Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Función picadillo determinista al grupo $\mathbb{G}_2$ y su aplicación en autenticación para dispositivos móviles

Tesis que presenta

José Eduardo Ochoa Jiménez

para obtener el Grado de

Maestro en Ciencias en Computación

Asesores de tesis:

Dr. Francisco Rodríguez Henríquez

Dra. María de Lourdes López García

México, D.F. **Diciembre 2013**

Tell me and I Forget,

Show me and I may Remember,

Involve me and I Understand.

~Benjamin Franklin~

Dedicatoria

Esta tesis la dedico mi familia quienes hicieron todo para que pudiera lograr mis sueños y que me han acompañado durante toda mi vida brindándome su sincero amor y apoyo. En particular a mi madre Ana María que está con migo incondicionalmente, que cree en mí y a quien agradezco su amor, consejos y paciencia, ya que me han ayudado a convertirme en la persona que soy, y que sin ella no hubiera podido lograr esta meta; a mi padre Agustín que admiro profundamente porque cada día me muestra que es posible lograr lo que uno se propone sin importar las adversidades y que me ha enseñado a nunca darme por vencido; a mis hermanos Luis Ángel y Leslie de quienes estoy muy orgulloso, que amo con todo el corazón, que me han apoyado a cada paso y que han sido el motor que me impulsa cada día ya que son la alegría de mi vida.

mu mu.

Agradecimientos

Comenzare agradeciendo a mi familia que son la fuerza que me impulsa, por todo su amor, paciencia y apoyo durante esta etapa de mi vida.

A mis profesores de licenciatura M. en C. Oscar Alvarado y Dr. Francisco Zaragoza por su apoyo durante la etapa de ingreso al CINVESTAV.

A mi asesor de tesis, el doctor Francisco Rodríguez Henríquez que creyó en mí y que me brindo su confianza durante la realización de este proyecto.

A mi asesora, la doctora María de Lourdes López García por sus consejos y por haberme apoyado durante este proceso.

A mis revisores de tesis Dr. Debrup Chakraborty y Dr. Luis Gerardo de la Fraga, así como al Dr. Miguel Ángel León Chávez por sus comentarios.

Al alumno de doctorado Gora Adj por sus comentarios y contribuciones a este trabajo de tesis.

A Sofía Reza por su amabilidad, paciencia y eficiencia.

A mis compañeros de generación Marco, Rogelio y Michel, por compartir sus conocimientos y haber hecho tan amena la maestría.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por haberme brindado el apoyo económico a partir del cual he culminado mis estudios de postgrado en tan prestigiosa institución.

Al CINVESTAV, quien se ha convertido en mi *alma mater*, por permitirme crecer en lo personal y académico.

Resumen

Actualmente los emparejamientos bilineales definidos sobre curvas elípticas han sido utilizados como una primitiva en la construcción de protocolos criptográficos, los cuales se consideran prácticos, si es posible calcular de manera eficiente y segura todas las operaciones involucradas en ellos. Sin embargo, la mayoría de las implementaciones encontradas en el estado del arte sólo toman en cuenta el cálculo del emparejamiento y no ponen atención a otras primitivas importantes, como es la generación de puntos aleatorios en una curva elíptica del problema conocido como picadillo al grupo $\mathbb{G}_2$, fundamental en protocolos basados en emparejamientos y basados en la identidad.

Por otro lado, uno de los problemas principales en cuanto a seguridad es la autenticación, la cual en muchos sistemas se realiza verificando la identidad de los usuarios por medio de contraseñas, mismas que presentan importantes problemas de vulnerabilidad ya que los usuarios tienden a utilizar contraseñas fáciles de adivinar. En este contexto, la autenticación de dos factores ofrece una solución, ya que se considera más fuerte y más segura que la tradicional autenticación de un factor.

En esta tesis, se consideran ambas áreas, proponiendo una función determinista denominada H_2 para calcular el picadillo al grupo $\mathbb{G}_2$, en la familia de curvas de Barreto-Naehrig. Tal función a diferencia del método probabilista existente, es capaz de evitar ataques de análisis de tiempo con un costo computacional insignificante. Además, se realiza una implementación eficiente de un par de protocolos de autenticación de dos factores basados en emparejamientos, los cuales, utilizan la función H_2.

Abstract

Nowadays, bilinear pairings over elliptic curves have been used as a primitive in the construction of cryptographic protocols, which are considered useful as long as the operations involved are efficient and secure. Nevertheless, most of the proposed implementations in literature consider just the calculation of the bilinear pairing function without pay attention in other important primitives, such as, the generation of random points on an elliptic curves, that is, a problem known as hash to $\mathbb{G}_2$, essential in cryptography-based protocols and identity-based protocols.

On the other hand, a main problem in the area of information security is the authentication, which is used in many security systems to verify the identity of the user through passwords that, in many cases, are vulnerable due to, are easily guess. In this sense, the two-factor authentication is a better solution, since, it is considered stronger and safer than the traditional one-factor authentication.

In this thesis, both areas are considered, to propose a deterministic function called H_2 to compute the hash to $\mathbb{G}_2$, over the Barreto-Naehrig family curves. Unlike the probabilistic method, this deterministic function is capable to avoid the timing attacks in a low computationally cost. Besides, an efficient implementation of two protocols of two-factor authentication based on bilinear pairing, which use the H_2 function, is performed.

Índice general

Lista de Algoritmos

Índice de figuras

Índice de tablas

1 Introducción

"El primer paso para el conocimiento es reconocer que somos ignorantes."
∼Sócrates (470-399 a.c.)∼

1.1. Antecedentes

Con el rápido crecimiento de la tecnología en los teléfonos inteligentes y otros dispositivos móviles, así como el incremento de su uso en la vida cotidiana, se ha desarrollado un gran interés en el cómputo móvil para lograr el acceso a distinta información en cualquier lugar y en cualquier momento, esto mediante el uso de diversas fuentes de información digital a través de un canal de comunicación inalámbrico [1].

A pesar de que este hecho ha presentado muchas ventajas al mejorar el servicio en aplicaciones donde la velocidad de atención es crítica, es necesario considerar que los canales de comunicación inalámbricos son inseguros, y que debido a esta inseguridad se deben tener en cuenta aspectos de seguridad, que protejan a los sistemas que requieren de intercambio y acceso seguro a información de importancia. En este sentido, la criptografía es una herramienta que ayuda a contrarrestar posibles amenazas, a través de servicios de seguridad como:

- *Autenticación*: Permite certificar que la identidad de las entidades participantes en la comunicación es verdadera. La autenticación se logra verificando dichas entidades usando mecanismos como firmas digitales, certificados digitales o características biométricas. La autenticación puede realizare verificando alguno de los siguientes tres factores:

  1. Algo que el usuario tiene, por ejemplo, una llave privada con la cual puede emitir firmas digitales.
  2. Algo que él sabe, esto es, pedirle una contraseña.

3. Algo que el usuario es, por ejemplo, analizar sus huellas dactilares.

- *Confidencialidad*: Asegura que la información privada sólo puede ser consultada o manipulada por usuarios o entidades autorizadas.

- *Integridad*: Da la certeza de que la información no ha sido modificada por entidades no autorizadas para hacerlo. Dentro de las posibles modificaciones están la escritura, modificación o borrado de segmentos de datos.

- *No repudio*: Ofrece protección a un usuario o entidad respecto a que otro participante en la comunicación niegue, posteriormente, que en realidad se realizó cierta transacción, es decir, impide que una entidad niegue las acciones realizadas.

- *Control de acceso:* Proporciona la habilidad de permitir o denegar el uso de un recurso particular a una entidad en particular.

Para brindar estos servicios de seguridad, por lo general, es necesario emplear esquemas criptográficos, los cuales permiten establecer la comunicación entre dos o más entidades de forma segura, a través de un canal de comunicación inseguro.

En esta tesis se abordará el estudio y aplicación de la criptografía basada en emparejamientos, en específico el servicio de autenticación, aprovechando la gran capacidad de cómputo que ofrecen los dispositivos móviles de hoy en día.

1.1.1. Criptografía

La criptografía (del griego *kriptos* que significa ocultar y *graphos* que significa escritura) ha sido empleada durante miles de años con el objetivo de proveer comunicaciones confiables sobre canales inseguros. Concretamente, al proceso de diseñar sistemas para obtener una comunicación segura, en un canal que no lo es, se le conoce como criptografía [2].

De una manera muy general, la configuración básica de un esquema criptográfico se modela con dos entidades (Alicia y Beto) que desean comunicarse de forma segura, de tal manera que una tercera entidad (Eva) no pueda entender la comunicación entre ellos. En este contexto Alicia, con ayuda de un algoritmo denominado de cifrado, transforma el mensaje conocido como texto en claro, produciendo un mensaje llamado texto cifrado, incomprensible para Eva. Después, Alicia envía el texto cifrado a Beto, quien utiliza un algoritmo de descifrado que transforma el texto cifrado nuevamente al texto en claro del mensaje original. La clave del éxito en la comunicación segura es que los algoritmos de cifrado y descifrado son operaciones inversas que transforman el texto en claro a cifrado y viceversa, con la ayuda de una llave secreta conocida únicamente por Alicia y Beto. La criptografía puede ser clasificada en simétrica y asimétrica [3].

1.1.2. Criptografía simétrica

La criptografía simétrica o de llave privada es la más antigua, y debido a su eficiencia es útil para cifrar grandes cantidades de información; su característica principal consiste en que utiliza una misma llave para cifrar y descifrar los mensajes. Alicia y Beto deben quedar de acuerdo en el valor de la llave secreta a utilizar, antes de iniciar la comunicación segura. Algunos sistemas conocidos de llave secreta son DES (*Data Encryption Standard*) desarrollado por IBM [4] y AES (*Advanced Encryption Standard*) [5].

Los esquemas de cifrado simétrico se dividen en dos: *cifradores por flujo de datos*, que realizan el cifrado de la información bit a bit y *cifradores por bloques*, que procesan los datos por grupos de bits de longitud fija llamados bloques. En este tipo de esquemas, el hecho de utilizar la misma llave para el cifrado y descifrado de los mensajes produce diversos problemas como:

- La distribución de llaves: En un grupo de n entidades, si una entidad desea comunicarse con cada una de las $n-1$ entidades restantes, deberá manejar $n-1$ llaves distintas. En total el número de llaves requeridas es de $n(n-1)/2$.
- El intercambio de llaves: Si dos entidades se encuentran físicamente en lugares distintos, surge el problema de cómo intercambiar las llaves de manera segura.

1.1.3. Criptografía asimétrica

La criptografía asimétrica o de llave pública es el área de interés de esta tesis, este tipo de criptografía se basa en problemas matemáticos difíciles de resolver y a diferencia de la criptografía simétrica, se utiliza un par llaves, una para cifrar (llave pública) y otra para descifrar (llave privada). Ambas, tienen una relación estrecha, debido a que la llave pública se deriva de la llave privada. Así, a cada entidad le pertenece un par de llaves, donde la llave privada es sólo conocida por su entidad dueño y la llave pública es difundida a todas las entidades, de tal manera que para un grupo de n entidades, sólo se requieren dos llaves por cada entidad.

Es importante mencionar que como todas las entidades conocen la llave pública, debe ser imposible computacionalmente hablando, deducir la llave privada a partir de la pública. Los sistemas de llave pública más famosos son RSA (Rivest, Shamir y Adleman) [6], cuya seguridad se basa en la dificultad de la factorización de números enteros; El Gamal [7], el cual descansa en la dificultad del problema del logaritmo discreto; y criptografía de curvas elípticas, que se basa en la dificultad del problema de logaritmo discreto en curvas elípticas.

A pesar de que la criptografía asimétrica dio pie a la elaboración de nuevos protocolos criptográficos, tiene la desventaja de que el tamaño de las llaves es mayor y las operaciones de cifrado y descifrado son considerablemente más costosas que las involucradas en la criptografía simétrica. Además, el hecho de contar con una llave pública requiere que una tercera entidad "confiable" determine que dicha llave es de quien se dice ser, evitando problemas como la usurpación de la identidad. Debido a

que los sistemas de llave pública son más lentos que los sistemas de llave privada, es común usar sistemas de llave pública para establecer una llave que es usada en un sistema de llave privada para cifrar y descifrar los mensajes. El interés en la velocidad es importante en los esquemas que requieren de la transmisión de una gran cantidad de datos.

Cronológicamente hablando, la criptografía asimétrica fue introducida por los investigadores Whitfield Diffie y Martin Hellman en 1976 [8], quienes propusieron una manera interesante de resolver el problema de intercambio de llaves y cuyo trabajo fue considerado como el desarrollo más impactante en la criptografía. Dos años después, Rivest, Shamir y Adleman [6] dieron a conocer el primer esquema de llave pública: RSA, cuya seguridad se basa en la dificultad de la factorización de números enteros.

Años después en 1985, Victor Miller [9] y Neal Koblitz [10], de manera independiente, propusieron el uso de curvas elípticas para el diseño de criptosistemas de llave pública, observando que su seguridad está garantizada por la complejidad computacional del problema del logaritmo discreto definido sobre un grupo abeliano, generado por los puntos de una curva elíptica, el cual con una selección adecuada de parámetros, es un problema matemático difícil de resolver. Hasta este momento, no se conoce ningún ataque que vulnere este enfoque con tiempo de ejecución polinomial, además tiene la ventaja de que para obtener el mismo nivel de seguridad que brindan otros esquemas criptográficos, su espacio de llaves es mucho más pequeño, lo que la convierte en una tecnología adecuada para ser utilizada en ambientes con recursos restringidos (memoria, velocidad, ancho de banda, etc).

Los emparejamientos bilineales fueron inicialmente introducidos en la criptografía en 1993 por Alfred J. Menezes, Tatsuaki Okamoto y Scott A. Vanstone [11], como un ataque a los esquemas de criptografía de curvas elípticas. Este ataque consiste en reducir el problema de logaritmo discreto en una curva elíptica, al problema del logaritmo discreto en una extensión del campo en donde se define la curva, esto se logra mediante el establecimiento de una proyección de un punto en la curva a un elemento en la extensión del campo, utilizando el emparejamiento bilineal de Weil.

Los primeros autores en utilizar los emparejamientos bilineales para dar soluciones a problemas criptográficos y no como herramientas para ataques fueron Antoine Joux [12], Shiego Mitsunari *et al.* [13] y Ryuichi Sakai *et al.* [14], quienes aprovecharon las propiedades de los emparejamientos, abriendo así la posibilidad de crear nuevos sistemas criptográficos.

1.1.4. Criptografía basada en la identidad

En 1984, Adi Shamir introdujo el concepto de criptografía basada en la identidad [15], para evitar los problemas de usurpación en la criptografía clásica de llave pública. En este esquema, Shamir propuso la idea de que una cadena arbitraria, tal como la dirección de correo electrónico o un número telefónico, servía como llave pública en un esquema de criptografía asimétrica.

La idea general consiste en que si Alicia desea enviar un mensaje cifrado a Beto,

bastaría con utilizar la dirección de correo electrónico de él como llave pública para cifrarlo, por ejemplo: "beto@correo.com". De esta manera, Beto al recibir el mensaje cifrado contacta a una tercera autoridad llamada "Generador de llaves privadas", con quien se autentica y obtiene su llave privada, a partir de la cual descifra el mensaje de Alicia.

En el año 2001 Dan Boneh y Matt Franklin presentaron una solución al problema de cómo desarrollar de forma práctica la criptografía basada en la identidad, a través del uso de emparejamientos bilineales [16].

1.2. Planteamiento del problema

En los últimos años, el crecimiento de las tecnologías de computo móvil y la creciente apertura de las redes corporativas (vía Internet) han facilitado las transacciones electrónicas, compras en línea y muchos otros servicios financieros. Sin embargo, llevar a cabo estos objetivos en un dispositivo móvil trae consigo ciertos problemas de seguridad, los cuales deben ser tomados en cuenta, sobre todo en sistemas que manejan información crítica que viaja a través de un canal inseguro.

Uno de los problemas principales en cuanto a seguridad es la autenticación, la cual en muchos sistemas se realiza verificando la identidad de los usuarios por medio de contraseñas. En estos sistemas las contraseñas presentan importantes problemas de seguridad ya que los usuarios tienden a utilizar contraseñas fáciles de adivinar, a utilizar la misma contraseña en varias cuentas, a escribir las contraseñas o a almacenarlas en sus máquinas, etc. En este sentido la autenticación de dos factores ofrece una solución ya que es un mecanismo que implementa dos de los factores mencionados en la Sección 1.1 y por tanto se considera más fuerte y más seguro que la tradicional autenticación de un factor.

En la mayoría de los protocolos basados en la identidad se utiliza una función picadillo, la cual se puede definir informalmente como el proceso de transformar una cadena de longitud arbitraria a una cadena de longitud fija. En los protocolos de autenticación basados en emparejamientos bilineales se requiere de una función picadillo especial, que traslade una cadena de longitud arbitraria a un punto perteneciente a un grupo de puntos sobre una curva elíptica. Tal grupo normalmente es denotado en la literatura como $\mathbb{G}_1$ o $\mathbb{G}_2$, dependiendo de las características de la curva elíptica.

La función picadillo especial se denota como H_1 y se denomina picadillo a $\mathbb{G}_1$ o *map-to-point*, cuando la cadena se proyecta a un punto en el grupo $\mathbb{G}_1$. H_2 y picadillo a $\mathbb{G}_2$, corresponden al caso cuando el grupo es $\mathbb{G}_2$. Es significativo notar que el método existente para trasladar una cadena a un punto en $\mathbb{G}_2$ es probabilista, considerando que su probabilidad de éxito depende de la entrada del algoritmo. Esta característica es considerada una vulnerabilidad ya que puede derivar en un ataque de análisis de tiempo. Una posible forma de contrarrestar este tipo de ataques sería utilizar un método determinista, en otras palabras, que se realice el cálculo con un número constante de operaciones. Así, las comunicaciones entre entidades, principalmente de forma inalámbrica, serían seguras, al utilizar sistemas electrónicos que apliquen

protocolos de autenticación de dos factores basados en emparejamientos bilineales que no sean susceptibles al ataque de análisis de tiempo.

Además de las funciones H_1 y H_2, los protocolos basados en emparejamientos, como su nombre lo indica, usan la función de emparejamiento bilineal, misma que debe ser eficientemente implementada en cualquier dispositivo ya sea de escritorio o móvil, con la finalidad de realizar las operaciones de los protocolos de manera eficiente y así hacer de estos protocolos una opción viable. Dado que los bloques principales que componen estos protocolos han sido estudiados por diferentes autores, exceptuando la función determinista H_2, se puede suponer que si se contará con un algoritmo determinista para H_2 y una función de emparejamiento bilineal implementada eficientemente, entonces la implementación de un protocolo de autenticación de dos factores basado en emparejamientos bilineales, utilizando un dispositivo móvil, puede ser además de seguro, eficiente.

