neutre són generadors. A més a més també crida als tres següents mètodes per a generar les claus dels participants i retornar una llista de Participant, amb tota la informació necessària de cada participant de la comunicació.

- **generatePrivateKeys()** : Per generar les claus privades, tal com ja hem explicat a la secció 3.2, únicament hem d'agafar un Element de $\mathbb{Z}_r$ per a cada participant.

- **generatePublicKeys()** : Per generar les claus públiques però, hem d'agafar la clau privada de cada participant i multiplicar-la pel generador de G_1 que hem trobar anteriorment.

- **generateSharedKeys()** : Per generar la clau compartida, cada participant el que haurà de fer serà :

  ```
  Element e = p.pairing(p1, p2).getImmutable();
  Element sharedKey = e.powZn(pk);
  ```

  és a dir, obtenir el pairing de p_1 i p_2 (en aquest cas aquests dos elements són les claus públiques dels altres dos participants) i elevar-ho a la clau privada pròpia.
  Cada participant farà el mateix i aquest Element sharedKey serà igual en els tres participants, demostrant doncs la certesa de la propietat de la bilinealitat dels pairings:

$$e(bP, cP)^a = e(aP, cP)^b = e(aP, bP)^c = e(P, P)^{abc}$$

5.4 Desenvolupament d'una aplicació d'e-mail

En aquesta secció s'explicarà de manera més detallada l'aplicació que dóna nom al treball; una aplicació implementada amb Java aprofitant-nos de les facilitats que ens dóna la llibreria jPBC explicada en la secció anterior, que intenta simular una aplicació de correu electrònic que s'alimenta de la seguretat que ens dóna l'encriptació basada en la Identitat. El codi està disponible en el projecte que porta per nom IBEMail, adjunt al present document. En la següent secció no s'explicarà el codi línia per línia, si no que s'explicarà la lògica del programa així com les solucions que s'han trobat a alguns problemes durant la implementació i quines decisions s'han pres i per quins motius.

5.4.1 Encriptació basada en Identitat - IBEMail

Aquesta aplicació és una mica més elaborada que la de Three-Party, tant per lògica de programa com per la utilització de pairings, i per tant, hem de fer ènfasi en unes quantes qüestions més:

1. **Classes principals de la implementació, situades en el paquet ibemail.**

   (a) **La classe PKG.** És la classe encarregada de generar les claus privades i públiques dels usuaris i emmagatzemar-les a la base de dades corresponent.

   (b) **La classe User**. Aquesta classe és la que representa un usuari del sistema. A més a més és la que s'encarregarà de tenir els mètodes de encriptar i desencriptar els missatges.

   (c) **La classe Pair.** És la classe encarregada de representar una parella de claus (pública i privada).

   (d) **La classe ElementPair.** És la classe encarregada d'emmagatzemar la parella (U,V) que és la que es genera en el mètode d'encriptació i que a més a més és la que s'envia al receptor.

   (e) **La classe MasterInfo.** És la classe que guarda la informació del master del sistema. Aquesta informació és clau púbplica, clau privada i l'enter s secret, que servirà per a generar les claus privades dels usuaris del sistema.

(f) **La classe ContentDB.** Aquesta classe s'explicarà més detalladament en el proper apartat, però pel nom ja es pot deduïr que és la classe que contindrà tota la informació necessària per treballar amb una Base de dades.

2. **Classes del paquet frontEnd.** Les classes que estan en aquest paquet són les classes que formen els Frames que serviran per a la part visual del programa, i que seran les que permetran que es pugi interactuar amb el programa.

3. **Ús de Bases de Dades** per emmagatzemar claus dels usuaris registrats en l'aplicació. En la primera part de la implementació, vam començar utilitzant el més senzill, que era un fitxer SQLITE el qual suportava la insersió d'elements i la seva recuperació, però el principal problema era que no permitia fer consultes concurrents, ja que el fitxer acabava bloquejant-se. Davant aquest problema es va optar per crear una Base de Dades en local utilitzant MySQL, que un dels advantatges que tenia era la fàcil insersió dins del projecte Netbeans i la fàcil utilització. Per gestionar la base de dades s'ha utilitzat el software MySQLWorkbench per a Mac OS X El Capitan 10.11.4.