1.3. Objetivos

Esta tesis tiene como objetivos principales: *El desarrollo de un algoritmo determinista para el cálculo de la función picadillo al grupo $\mathbb{G}_2$ y la implementación de protocolos de autenticación de dos factores, sobre dispositivos móviles, basados en emparejamientos bilineales que utilicen como bloque criptográfico la función picadillo desarrollada.*

Los objetivos particulares son:

- Revisión de la literatura existente sobre funciones picadillo a curvas elípticas.
- Análisis de los algoritmos de picadillo al grupo $\mathbb{G}_1$ y $\mathbb{G}_2$.
- Desarrollo de la función picadillo al grupo $\mathbb{G}_2$ de manera determinista.
- Implementación de los algoritmos de picadillo a $\mathbb{G}_2$ tanto probabilista como determinista y análisis comparativo entre ambos métodos.
- Análisis de los protocolos de autenticación de dos factores propuestos en [17,18].
- Implementación de las primitivas criptográficas necesarias para los protocolos de autenticación en el servidor.
- Implementación de las primitivas criptográficas necesarias para los protocolos de autenticación en el cliente (dispositivo móvil).
- Implementación de los protocolos de autenticación y su análisis de eficiencia.

1.4. Contribuciones

1. El desarrollo de un algoritmo para el cálculo del picadillo al grupo $\mathbb{G}_2$ de manera determinista.

2. Análisis comparativo entre el método existente y el método propuesto para el picadillo al grupo $\mathbb{G}_2$.

3. Análisis de seguridad de los protocolos de autenticación de dos factores propuestos en [17, 18].

4. La implementación de los protocolos de autenticación basados en emparejamientos descritos en [17, 18] sobre un dispositivo móvil, los cuales utilizan el algoritmo de picadillo al grupo $\mathbb{G}_2$ como una de sus primitivas.

1.5. Metodología

Para alcanzar los objetivos planteados en esta tesis, se sigue el modelo de capas de la Figura 1.1 en donde la primer capa contiene la aritmética de campos finitos, que incluye las operaciones básicas como la suma, resta, multiplicación, inversión y exponenciación en campos. La capa siguiente corresponde a las curvas elípticas, en donde las operaciones de suma y doblado de puntos son requeridas para el cálculo de la multiplicación escalar, que es la operación principal de esta sección. La capa tres corresponde a las funciones picadillo especiales que trasladan una cadena de longitud arbitraria a un punto que satisface la curva elíptica, tanto H_1 como H_2. La capa siguiente contiene los emparejamientos bilineales, donde se encuentran las funciones de emparejamiento de Weil y Tate, y versiones posteriores de éste último tales como ate, R-ate y ate óptimo entre otros. La capa de criptografía basada en la identidad que utiliza la multiplicación escalar para generar los secretos, las funciones H_1 y H_2 para proyectar las identidades a puntos en los grupos correspondientes, además utiliza la capa de emparejamientos cuando hace el cálculo y la verificación de la identidad.

Por último se tiene la capa de los esquemas de autenticación, que apoyan su seguridad en la criptografía basada en la identidad y consiguen una implementación eficiente de acuerdo a los campos definidos para establecer la aritmética en las capas anteriores.

1.6. Organización de la tesis

El trabajo de tesis ha sido organizado en seis capítulos, los cuales se describen a continuación: en el Capítulo 2 se introduce al lector en el tema, a través de la definición de conceptos generales que son necesarios para la comprensión de lo desarrollado en esta tesis; posteriormente, dado que uno de los intereses de este trabajo es el desarrollo de la función picadillo al grupo $\mathbb{G}_2$ de manera determinista, se presentan los detalles

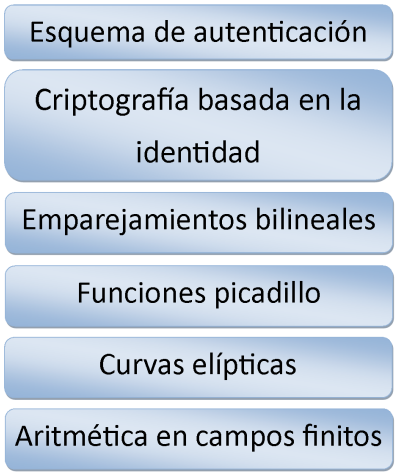


Figura 1.1: Modelo de capas de un protocolo de autenticación basado en emparejamientos.

de su construcción e implementación en el Capítulo 3; en el Capítulo 4 se describen los protocolos propuestos por Michael Scott en [17, 18], además se presentan los análisis realizados a estos protocolos; ya que la implementación de dichos protocolos es otro interés en este trabajo, en el Capítulo 5 se detalla la forma en que se realizó la implementación y se muestran los resultados obtenidos; por último en el Capítulo 6 se presentan las conclusiones derivadas del trabajo realizado, así como el trabajo futuro.

2

Fundamentos matemáticos

"Lo más importante no es el conocimiento,
si no saber donde encontrarlo."
∼Samuel Johnson (1709-1784)∼

En este capítulo se introducen los conceptos que serán utilizados a lo largo de esta tesis, los cuales ayudan a la construcción de los emparejamientos bilineales [3, 19–21]. Se presentan las definiciones asociadas a los conceptos de curvas elípticas en la Sección 2.1, las curvas elípticas sobre campos finitos en la Sección 2.2, los emparejamientos bilineales en la Sección 2.3 y una descripción de los problemas del logaritmo discreto en grupos en la Sección 2.4. Algunos conceptos básicos de teoría de números no se incluyen en este capítulo, sin embargo, se presentan en el Apéndice A.

2.1. Curva elíptica

Una curva elíptica E sobre un campo $\mathbb{F}$, denotada como $E/\mathbb{F}$, en el espacio afín, está definida por la ecuación de Weierstrass:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{2.1}$$

donde $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}$. Si la característica del campo $\mathbb{F}$ es distinta de 2 y 3, se permite el siguiente cambio de variables [20]:

$$(x, y) \to \left(\frac{x - 3a_1^2 - 12a_2}{36}, \frac{y - 3a_1x}{216} - \frac{a_1^3 + 4a_1a_2 - 12a_3}{24} \right)$$

el cual transforma a $E/\mathbb{F}$ en una curva elíptica definida por la ecuación

$$y^2 = x^3 + ax + b, \tag{2.2}$$

con discriminante $\Delta = -16(4a^3 + 27b^2)$, la cual es conocida como la ecuación simplificada de Weierstrass [20], donde $a, b \in \mathbb{F}$.

2.1.1. Puntos en la curva elíptica

Definición 2.1. *(Punto al infinito). El punto correspondiente a* (∞, ∞) *es conocido como punto al infinito y está denotado por* $\mathcal{O}$*. El punto al infinito se encuentra en el extremo inferior y superior del eje de las ordenadas, de tal manera que la línea vertical al punto* $P = (x, y)$ *interseca a* $\mathcal{O}$*.*

Sea $\bar{\mathbb{F}}$ la cerradura algebraica de $\mathbb{F}$ y dada la curva elíptica $E/\mathbb{F}$ definida por la Ecuación 2.2, el conjunto de los puntos en la curva $E/\mathbb{F}$ está definido como:

$$E(\bar{\mathbb{F}}) = \{(x, y) \mid x, y \in \bar{\mathbb{F}},\ y^2 - x^3 - ax - b = 0\} \cup \{\mathcal{O}\}.$$

Además, para cualquier extensión $\mathbb{F}'$ del campo $\mathbb{F}$, el conjunto de los $\mathbb{F}'$-puntos racionales de la curva elíptica se define como:

$$E(\mathbb{F}') = \{(x, y) \mid x, y \in \mathbb{F}',\ y^2 - x^3 - ax - b = 0\} \cup \{\mathcal{O}\},$$

este conjunto forma un grupo abeliano escrito de manera aditiva, en donde $\mathcal{O}$ es el elemento identidad. En lo siguiente se utilizará $E(\mathbb{F}')$ para hacer referencia al grupo abeliano y no sólo al conjunto de los $\mathbb{F}'$-puntos racionales de $E/\mathbb{F}$.

2.1.2. Ley de grupo

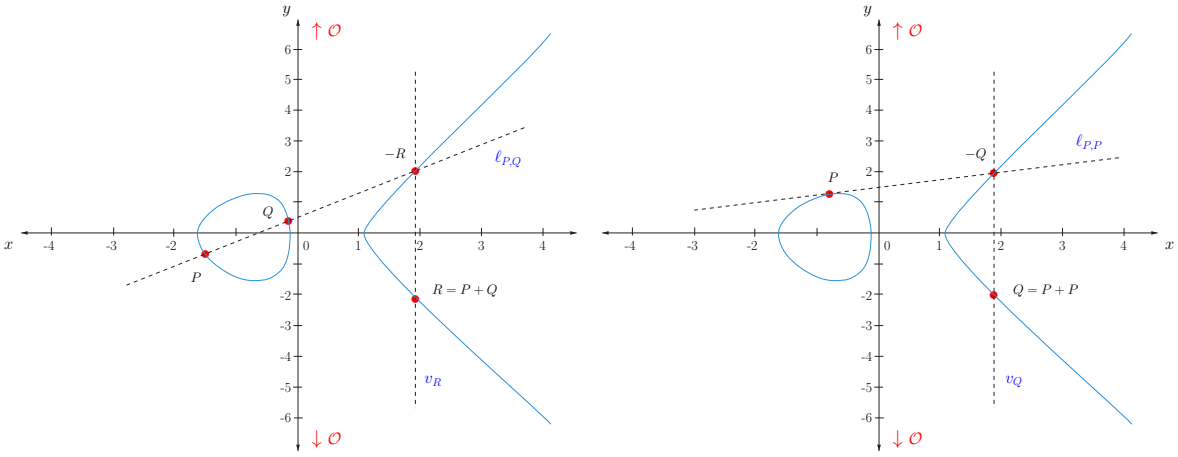
Como se mencionó anteriormente $E(\mathbb{F})$ es un grupo abeliano, bajo la operación de adición. La manera de hacer la suma de puntos frecuentemente es explicada geométricamente como:

- Sean los puntos P, Q y $R \in E(\mathbb{F})$. La suma R, de los puntos P y Q se realiza al trazar una línea recta, denotada como $\ell_{P,Q}$, a través de ellos, la cual interseca la curva elíptica en un tercer punto $-R$, entonces, el punto R es la reflexión de $-R$ sobre el eje de las abscisas descrita por la linea vertical v_R en la Figura 2.1(a).

- Sean los puntos P y $Q \in E(\mathbb{F})$. El doblado Q, del punto P se realiza al trazar una línea tangente, denotada como $\ell_{P,P}$, a la curva elíptica en P, esta línea interseca la curva elíptica en un segundo punto $-Q$, entonces, el punto Q es la reflexión de $-Q$ sobre el eje de las abscisas descrita por la linea vertical v_R en la Figura 2.1(b).

Las fórmulas algebraicas para la ley de grupo se pueden derivar de la descripción geométrica. Estas fórmulas se presentan a continuación para la curvas elíptica $E/\mathbb{F}$ de la forma simplificada de Weierstrass (Ecuación 2.2) en coordenadas afines, cuando la característica del campo base no es 2 ni 3.

1. Identidad: $P + \mathcal{O} = \mathcal{O} + P = P$ para todo $P \in E(\mathbb{F})$.



(a) Suma de puntos.

(b) Doblado de punto.

Figura 2.1: Operaciones en una curva elíptica definida sobre los reales.

2. Negativo: si $P = (x, y) \in E(\mathbb{F})$, entonces $(x, y) + (x, -y) = \mathcal{O}$. El punto $(x, -y)$ es denotado por $-P$ y es llamado el negativo de P. Nota que $-P$ es un punto en $E(\mathbb{F})$ al igual que $\pm\mathcal{O}$.

3. Suma: sean $P = (x_P, y_P) \in E(\mathbb{F})$ y $Q = (x_Q, y_Q) \in E(\mathbb{F})$, con $P \neq \pm Q$.
   Entonces $P + Q = (x, y)$, donde

$$x = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)^2 - x_P - x_Q \quad \text{and} \quad y = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)(x_P - x) - y_P.$$

4. Doblado: sea $P = (x_P, y_P) \in E(\mathbb{F})$, donde $P \neq -P$. Entonces $2P = (x, y)$, donde

$$x = \left(\frac{3x_P^2 + a}{2y_P}\right)^2 - 2x_P \quad \text{and} \quad y = \left(\frac{3x_P^2 + a}{2y_P}\right)(x_P - x) - y_P.$$

A partir de las operaciones de suma y doblado se define la operación conocida como *multiplicación escalar*, denotada como kP, la cual teniendo un entero k (escalar) y un punto $P \in E(\mathbb{F})$, consiste en repetir $k - 1$ veces la operación de adición sobre P.

$$kP = \underbrace{P + P + ... + P}_{k-1 \quad veces}.$$

Una forma de llevar a cabo la multiplicación escalar es a través de una adaptación del algoritmo de Horner. En este caso, se representa el escalar k de forma binaria y para obtener la multiplicación escalar, se recorre cada uno de los bits de k realizando el doblado de punto en cada paso y aplicando la suma de puntos si el i-ésimo bit de la representación es uno, este algoritmo es conocido como método binario de izquierda a derecha.

Existen otros métodos que a cambio de cierto nivel de precómputo, reducen el número de sumas y doblados requeridos, tal como el método de ventana ω-NAF; otros métodos en cambio, aprovechan las propiedades de la curva elíptica para reducir el número de operaciones, tales como el método GLV [22] y el método GLS [23].

2.1.3. Espacio proyectivo

En la sección anterior se presentaron las formulas para la suma y doblado de puntos, para la curva elíptica generada por la Ecuación 2.2, definida sobre el campo $\mathbb{F}$ de característica distinta a 2 y 3. Las formulas descritas requieren de una inversión y diversas multiplicaciones en el campo. Si la inversión en $\mathbb{F}$ es significativamente más costosa que una multiplicación, entonces puede ser mejor representar los puntos utilizando coordenadas proyectivas.

Sea $\mathbb{F}$ un campo y c, d dos enteros positivos. Se define la relación de equivalencia "$\sim$" en el conjunto $\mathbb{F}^3 \setminus \{(0,0,0)\}$ como: $(X_1, Y_1, Z_1) \sim (X_2, Y_2, Z_2)$ si $X_1 = \lambda^c X_2$, $Y_1 = \lambda^d Y_2$ y $Z_1 = \lambda Z_2$ para algún $\lambda \in \mathbb{F}^*$.

La clase de equivalencia que contiene a $(X, Y, Z) \in \mathbb{F}^3 \setminus \{(0,0,0)\}$ es:

$$(X : Y : Z) = \{(\lambda^c X, \lambda^d Y, \lambda Z) \mid \lambda \in \mathbb{F}^*\},$$

en donde $(X : Y : Z)$ es el punto proyectivo, mientras que (X, Y, Z) denota el representativo de $(X : Y : Z)$. El conjunto de todos los puntos proyectivos se denota como $\mathbb{P}(\mathbb{F})$. En particular, si $Z = 1$, se tiene que $(X/Z^c, Y/Z^d, 1)$ es el representativo del punto $(X : Y : Z)$, el cual es el único punto representativo con coordenada $Z = 1$. Por lo tanto, se tiene una correspondencia uno a uno entre el conjunto de puntos proyectivos:

$$\mathbb{P}(\mathbb{F})^* = \{(X : Y : Z) \mid X, Y, Z \in \mathbb{F},\ Z \neq 0\}$$

y el conjunto de puntos afines:

$$\mathbb{A}(\mathbb{F})^* = \{(x, y) \mid x, y \in \mathbb{F}\}.$$

El conjunto de puntos proyectivos

$$\mathbb{P}(\mathbb{F})^0 = \{(X : Y : Z) \mid X, Y, Z \in \mathbb{F},\ Z = 0\}$$

es denominado línea al infinito dado que estos puntos no corresponden a ninguno de los puntos afines.

La forma proyectiva de la ecuación de Weierstrass 2.2, puede ser obtenida sustituyendo $x = X/Z^c$ y $y = Y/Z^d$, y haciendo la reducción de los denominadores. Los puntos $\mathbb{P}(\mathbb{F})^*$ satisfacen la ecuación proyectiva, mientras que el conjunto de puntos $\mathbb{P}(\mathbb{F})^0$ corresponden al punto al infinito $\mathcal{O}$.

2.1.3.1. Coordenadas proyectivas estándar

En estas coordenadas los enteros c y d tienen un valor de uno. Por lo que el punto proyectivo $(X : Y : Z)$, en donde $Z \neq 0$, corresponde al punto afin $(X/Z, Y/Z)$. La ecuación proyectiva de la curva elíptica es:

$$Y^2Z = X^3 + aXZ^2 + bZ^3.$$

El punto al infinito $\mathcal{O}$ corresponde a $(0 : 1 : 0)$, mientras el negativo de $(X : Y : Z)$ es $(X : -Y : Z)$.

2.1.3.2. Coordenadas jacobianas

Para estas coordenadas los enteros c y d toman los valores dos y tres, respectivamente. Entonces el punto proyectivo $(X : Y : Z)$, en donde $Z \neq 0$, corresponde al punto afin $(X/Z^2, Y/Z^3)$. En estas coordenadas, la ecuación proyectiva de la curva elíptica es:

$$Y^2 = X^3 + aXZ^4 + bZ^6.$$

El punto al infinito $\mathcal{O}$ es el correspondiente a $(1 : 1 : 0)$, mientras que el negativo de $(X : Y : Z)$ es $(X : -Y : Z)$.

2.1.3.3. Suma y doblado de puntos

La forma más eficiente de realizar la operación de suma de puntos es a través de coordenadas mixtas, es decir, un punto en afines (con $Z = 1$) y otro en jacobianas. Mientras que la mejor forma de realizar el doblado de un punto es en coordenadas jacobianas. A continuación se presentan las fórmulas para estas operaciones y sus respectivos costos.

Sea $P = (X_1 : Y_1 : Z_1)$ con $Z_1 \neq 0$ y $Q = (X_2 : Y_2 : 1)$, para $P \neq \pm Q$, se tiene que la suma $R = P + Q = (X_3 : Y_3 : Z_3)$ se puede obtener a través de las ecuaciones [20]:

$$\begin{aligned} X_3 &= (Y_2Z_1^3 - Y_1)^2 - (X_2Z_1^2 - X_1)(X_1 + X_2Z_1^2) \\ Y_3 &= (Y_2Z_1^3 - Y_1)(X_1(X_2Z_1^2 - X_1)^2 - X_3) - Y_1(X_2Z_1^2 - X_1)^3 \\ Z_3 &= (X_2Z_1^2 - X_1)Z_1 \end{aligned}$$

A un costo de tres elevaciones al cuadrado y ocho multiplicaciones.

Dado un punto $P = (X_1 : Y_1 : Z_1)$, su doblado denotado por $2P = (X_2 : Y_2 : Z_2)$ se obtiene a través de las siguientes ecuaciones [20]:

$$\begin{aligned} X_2 &= (3X_1^2 + aZ_1^4)^2 - 8X_1Y_1^2 \\ Y_2 &= (3X_1^2 + aZ_1^4)(4X_1Y_1^2 - X_3) - 8Y_1^4 \\ Z_2 &= 2Y_1Z_1 \end{aligned}$$

Con un costo de seis elevaciones al cuadrado y tres multiplicaciones, sin embargo, cuando $a = 0$ esta operación tiene un costo de cuatro elevaciones al cuadrado y tres multiplicaciones.