   (a) **Taules** que formen la base de dades:

   Taula **USER**. És la taula encarregada de l'emmagatzemament de la informació del registre dels usuaris (nom, mail i contrasenya, on mail és la clau primària).

   Taula **KEYS**. És la taula encarregada de l'emmagatzemament de les claus (pública i privada) dels usuaris i del mail (que és la clau primària de la taula).

   Taula **MAIL**. És la taula encarregada de l'emmagatzemament dels correus que s'envien, els quals es formen d'un identificador (clau primària), el mail emisor, el mail receptor, l'assumpte, la data i el missatge encriptat (consta de dos parts U,V que en la

secció 4.1 s'explica com generar-les).

Taula **MASTER**. És la taula encarregada de l'emmagatzemament de les tres claus que poseeix el Master, que és la secretKey, la publicKey i la privateKey.

(b) **La classe ContentDB** és l'encarregada del maneig de la base de dades, desde la connexió/desconnexió fins a tots els mètodes de insert/select.
Una cosa que cal deixar clara i que segurament és el més important, és com guardar un Element (com podria ser la clau pública o la privada) dins de la BBDD.

L'única manera de guardar un Element és amb la seva representació en bytes, per a que després la recuperació sigui ràpida i directa. Per fer això l'únic que fem és el que hem dit en la secció 5.2 apartat 10 i guardar-ho a la BBDD en forma de Blob. A continuació es mostra el codi utilitzat per l'aplicació per a inserir en la base de dades un nou usuari i les seves claus:

```
addKeys(String user, Element pk, Element sk) {
  try {
    String query = "INSERT INTO IBEMailDB.KEYS
    (Mail,PublicKey,PrivateKey) VALUES(?,?,?)";
    PreparedStatement stmt =
            db.prepareStatement(query);
    stmt.setString(1, user);
    stmt.setBytes(2, pk.toBytes());
    stmt.setBytes(3, sk.toBytes());
    stmt.executeUpdate();
    stmt.close();
    db.commit();
  } catch (Exception ex) {
      System.out.println("Some error during
         adding new Mail");
  }
}
```

Per recuperar-lo hem de jugar amb el mètode descrit en la secció

5.2 apartat 11 i treure l'element de la BBDD en forma d'Array de bytes. A continuació es mostra el codi que s'ha emprat per a recuperar la clau privada d'un usuari a partir del seu mail:

```
Element getPublicKeyFromID(String mail) {
  Pairing e =
    PairingFactory.getPairing("params.params");
  Element epk = e.getG1().newOneElement();
  try {
      Statement stmt = db.createStatement();
      String query = "Select PublicKey FROM
      IBEMailDB.KEYS where Mail='"+mail+"';";
      ResultSet rs = stmt.executeQuery(query);
      if (rs.next()) {
          byte[] pk = rs.getBytes("PublicKey");
          epk.setFromBytes(pk);
      }
      rs.close();
      stmt.close();
  } catch (SQLException ex) {
      return null;
  }
  return epk;
}
```

(c) **Generació de la clau pública d'un usuari a partir de la seva identitat.** Tal com s'ha explicat en la secció 4.1.1, teòricament quan un usuari A vol enviar un missatge a un usuari B, A(emisor) genera la clau pública de B a partir de la seva identitat ID_B, que en aquest cas seria el mail de B. La variació que s'ha fet en el programa és que l'usuari B demana a PKG la clau pública a partir de la identitat del receptor, i així, a part de reduir el temps (no s'ha de generar cada vegada la clau pública, sinó que es va buscar a la Base de dades) també evitem un ús excessiu de l'aparellament. El problema d'això, és que el sistema és encara més centralitzat del que seria el descrit teòricament, ja que també hauran de servir les claus públiques dels usuaris i també que es podria no veure com un esquema basat en identitat, però sí que les claus públiques es creen a partir de la identitat (en aquest cas el correu electrònic)

de cada participant. En la implementació, això es redueix a fer la següent operació:

```
Element publicKey = e.getG1().newRandomElement
    ().getImmutable();
```

S'ha decidit fer d'aquesta manera perquè, tot i que la llibreria ens dóna el mètode:

```
Element f = e.getG1().newElementFromBytes(
      user.getMail()).getImmutable();
```

aquest, no funciona com és esperat i sigui quina sigui l'entrada, es genera el punt infinit en tot moment. Així doncs el PKG utilitza el mètode anterior cada vegada que algún usuari es registri per a generar la clau pública associada a l'usuari. Aleshores, un cop es té la clau pública, per aconseguir la clau privada es realitza la següent operació:

```
Element privateKey = publicKey.mul(new
    BigInteger(String.valueOf(
      this.s))).getImmutable();
```

(d) **Funció de hash utilitzada.** Com a funció de hash s'ha utilitzat SHA-256. Com a característiques principals d'aquesta funció tenim:

i. Longitud de paraula de 32 bits.

ii. Longitud de sortida de 256 bits.

iii. La funció de compressió de SHA-256 opera un bloc de missatge de 512 bits i un valor de hash intermedi de 256 bits. Es tracta essencialment d'un algoritme de xifrat de blocs de 256 bits que encripta el valor intermedi del hash usilitzant el bloc del missatge com a clau.

iv. Operadors :

$\oplus$	XOR
$\wedge$	AND
$\vee$	OR
$\neg$	Operació bit a bit (bitwise complement)
$+$	Addició mod 2^{32}
R^n	Desplaçament a la dreta n bits
S^n	Rotació a la dreta de n bits

v. En el protocol Bitcoin, SHA-256 s'utilitza en la creació de claus o direccions públiques i en la mineria de Bitcoin.

(e) **Operador lògic XOR** utilitzat per a l'encriptació/desencriptació. Per definició, l'operador lògic XOR és una disjunció que únicament és certa quan les dues frases són diferents. Aquí com que el que havíem de fer era fer la operació amb els bytes procedents del missatge i els bytes procedents de fer l'aparellament havíem de tenir en compte que les longituds eren diferents i per tant treballar sobre això. El codi resultant utilitzat en la implementació, és el següent:

```
byte[] xor(byte[] a, byte[] key) {
  byte[] out = new byte[a.length];
  for (int i = 0; i < a.length; i++) {
    out[i] = (byte)(a[i]^key[i % key.length]);
  }
  return out;
}
```

4. **Aplicacions reals** que utilitzen aquest tipus d'encriptació.
Després d'estar buscant informació per internet, vaig topar amb que l'empresa Hewlett Packard Enterprise [2] (un dels fundadors és Dan Boneh) va basar-se en IBE per a crear el seu propi sistema anomenat HPE Identity-Based Encryption. El que fan en aquesta empresa americana és garantir la protecció de dades procedents de correus electrònics, carpetes, documents i bases de dades i dades en el cloud en general, sense la necessitat d'utilitzar certificats de claus, i oferir els seus serveis a empreses o particulars.
Actualment estan treballant amb dades procedents de 100 milions d'usuaris arreu del món, garanteixen la seguretat de bilions de transaccions i permeten que més de mil empreses s'aprofitin dels seus productes i serveis.

El que vull dir amb aquest exemple d'aplicació és que el tema sobre el qual ha estat girant tot el meu treball no és un tema teòric sense aportacions pràctiques, si no que s'utilitza i facilita la tasca de garantir la seguretat i l'integritat de les dades en el món real.

5.4.2 Funcionament de l'aplicació de correu

En aquesta secció mostrarem algunes imatges del funcionament de l'aplicació de correu en una possible execució utilitzant com a paràmetres de l'aparellament els que hem de finit en la secció 5.2 en l'exemple de fitxer.

Generació de la clau privada i pública del PKG

Quan iniciem el programa el primer que es fa és generar un parell de claus pública/privada per al PKG així com l'enter s.