2.2. Curvas elípticas sobre campos finitos

Sea p un número primo y sea $q = p^n$, donde $n \in \mathbb{Z}^+$, dado un campo finito $\mathbb{F}_q$ de característica p, los $\mathbb{F}_q$-puntos racionales de una curva elíptica forman un grupo finito $E(\mathbb{F}_q)$, tal que para todo punto $P = (x_P, y_P) \in E(\mathbb{F}_q)$, los valores x_P y y_P son elementos en $\mathbb{F}_q$.

2.2.1. Orden de la curva elíptica

El orden del grupo $E(\mathbb{F}_q)$ es el número de puntos en él. Dado que la Ecuación 2.2 tiene a lo más dos soluciones por cada $x \in \mathbb{F}_q$, sabemos que $\#E(\mathbb{F}_q) \in [1, 2q+1]$. Sin embargo, el teorema de Hasse establece limites más precisos para $\#E(\mathbb{F}_q)$:

Teorema 2.1. *(Teorema de Hasse). Sea E una curva elíptica definida sobre el campo $\mathbb{F}_q$, el intervalo*

$$q + 1 - 2\sqrt{q} \leq \#E(\mathbb{F}_q) \leq q + 1 + 2\sqrt{q},$$

es llamado intervalo de Hasse.

Alternativamente, se puede escribir $\#E(\mathbb{F}_q) = q + 1 - t$, donde $|t| \leq 2\sqrt{q}$. El parámetro t se define como la *traza* de E sobre $\mathbb{F}_q$ y dado que t es relativamente más pequeño que q, $\#E(\mathbb{F}_q) \approx q$.

Dada una curva elíptica $E/\mathbb{F}_q$ con $q = p^n$ y $\#E(\mathbb{F}_q) = q + 1 - t$, se dice que $E/\mathbb{F}_q$ es *supersingular* [3] si p divide a t, es decir, E es textitsupersingular si y sólo si $t \equiv 0 \mod p$, lo cual es cierto si y sólo si $\#E(\mathbb{F}_q) \equiv 1 \mod p$; de otra forma, la curva es llamada *ordinaria*.

2.2.2. Puntos de torsión

Dada una curva elíptica $E/\mathbb{F}_p$ y sea $\bar{\mathbb{F}}_p$ la cerradura algebraica de $\mathbb{F}_p$. Para cualquier entero positivo r, definimos el conjunto de los puntos de torsión r de $E(\bar{\mathbb{F}}_p)$, denotado como $E(\bar{\mathbb{F}}_p)[r]$, como el conjunto de puntos en $E(\bar{\mathbb{F}})$ de orden r, es decir,

$$E(\bar{\mathbb{F}}_p)[r] = \{P \in E(\bar{\mathbb{F}}_p) | rP = \mathcal{O}\}.$$

Sea $n \in \mathbb{Z}^+$, el conjunto de los F_{p^n}-puntos racionales de torsión r, para $\mathbb{F}_p \subseteq \mathbb{F}_{p^n} \subset \bar{\mathbb{F}}_p$, denotado por $E(\mathbb{F}_{p^n})[r]$, es:

$$E(\mathbb{F}_{p^n})[r] = \{P \in E(\mathbb{F}_{p^n}) | rP = \mathcal{O}\}.$$

2.2.3. Grado de encajamiento

Definición 2.2. *(Grado de encajamiento). Para dos números primos p y r, dado el campo finito $\mathbb{F}_p$, considérese una curva elíptica $E/\mathbb{F}_p$ tal que $\#E(\mathbb{F}_p) = h \cdot r$, donde*

$h \in \mathbb{Z}^+$. Sea k un entero positivo, se dice que k es el grado de encajamiento de $E/\mathbb{F}_p$ con respecto a p y r, si k es el menor entero positivo que satisface

$$r|p^k - 1.$$

Sea $\Phi_k(\cdot)$ el k-ésimo polinomio ciclotómico, por definición se cumple que $\Phi_k(p) \mid p^k - 1$ y por lo tanto $r \mid \Phi_k(p)$. Dado que $p \equiv t - 1 \mod r$, donde t es la traza de E sobre $\mathbb{F}_p$, el grado de encajamiento puede ser definido como el menor entero positivo k, tal que

$$r \mid \Phi_k(t-1).$$

2.2.4. Curva enlazada (*twist*)

Definición 2.3. *(Invariante-j). Dada la curva elíptica $E : y^2 = x^3 + ax + b$, el invariante-j de E, denotado como $j(E)$, determina la clase de isomorfismo de E y es definido como*

$$j = -1728\frac{(4a)^3}{\Delta},$$

donde $\Delta = -16(4a^3 + 27b^2)$ es el discrimintante de la curva.

Definición 2.4. *(Curva enlazada) Sean E y E' dos curvas elípticas, se dice que E' es la curva enlazada de E, si y sólo si E y E' tienen el mismo invariante-j y son isomórficas sobre la cerradura algebraica de un campo finito $\mathbb{F}_p$.*

En particular, dada la curva elíptica $E/\mathbb{F}_p$ con grado de encajamiento k, si el grupo finito $E(\mathbb{F}_p)$ tiene un subgrupo de orden primo r, Hess *et al.* [24] demostraron que, existe una curva enlazada E' de E, definida sobre el campo $\mathbb{F}_{p^{k/d}}$, donde $d|k$, con $r|\#E'(\mathbb{F}_{p^{k/d}})$, tal que existe un isomorfismo:

$$\phi : E'(\mathbb{F}_{p^{k/d}}) \to E(\mathbb{F}_{p^k}),$$

en donde el entero d es el grado de la curva enlazada E'.

2.2.5. Endomorfismo de Frobenius

Definición 2.5. *(Endomorfismo de Frobenius). Sea $E/\mathbb{F}_p$ una curva elíptica con grado de encajamiento k y sea $E(\mathbb{F}_p)$ el grupo de los $\mathbb{F}_p$-puntos racionales en $E/\mathbb{F}_p$, tal que $E(\mathbb{F}_p)$ tiene un subgrupo de orden primo r. El endomorfismo de Frobenius π actúa sobre $E(\mathbb{F}_{p^k})$ de la siguiente manera:*

$$\pi : E(\mathbb{F}_{p^k}) \to E(\mathbb{F}_{p^k}), \quad \textit{tal que} \quad \pi(X,Y) = (X^p, Y^p) \in E(\mathbb{F}_{p^k}).$$

Sea t la traza de E sobre $\mathbb{F}_p$, $\sigma(u) = u^2 - tu + p$ es el polinomio característico del endomorfismo de Frobenius, para el que todo punto $Q \in E(\mathbb{F}_{p^k})$ satisface la igualdad

$$\pi^2(Q) - t\pi(Q) + pQ = \mathcal{O}.$$

Si $\sigma(u)$ se factoriza modulo r, entonces

$$\sigma(u) = (u-1)(u-p) \mod r.$$

Por lo tanto, existen dos conjuntos de puntos en $E(\mathbb{F}_{p^k})[r]$ definidos como $\{P \in E(\mathbb{F}_{p^k})[r] \mid \pi(P) = P\}$ y $\{Q \in E(\mathbb{F}_{p^k})[r] \mid \pi(Q) = pQ\}$ [25].

El grupo cíclico $E(\mathbb{F}_p)[r]$ corresponde al primer conjunto, ya que para todo punto $P = (x, y) \in E(\mathbb{F}_p)$ se cumple que $(x^p, y^p) = (x, y)$. Por otra parte, si la curva elíptica $E/\mathbb{F}_p$ tiene una curva enlazada $E'/\mathbb{F}_{p^{k/d}}$ de grado d, con $r|\#E'(\mathbb{F}_{p^{k/d}})$, tal que E y E' son isomórficas bajo $\phi : E'(\mathbb{F}_{p^{k/d}}) \to E(\mathbb{F}_{p^k})$, Barreto *et al.* [25] demostraron que el segundo conjunto es el subgrupo de $E(\mathbb{F}_{p^k})[r]$, formado por el conjunto $\phi(E'(\mathbb{F}_{p^{k/d}})[r]) = \{\phi(Q') \mid Q' \in E'(\mathbb{F}_{p^{k/d}})[r]\}$.

2.3. Emparejamientos bilineales

En la actualidad los emparejamientos bilineales han sido estudiados por diversos investigadores en el campo de la criptografía, ya que sus propiedades han permitido la creación de novedosos esquemas criptográficos. En esta sección, se presentan algunos de los conceptos relacionados con los emparejamientos bilineales, los cuales son parte importante para la realización de la tesis.

2.3.1. Propiedades de los emparejamientos bilineales

Sean $\mathbb{G}_1 = (\mathbb{G}_1, +, 0)$, $\mathbb{G}_2 = (\mathbb{G}_2, +, 0)$ y $\mathbb{G}_T = (\mathbb{G}_T, \cdot, 1)$ grupos cíclicos de orden primo r, un emparejamiento bilineal se define como la proyección:

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T,$$

con las siguientes propiedades:

- **Computable**. Existe un algoritmo capaz de calcular eficientemente $\hat{e}(\cdot, \cdot)$.
- **No degenerado**. Un emparejamiento es no degenerado, si para todo $A \in \mathbb{G}_1$ existe un elemento $C \in \mathbb{G}_2$, tal que $\hat{e}(A, C) \neq 1$ con $A \neq 0$ y $C \neq 0$.
- **Bilinealidad**. Dados los elementos $A, B \in \mathbb{G}_1$ y $C, D \in \mathbb{G}_2$, en donde A, B, C y D son diferentes de cero. Esta propiedad implica que $\hat{e}(A + B, C) = e(A, C) \cdot \hat{e}(B, C)$ y del mismo modo, que $\hat{e}(A, C + D) = \hat{e}(A, C) \cdot \hat{e}(A, D)$. Por lo tanto,

  $$\hat{e}(A + A, C) = \hat{e}(A, C + C) = \hat{e}(A, C) \cdot \hat{e}(A, C)$$

  y en general, para todo $m \in [1, r-1]$ se cumple que

  $$\hat{e}(m \cdot A, C) = \hat{e}(A, m \cdot C) = \hat{e}(A, C)^m.$$

Considerando la curva elíptica ordinaria $E/\mathbb{F}_p$ con grado de encajamiento k, la cual define el grupo $E(\mathbb{F}_p)$ de orden $h \cdot r = p + 1 - t$, en donde r es un número primo y el entero h es conocido como el cofactor. Además, dada la curva enlazada E' de grado d, tal que $E'(\mathbb{F}_{p^{k/d}})$ tiene un subgrupo de orden r, suponemos que E y E' son isomórficas sobre el campo $\mathbb{F}_{p^k}$, es decir, existe una proyección $\phi : E'(\mathbb{F}_{p^{k/d}}) \to E(\mathbb{F}_{p^k})$, los grupos $\mathbb{G}_1$ y $\mathbb{G}_2$ involucrados en el emparejamiento se definen como [19]:

- $\mathbb{G}_1$ es el grupo cíclico escrito de manera aditiva, formado por los puntos de torsión r en la curva elíptica $E(\mathbb{F}_p)$.
- $\mathbb{G}_2$ es el grupo cíclico generado por el punto Q, es decir, $\mathbb{G}_2 = \langle Q \rangle$, en donde dado el elemento $Q' \in E'(\mathbb{F}_{p^{k/d}})[r]$, tal que $\mathbb{G}'_2 = \langle Q' \rangle$ el punto Q se define como $Q = \phi(Q')$.
- $\mathbb{G}_T$ es el subgrupo de $\mathbb{F}^*_{p^k}$ escrito de manera multiplicativa, denotado como $\mathbb{F}^\times_{p^k}$, el cual está formado por el conjunto de las r-ésimas raíces primitivas de la unidad en el grupo cíclico $\mathbb{F}^*_{p^k}$.

2.3.2. Seguridad en los emparejamientos

Un sistema criptográfico basado en emparejamientos bilineales se considera seguro, si el problema del logaritmo discreto es computacionalmente intratable en el subgrupo $\mathbb{F}^\times_{p^k}$ y en el grupo formado por los puntos de torsión r en la curva elíptica E.

En el grupo definido por los puntos de torsión r en la curva E, el mejor ataque conocido para solucionar el problema del logaritmo discreto es el algoritmo paralelizado de Pollard rho, cuya complejidad es $O(\sqrt{r})$ [26, 27] y el mejor ataque conocido para solucionar el problema del logaritmo discreto en el grupo $\mathbb{F}^\times_{p^k}$ es el cálculo de índices, cuya complejidad es subexponencial con respecto al orden del campo, es decir, $O(\exp(1.92 \cdot (\ln p^k)^{1/3} \cdot (\ln \ln p^k)^{2/3}))$ [28].

De acuerdo a lo anterior, la seguridad del emparejamiento es medido con respecto al $\log_2(r)$ y $\log_2(p^k)$, la relación entre ambos parámetros está definida por $k \cdot \rho$, en donde $\rho = \log_2(p)/\log_2(r)$, y dada la complejidad de los ataques, se requiere que $\log_2(p^k)$ sea significativamente mayor a $\log_2(r)$. Las curvas elípticas aptas para la implementación de protocolos basados en emparejamientos, deben poseer un subgrupo de orden primo r "grande" y un grado de encajamiento k relativamente "pequeño", por lo que si se satisfacen ambas condiciones, se dice que las curvas son "amables" con los emparejamientos.

2.3.3. Curvas amables con los emparejamientos

Una definición formal para las curvas amables con los emparejamientos fue propuesta por Freeman *et al.* [29], la cual es presentada a continuación:

Definición 2.6. *Sea E una curva elíptica ordinaria definida sobre el campo finito primo $\mathbb{F}_p$. Se dice que E es amable con los emparejamientos si satisface las siguientes condiciones:*

- *Existe un número primo* r *tal que* $r \geq \sqrt{p}$ *y* $r|\#E(\mathbb{F}_p)$.
- *El grado de encajamiento* k *de la curva elíptica* $E/\mathbb{F}_p$ *con respecto a* r *es menor que* $\log_2(r)/8$.

Este tipo de curvas son construidas a través del método de multiplicación compleja (CM) [30]. En este método se fija el grado de encajamiento k y posteriormente se calculan los enteros p, r y t, los cuales satisfacen las siguientes condiciones:

1. Los números p, r y t deben ser primos, el número primo r debe satisfacer $r \geq \sqrt{p}$, además debe dividir a $p+1-t$ y el número primo t debe ser primo relativo de p.
2. k debe ser el menor entero positivo tal que $r|\Phi_k(t-1)$.
3. Para $D \in \mathbb{Z}^+$ y $f \in \mathbb{Z}$, se debe satisfacer la ecuación:

$$4p - t^2 = Df^2, \tag{2.3}$$

   la cual garantiza que $t \leq 2\sqrt{p}$, en donde D denota el discriminante CM [29].

Estas condiciones definen una curva elíptica ordinaria E sobre el campo $\mathbb{F}_p$ con grado de encajamiento k y $\#E(\mathbb{F}_p) = p+1-t$, tal que $r|\#E(\mathbb{F}_p)$. Además, la ecuación de la curva es determinada a partir del valor D de la Ecuación 2.3, en donde los casos más comunes son [31]:

- $D = 1$, que define la ecuación de la curva $E : y^2 = x^3 + ax$.
- $D = 3$, que define la ecuación de la curva $E : y^2 = x^3 + b$.

Las familias de curvas elípticas son parametrizadas por la terna de funciones $(p(z), r(z), t(z))$ y construidas mediante las condiciones del método descrito previamente, para un entero z que garantice que las evaluaciones de $p(z), r(z)$ y $t(z)$ dan como resultado números primos. Con base en la Ecuación 2.3 se dice que una familia de curvas elípticas es completa, si existe un polinomio $f(z)$ tal que $4p(z)-t(z)^2 = Df(z)^2$, en caso contrario es llamada dispersa [29].

Algunos ejemplos de familias completas de curvas elípticas amables con los emparejamientos son: BN (Barreto-Naehrig) [32], BW (Brezing-Weng) [33], KSS (Kachisa-Schaefer-Scott) [34] y BLS (Barreto-Lynn-Scott) [35], las cuales han sido estudiadas y consideradas para la implementación eficiente de emparejamientos bilineales. En particular el trabajo realizado en esta tesis está enfocado a la familia de curvas BN.

2.3.3.1. Curvas elípticas BN

La familia de curvas BN [32] tiene grado de encajamiento $k = 12$ y define curvas elípticas de orden primo r, es decir, $\#E(\mathbb{F}_p) = r$. La característica del campo, el

orden del grupo y la traza de Frobenius se encuentran parametrizados por:

$$p(z) = 36z^4 + 36z^3 + 24z^2 + 6z + 1$$
$$r(z) = 36z^4 + 36z^3 + 18z^2 + 6z + 1$$
$$t(z) = 6z^2 + 1$$

En las curvas BN la Ecuación 2.3 se cumple para $f(z) = 6z^2 + 4z + 1$ y $D = 3$; por lo tanto, dado $z_0 \in \mathbb{Z}$, si $p = p(z_0)$ y $r = r(z_0)$ son números primos, la ecuación de la curva es $E/\mathbb{F}_p : y^2 = x^3 + b$ y es isomórfica a la curva enlazada de grado $d = 6$, definida como $E'/\mathbb{F}_{p^2} : Y^2 = X^3 + b/\xi$, donde los elementos $b \in \mathbb{F}_p$ y $\xi \in \mathbb{F}_{p^2}$ no tienen residuos cuadráticos ni residuos cúbicos en $\mathbb{F}_p$ y $\mathbb{F}_{p^2}$, respectivamente.

2.3.4. Funciones racionales de la curva elíptica

Dada una curva elíptica E definida sobre un campo finito $\mathbb{F}_q$, donde $q = p^n$ y $n \in \mathbb{Z}^+$, sea $\bar{\mathbb{F}}_q$ la cerradura algebraica de $\mathbb{F}_q$, se dice que $f(x, y)$ es una función racional en $E/\mathbb{F}_q$, si existe un punto $P = (x_P, y_P) \in E(\bar{\mathbb{F}}_q)$, tal que $f(x_P, y_P) \neq \infty$. El conjunto de funciones racionales en $E/\mathbb{F}_q$ está denotado por $\bar{\mathbb{F}}_q(E)$ y para todo $f \in \bar{\mathbb{F}}_q(E)$ se cumple que $f(P)$ es un elemento en el conjunto $\{\bar{\mathbb{F}}_q \cup \infty\}$ [3].