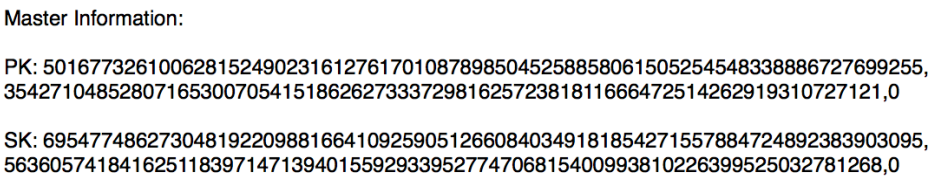

Figura 5.1: Generació de claus per al PKG

Pantalla d'inici de l'aplicació

Després de mostrar les claus generades per al master, entrem a l'aplicació i el primer que ens trobem és la pantalla inicial:

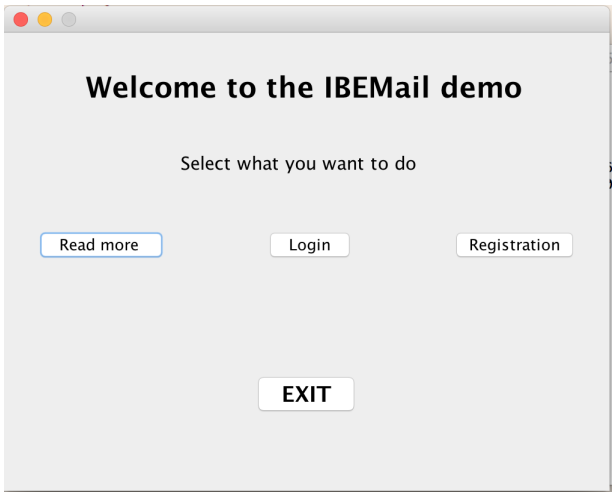

Figura 5.2: Pantalla d'inici de l'aplicació

Regitre d'un nou usuari

Entrant a l'apartat de *Registration* de la pantalla principal ens trobem un formulari com el següent:

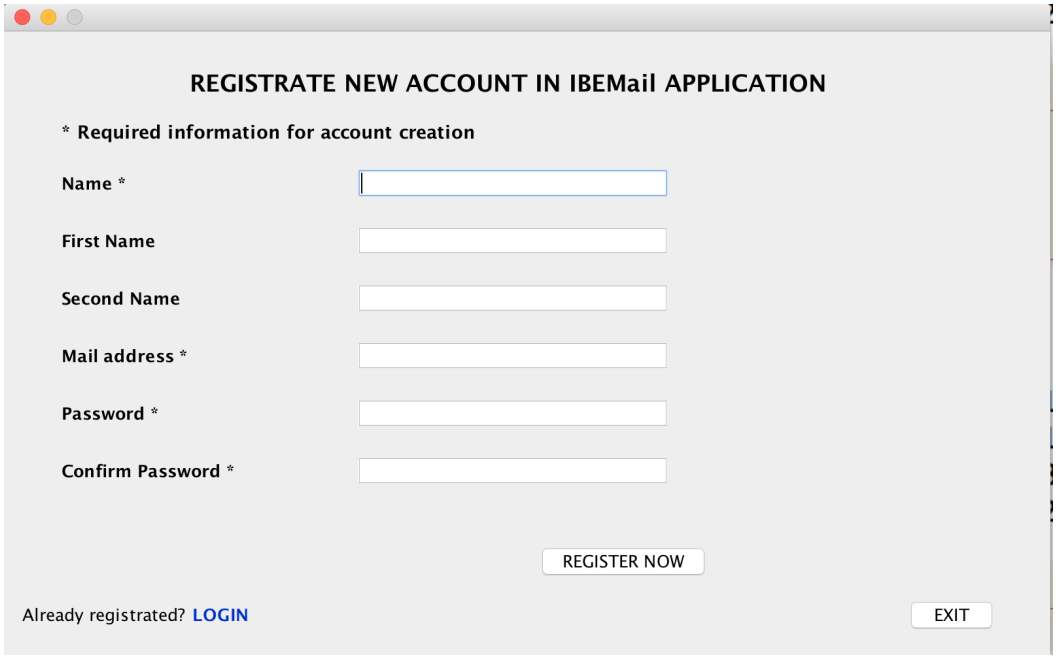

Figura 5.3: Formulari de registre de l'aplicació de correu

Generació de claus d'un nou usuari

Després d'escriure les nostres dades personals i de rebre la confirmació per pantalla de la correcta creació del nou usuari, es crearà un parell de claus. En aquest cas les mostrem per consola per a que es vegi el que va passant en cada pas de l'execució.

```
User meritxell@ibemail.com added. Keys generated:
Public key from user ---> 617801431772555411879440588869763291490074440706347063369519790948517210257 3,
591666745838863870364288257157387187453814608089636734358131804394725046323 5,0

Private Key from user ---> 276103539955709301566877230782676894840422285744588429001803042423723510996,
397100989056164046939891098347608335106481357225431457453550120088725077866 6,0
```

Figura 5.4: Generació de claus per a un nou usuari

Safata d'entrada del correu electrònic

A l'inici, tenim la safata d'entrada buida, per tant, l'aspecte d'aquesta serà el següent:

Figura 5.5: Safata d'entrada buida a l'inici

Encriptació i enviament d'un nou correu

Suposem que tenim un amic que té com a identitat, en aquest cas correu electrònic, la següent: *pep@ibemail.com*. En aquest cas haurem de prémer sobre *send new email* i s'obrirà la pantalla que podem veure en la figura 5.6. Un cop omplim les dades corresponents (inclòs el cos del missatge) premem *Encrypt* i el missatge s'encriptarà com es pot veure en la figura 5.7.

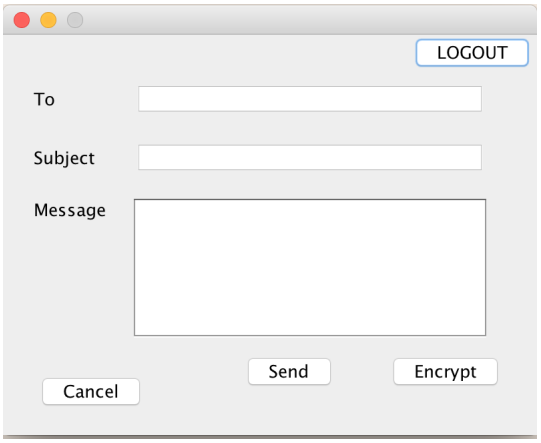

Figura 5.6: Pantalla d'enviament d'un nou correu

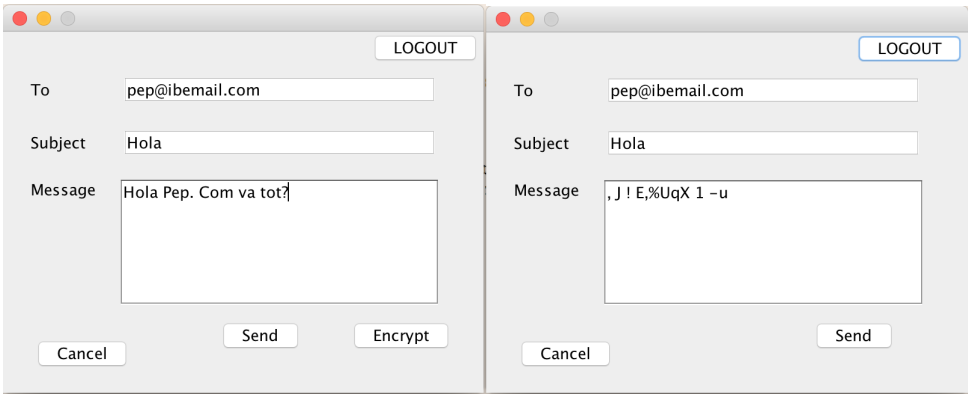

Figura 5.7: Procés d'encriptació d'un nou correu

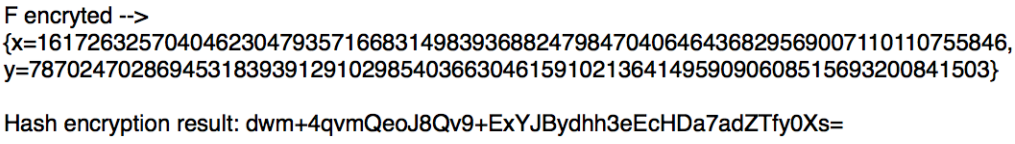

Figura 5.8: Resultat de l'encriptació del missatge

Aleshores, recordant que el que realment s'envia en aquest cas és la tupla $(U, V) = (r \cdot Pk, m \oplus h(e(Q_A, S_k)^r))$, es mostra per terminal el resultat del pairing, és a dir, $(e(Q_A, S_k)^r)$ que en aquest cas l'anomenem F i també mostrem el resultat del hash, és a dir, el resultat de fer $m \oplus (e(Q_A, S_k)^r)$. En aquest cas, després de l'encriptació realitzada en la figura 5.7, es mostra per pantalla el resultat que es pot veure en la figura 5.8.