Sean P y Q puntos en la curva elíptica $E/\mathbb{F}$, una función racional $f \in \bar{\mathbb{F}}_q(E)$ tiene un *cero* en P y un *polo* en Q, si y sólo si $f(P) = 0$ y $f(Q) = \infty$, respectivamente. En general, la evaluación de f en un punto P puede ser representada a partir de la siguiente igualdad:

$$f(P) = (u(P))^m \cdot g(P),$$

en donde $m \in \mathbb{Z}$, $u(P) = 0$ y $g(P) \neq \{0, \infty\}$. Por lo que si $m > 0$ entonces f tiene un cero en P y si $m < 0$ entonces f tiene un polo en P. Cabe mencionar que $u(P)$ es llamada la función *uniformadora* y el número entero m es el *orden* de f en P, denotado como $\text{ord}_P(f) = m$.

2.3.5. Divisores

Sea $E/\mathbb{F}_q$ una curva elíptica, a cada punto $P \in E(\bar{\mathbb{F}}_q)$ se le asigna el símbolo formal $[P]$. Un divisor $\mathcal{D}$ sobre $E/\mathbb{F}_q$, es una combinación lineal finita de dichos símbolos con coeficientes en $\mathbb{Z}$ [3].

$$\mathcal{D} = \sum_j a_j[P_j], \qquad a_j \in \mathbb{Z}.$$

Por lo tanto un divisor es un elemento del grupo abeliano generado por los símbolos $[P]$, en donde el grupo de divisores es denotado por $Div(E)$ y las operadores que

definen un divisor son el grado, la suma y el soporte, las cuales se calculan como:

$$\begin{aligned} deg\left(\sum_j a_j[P_j]\right) &= \sum_j a_j \in \mathbb{Z}, \\ sum\left(\sum_j a_j[P_j]\right) &= \sum_j a_j P_j \in E, \\ supp\left(\sum_j a_j[P_j]\right) &= \{P_j \in E | a_j \neq 0\}. \end{aligned}$$

2.3.5.1. Divisores principales

Un divisor $\mathcal{D}$ sobre la curva elíptica $E/\mathbb{F}_q$ con $\deg(\mathcal{D}) = 0$ y $\text{sum}(\mathcal{D}) = \mathcal{O}$, es llamado principal si existe una función racional $f \in \bar{\mathbb{F}}_q(E)$, tal que $\mathcal{D} = \text{div}(f)$, en donde

$$\text{div}(f) = \sum_{P_j \in E} \text{ord}_{P_j}(f)[P_j].$$

Dadas las funciones f y $g \in \bar{\mathbb{F}}_q(E)$, los divisores principales cumplen con las siguientes propiedades:

- $\text{div}(f \cdot g) = \text{div}(f) + \text{div}(g)$.
- $\text{div}(f/g) = \text{div}(f) - \text{div}(g)$.
- La función f es una constante, si y sólo si $\text{div}(f) = 0$.

Una función racional f puede ser evaluada en un divisor $\mathcal{D}$, a través de la fórmula:

$$f(\mathcal{D}) = \prod_{P_j \in supp(\mathcal{D})} f(P_j)^{a_j},$$

de tal manera que, para todo $n \in \mathbb{Z}$, se cumple $f(\mathcal{D})^n = f(n\mathcal{D})$, con $n\mathcal{D} = \sum_j n \cdot a_j[P_j]$.

Algunos conceptos importantes para la definición de los emparejamientos bilineales se enuncian a continuación:

Definición 2.7. *(Reciprocidad de Weil) [36]. Sea $E/\mathbb{F}_q$ una curva elíptica y sean $f, g \neq 0$ funciones racionales en $\bar{\mathbb{F}}_q(E)$ con $supp(div(f)) \cap supp(div(g)) = \emptyset$, entonces:*

$$f(div(g)) = g(div(f))$$

Definición 2.8. *(Relación de equivalencia). Dos divisores $\mathcal{D}_1$ y $\mathcal{D}_2$ son linealmente equivalentes, $\mathcal{D}_1 \sim \mathcal{D}_2$, si y sólo si $\mathcal{D}_1 - \mathcal{D}_2$ es un divisor principal, es decir,*

$$\mathcal{D}_1 - \mathcal{D}_2 = div(f)$$

Definición 2.9. *(Función de Miller) [36]. Una función de Miller de longitud $s \in \mathbb{Z}$ denotada por $f_{s,R}$, es una función racional en $\bar{\mathbb{F}}_q(E)$ con divisor $div(f_{s,R}) = s[R] - [sR] - (s-1)[\mathcal{O}]$.*

Lema 2.1. *Sea $f_{s,R}$ una función de Miller y sea v_R la línea vertical que corta a la curva elíptica E en el punto R, para todo $a, b \in \mathbb{Z}$ se cumple que:*

I) $f_{a+b,R} = f_{a,R} \cdot f_{b,R} \cdot \ell_{aR,bR}/v_{(a+b)R}$

II) $f_{ab,R} = f_{b,R}^{a} \cdot f_{a,bR}$

III) $f_{1,R} = c$, *donde c es una constante, por ejemplo $c = 1$.*

2.3.6. Emparejamiento de Weil

Definición 2.10. *(Emparejamiento de Weil) [36]. Sea un número entero $r > 1$ y sean $\mathcal{D}_1$ y $\mathcal{D}_2$ divisores en una curva elíptica E, con $supp(\mathcal{D}_1) \cap supp(\mathcal{D}_2) = \emptyset$, existen dos funciones racionales f_1 y f_2 en E, tales que $div(f_1) = r\mathcal{D}_1$ y $div(f_2) = r\mathcal{D}_2$. El emparejamiento de Weil es definido como*

$$e_W(\mathcal{D}_1, \mathcal{D}_2) = \frac{f_1(\mathcal{D}_2)}{f_2(\mathcal{D}_1)},$$

es un emparejamiento bilineal no degenerado.

En particular, dados los puntos $P \in \mathbb{G}_1$ y $Q \in \mathbb{G}_2$, f_1 y f_2 son funciones racionales en $\mathbb{F}_{p^k}(E)$ con divisores $\mathrm{div}(f_1) = r[P] - r[\mathcal{O}]$ y $\mathrm{div}(f_2) = r[Q] - r[\mathcal{O}]$, respectivamente, tales que $\mathcal{D}_1 \sim [P] - [\mathcal{O}]$ y $\mathcal{D}_2 \sim [Q] - [\mathcal{O}]$. Por lo tanto, el cálculo de $f_i(\mathcal{D}_j)$ toma valores en el grupo multiplicativo $\mathbb{F}_{p^k}^*$ y además, a través de la reciprocidad de Weil, se cumple que

$$\left(\frac{f_1(\mathcal{D}_2)}{f_2(\mathcal{D}_1)}\right)^r = \frac{f_1(r\mathcal{D}_2)}{f_2(r\mathcal{D}_1)} = \frac{f_1(\mathrm{div}(f_2))}{f_2(\mathrm{div}(f_1))} = 1,$$

es decir, $g = e_W(\mathcal{D}_1, \mathcal{D}_2)$ es un elemento en el subgrupo de las r-ésimas raíces primitivas de la unidad en $\mathbb{F}_{p^k}^*$.

En los últimos años se han hecho distintas mejoras al calculo de los emparejamientos bilineales, se ha demostrado que $f_1(\mathcal{D}_2)$ puede ser reemplazado por $f_1(Q)$ y del mismo modo $f_2(\mathcal{D}_1)$ es sustituido por $f_2(P)$. Por lo que, si P y Q son puntos de torsión r, entonces:

$$\begin{aligned} \mathrm{div}(f_1) &= r[P] - r[\mathcal{O}] = r[P] - [rP] - (r-1)[\mathcal{O}] \quad \text{y} \\ \mathrm{div}(f_2) &= r[Q] - r[\mathcal{O}] = r[Q] - [rQ] - (r-1)[\mathcal{O}], \end{aligned}$$

es decir, f_1 y f_2 se pueden expresar como las funciones de Miller $f_{r,P}$ y $f_{r,Q}$. Lo cual produce,

$$\begin{aligned} e_W : \mathbb{G}_1 \times \mathbb{G}_2 &\to \mathbb{G}_T, \\ (P, Q) &\mapsto \frac{f_{r,P}(Q)}{f_{r,Q}(P)} \end{aligned} \tag{2.4}$$

2.3.7. Emparejamiento de Tate

Definición 2.11. *(Emparejamiento de Tate) [25]. Dados los puntos $P \in \mathbb{G}_1$ y $Q \in \mathbb{G}_2$, consideremos al divisor $\mathcal{D}_Q \sim [Q] - [\mathcal{O}]$ y a la función de Miller $f_{r,P}$. El emparejamiento de Tate no degenerado y bilineal está definido como:*

$$\begin{aligned} \hat{t} : \mathbb{G}_1 \times \mathbb{G}_2 &\rightarrow \mathbb{G}_T, \\ (P, Q) &\mapsto f_{r,P}(Q)^{\frac{p^k-1}{r}} \end{aligned} \tag{2.5}$$

Sea $g = f_{r,P}(Q)$ un elemento en el grupo multiplicativo $\mathbb{F}^*_{p^k}$, a diferencia del emparejamiento de Weil, $\hat{t}(P, Q)$ requiere del cómputo de una exponenciación final, tal que

$$(g^{\frac{p^k-1}{r}})^r = 1,$$

es decir, $g = \hat{t}(P, Q)$ es un elemento en el subgrupo de las r-ésimas raíces primitivas de la unidad en $\mathbb{F}^*_{p^k}$.

2.3.8. Emparejamiento ate

El emparejamiento ate es una derivación del emparejamiento Tate, el cual se define de la siguiente manera:

Definición 2.12. *(Emparejamiento de ate) [24]. Dada una curva elíptica $E/\mathbb{F}_p$ con grado de encajamiento k, sea $E(\mathbb{F}_p)$ el grupo de los $\mathbb{F}_p$-puntos racionales de $E/\mathbb{F}_q$, con orden $\#E(\mathbb{F}_q) = h \cdot r = p + 1 - t$, donde t la traza de E sobre $\mathbb{F}_p$. Dados dos puntos $P \in E(\mathbb{F}_p)[r]$ y $Q \in E'(\mathbb{F}_{p^{k/d}})[r]$, el emparejamiento ate está definido como*

$$\hat{a}(Q, P) = f_{t-1,Q}(P)^{\frac{p^k-1}{r}} \tag{2.6}$$

2.3.8.1. Emparejamiento óptimo ate

Sea $f_{s,R}$ la función de Miller de longitud $s \in \mathbb{Z}$, esta función es calculada mediante el algoritmo de Miller, el cual será descrito en la Sección 2.3.9; en general este algoritmo requiere de $\log_2(s)$ iteraciones para ser computado. De acuerdo con Vercauteren [37], un emparejamiento bilineal es considerado óptimo si puede ser computado con $\log_2(r)/\varphi(k) + \varepsilon(k)$ iteraciones del algoritmo de Miller, donde $\varepsilon(k) \leq \log_2(k)$. En el caso del emparejamiento de Tate, se tiene que el número de iteraciones está relacionado con el orden r de la curva elíptica, mientras que para el emparejamiento ate, se observa que el número de iteraciones depende de la traza t de la curva. Dado que $t \approx \sqrt{r}$, el emparejamiento ate presenta una mejora ya que el número de iteraciones se reduce a la mitad.

Una mejor forma para el cálculo del emparejamiento ate es el conocido como emparejamiento óptimo ate [37], el cual se define como:

$$\begin{aligned} \hat{a}_{opt} : \mathbb{G}_2 \times \mathbb{G}_1 &\rightarrow \mathbb{G}_T \\ (Q, P) &\mapsto \left[f_{6z+2}(P) \cdot \ell_{(6z+2)Q,\pi(Q)}(P) \cdot \ell_{(6z+2)Q+\pi(Q),\pi^2(Q)}(P) \right]^{(p^{12}-1)/r} \end{aligned} \tag{2.7}$$

en donde $\pi : (x, y) \mapsto (x^p, y^p)$ es el endomorfismo de Frobenius sobre el punto Q. Dado que $z \approx \sqrt[4]{t}$, el emparejamiento óptimo ate sólo requiere una cuarta parte de iteraciones en el ciclo de Miller en relación con el emparejamiento Tate.

2.3.9. Ciclo de Miller

Una de las partes principales en el cálculo del emparejamiento es la evaluación de la función de Miller $f_{r,P}$, la cual es realizada utilizando repetidamente el Lema 2.1 en el método de suma y doblado de líneas (Ecuaciones 2.8), sobre la representación binaria de $r = (r_{l-1}, \ldots, r_1, r_0)$.

$$f_{r+1,P} = f_{r,P} \cdot \ell_{rP,P}/v_{(r+1)P} \quad \text{y} \quad f_{2r,P} = f_{r,P}^2 \cdot \ell_{rP,rP}/v_{(2r)P} \tag{2.8}$$

El algoritmo para el cálculo de $f_{r,P}$ es conocido como *ciclo de Miller*, a continuación se presenta este algoritmo de acuerdo con el trabajo de Barreto *et al.* [25], en donde se demuestra que las líneas verticales $v_{(r+1)P}$ y $v_{(2r)P}$ pueden ser omitidas.

Algoritmo 2.1 Ciclo de Miller.

Entrada: $r = (r_{l-1}, \ldots, r_1, r_0)_2$, $Q, P \in E(\bar{\mathbb{F}}_p)$ tal que $P \neq Q$
Salida: $f_{r,Q}(P)$
1: $T \leftarrow Q$, $f \leftarrow 1$
2: **for** $i = l - 2 \rightarrow 0$ **do**
3: $\quad f \leftarrow f^2 \cdot \ell_{T,T}(P)$, $T \leftarrow 2T$
4: $\quad$ **if** $r_i = 1$ **then**
5: $\quad\quad f \leftarrow f \cdot \ell_{T,Q}(P)$, $T \leftarrow T + Q$
6: $\quad$ **end if**
7: **end for**
8: **return** f

En el Algoritmo 2.1, se puede observar que en el paso 3 es necesario hacer la evaluación de la línea $\ell_{T,T}$ y el doblado del punto T, mientras que en el paso 5 es necesario evaluar la línea $\ell_{T,P}$ y realizar la suma de los puntos $T + P$. De acuerdo con Costello *et al.* [38], la forma más eficiente para realizar estas operaciones es a través del uso de coordenadas proyectivas estándar.

Sea E una curva elíptica BN como se definió en la Sección 2.3.3.1 y sea $E/\mathbb{F}_p : Y^2Z = X^3 + bZ^3$ su forma proyectiva. Dado un punto $T = (X_1 : Y_1 : Z_1)$ en la curva enlazada $E'/\mathbb{F}_{p^2}$ de $E/\mathbb{F}_p$, se puede calcular $2T = (X_2 : Y_2 : Z_2)$ a través de las siguientes fórmulas [39]:

$$X_2 = \frac{X_1Y_1}{2}(Y_1^2 - 9b'Z_1^2), \qquad Y_2 = \left[\frac{1}{2}(Y_1^2 + 9b'Z_1^2)\right]^2 - 27b'^2Z_1^4, \qquad Z_2 = 2Y_1^3Z_1.$$

Además la línea $\ell_{T,T}$ evaluada en $P = (x_P, y_P) \in E(\mathbb{F}_p)$, puede ser calculada simultáneamente con el doblado $2T$ debido a que esta dada por la ecuación:

$$\ell_{T,T}(P) = -2Y_1Z_1y_P + (3X_1x_P)w + (3b'Z_1^2 - Y_1^2)w^3 \quad \in \mathbb{F}_{p^k}^*.$$

De la misma forma, dados los puntos $T = (X_1, Y_1, Z_1)$, $Q = (X_2, Y_2, 1) \in E'(\mathbb{F}_{p^2})$ y $P = (x_P, y_P) \in E(\mathbb{F}_p)$, se puede calcular la suma de puntos $R = T + Q = (X_3, Y_3, Z_3)$ y $\ell_{T,Q}(P)$ a través de las siguientes fórmulas [39]:

$$X_3 = \lambda(\lambda^2 + Z_1\theta^2 - 2X_1\lambda^2) \qquad Y_3 = \theta(3X_1\lambda^2 - \lambda^3 - Z_1\theta^2) - Y_1\lambda^3 \qquad Z_3 = Z_1\lambda^3$$

$$\ell_{T,Q}(P) = \lambda y_P - (\theta x_P)w + (\theta X_2 - \lambda Y_2)w^3,$$

en donde $\theta = Y_1 - Y_2Z_1$ y $\lambda = X_1 - X_2Z_1$. La suma de puntos $R = T + Q$ y la evaluación de la línea $\ell_{T,Q}(P)$ también pueden ser realizadas de manera simultánea.

Por otra parte, en el *ciclo de Miller* se requiere de la multiplicación del elemento $f \in \mathbb{F}_{p^{12}}$ por la evaluación de las líneas $\ell_{T,T}$ y $\ell_{T,Q}$ en los pasos 3 y 5 del Algoritmo 2.1, respectivamente. La evaluación de estas líneas definen un elemento en el grupo $\mathbb{F}^*_{p^k}$ con la mitad de los coeficientes en 0, por lo que el producto de f por $\ell_{T,T}$ y $\ell_{T,Q}$ se puede realizar de manera eficiente a través el método conocido como *multiplicación dispersa*.

2.3.10. Exponenciación final

Como se observó en la Sección 2.3.7, los emparejamientos bilineales derivados del emparejamiento de Tate requieren del cómputo de la *exponenciación final* $f^{(p^k-1)/r} \in \mathbb{F}^{\times}_{p^k}$, la cual garantiza que el resultado obtenido en el cálculo del emparejamiento es único. Realizar la exponenciación final directamente resulta ser muy costoso computacionalmente, por lo que una forma más eficiente de efectuar esta operación es representando $(p^k - 1)/r$ como producto de dos exponentes:

$$\frac{p^k - 1}{r} = \frac{p^k - 1}{\Phi_k(p)} \cdot \frac{\Phi_k(p)}{r},$$

en donde $\Phi_k(p)$ es el k-ésimo polinomio ciclotómico evaluado en p. Para el caso particular de las curvas BN (Sección 2.3.3.1), $k = 12$ y por tanto el exponente puede ser expresado como:

$$\frac{p^{12} - 1}{r} = (p^6 - 1) \cdot (p^2 + 1) \cdot \frac{p^4 - p^2 + 1}{r}.$$

Si se define $f \in \mathbb{F}_{p^{12}}$ como una extensión cuadrática de una extension séxtica, el elemento f es representado por $f = f_0 + f_1W$, entonces se define el conjugado de f como $\bar{f} = f_0 - f_1W$. Esta observación es de ayuda para el cálculo del primer factor ya que $f^{p^6} = \bar{f}$, por lo que $f^{p^6-1} = \bar{f} \cdot f^{-1}$. Después de realizar esta operación $g = f^{p^6-1}$ se convierte en un elemento del grupo ciclotómico $\mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$ en donde las operaciones tienen un menor costo computacional.