Recepció i desencriptació d'un correu rebut

Suposem ara que el nostre correu és *pep@ibemail.com*. Quan entrem a la nostra safata d'entrada ens trobem el que es pot veure en la figura 5.9. Després de donar doble click ens trobem amb la mateixa situació que teníem en la figura 5.7 després d'encriptar, com es pot veure en la figura 5.10. Després de prémer *Decrypt* tornem a recuperar el missatge original com es pot veure en la figura 5.11. Tal com hem mostrat en l'apartat anterior, en la figura 5.12 mostrem el resultat de la F desencriptada així com el resultat del hash.

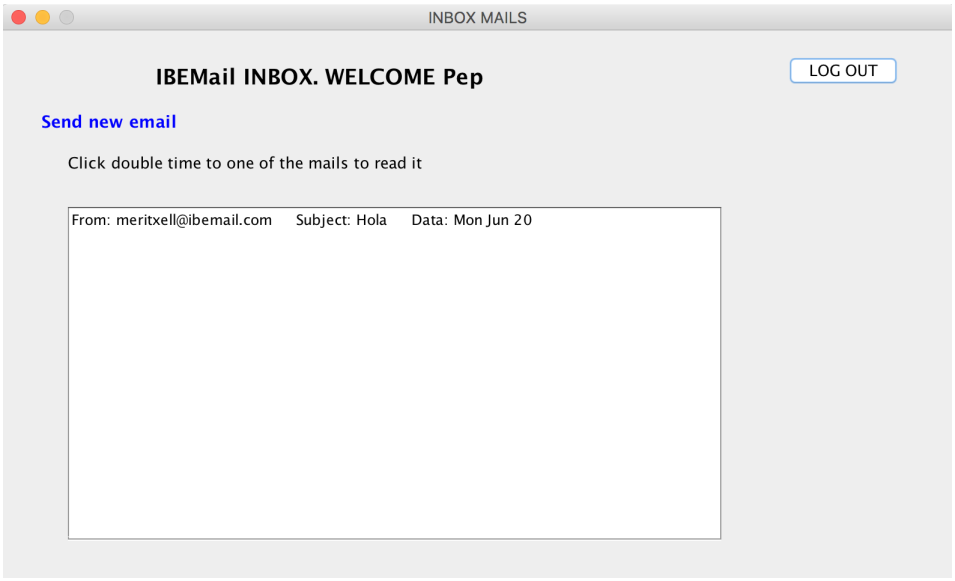

Figura 5.9: Safata d'entrada després de rebre un missatge

Figura 5.10: Missatge rebut abans de desencriptar-lo

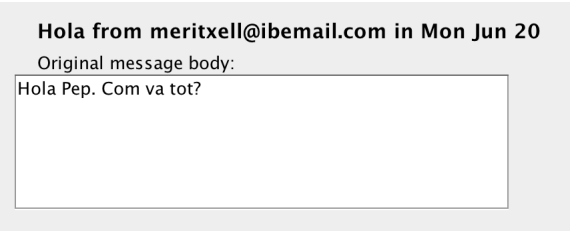

Figura 5.11: Missatge rebut després de desencriptar-lo

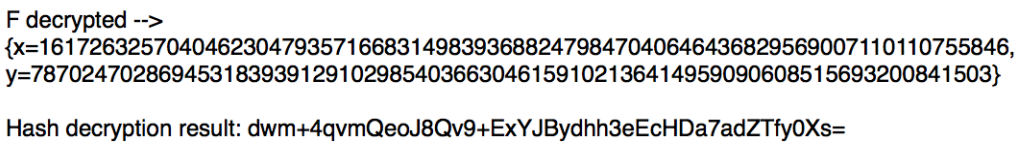

Figura 5.12: Resultat de desencriptar el missatge rebut

Com es pot observar, el resultat del hash del resultat obtingut, és el mateix que el hash que hem obtingut prèviament després d'encriptar, per tant, al aplicar el mètode contrari, trobarem al final el mateix missatge.