La exponenciación $g = g^{p^2+1}$ puede realizarse por medio de una multiplicación y la aplicación del operador de Frobenius. Finalmente al cálculo de f^d, en donde $d = (p^4 - p^2 + 1)/r$, se le conoce como *parte difícil de la exponenciación final*, debido

a que requiere un mayor número de operaciones. Para el caso de este trabajo, la exponenciación es realizada por $d = (p^4 - p^2 + 1)/r$ de acuerdo a la mejora descrita por Fuentes-Castañeda *et al.* [40], en la que es posible reducir el número de operaciones utilizando un múltiplo d' de d siempre que $r \nmid d$.

El método consiste en expresar el exponente $d(z) = (p(z)^4 - p(z)^2 + 1)/r(z)$, en donde $p(z)$ y $r(z)$ son los parámetros de la curva BN, como:

$$\begin{aligned} d(z) &= (-36z^4 - 30z^2 + 18z - 2) &+ \\ &= (-36z^3 - 18z^2 - 12z + 1)p(z) &+ \\ &= (6z^2 + 1)p(z)^2 &+ \\ &= p(z)^3 \end{aligned}$$

Posteriormente utilizando el método de rejillas y el algoritmo LLL [41] se obtiene el múltiplo $d'(z) = \lambda_0 + \lambda_1 p + \lambda_2 p^2 + \lambda_3 p^3 = (12z^3 + 6z^2 + 2z)d(z)$, en donde:

$$\begin{aligned} \lambda_0 &= 12z^3 + 12z^2 + 6z + 1 \\ \lambda_1 &= 12z^3 + 6z^2 + 4z \\ \lambda_2 &= 12z^3 + 6z^2 + 6z \\ \lambda_3 &= 12z^3 + 6z^2 + 4z - 1 \end{aligned}$$

En la exponenciación $g^{d'(z)}$ se requiere el cálculo las siguientes variables:

$$f^z \mapsto f^{2z} \mapsto f^{4z} \mapsto f^{6z} \mapsto f^{6z^2} \mapsto f^{12z^2} \mapsto f^{12z^3},$$

las cuales requieren de 3 exponenciaciones por z, 3 cuadrados y una multiplicación en $\mathbb{F}_{p^{12}}$. Finalmente, dadas las variables $a = g^{12z^3} \cdot g^{6z^2} \cdot g^{6z}$ y $b = a \cdot (g^{2z})^{-1}$, la exponenciación $g^{d'(z)}$ es calculada como:

$$g^{d'(z)} = \left[a \cdot g^{6z^2} \cdot g\right] \cdot [b]^p \cdot [a]^{p^2} \cdot \left[b \cdot g^{-1}\right]^{p^3} \in \mathbb{F}_{p^{12}}^{\times}.$$

Dado que $g \in \mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$, el costo de $g^{d'(z)}$ es de 3 aplicaciones del operador de Frobenius, 3 exponenciaciones por z, 10 multiplicaciones en $\mathbb{F}_{p^{12}}$ y tres cuadrados en el grupo $\mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$.

2.4. Problema del logaritmo discreto en grupos

En esta sección se utiliza la notación multiplicativa de un grupo para definir el problema del logaritmo discreto (PLD).

Definición 2.13. *(Problema del logaritmo discreto). Dado un grupo abeliano* $\mathbb{G} = (\mathbb{G}, \cdot, 1)$ *y un generador* $g \in \mathbb{G}$*, el problema del logaritmo discreto consiste en encontrar la solución a la ecuación* $g^x = h \in \mathbb{G}$*, denotada como* $x = \log_g(h)$.

- **Problema computacional de Diffie-Hellman (CDH)**. Dado el grupo cíclico $\mathbb{G} = (\mathbb{G}, \cdot, 1)$, tal que $\mathbb{G} = \langle g \rangle$ y dados dos elementos $h_1, h_2 \in \mathbb{G}$, el problema computacional de Diffie-Hellman consiste en calcular $g^{\log_g(h_1)\cdot\log_g(h_2)}$.

- **Problema de decisión de Diffie-Hellman (DDH)**. Dados los elementos g, g^a, g^b, b^c en el grupo $\mathbb{G} = \langle g \rangle$, donde $a, b, c \in \mathbb{Z}^+$, consiste en determinar si $g^{ab} = g^c$.

Una variación de estos problemas, es la correspondiente al contexto de los emparejamientos bilineales, en la que se define una instancia del problema de emparejamiento como la tupla $\Gamma = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, \hat{e})$, en donde $\mathbb{G}_1, \mathbb{G}_2$ y $\mathbb{G}_T$ corresponden a los grupos de orden primo q involucrados en el emparejamiento $\hat{e}$ y P_1, P_2 denotan a los generadores de $\mathbb{G}_1$ y $\mathbb{G}_2$ respectivamente. Los problemas pueden ser planteados para el caso en que los emparejamientos se definen como $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, con $\mathbb{G}_1 = \mathbb{G}_2$ o $\mathbb{G}_1 \neq \mathbb{G}_2$.

Dada una instancia del problema de emparejamiento Γ y los valores $i, j, k \in \{1, 2\}$, se pueden definir los siguientes problemas de manera general para ambos casos:

- **Problema Bilineal de Diffie-Hellman (BDH)**. Dados los puntos aP_i, bP_j y cP_k, con $a, b, c \in \mathbb{F}_q$, consiste en calcular $\hat{e}(P_1, P_2)^{abc}$, a partir de los puntos dados.

- **Problema co-bilineal de Diffie-Hellman (coBDH)**. Dados los puntos aP_1, aP_2, bP_i y cP_j, con $a, b, c \in \mathbb{F}_q$, consiste en calcular $\hat{e}(P_1, P_2)^{abc}$, a partir de los puntos dados.

- **Problema eXterno de Diffie-Hellman (XDH)**. Establece formalmente que:

  1. El problema del logaritmo discreto, el problema computacional de Diffie-Hellman y el problema computacional co-Diffie-Hellman son intratables en $\mathbb{G}_1$ y $\mathbb{G}_2$.
  2. El problema de decisión de Diffie-Hellman es intratable en $\mathbb{G}_1$.

3

Funciones picadillo deterministas hacia los grupos $\mathbb{G}_1$ y $\mathbb{G}_2$

"La irrealidad de lo mirado, da realidad a la mirada."
∼Octavio Paz (1914-1998)∼

En este capítulo en la Sección 3.1 se presenta una introducción al cálculo de funciones picadillo hacia grupos, en la Sección 3.2 se describen los antecedentes del cálculo del picadillo a curvas elípticas, también en la Sección 3.4 se presentan las codificaciones existentes para obtener puntos en curvas elípticas, en la Sección 3.5 además se muestran los trabajos relacionados con el cálculo del picadillo a $\mathbb{G}_1$ de manera determinista y por último se propone una manera de realizar el cálculo del picadillo a $\mathbb{G}_2$ de manera determinista en la Sección 3.6.

3.1. Función picadillo hacia grupos

La mayoría de los protocolos criptográficos para firmas digitales, intercambio de llaves autenticadas basados en contraseñas, cifrado basado en la identidad, etc., requieren de una operación que proyecte un valor arbitrario a un elemento en el grupo en donde se realizan los cálculos del protocolo, esta operación puede ser realizada en dos fases:

La primera fase, es la transformación del valor arbitrario a un valor de longitud fija, el cual se espera que sea aleatorio, a través de la utilización de una función picadillo definida como:

Definición 3.1. *Una función picadillo* $\mathfrak{h} : \{0,1\}^* \to \{0,1\}^n$ *es una función de sólo ida, que convierte una cadena* x *de longitud arbitraria a una cadena binaria de longitud fija* n*, el resultado de la aplicación de* $\mathfrak{h}$ *sobre* x *es conocido como digesto.*

Una función picadillo es considerada segura si los siguientes problemas son difíciles de resolver:

- *Preimagen.* Dado un elemento $y \in \{0,1\}^n$, el problema consiste en encontrar $x \in \{0,1\}^*$ tal que $\mathfrak{h}(x) = y$.

- *Segunda preimagen.* Dado $x \in \{0,1\}^*$, el problema radica en encontrar un elemento $x' \in \{0,1\}^*$ tal que $x \neq x'$, pero $\mathfrak{h}(x) = \mathfrak{h}(x')$.

- *Colisión.* El problema consiste en encontrar elementos $x, x' \in \{0,1\}^*$, tal que $\mathfrak{h}(x) = \mathfrak{h}(x')$.

Además una función picadillo debe poder calcularse de manera eficiente y asegurar que el cambio de incluso un bit en x, producirá un digesto diferente.
La segunda fase es la correspondiente al grupo en donde se realizan los cálculos del protocolo, por lo que es específica para el grupo en donde se esté trabajando.

Ejemplo 1. Si se considera que los cálculos en un protocolo son realizados en el grupo $\mathbb{G} = \mathbb{Z}_p^*$, con p un número primo. Una manera de realizar el cálculo de la función picadillo hacia $\mathbb{G}$ es la siguiente:

Fase 1: Dado el valor arbitrario x, calcular $d = \mathfrak{h}(x)$, tal que d tenga un tamaño fijo conveniente.

Fase 2: Reducir d módulo p.

Sin embargo, no se puede seguir una técnica equivalente a la presentada en el Ejemplo 1 para el caso de los grupos en curvas elípticas.

Ejemplo 2. Si se considera que los cálculos en un protocolo son realizados en el grupo $\mathbb{G}$ definido por la curva elíptica $E(\mathbb{F}_p)$, con p un número primo, una forma de realizar el cálculo de la función picadillo hacia $\mathbb{G}$ es:

Fase 1: Dado el valor arbitrario x, calcular $d = \mathfrak{h}(x)$, tal que d tenga un tamaño fijo conveniente.

Fase 2: Reducir d módulo p, colocarlo en la coordenada x del punto en la curva elíptica y calcular la coordenada y a partir de la evaluación de la ecuación de la curva en x.

Aunque esta técnica es simple, no es muy apropiada, ya que sólo la mitad de los posibles valores de la coordenada x corresponden a un punto real en la curva elíptica. Por lo que una parte importante del trabajo de tesis ésta enfocada al estudio de las funciones picadillo hacia grupos en curvas elípticas, específicamente la correspondiente al grupo denominado $\mathbb{G}_2$.

3.2. Antecedentes

Muchos protocolos criptográficos, en específico los que utilizan curvas elípticas, requieren la función picadillo hacia el grupo $\mathbb{G}$ definido por los puntos en la curva utilizada, esta operación es denotada como $\mathfrak{H} : \{1,0\}^* \to \mathbb{G}$. Tales protocolos involucran funciones picadillo que proyectan valores arbitrarios a los grupos $\mathbb{G}_1$ y $\mathbb{G}_2$, utilizados en el cálculo de emparejamientos bilineales (Sección 2.3 en la página 16). Por ejemplo, en el esquema de cifrado basado en la identidad propuesto por Boneh-Franklin [16] la llave pública para la identidad $\mathrm{id} \in \{0,1\}^*$ es el punto $Q_{\mathrm{id}} = \mathfrak{H}(\mathrm{id})$. Este también es el caso de muchos otros sistemas criptográficos basados en emparejamientos, entre los cuales se encuentran los esquemas: IBE y HIBE [42–44]; de firma [45–47]; y esquemas que aplican firma y cifrado basados en la identidad [48]. La función picadillo hacia grupos es también requerida en algunos protocolos de autenticación basados en contraseñas, tales como, SPEKE [49], PAKE [50] y en los protocolos propuestos recientemente por Michael Scott [17, 18] estudiados en esta tesis.

En todos los sistemas criptográficos mencionados, las funciones picadillo son modeladas como oráculos aleatorios [51] en las pruebas de seguridad. Sin embargo, no queda claro como deben ser aplicadas en la práctica. De hecho, los oráculos aleatorios a grupos como el del Ejemplo 1 pueden ser fácilmente construidos a partir de oráculos aleatorios de cadenas de bits de longitud fija, por lo que las funciones picadillo criptográficas convencionales suelen proporcionar sustitutos aceptables. Por otra parte, construir oráculos aleatorios a curvas elípticas, incluso a partir de oráculos aleatorios de cadenas de bits, parece difícil en general y algunas de las más obvias técnicas rompen completamente la seguridad.

3.2.1. Construcción ingenua de la función picadillo a grupos

La forma más simple de construir una función picadillo $\mathfrak{H} : \{1,0\}^* \to \mathbb{G}$ hacia un grupo $\mathbb{G}$ de orden r, el cual es generado por el punto G, es probablemente empezar por aplicar una función picadillo $\mathfrak{h} : \{0,1\}^* \to \mathbb{Z}_r$ y definir $\mathfrak{H}$ como:

$$\mathfrak{H}(m) = \mathfrak{h}(m) \cdot G, \tag{3.1}$$

en donde $m \in \{0,1\}^*$. Sin embargo, al utilizar la función picadillo $\mathfrak{H}$ en un protocolo, se permite a un atacante calcular el logaritmo discreto de $\mathfrak{H}(m)$, siempre que conozca el valor de m, lo que sin duda no es deseable desde una perspectiva de seguridad.

Para mostrar las consecuencias de la construcción ingenua presentada, se utilizará el esquema de firma corta BLS propuesto por Boneh, Lynn y Shacham [46], el cual se vuelve completamente inseguro si la función picadillo involucrada es aplicada de acuerdo a la Ecuación 3.1 [52].

Los parámetros públicos del esquema BLS son: el grupo cíclico $\mathbb{G}$ de orden primo r dotado con un emparejamiento bilineal simétrico no degenerado $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ (aunque también puede ser extendido a emparejamientos asimétricos), un generador G del grupo y una función picadillo $\mathfrak{H} : \{0,1\}^* \to \mathbb{G}$. El esquema de firma consta de tres algoritmos, los cuales se describen a continuación:

- **Genera llaves():** Elige $x \xleftarrow{\$} \mathbb{Z}_p$ como llave privada y $P \leftarrow x \cdot G$ como llave pública.
- **Firma(m, x):** Calcula la firma del mensaje m como $S \leftarrow x \cdot \mathfrak{H}(m)$
- **Verifica(m, S, P):** Acepta el mensaje si y sólo si $e(\mathfrak{H}(m), P) = e(S, G)$

Boneh, Lynn y Shacham demostraron que si el problema computacional de Diffie-Hellman es difícil de resolver en $\mathbb{G}$, entonces el esquema es seguro cuando $\mathfrak{H}$ es modelada como un oráculo aleatorio. Si se considera el caso cuando $\mathfrak{H}$ es definida como en la Ecuación 3.1, la firma sobre un mensaje m puede ser escrita como:

$$S = x \cdot \mathfrak{H}(m) = x \cdot \mathfrak{h}(m) \cdot G = \mathfrak{h}(m) \cdot P,$$

y entonces, un atacante puede falsificar una firma sobre cualquier mensaje usando sólo datos públicos. Por lo que no hay seguridad cuando se construye la función picadillo de manera trivial.

3.3. Intentar e incrementar

Una construcción clásica de la función picadillo a curvas elípticas, que funciona para los grupos $\mathbb{G}_1$ y $\mathbb{G}_2$, es el algoritmo conocido como *try-and-increment*. Este algoritmo fue presentado por Boneh, Lynn y Shacham [46] y es considerado como la primer construcción genérica de una función picadillo segura a curvas elípticas, considerada genérica en el sentido de que puede ser adaptada a cualquier curva elíptica.

Suponiendo una curva elíptica E sobre un campo finito $\mathbb{F}_q$ de característica prima impar q, con una ecuación de Weierstrass de la forma $E : y^2 = x^3 + ax + b$, para algunas constantes $a, b \in \mathbb{F}_q$. Esta construcción probabilista de puntos en $E(\mathbb{F}_q)$, se realiza de acuerdo a los siguientes pasos: se elige un valor $x \in \mathbb{F}_q$ de manera aleatoria, se verifica que $t = x^3 + ax + b$ es un residuo cuadrático[1] en $\mathbb{F}_q$, si lo es entonces se regresa el punto $(x, \pm\sqrt{t})$. Si t no es un residuo cuadrático significa que x no es la abscisa de un punto en $E(\mathbb{F}_q)$, entonces se elige otro valor $x \in \mathbb{F}_q$ de manera aleatoria y se intenta nuevamente. Dado que se sabe que en $\mathbb{F}_q^*$ existen exactamente $(q-1)/2$ residuos cuadráticos, entonces la probabilidad de éxito de un sólo intento es cercana a $1/2$.

Es posible convertir la construcción de puntos anterior en una función picadillo a curvas elípticas basada en el oráculo aleatorio $\mathfrak{h} : \{0,1\}^* \to \mathbb{F}_q$. La idea para hacer el picadillo de un mensaje m, es escoger la coordenada x como $\mathfrak{h}(m)$ y regresar como resultado la construcción de puntos anterior. Sin embargo, ya que debe ser posible intentar más de una vez, en caso de que la coordenada x no sea la abscisa de un punto en $E(\mathbb{F}_q)$, podemos tomar x como $\mathfrak{h}(m||c)$, en donde $m||c$ representa la concatenación

[1] Dentro de la Teoría de Números se denomina residuo cuadrático módulo p a cualquier entero a, con $gcd(a, p) = 1$, tal que satisfaga la ecuación $x^2 \equiv a \bmod p$, es decir, a es una cuadrado perfecto módulo p, y por tanto tiene una raíz cuadrada en la aritmética de módulo p.

del mensaje con un contador inicialmente puesto a 0 y el cual es incrementado en caso de fallo. El algoritmo resultante es el conocido como *try-and-increment* y se presenta a continuación.

Algoritmo 3.1 Función picadillo a curvas elípticas *try-and-increment*

Entrada: Un mensaje m y un entero k
Salida: El punto $(x, y) \in E(\mathbb{F}_q)$

```
1: c ← 0
2: for c < k do
3:    x ← 𝔥(m||c)
4:    y ← x^3 + ax + b
5:    if χ_q(y) = 1 then
6:       return (x, ±√y)
7:    end if
8:    c ← c + 1
9: end for
```

En el Algoritmo 3.1 la función $\chi_q(\cdot)$ representa el criterio de Euler extendido a cero, en donde $\chi_q(0) = 0$ y para $a \neq 0$ y $a \in \mathbb{F}_q$, $\chi_q(a) = 1$ si a es un residuo cuadrático y -1 en caso contrario. La probabilidad de fallo del algoritmo después de k iteraciones es de 2^{-k}, por lo que la elección de la longitud del contador k para un máximo de 128 iteraciones, es suficiente para asegurar que el algoritmo tenga éxito excepto con una probabilidad insignificante, aunque esto es algo ineficiente debido a que se pueden necesitar muchas iteraciones antes de encontrar un punto adecuado.

Boneh, Lynn y Shacham probaron que esta construcción puede remplazar el oráculo aleatorio $\mathfrak{H} : \{0,1\}^* \to E(\mathbb{F}_q)$ en las firmas BLS sin comprometer la seguridad, y en realidad no es difícil ver que la construcción no puede ser diferenciada de un oráculo aleatorio, de acuerdo con Maurer *et al.* [53]. Esto asegura que la construcción anterior pueda ser introducida en muchos protocolos que requieran de $\mathfrak{H}$, preservando las pruebas de seguridad.

Sin embargo, existen varias razones por las cuales el Algoritmo 3.1 no es una construcción de función picadillo a curvas elípticas completamente satisfactoria. Una de ellas es la falta de elegancia matemática, provocada por la idea de elegir la coordenada x de manera aleatoria hasta encontrar una coordenada correcta. Pero la razón más importante puede tener consecuencias adversas para la seguridad, cuando se implementa un protocolo que utiliza esta construcción, ya que el número de iteraciones en el Algoritmo 3.1 depende de la entrada m y un atacante puede obtener información acerca de la entrada midiendo el tiempo de ejecución o el consumo de potencia de la implementación, en especial en protocolos de intercambio de llaves autenticadas.

3.3.1. Ataque de análisis de tiempo

Una circunstancia real en la que la variación del tiempo de ejecución en el Algoritmo 3.1 puede ser un problema serio, se presenta en el caso de la implementación

de un protocolo de intercambio de llaves autenticadas basado en contraseñas[2] [52].

Es importante señalar, que un protocolo de este tipo es considerado vulnerable si un atacante, que sólo puede escuchar el canal de comunicación, es capaz de obtener cualquier información relacionada con la contraseña. Por ejemplo, el protocolo propuesto por Jablon [54], modificado en [52], presentado en la Figura 3.1, cuyos parámetros públicos son: un grupo $\mathbb{G}$ definido sobre una curva elíptica de orden primo r y una funcion picadillo $\mathfrak{H} : \{0,1\}^* \to \mathbb{G}$. El cual tiene como característica que ambas partes comparten una contraseña π y derivan un secreto $k \in \mathbb{G}$, utilizando un acuerdo de llaves de Diffie-Hellman con un generador $G \in \mathbb{G}$ variable obtenido mediante la aplicación de la función picadillo $\mathfrak{H}$ a la contraseña.

Alicia		**Beto**
	$\xleftarrow{s}$	$s \xleftarrow{\$} \{0,1\}^k$
$G \leftarrow \mathfrak{H}(s\|\|\pi)$		$G \leftarrow \mathfrak{H}(s\|\|\pi)$
$r_A \xleftarrow{\$} \mathbb{Z}_r$		$r_B \xleftarrow{\$} \mathbb{Z}_r$
$A \leftarrow r_A \cdot G$	$\xrightarrow{A}$	
	$\xleftarrow{B}$	$B \leftarrow r_B \cdot G$
$K \leftarrow r_A \cdot B$		$K \leftarrow r_B \cdot A$

Figura 3.1: Protocolo de intercambio de llaves autenticadas

Si la función picadillo $\mathfrak{H}$ es implementada de acuerdo al Algoritmo 3.1 y el atacante es capaz de medir el tiempo de ejecución de una de las partes involucradas en el protocolo, encontrará diferentes tiempos de ejecución dependiendo de cuantos intentos fueron necesarios para obtener una coordenada x adecuada en el cálculo de $\mathfrak{H}(s||\pi)$. Dado que una iteración tiene una probabilidad cercana a $1/2$, una ejecución del protocolo provee al menos un bit de información relacionada con π al atacante. En consecuencia el atacante puede contar el número de iteraciones necesarias para calcular $\mathfrak{H}(s||\pi)$ para cada contraseña π_0 en su diccionario de contraseñas, manteniendo sólo aquellas que coincidan con la medición realizada. Esto reduce el espacio de búsqueda por un factor de al menos 2, para cada ejecución del protocolo ya que los tiempos de ejecución para diferentes valores de s son independientes. Como resultado, el atacante puede reducir su espacio de búsqueda a una única contraseña después de algunas pocas docenas de ejecuciones del protocolo.

[2]Una definición de protocolo de intercambio de llaves autenticadas es presentado en la Sección 4.4 en la página 51

3.4. Codificaciones deterministas a curvas elípticas

Una forma natural para construir una función picadillo determinista a una curva elíptica $E(\mathbb{F}_p)$, donde p es un número primo, sería usando como bloque de construcción una función $f : \mathbb{F}_p \to E(\mathbb{F}_p)$, la cual pueda calcularse eficientemente en un tiempo constante. Después, combinando f con una función picadillo $\mathfrak{h} : \{0,1\}^* \to \mathbb{F}_p$, esperar obtener una función picadillo hacia $E(\mathbb{F}_p)$ bien comportada. En lo siguiente, nos referiremos a la función f como codificación.

El desafío de una codificación determinista ha estado por lo menos desde 1985, cuando fue planteado por René Schoof [55], quien dijo que el problema presumiblemente fácil de construir un sólo punto distinto a la identidad en una curva elíptica sobre un campo finito, era un problema abierto. El primer resultado significativo para la obtención de una codificación determinista, fue presentado por Schinzel y Skałba [56], quienes, dada la curva elíptica:

$$E : y^2 = g(x) = x^3 + ax^2 + bx + c, \tag{3.2}$$

sobre $\mathbb{F}_p$, con p un número primo impar, introdujeron el triple algebraico $V \subset \mathbb{P}^4$ con ecuación afín:

$$V : y^2 = g(x_1) \cdot g(x_2) \cdot g(x_3), \tag{3.3}$$

y observaron que si (x_1, x_2, x_3, y) es un punto racional en V sobre $\mathbb{F}_p$, entonces al menos uno de los valores x_1, x_2, x_3 es la abscisa de un punto en la curva elíptica E. En realidad, el producto $g(x_1) \cdot g(x_2) \cdot g(x_3) \in \mathbb{F}_p$ es un residuo cuadrático, por lo que al menos uno de los factores debería ser un residuo cuadrático también.

El primer ejemplo de una codificación determinista es la presentada por Boneh y Franklin [16] para curvas elípticas supersingulares. Mientras que para curvas elípticas ordinarias, las principales funciones de codificación se deben al trabajo realizado por Shallue y Woestijne [57] y al trabajo realizado por Icart [58].

3.4.1. Codificación de Shallue y Woestijne

El primer algoritmo para la generación de puntos, en curvas elípticas ordinarias de manera determinista en tiempo polinomial, fue publicado por Shallue y Woestijne [57]. El trabajo presentado por los autores se centra en el siguiente teorema:

Teorema 3.1. *[57, Teorema 1]. Existe un algoritmo determinista que, dado un campo finito $\mathbb{F}_p$ y una ecuación de Weierstrass definida como en la Ecuación 3.2:*

I) *si E es singular, calcula los puntos singulares y proporciona una parametrización de todos los puntos en la curva elíptica $E = 0$;*

II) *si E es ordinaria y el ord$(\mathbb{F}_p) > 5$, calcula una proyección racional ρ de la línea afín sobre $\mathbb{F}_p$, hacia un triple afín V que esta dado explícitamente en términos de los coeficientes de E;*

III) *dado un punto racional en el triple V (Ecuación 3.3), calcular un punto racional sobre la curva elíptica E, de tal forma que al menos $(p-4)/8$ puntos racionales en E son obtenidos de la imagen de la proyección ρ, y al menos $(p-4)/3$ si la característica del campo es 2.*

A continuación se presenta la codificación usada para construir puntos sobre curvas elípticas ordinarias, a partir del teorema anterior.

Dada la curva elíptica E definida como en la Ecuación 3.2, sobre un campo finito $\mathbb{F}_p$, de característica impar y con $ord(\mathbb{F}_p) > 5$. Shallue y Woestijne siguiendo las ideas de Schinzel y Skałba [56], establecieron el siguiente resultado:

Lema 3.1. *[57, Lema 6] Dada la función $h(u,v) = u^2 + uv + v^2 + a(u+v) + b$, y la definición de:*

$$\begin{aligned} S &: y^2 \cdot h(u,v) = -g(u), \\ \psi &: (u,v,y) \mapsto (v, -a-u-v, u+y^2, g(u+y^2)\cdot h(u,v)\cdot y^{-1}), \end{aligned}$$

entonces ψ es una proyección racional de la superficie S hacia V, la cual es invertible en su imagen.

En particular, cualquier punto en $S(\mathbb{F}_p)$ en donde ψ es bien definida, es decir $y \neq 0$, proyecta a un punto en $V(\mathbb{F}_p)$, y produce un punto en $E(\mathbb{F}_p)$. Finalmente, para construir puntos en S, los autores notaron que cualquier sección plana de S de la forma $u = u_0$, es birracional a una cónica que es no degenerada, siempre y cuando:

$$g(u_0) \neq 0 \qquad y \qquad 3u_0^2 + 2au_0 + 4b - a^2 \neq 0.$$

Si se fija el valor de u_0, la cónica correspondiente admite una parametrización racional, la cual produce una proyección racional $\phi : \mathbb{A}^1 \to S$. La función de codificación $f : \mathbb{F}_p \to E(\mathbb{F}_p)$ se obtiene proyectando un valor $t \in \mathbb{F}_p$ a uno de los puntos sobre la curva $E(\mathbb{F}_p)$ con abscisa x_i, en donde $\psi(\phi(t)) = (x_1, x_2, x_3, y)$ y con $i \in \{1,2,3\}$ como el índice más pequeño tal que $g(x_i)$ es un residuo cuadrático.

3.4.1.1. Codificación de Ulas

La proyección racional en [57] fue posteriormente simplificada y generalizada a curvas hiperelípticas por Ulas [59], este algoritmo es conocido como Shallue-Woestijne-Ulas y presenta a continuación:

Lema 3.2. *[59] Sea $\mathbb{F}_p$ un campo finito y sea $g(x) = x^3 + ax + b$, donde $a, b \neq 0$. Se tiene que:*

$$X_1(t,u) = u \qquad X_2(t,u) = \frac{-a}{b}\left(1 + \frac{1}{t^4 g(u)^2 + t^2 g(u)}\right)$$

$$X_3(t,u) = t^2 g(u) X_2(t,u) \qquad U(t,u) = t^3 g(u)^2 g(X_2(t,u))$$

entonces

$$U(t,u)^2 = g(X_1(t,u)) \cdot g(X_2(t,u)) \cdot g(X_3(t,u)) \tag{3.4}$$

De la Ecuación 3.4 almenos uno de $g(X_1(t,u))$, $g(X_2(t,u))$ y $g(X_3(t,u))$ debería ser residuo cuadrático. Por lo tanto cualquiera de $X_1(t,u)$, $X_2(t,u)$ o $X_3(t,u)$ es la abscisa de un punto en la curva $y^2 = g(x)$.

3.4.2. Codificación de Icart

Un algoritmo para curvas elípticas ordinarias fue publicado por Icart [58]. Este algoritmo funciona para $p \equiv 2 \mod 3$ y se describe a continuación.

Considere una curva elíptica E sobre un campo finito $\mathbb{F}_p$, con p impar y congruente con 2 módulo 3, con ecuación $y^2 = x^3 + ax + b$. La función de Icart es definida como la proyección $f_{a,b} : \mathbb{F}_p \to E(\mathbb{F}_p)$ tal que $f_{a,b}(u) = (x, y)$ donde:

$$x = \left(v^2 - b - \frac{u^6}{27}\right)^{1/3} + \frac{u^2}{3}, \qquad y = ux + v, \qquad v = \frac{3a - u^4}{6u},$$

para $u \neq 0$ y $f(0) = \mathcal{O}$, el elemento identidad de la curva elíptica. Cuando $p \equiv 2 \mod 3$ se tiene que $x \mapsto x^3$ es una biyección en $\mathbb{F}_p$ por lo que las raíces cúbicas son únicamente definidas por $x^{1/3} = x^{(2q-1)/3}$.

3.4.3. Sumario de codificaciones

En la Tabla 3.1 se muestra un sumario de las codificaciones existentes para curvas elípticas ordinarias, en las que la característica del campo es distinta a 2 y 3, además se presentan algunas de sus propiedades.

Ecuación de la curva	Discriminante Δ	Codificación	Condición
$y^2 = x^3 + ax + b$	$-16(4a^3 + 27b^2)$	Icart [58]	$p \equiv 2 \mod 3$
		SW [57]	-
		SWU [59]	-
		SWU [60]	$p \equiv 3 \mod 4$

Tabla 3.1: Sumario de codificaciones a curvas elípticas ordinarias de característica distinta a 2 y 3.

3.5. Picadillo determinista hacia el grupo $\mathbb{G}_1$ en curvas BN

Investigaciones realizadas en la construcción de funciones de codificación, más o menos prácticas, hacia curvas elípticas, han mostrado que los resultados propuestos excluyen el caso en el que la curva $E : y^2 = x^3 + b$ es definida sobre un campo $\mathbb{F}_q$, con $q \equiv 1 \mod 3$. Un ejemplo importante de este caso son la familia de curvas

elípticas BN, las cuales son la familia de curvas preferidas para la implementación de emparejamientos bilineales asimétricos, hoy en día.

Esta situación puede verse claramente en las codificaciones derivadas de la planteada por Icart, tales como las propuestas realizadas por Farashahi [61], Kammerer *et al.* [62] y Couveignes *et al.* [63]. Dado que Icart hizo una extensión de la codificación propuesta por Boneh y Franklin [16] al caso de curvas ordinarias y esta codificación se basa en la posibilidad de calcular raíces cúbicas en el campo base $\mathbb{F}_p$, se requiere que la característica del campo p satisfaga $p \equiv 2 \mod 3$, con la finalidad de calcular la raíz de manera eficiente. De acuerdo con Mehdi Tibouchi [52], la única códificación que se puede utilizar es la codificación propuesta por Shallue y Woestijne [57] ya que es la única permite que el invariante-j sea cero, lo que es una característica de las curvas BN.

En el año 2012, Pierre-Alain Fouque y Mehdi Tibouchi [64] realizaron la especialización de la construcción propuesta por Shallue y Woestijne [57] a curvas BN alcanzando 9/16 de todos los puntos en la curva elíptica. Su propuesta es válida para curvas de la forma $y^2 = x^3 + b$, con $b \neq -1$, sobre un campo finito $\mathbb{F}_q$ en donde $q \equiv 7 \mod 12$, aunque también es válida para el caso en que $q \equiv 1 \mod 12$. A continuación se muestra su construcción:

Considerando la curva elíptica $E : y^2 = g(x) = x^3 + b$ sobre el campo finito $\mathbb{F}_q$, con $q \geq 5$ y suponiendo que $q \equiv 7 \mod 12$ y $g(1) = 1 + b$ es un cuadrado diferente de cero en $\mathbb{F}_q$. La ecuación de la superficie S definida en el Lema 3.1 queda como:

$$S : y^2 \cdot (u^2 + uv + v^2) = -u^3 - b.$$

Dada la sección definida por el plano de ecuación $u = u_0 = 1$, se produce la curva con ecuación: $y^2 \cdot (v^2 + v + 1) = -1 - b$ y completando el cuadrado de $v^2 + v + 1$ se obtiene:

$$y^2 \cdot \left(\frac{3}{4} + \left(v + \frac{1}{2} \right)^2 \right) = -1 - b,$$

con el cambio de variables $z = v + 1/2$ y $w = 1/y$ se produce la cónica no degenerada, debido a que $g(1) = 1 + b \neq 0$, con ecuación:

$$z^2 + (1 + b)w^2 = -\frac{3}{4}.$$

Es posible dar una parametrización de la cónica como se explica a continuación: dado que $q \equiv 1 \mod 3$, -3 es un residuo cuadrático en $\mathbb{F}_q$, de esta manera las coordenadas $(z_0, w_0) = (\sqrt{-3}/2, 0)$ pertenecen a un $\mathbb{F}_q$-punto racional en la cónica. Entonces los otros puntos pueden ser parametrizados poniendo $z = z_0 + tw$, con lo que se obtiene:

$$\sqrt{-3} \cdot t + t^2 \cdot w + (1 + b) \cdot w = 0,$$

entonces, despejando w de la ecuación anterior se obtiene:

$$y = \frac{1}{w} = -\frac{1 + b + t^2}{\sqrt{-3} \cdot t}$$

y

$$v = z_0 + tw + \frac{1}{2} = \frac{-1+\sqrt{-3}}{2} + \frac{\sqrt{-3}\cdot t^2}{1+b+t^2},$$

estas ecuaciones son bien definidas si $t \neq 0$ y $t^2 \neq -1-b$. La última condición siempre se verifica ya que $\chi_q(-1-b) = -\chi_q(1+b) = -1$.

De acuerdo con el Lema 3.1 al menos uno de los siguientes tres valores es la abscisa de un punto en $E(\mathbb{F}_q)$.

$$x_1 = v = \frac{-1+\sqrt{-3}}{2} - \frac{\sqrt{-3}\cdot t^2}{1+b+t^2}, \tag{3.5}$$

$$x_2 = -1-v = \frac{-1-\sqrt{-3}}{2} + \frac{\sqrt{-3}\cdot t^2}{1+b+t^2}, \tag{3.6}$$

$$x_3 = 1+y^2 = 1 - \frac{(1+b+t^2)^2}{3t^2}. \tag{3.7}$$

Además, estos valores dependen sólo de t^2 y por tanto son invariantes al cambio de signo de t. De esta manera, se proyectan los valores t y $-t$ hacia puntos opuestos en $E(\mathbb{F}_q)$ con alguna de las coordenadas previas.

Por lo tanto, la construcción de puntos de Shallue y Woestijne en la curva E se define como sigue:

Definición 3.2. *Para todo $t \in \mathbb{F}_q^*$, sean x_1, x_2, $x_3 \in \mathbb{F}_q$, como en las ecuaciones 3.5, 3.6 y 3.7. La codificación Shallue y Woestijne a la curva BN, denotada por E, es la proyección:*

$$\begin{aligned} f : \mathbb{F}_q^* &\to E(\mathbb{F}_q) \\ t &\mapsto \left(x_i, \chi_q(t)\cdot\sqrt{g(x_i)}\right) \end{aligned}$$

en donde para cada t, el índice $i \in \{1,2,3\}$ es tal que $g(x_i)$ es un cuadrado en $\mathbb{F}_q$.

La codificación puede ser extendida a todo $\mathbb{F}_q$ enviando 0 a un punto arbitrario en $E(\mathbb{F}_q)$. Dado que x_1 es bien definido e igual a $(-1+\sqrt{-3})/2$ para $t = 0$ y $g(x_1) = 1+b$ es un cuadrado, una elección natural sería:

$$f(0) = \left(\frac{-1+\sqrt{-3}}{2}, \sqrt{1+b}\right)$$

.