Capítol 6

Conclusions i futures línies de treball

En aquest treball s'ha realitzat un estudi de les corbes el·líptiques, dels aparellaments (o pairings) i s'han implementat dues aplicacions per a demostrar la utilitat dels aparellaments. La primera implementació, més teòrica, és la d'un intercanvi de claus a tres bandes. La segona és la implementació d'un prototip de correu electrònic.

En aquest cas no podem extreure conclusions a nivell de resultats obtinguts en execucions de l'aplicació perquè la finalitat d'aquesta no era estudiar l'eficiència del programa, si no estudiar com evitar els certificats de clau pública amb pairings, no sense abans haver estudiat totes les característiques de les corbes el·líptiques i els pairings. Així doncs les conclusions que es poden extreure són més a nivell teòric i del que s'ha pogut extreure després de tot el treball que s'ha fet.

Sabem que la criptografia cada vegada va tenint un paper més important en la societat present, i que el que sempre demanem els usuaris és que es garanteixi seguretat en les nostres dades, així que no és estrany que cada vegada l'estudi de seguretat sigui més important i més freqüent. Durant tot el treball ja s'han anat treient conclusions de cada secció, així que tampoc cal que ens repetim aquí; només cal remarcar que utilitzar els criptosistemes com a mesures de seguretat de les nostres dades és una feina costosa però que simplement ens permet protegir les nostres dades de manera eficient i ràpida. També s'ha vist la gran utilitat que tenen les corbes el·líptiques i els

aparellaments en la implementació de protocols de seguretat i com d'útil i senzilla és la llibreria jPBC que hem utilitzat per a realitzar demostracions.

Com a futures línies de treball relacionades amb aquest projecte es podria seguir treballant amb criptografia basada en pairings, ja que cada vegada van sorgint més implementacions que els utilizen, així que són una font d'investigació bastant gran. Pel que fa a la implementació de l'aplicació de correu, no crec que sigui necessari ampliar la implentació d'aquesta ja que, al ser un prototip, les parts més importants que es volien mostrar ja es veuen reflexades en la present implementació funcional. Com a millores d'aquestes ja s'ha dit que es podria millorar la part visual i posar una Base de Dades remota enlloc de local per a poder utilitzar-la remotament, però no és necessari per a la finalitat del protostip, que era l'estudi dels aparellaments en aplicacions reals.

Bibliografia

[1] I. Blake; G. Seroussi; N. Smart (2000). Elliptic Curves in Cryptography. LMS Lecture Notes. Cambridge University Press

[2] https://www.voltage.com/technology/data-encryption/identity-based-encryption/

[3] L. Huguet, J. Rifà, J.G. Tena. Criptografia con curvas elípticas. UOC. https://www.exabyteinformatica.com/uoc/Informatica/Criptografia_avanzada/Criptografia_avanzada_(Modulo_4).pdf

[4] D. Boneh and M. Franklin. Identity-based Encryption from the Weil Pairing. In CRYPTO 2001, volume 2139 of Lecture Notes in Computer Science, pages 213-229. Springer Verlag, 2001.

[5] I.F. Blake, G. Seroussi, and N.P. Smart. Elliptic Curves in Cryptography. London Mathematical Society Lecture Note Series. Cambridge University Press, Cambridge, 1999.

[6] (Edited by I.F. Blake, G. Seroussi and N.P. Smart). Advances in Elliptic Curve Cryptography. London Mathematical Society Lecture Note Series. Cambridge University Press, 2004.

[7] D. Boneh, X. Boyen, Efficient selective-ID secure identity-based encryption without random oracles, in: Advances in Cryptology—Eurocrypt 2004, in: LNCS, vol. 3027, Springer-Verlag, 2004, pp. 223–238.