La proyección $f : \mathbb{F}_q^* \to E(\mathbb{F}_q)$, combinada con un oráculo aleatorio $\mathfrak{h} : \{0,1\}^* \to \mathbb{F}_q^*$, produce como resultado la función picadillo determinista definida como:

$$\begin{aligned} H_1 &: \{0,1\}^* \to \mathbb{G}_1 \\ H_1(t) &= f(\mathfrak{h}(t)) \qquad \text{con} \quad t \in \{0,1\}^*. \end{aligned}$$

3.6. Propuesta de picadillo determinista hacia el grupo $\mathbb{G}_2$ en curvas BN

Los protocolos basados en emparejamientos son considerados prácticos si es posible calcular de manera eficiente y segura todas las operaciones involucradas en ellos. Sin embargo, la mayoría de las implementaciones encontradas en el estado del arte sólo toman en cuenta el cálculo del emparejamiento bilineal, sin poner atención a otras primitivas importantes, como la generación de puntos aleatorios en una curva elíptica del problema conocido como picadillo al grupo $\mathbb{G}_2$. Por esta razón, uno de los objetivos de este trabajo de tesis es la construcción de la función picadillo a $\mathbb{G}_2$ de manera determinista. Este problema se describe a continuación:

De acuerdo a la definición del grupo $\mathbb{G}_2$ presentada en la Sección 2.3.1en la página 16, un elemento $Q \in \mathbb{G}_2$ es un punto en el grupo $E(\mathbb{F}_{q^k})[r]$, tal que $\pi(Q) = pQ$, en donde π representa el endomorfismo de Frobenius, k el grado de encajamiento, p y r números primos y q una potencia impar de p. En este contexto, el problema del picadillo al grupo $\mathbb{G}_2$ consiste en encontrar un punto $Q \in \mathbb{G}_2$ a partir de un punto aleatorio en $E(\mathbb{F}_{q^k})$. Utilizando la definición de curvas enlazadas, si el grupo $E'(\mathbb{F}_{q^{k/d}})[r]$ definido por la curva enlazada $E'/\mathbb{F}_{q^{k/d}}$ de grado d es isomórfico al grupo $E(\mathbb{F}_q)[r]$, el problema se reduce a encontrar un punto $Q' \in E'(\mathbb{F}_{q^{k/d}})[r]$.

A continuación se describe la forma de obtener un punto $Q' \in E'(\mathbb{F}_{q^{k/d}})$ y posteriormente se detalla el procedimiento para obtener Q' de orden r.

3.6.1. Construcción determinista de puntos en $E'(\mathbb{F}_{q^2})$

Con base en la definición de la familia de curvas BN de la Sección 2.3.3.1 en la página 18, se sabe que las curvas BN tienen grado de encajamiento $k = 12$ y la curva enlazada correspondiente tiene grado $d = 6$, lo que permite definir el grupo $\mathbb{G}_2$ como:

$$\mathbb{G}_2 = E'(\mathbb{F}_{q^{k/d}})[r] = E'(\mathbb{F}_{q^2})[r].$$

Para este grupo, las construcciones presentadas en la Sección 3.4 en la página 33 podrían brindar una opción viable para solucionar el problema, sin embargo, requieren que la característica del campo o la curva elíptica tengan ciertas propiedades, las cuales no coinciden con las correspondientes a la familia de curvas BN en el grupo $\mathbb{G}_2$. Por otro lado, en el trabajo realizado por Pierre-Alain Fouque y Mehdi Tibouchi en [64], descrito en la sección anterior, muestran que sus resultados aplican a curvas elípticas de la forma $y^2 = x^3 + b$ con $b \neq -1$ sobre un campo finito $\mathbb{F}_q$ con $q \equiv 7 \mod 12$ o $q \equiv 1 \mod 12$.

Considerando que la característica del campo sobre el que se construyen las curvas elípticas BN tiene la propiedad que $q \equiv 7 \mod 12$, al hablar del grupo $\mathbb{G}_2$ se puede observar que:

$$q \equiv 7 \mod 12 \Longrightarrow q = 12k + 7$$

y dada la extensión cuadrática de $\mathbb{F}_q$, se obtiene que:

$$q^2 = (12k + 7)^2,$$

si reducimos la ecuación módulo 12, se puede ver que:

$$\begin{aligned} q^2 &= (12k+7)^2 \mod 12 \\ &= (12^2k^2 + 2\cdot(12\cdot 7k) + 49) \mod 12 \\ &= 1 \mod 12 \\ \implies & q^2 \equiv 1 \mod 12. \end{aligned}$$

Con lo que se puede asegurar que para realizar el cálculo de la función picadillo al grupo $\mathbb{G}_2$, se puede utilizar el procedimiento propuesto por Pierre-Alain Fouque y Mehdi Tibouchi. Sin embargo, dado que $q^2 \equiv 1 \mod 12$ es necesario contar con un método para el cálculo determinista de la raíz cuadrada. A continuación se presentan los detalles de la implementación:

La base de la implementación es la aritmética en los campos $\mathbb{F}_q$ y $\mathbb{F}_{q^2}$, la cual se detalla en el Apéndice B en la página 113. El algoritmo principal es el derivado del trabajo Pierre-Alain Fouque y Mehdi Tibouchi modificado para el cálculo de $\mathbb{G}_2$, el cual se presenta a continuación:

Algoritmo 3.2 Codificación de Shallue y Woestijne a curvas BN en $\mathbb{G}_2$

Entrada: $t \in \mathbb{F}_q^*$, el parámetro $B = b_0 + b_1 u \in \mathbb{F}_{q^2}$ de $E' : y^2 = x^3 + B$
Salida: El punto $Q = (x, y) \in E'(\mathbb{F}_{q^2})$

1: $r \leftarrow \sqrt{-3}$ (en $\mathbb{F}_q$, Algoritmo 3.3)
2: $a_0 \leftarrow 1 + b_0 + t^2$ (en $\mathbb{F}_q$)
3: $a_1 \leftarrow b_1 u$
4: $A \leftarrow 1/A$ (en $\mathbb{F}_{q^2}$ con $A = a_0 + a_1 u$, Algoritmo B.12)
5: $w_0 \leftarrow r \cdot t$ (en $\mathbb{F}_q$)
6: $W \leftarrow A \cdot w_0$ (en $\mathbb{F}_{q^2}$, Algoritmo B.10)
7: $A \leftarrow W \cdot t$ (en $\mathbb{F}_{q^2}$, Algoritmo B.10)
8: $x1_0 \leftarrow (-1 + r)/2$ (en $\mathbb{F}_q$)
9: $x1_1 \leftarrow 0$
10: $X1 \leftarrow X1 - A$ (en $\mathbb{F}_{q^2}$ con $X1 = x1_0 + x1_1 u$, Algoritmo B.12)
11: $X2 \leftarrow -1 - X1$ (en $\mathbb{F}_{q^2}$)
12: $X3 \leftarrow 1/W^2$ (en $\mathbb{F}_{q^2}$)
13: $X3 \leftarrow 1 + X3$
14: $\alpha \leftarrow \chi_{q^2}(x_1^3 + B)$
15: $\beta \leftarrow \chi_{q^2}(x_2^3 + B)$
16: $i \leftarrow [(\alpha - 1) \cdot \beta \mod 3] + 1$
17: $x \leftarrow Xi$
18: $y \leftarrow \chi_q(t) \cdot \sqrt{Xi^3 + B}$
19: **return** (x, y)

En la línea 1 del Algoritmo 3.2 dado que $q \equiv 7 \mod 12 \implies q \equiv 3 \mod 4$, se puede calcular la raíz cuadrada x de un elemento $a \in \mathbb{F}_q$ como $x = a^{(p+1)/4}$, como se muestra en el Algoritmo 3.3.

Algoritmo 3.3 Algoritmo de Shanks

Entrada: $a \in \mathbb{F}_q^*$
Salida: falso o $x \in \mathbb{F}_q^*$ tal que $x^2 = a$

1: $a_1 \leftarrow a^{\frac{q-3}{4}}$
2: $a_0 \leftarrow a_1^2 a$
3: **if** $a_0 = -1$ **then**
4: **return** falso
5: **end if**
6: $x \leftarrow a_1 a$
7: **return** x

En las líneas 14 y 15 del Algoritmo 3.2, para realizar de forma eficiente el cálculo del caracter cuadrático $\chi_{q^2}(\cdot)$, correspondiente al campo $\mathbb{F}_{q^2}$, se tomó la mejora realizada por Gora Adj y Francisco Rodríguez Henríquez [65] al trabajo de Bach [66], en lugar de la exponenciación de $a^{\frac{q^2-1}{2}}$ que es la manera más costosa computacionalmente hablando.

En el trabajo de Bach se mostró que el símbolo de Legendre puede ser usado para calcular el caracter cuadrático de un elemento $a \in \mathbb{F}_q^*$, con $q = p^n$, p un número primo impar y $n > 1$. Invocando recursivamente la ley de reciprocidad. Posteriormente en el trabajo de Gora Adj y Francisco Rodríguez Henríquez se presentó una forma de calcular el caracter cuadrático descendiendo el cálculo al campo base $\mathbb{F}_p$ más algunas evaluaciones del operador Frobenius, utilizando la siguiente factorización:

$$\frac{q-1}{2} = \frac{p-1}{2}\sum_{i=0}^{n-1} p^i.$$

Por lo tanto el cálculo de $\chi_{q^2}(a)$, con $a \in \mathbb{F}_{q^2}^*$ queda de la siguiente manera:

$$\begin{aligned} a^{\frac{q^2-1}{2}} &= (a \cdot a^q)^{\frac{q-1}{2}} \\ &= (a \cdot \bar{a})^{\frac{q-1}{2}} \end{aligned}$$

en donde $\bar{a}$ es el conjugado de a. Este cálculo tiene un costo de una multiplicación y una exponenciación en $\mathbb{F}_q$.

En la línea 18 del Algoritmo 3.2 es necesario el cálculo de χ_q, que representa el caracter cuadrático en el campo $\mathbb{F}_q$, este procedimiento se muestra en el Algoritmo 3.4. Dado que el tiempo de ejecución del algoritmo depende del parámetro de entrada, se optó por realizar la operación de la siguiente manera: primero se elige un valor $s \in \mathbb{F}_q^*$ de forma aleatoria y se calcula $\chi_q(s^2 \cdot t)$, con el fin de realizar el cómputo en un tiempo constante, independientemente de la entrada del algoritmo, para evitar fugas de información.

Además es necesario el cálculo de la raíz cuadrada en $\mathbb{F}_{q^2}$ (línea 18 del Algoritmo 3.2) y dado que $q^2 \equiv 1 \bmod 12 \Rightarrow q^2 \equiv 1 \bmod 4$, podría pensarse que no se puede

Algoritmo 3.4 Cálculo de χ_q en el campo $\mathbb{F}_q$

Entrada: $a \in \mathbb{F}_q^*$, la característica del campo q y el conjunto $tab = \{0, 1, 0, -1, 0, -1, 0, 1\}$
Salida: $k = 1$ si a es un cuadrado en $\mathbb{F}_q$, $k = -1$ en caso contrario.

```
1: k ← 1
2: while (a > 0) do
3:    v ← 0
4:    while a es par do
5:       a ← a/2
6:       v ← v + 1
7:    end while
8:    if (v es impar) then
9:       k ← tab[q&7] * k
10:   end if
11:   if (a&q&2) then
12:      k ← −k
13:   end if
14:   t ← a
15:   a ← q mod t
16:   q ← t
17: end while
18: return k
```

realizar de forma determinista ya que los algoritmos existentes para el caso en que $q \equiv 1 \bmod 4$ dependen del parámetro de entrada. Sin embargo, es posible utilizar el método complejo descrito por Scott en [67], que es el método más eficiente para el cálculo de raíces cuadradas en extensiones de campo pares, debido a que desciende el cálculo al campo base, en este caso al campo $\mathbb{F}_q$ cuya caracerística $q \equiv 3 \bmod 4$. Este método se presenta en el Algoritmo 3.5.

El Algoritmo 3.2 hace uso de la función τ para la elección del índice i (línea 16), la cual realiza la selección de $X1$, $X2$ o $X3$ según cual sea un cuadrado en $\mathbb{F}_{q^2}$:

$$\tau(1,1) = \tau(1,-1) = 1, \quad \tau(-1,1) = 2, \quad \tau(-1,-1) = 3$$

la funcion τ esta dada por:

$$\tau(\alpha, \beta) = [(\alpha - 1) \cdot \beta \mod 3] + 1$$

3.6.2. Obtención de puntos en $E'(\mathbb{F}_{q^2})$ de torsión r

Sea $E'(\mathbb{F}_{p^{k/d}})$ un grupo abeliano finito de orden $\#E'(\mathbb{F}_{p^{k/d}}) = c \cdot r$, donde c es un número compuesto conocido como el cofactor de la curva enlazada. A partir de los Teoremas A.1, A.2 y A.3, se sabe que

$$\{c\tilde{Q} \mid \tilde{Q} \in E'(\mathbb{F}_{p^{k/d}})\} = \{Q' \in E'(\mathbb{F}_{p^{k/d}}) \mid rQ' = \mathcal{O}\},$$

Algoritmo 3.5 Método complejo para el cálculo de raíces cuadradas sobre $\mathbb{F}_{q^2}$

Entrada: Binomio irreducible $f(u) = u^2 - \beta$ tal que $\mathbb{F}_{q^2} \cong \mathbb{F}_q[u]/(u^2 - \beta)$, $\beta \in \mathbb{F}_q$, con $q = p^n$, $a = a_0 + a_1 u \in \mathbb{F}_{q^2}^*$.
Salida: Si existe, $x = x_0 + x_1 u \in \mathbb{F}_{q^2}$ satisfaciendo $x^2 = a$, Falso otra cosa.

1: **if** $a_1 = 0$ **then**
2: **return** $\sqrt{a_0}$ (en $\mathbb{F}_q$)
3: **end if**
4: $\alpha \leftarrow a_0^2 - \beta \cdot a_1^2$
5: $\gamma \leftarrow \chi_q(\alpha)$
6: **if** $\gamma = -1$ **then**
7: **return** falso
8: **end if**
9: $\alpha \leftarrow \sqrt{\alpha}$ (en $\mathbb{F}_q$)
10: $\delta \leftarrow \frac{a_0+\alpha}{2}$
11: $\gamma \leftarrow \chi_q(\delta)$
12: **if** $\gamma = -1$ **then**
13: $\delta \leftarrow \frac{a_0-\alpha}{2}$
14: **end if**
15: $x_0 \leftarrow \sqrt{\delta}$ (en $\mathbb{F}_q$)
16: $x_1 \leftarrow \frac{a_1}{2x_0}$
17: $x \leftarrow x_0 + x_1 u$
18: **return** x

es decir, al tomar un punto aleatorio $\tilde{Q}$ de la curva enlazada y multiplicarlo por el cofactor c, se obtiene un punto Q' de torsión r en $E'(\mathbb{F}_{p^{k/d}})[r]$ [40]. La solución es sencilla, sin embargo, c es un número considerablemente grande, lo que provoca que esta solución sea costosa. A continuación se muestra la solución más eficiente hasta el momento para realizar el cómputo del problema descrito, desarrollada por Laura Fuentes Castañeda *et al.* [40].

Con base en la observación de que dado un múltiplo c' del cofactor c, tal que $c' \not\equiv 0 \mod r$, se cumple que $c'\tilde{Q} \in E(\mathbb{F}_{p^{k/d}})[r]$, proponen el siguiente teorema.

Teorema 3.2. *Suponiendo que $E'(\mathbb{F}_{p^{k/d}})$ es un grupo cíclico y $p \equiv 1 \mod d$. Existe un polinomio $h(w) = h_0 + h_1 w + \cdots + h_{\varphi(k)-1} w^{\varphi(k)-1} \in \mathbb{Z}[w]$ tal que $h(\psi(\tilde{Q}))$ es un múltiplo de $c\tilde{Q}$, para todo $\tilde{Q} \in E(\mathbb{F}_{p^{k/d}})$, y $|h_i|^{\varphi(k)} \leq \#E'(\mathbb{F}_{p^{k/d}})/r$, donde $i \in \mathbb{Z}$.*

La prueba del teorema está basada en dos lemas principales:

Lema 3.3. *Sea d el grado de la curva enlazada E', si $p \equiv 1 \mod d$, entonces $\psi(\tilde{Q}) \in E'(\mathbb{F}_{p^{k/d}})$, para todo $\tilde{Q} \in E'(\mathbb{F}_{p^{k/d}})$.*

Lema 3.4. *Sea $t^2 - 4p = Df^2$ y $\tilde{t}^2 - 4q = D\tilde{f}^2$, para algún valor de f y $\tilde{f}$, donde $q = p^{k/d}$ y D es el discriminante CM; sea además $\tilde{n} = \#E'(\mathbb{F}_{p^{k/d}})$, si se cumplen las condiciones:*

- $p \equiv 1 \mod d$,
- $mcd(\tilde{f}, \tilde{n}) = 1$,
- $E'(\mathbb{F}_{p^{k/d}})$ *es un grupo cíclico,*

entonces $\psi(\tilde{Q}) = a\tilde{Q}$ *para todo* $\tilde{Q} \in E'(\mathbb{F}_{p^{k/d}})$, *donde:*

$$a = (t \pm f(\tilde{t} - 2)/\tilde{f})/2.$$

Una vez calculado el valor de a, tal que $a\tilde{Q} = \psi(\tilde{Q})$, es necesario encontrar el polinomio $h \in \mathbb{Z}[w]$ con los menores coeficientes, tal que $h(a) = 0 \mod c$. Esto se realiza construyendo una matriz M cuyas filas representan a los polinomios $h_i(w) = w^i - a^i$, tal que $h_i(a) \equiv 0 \mod c$. De esta manera, cualquier combinación lineal de las filas de M corresponde con un polinomio $h'(w)$ que satisface dicha condición.

$$M = \begin{array}{c} \begin{matrix} a^0 & a^1 & a^2 & \cdots & a^{\varphi(k)-1} \end{matrix} \\ \left(\begin{array}{ccccc} c & 0 & 0 & \cdots & 0 \\ -a & 1 & 0 & \cdots & 0 \\ -a^2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \\ -a^{\varphi(k)-1} & 0 & 0 & \cdots & 1 \end{array} \right) \end{array} \dashrightarrow \begin{array}{ccccc} c & \equiv & 0 & \mod c \\ -a + a & \equiv & 0 & \mod c \\ -a^2 + a^2 & \equiv & 0 & \mod c \\ \vdots & \vdots & & \vdots \\ -a^{\varphi(k)-1} + a^{\varphi(k)-1} & \equiv & 0 & \mod c \end{array}$$

Las filas de la matriz M son vistas como vectores, los cuales forman la base de una rejilla L. Al aplicar el algoritmo $LLL(M)$, se obtiene un vector v formado por la combinación lineal de la base de L, el cual corresponde al polinomio h con coeficientes menores a $|c|^{1/\varphi(k)}$.