[8] R. Sakai, M. Kasahara, ID based cryptosystems with pairing on elliptic curve, Cryptology ePrint Archive, Report 2003/054, 2003.

[9] A. Shamir. Identity-based cryptosystems and signature schemes. In Advances in Cryptology – CRYPTO 1984, vol. 196 of LNCS, pp.

[10] J.M. Miret. Elliptic Curve Cryptography. Máster en Matemática avanzada, Universidad de Murcia 2010.

[11] Reńe Schoof, Elliptic curve over finite fields and the computation of square roots mod p, Mathematics of Computation 44 (1985), no. 170, 483–495.

[12] F. Brezing, A. Weng, "Elliptic curves suitable for pairings based cryptography", Designs, Codes and Cryptography 37, pp. 133–141, 2005.

[13] P.S.L.M. Barreto, M.Naehrig. Pairing-Friendly Elliptic Curves of Prime Order. http://link.springer.com/chapter/10.1007%2F11693383_22

[14] B.Lynn. On the Implementation of Pairing-Based Cryptography. Final Thesis, 2007.

[15] Documentació de la llibreria jPBChttp://gas.dia.unisa.it/projects/jpbc/java-docs/api/index.html

[16] M.B.Barbosa, Identity Based Cryptography From Bilinear Pairings, 2005. https://repositorium.sdum.uminho.pt/bitstream/1822/3813/1/report.pdf

[17] R.Crandall, C.Pomerance. Prime Numbers, a Computational perspective (Second Edition) (2000), pp. 348-371.

[18] J. Baek, J. Newmarch, R. Sfavi-Naini,W. Susilo, A Survey of Identity-Based Cryptography. https://jan.newmarch.name/publications/auug_id_survey.pdf

[19] R. Barua, R. Dutta, Pairing-based Cryptography. http://math.iisc.ernet.in/~nmi/downloads/RBC.pdf

[20] S. Schmitt, H.G. Zimmer. Elliptic Curves: A Computational Approach, 2003.

[21] A. Roy, A. Datta, J.C. Mitchell. Formal Proofs of Cryptographic Security of Diffie-Hellman-based Protocols. http://www-cs.stanford.edu/people/jcm/papers/rdm-tgc-07.pdf

[22] Sakai, Ryuichi; Kasahara, Masao (2003). "ID Based cryptosystems with pairing on elliptic curve". Cryptography ePrint Archive. 2003/054. https://eprint.iacr.org/2003/054.pdf

[23] Certicom Research, Standards for efficient cryptography, SEC 1: Elliptic Curve Cryptography, Version 2.0, May 21, 2009. http://www.secg.org/sec1-v2.pdf

[24] S.Risco. Estudi d'esquemes de signatura per a Smart Meters. Treball final de master, Setembre de 2014.

[25] Complex multiplication method https://en.wikipedia.org/wiki/Complex_multiplication

[26] S.Blanch . Criptografia lliure amb corbes el·líptiques. Treball final de grau, Setembre 2008.

[27] Diffie, W. y M.E.Hellman. "New directions in cryptography", IEEE Transactions on Information Theory 22 (1976), pp. 644-654.

[28] V. Miller, Use of elliptic curves in cryptography, CRYPTO 85, 1985.

[29] N. Koblitz. Elliptic curve cryptosystems, Mathematics of Computation 48, 1987, pp:203-209.

[30] S. Galbraith, Pairings, in Advances in elliptic curve cryptography, I. F. Blake, G. Seroussi, N. P. Smart, eds., London Math. Soc. Lect. Note Series 317, Cambridge Univ. Press, Cambridge, 2005, pp:211

[31] B. Lynn. On the implementation of pairing-based cryptosystems. PhD thesis, Stanford University, 2007.

[32] V. S. Miller. Short Programs for functions on Curves. Unpublished manuscript, 1986.

[33] D.Freeman. Constructing Pairing-Friendly Elliptic Curves with Embedding Degree 10.

[34] D.Boneh, B.Lynn, and H.Shacham. Short signatures from the Weil pairing. https://www.iacr.org/archive/asiacrypt2001/22480516.pdf