En la familia de curvas BN, el orden del grupo $\tilde{n} = \#E'(\mathbb{F}_{q^2})$ y la traza de E' sobre $\mathbb{F}_{q^2}$, son parametrizadas por:

$$\begin{aligned} \tilde{n} &= (36z^4 + 36z^3 + 18z^2 + 6z + 1)(36z^4 + 36z^3 + 30z^2 + 6z + 1) \\ \tilde{t} &= 36z^4 + 1 \end{aligned}$$

en donde $\tilde{n}(x) = r(x)c(x)$. Utilizando el Lema 3.4, se obtiene:

$$a(x) = -\frac{1}{5}(3456z^7 + 6696z^6 + 7488z^5 + 4932z^4 + 2112z^3 + 588z^2 + 106z + 6).$$

Siguiendo el método descrito anteriormente, se construye la rejilla reduciendo $-a(z)^i$ modulo $c(z)$:

$$\left[\begin{array}{c|ccc} c(z) & 0 & 0 & 0 \\ -a(z) & 1 & 0 & 0 \\ -a(z)^2 & 0 & 1 & 0 \\ -a(z)^3 & 0 & 0 & 1 \end{array}\right] \rightarrow \left[\begin{array}{c|ccc} 36z^4 + 36z^3 + 30z^2 + 6z + 1 & 0 & 0 & 0 \\ 48/5z^3 + 6z^2 + 4z - 2/5 & 1 & 0 & 0 \\ 36/5z^3 + 6z^2 + 6z + 1/5 & 0 & 1 & 0 \\ 12z^3 + 12z^2 + 8z + 1 & 0 & 0 & 1 \end{array}\right].$$

Para la rejilla construida a partir de esta matriz, se puede ver que $h(w) = z + 3zw + zw^2 + w^3$. Al hacer la reducción módulo $\tilde{n}$, se encuentra que

$$h(a) = -(18z^3 + 12z^2 + 3z + 1)c(z)$$

y dado que $\text{mcd}(18z^3 + 12z^2 + 3z + 1, r(x)) = 1$:

$$\tilde{Q} \mapsto z\tilde{Q} + \psi(3z\tilde{Q}) + \psi^2(z\tilde{Q}) + \psi^3(\tilde{Q}).$$

El cálculo de $\tilde{Q} \mapsto Q'$, donde $Q' \in E'(\mathbb{F}_{q^2})[r]$, tiene un costo de un doblado, cuatro sumas de puntos, una multiplicación escalar por z y tres aplicaciones del operador ψ.

Con base en la codificación descrita en la sección anterior (Algoritmo 3.2) es posible obtener un punto en el grupo definido por la curva enlazada $E'/\mathbb{F}_{q^{k/d}}$, sin embargo, este punto no tiene el orden deseado, por lo que se requiere de la multiplicación por el cofactor descrita en párrafos anteriores. Una vez que se han expuesto lo procedimientos con los que se obtiene un punto $Q \in E'(\mathbb{F}_{q^{k/d}})[r]$, es posible construir la función picadillo determinista hacia el grupo $\mathbb{G}_2$, denotada por H_2, con ayuda de una función picadillo genérica $\mathfrak{h} : \{0,1\}^* \to \mathbb{F}_q$, de la misma forma que en el caso de H_1.

3.7. Resultados

Se realizaron las implementaciones en C de las funciones picadillo deterministas, tanto al grupo $\mathbb{G}_1$ como al grupo $\mathbb{G}_2$, en una máquina de 64 bits con un procesador Intel Core i7-2630QM CPU a 2.0 GHz. El parámetro x para la evaluación de las funciones que generan los parámetros p, r de las curvas BN fue: $x = -(2^{62} + 2^{55} + 1)$.

En la Tabla 3.2 se presentan los tiempos obtenidos en ciclos de reloj, para las operaciones involucradas en el cálculo de las funciones picadillo hacia los grupos $\mathbb{G}_1$ y $\mathbb{G}_2$. En la que se puede observar para el cálculo de un punto en el grupo $E'(\mathbb{F}_{p^2})$, que la propuesta realizada en esta tesis logra un mejor desempeño ya que en comparación con la no determinista, nuestra propuesta realiza el cálculo 13.3 % más rápido.

Operación	**Algoritmo**	**Ciclos de reloj** ($\times 10^3$)
Raíz cuadrada $\mathbb{F}_q$	Algoritmo 3.3	27.61
Raíz cuadrada $\mathbb{F}_{q^2}$	Algoritmo 3.5	61.10
Cálculo de $E'(\mathbb{F}_{p^2})$ no determinista	Algoritmo 3.1	331.85
Cálculo de $E'(\mathbb{F}_{p^2})$ determinista	Algoritmo 3.2	287.66
Multiplicación por el cofactor		165.5

Tabla 3.2: Resumen de tiempos de los algoritmos involucrados en el cálculo de las funciones picadillo.

Funcion	**Ciclos de reloj** ($\times 10^3$)
Picadillo a $\mathbb{G}_1$ no determinista	98.20
Picadillo a $\mathbb{G}_1$ determinista	83.07
Picadillo a $\mathbb{G}_2$ no determinista	445.65
Picadillo a $\mathbb{G}_2$ determinista	431.65

Tabla 3.3: Resumen de tiempos de las funciones picadillo hacia los grupos $\mathbb{G}_1$ y $\mathbb{G}_2$.

En la Tabla 3.3 se presentan los resultados obtenidos de las implementaciones de las funciones picadillo. La función picadillo no determinista hacia el grupo $\mathbb{G}_1$, fue implementada de acuerdo al Algoritmo 3.1, mientras que la determinista fue implementada conforme a lo descrito en la Sección 3.5. Por otro lado, la función picadillo no determinista hacia el grupo $\mathbb{G}_2$ fue también implementada con el algoritmo Algoritmo 3.1 y posteriormente se realizo la multiplicación por el cofactor, mientras que la función determinista fue implementada de acuerdo a lo propuesto en la sección anterior. En la tabla se puede observar que aún después de realizar la multiplicación por el cofactor, nuestra propuesta de función picadillo hacia el grupo $\mathbb{G}_2$ muestra un mejor desempeño, además de que ofrece protección en contra de ataques de análisis de tiempo. Lo que la convierte en una opción viable para su utilización en protocolos basados en la identidad que utilizan emparejamientos bilineales.

4

Autenticación de dos factores

"Ni la contradicción es indicio de falsedad, ni la falta de contradicción es indicio de verdad."
∼Blaise Pascal (1623-1662)∼

El objetivo de este capítulo es presentar algunos de los conceptos relacionados con la autenticación de dos factores. En la Sección 4.1 se define el concepto de autenticación, en la Sección 4.2 se presentan los antecedentes de los protocolos de este tipo, en la Sección 4.3 se define la autenticación multifactor, en la Sección 4.4 se describe el proceso de intercambio de llaves autenticadas, en la Sección 4.5 se muestran los ataques existentes a protocolos de autenticación y en la Sección 4.6 se presentan las características deseables en un protocolo de autenticación de dos factores.

Además, se presenta el análisis de dos protocolos de autenticación, en la Sección 4.7 el protocolo "Software-only two-factor authentication" y en la Sección 4.8 el protocolo "M-PIN technology".

4.1. Autenticación

Como se mencionó en el Capítulo 1, la autenticación es el proceso utilizado para asegurar que la identidad de los participantes en un protocolo de comunicación es verdadera, mediante la evaluación de tres aspectos: verificar algo que el participante tiene, ponerlo a prueba sobre algo que sabe y verificar algo que es. Este proceso consta de dos partes:

- **Identificación:** Presentación de un identificador, el cual debe ser asignado cuidadosamente, ya que las identidades autenticadas son la base de otros servicios de seguridad, tal como el control de acceso.

- **Verificación:** Presentar o generar información que corrobore la unión entre el participante y el identificador.

De acuerdo a la manera en que éste proceso es realizado, la autenticación se puede clasificar como: "simple", si el proceso utiliza una contraseña como información necesaria para verificar la identidad afirmada por una entidad; "fuerte", si el proceso utiliza la criptografía para verificar la identidad afirmada por la entidad, por ejemplo, la autenticación mediante credenciales derivadas de manera criptográfica.

En el contexto de un protocolo de comunicación, además de poder autenticar a los participantes, es posible autenticar:

- **El mensaje:** Consiste en verificar que una de las partes involucradas es la fuente original del mensaje. Este tipo de autenticación asegura su integridad.
- **La llave:** Permite asegurar a alguna de las partes que ninguna entidad, no confiable, pueda tener acceso a la llave privada correspondiente.
- **La transacción:** Provee la autenticación del mensaje y además garantiza la existencia única y temporal de los datos.

Como se puede observar, aunque los servicios de autenticación e integridad se definen por separado, están estrechamente relacionados. El servicio de autenticación depende, por definición, del servicio de integridad, por ejemplo, el servicio de autenticación del mensaje comprueba que la identidad de la fuente original del mensaje recibido es de quien se afirmó; no puede haber dicha comprobación si el mensaje ha sido alterado.

4.2. Antecedentes

Es básicamente un problema resuelto para un servidor autenticarse ante un cliente usando métodos de criptografía de llave pública. La infraestructura de llave pública (PKI) soporta el protocolo SSL, el cual a su vez permite esta funcionalidad. El único inconveniente (o punto de quiebre) en PKI, es la autoridad certificadora y por lo tanto es el objetivo de los atacantes. Sin embargo, se considera que esta entidad se encuentra comúnmente fuera de línea y bien protegida.

Por otro lado, el proceso que sigue un cliente para autenticarse ante un servidor es mucho más problemático. En este sentido, existen numerosos protocolos criptográficos que confían en contraseñas seleccionadas por los usuarios para aplicar la autenticación. En estos protocolos se encuentra frecuentemente la situación de que a los usuarios les resulta incómodo recordar contraseñas largas y típicamente las eligen cortas y fáciles de recordar, lo que provoca que el espacio muestral sea suficientemente pequeño para ser enumerado por un atacante, haciendo los protocolos vulnerables a los ataques de diccionario. Dado que la verificación de contraseñas en un canal inseguro ha sido un problema particularmente difícil, ante la amenaza siempre presente de un ataque de diccionario, y debido a que estos problemas han existido por demasiado tiempo, muchos han supuesto que la autenticación remota segura utilizando una contraseña corta es imposible. Aunque parece paradójico que las contraseñas cortas sean importantes

para la autenticación, cuando claramente, las criptográficamente grandes serían mejor elección, si tan sólo los usuarios pudieran recordarlas.

El problema de la autenticación basada en contraseñas fue estudiado primero por Gong *et al.* [68] quienes usaron cifrado de llave pública para protegerse de los ataques de diccionario fuera de línea. Otro trabajo relacionado, es el perteneciente a Bellovin y Merritt [69], quienes introdujeron el EKE (*Encrypted Key Exchange*) y el cual se convirtió en la base de muchos de los subsecuentes trabajos en esta área. Esos trabajos incluyen SPEKE (*Simple Password Exponential Key Exchange*) y SRP (*Secure Remote Password*).

En la literatura, se han propuesto varias estrategias para el uso "adecuado" de contraseñas [70]. Algunas de las cuales son muy difíciles de usar y otras podrían no responder a las preocupaciones de seguridad del ambiente en donde se quieren aplicar. La autenticación de dos factores, usando dispositivos tales como tokens y *smart cards* ha sido propuesta para resolver el problema de las contraseñas y ha demostrado ser difícil de romper. Sin embargo, este tipo de autenticación también tiene desventajas, que incluyen el costo de la compra, emisión y gestión de los tokens o *smart cards* y desde el punto de vista del cliente, el uso de más de un sistema de autenticación de dos factores requiere manejar varios tokens o *smart cards*, los cuales pueden perder o les pueden ser robados.

En el 2003 Kim *et al.* [71] propusieron un esquema, que prometía la "autenticación usando contraseñas basado en la identidad mediante *smart cards* y huellas digitales". Sin embargo, Scott [72] mostró cómo podría ser vulnerado por un atacante quien pasivamente espíara una sola transacción. En un trabajo más reciente Martínez-Pelaez y Rico-Novella [73] mostraron que el esquema propuesto por Sood, Sarje y Singh, descrito en el artículo, "Una mejora del sistema de autenticación de Liao *et al.* con *smart cards*" [74] es vulnerable a ataques como usuario malicioso, hombre de en medio, *smart card* robada y suposición de identidad. Este esquema fue esencialmente una propuesta de autenticación de dos factores.

Muchos otros esquemas han caído dentro del ciclo de romperlo y arreglarlo, el cual genera mucha literatura, pero a la vez menos confianza. Este problema fue reconocido por Hao y Clarke [75], quienes lo explican diciendo: "Los pasados treinta años de investigación en el área del intercambio de llaves autenticadas, ha probado que es increíblemente difícil conseguir incluso un esquema correcto de intercambio de llaves de un factor. Por lo que diseñar un protocolo de intercambio de llaves autenticadas de múltiples factores sólo puede ser más difícil". Los trabajos mencionados y la afirmación anterior muestran que este todavía es un campo joven, y por tanto, es necesaria más investigación.

En trabajos más recientes Michael Scott [17, 18] hizo una revisión de la literatura existente correspondiente a protocolos de autenticación de dos factores, Michael Scott presentó una lista de características deseables en estos protocolos para que puedan ser considerados seguros, además propuso un par de protocolos de autenticación de dos factores basados en emparejamientos bilineales, los cuales podrían ser utilizados en un ambiente sin *smart cards*. En las secciones siguientes nos enfocaremos en estos dos

trabajos, con el fin de aplicarlos en el proceso de autenticación utilizando dispositivos móviles.

4.3. Autenticación multifactor

Mientras que los trabajos descritos en la sección anterior corresponden a los factores: "algo que se sabe" o "algo que se tiene", existe un tercer factor para la autenticación: "algo que se es". Este factor corresponde al proceso de generar información, para la autenticación, mediante la medición de características físicas o de comportamiento, únicas de una persona.

Como se ha visto, la autenticación es sin duda uno de los objetivos más importantes de la criptografía moderna, en este contexto la autenticación multifactor consiste comúnmente en verificar dos o los tres factores antes mencionados. La forma en que esto sería más familiar, es la autenticación de dos factores que se lleva a cabo en el cajero automático, la tarjeta bancaria y el PIN de cuatro dígitos. A continuación se presentan ejemplos de cada uno de los tres factores que pueden ser verificados en la autenticación:

1. *Algo que se tiene*: este factor puede ser un token físico almacenando datos estáticos, tal vez los datos grabados en una cinta magnética, en un código QR o en una memoria USB.

   Puede ser también una *smart card*, pero cabe mencionar que una *smart card* tendrá una funcionalidad adicional, puede tener sus propios secretos protegidos, capacidad de cómputo y además no puede ser clonada. Sin embargo, una *smart card* es mucho más cara y su pérdida requiere de un reemplazo costoso.

   También, puede ser un dispositivo móvil, con datos estáticos almacenados en su memoria protegida contra manipulaciones. Esta opción es menos costosa, se puede aprovechar su poder de computo y además es fácil tratar con diferentes sistemas que implementen autenticación multifactor, utilizando un mismo dispositivo.

2. *Algo que se sabe*: este factor generalmente es una contraseña, la cual puede ser de dos tipos:

   a) De alta entropía: Es actualmente la de mayor demanda, por ejemplo la contraseña debe estar formada por ocho o más caracteres e involucrar mayúsculas, minúsculas y por lo menos un dígito.

   b) De baja entropía: Este se refiere a un PIN de cuatro dígitos.

   A partir de este momento se utilizará la palabra contraseña exclusivamente para las de alta entropía y la palabra PIN para las de baja entropía. Una manera alternativa de distinguirlas es observando que esta última puede ser encontrada fácilmente utilizando un ataque de diccionario fuera de línea, mientras que la primera con suerte no.

3. *Algo que se es*: este factor es capturado como una característica biométrica, quizá una huella dactilar o un escaneo del iris. Este factor comúnmente es inexacto y debe ser considerado un cierto rango de valores como aceptable. La biometría puede también ser de alta o baja entropía, la primera tiende a ser costosa, mientras que la segunda comúnmente es más accesible.

4.4. Intercambio de llaves autenticadas

Los protocolos de intercambio de llaves autenticadas, AKE por sus siglas en inglés, son protocolos fundamentales para los sistemas de comunicación segura. Su objetivo es derivar una llave de sesión de alta entropía para asegurar la comunicación subsecuente, mediante la utilización de un secreto común de baja entropía. Dependiendo de cómo sea definida la autenticación, los esquemas AKE se dividen en dos: intercambio de llaves autenticadas basadas en contraseñas, PAKE por sus siglas en inglés, e intercambio de llaves autenticadas basadas en PKI [76]. En el primer caso, la autenticación se basa en el conocimiento de una contraseña compartida, sin necesidad de la infraestructura de llave pública (PKI). En el segundo caso, cada parte tiene un único par de llaves (pública y privada), la autenticación se basa en la posesión de la llave privada, la cual es normalmente almacenada en un dispositivo a prueba de manipulaciones, y PKI es necesaria para la distribución de forma segura de las llaves autenticadas a todos los usuarios [77].

El primer protocolo de intercambio de llaves basado en criptografía asimétrica fue el propuesto por Diffie-Hellman [78]. Es una técnica fundamental para obtener un intercambio de llaves sin autenticación usando una exponenciación. Su seguridad recae en la dificultad del problema de Diffie-Hellman y el problema del logaritmo discreto. Muchos protocolos de intercambio de llaves se basan en las ideas de Diffie-Hellman y tales protocolos pueden ser descritos sobre cualquier grupo, en el cual el problema del logaritmo discreto es difícil y el cálculo de la exponenciación es eficiente. Estos grupos incluyen los grupos multiplicativos $\mathbb{Z}_p$ con p como un número primo, el campo $\mathbb{F}_{2^m}$, o el grupo de puntos en una curva elíptica sobre un campo finito.

Se han realizado muchos intentos de añadir autenticación al protocolo de Diffie-Hellman. Uno de los protocolos AKE más conocido en la familia Diffie-Hellman, es el protocolo propuesto por Menezes, Qu y Vastone [79]. En él, se afirma que el protocolo proporciona un intercambio de llaves mutuamente autenticadas. Sin embargo, como señala Burton y Kaliski en [80], este protocolo es vulnerable al ataque de *unknown key share*.

Un protocolo de intercambio de llaves autenticadas, es denominado basado en la identidad, si los usuarios en el protocolo usan un par de llaves asimétricas basadas en su identidad, en lugar del par de llaves pública y privada tradicionales. En 1984, Shamir introdujo el concepto de criptografía basada en la identidad [15], para evitar los problemas de las llaves públicas (certificados y autoridades de certificación) en la criptografía clásica de llave pública (Sección 1.1.4en la página 4).

Algunos protocolos de intercambio de llaves basados en la identidad se han desa-

rrollado de acuerdo a la propuesta de Diffie-Hellman y usando la idea establecida por Shamir. Por ejemplo, Okamoto [81] presentó un esquema basado en la identidad el cual fue modificado ligeramente por Tanaka y Okamoto [82]. Girault y Pailles [83] desarrollaron un sistema basado en la identidad, el cual puede ser usado para esquemas no interactivos de intercambio de llaves. Otro esquema de este tipo es el propuesto en el ISO/IEC 11770-3 [84].

En el 2001, se publicaron las primeras soluciones viables para el cifrado basado en la identidad. Una de estas publicaciones fue la de Boneh y Franklin [16], la cual se basa en emparejamientos sobre curvas elípticas y es el primer esquema formalmente demostrado de cifrado basado en la identidad. Poco después a este trabajo, se desarrollaron protocolos de intercambio de llaves basados en la identidad, utilizando técnicas de emparejamientos como la propuesta de Antoine Joux [12], considerada como el primer esquema de intercambio de llaves; el trabajo de Sakai *et al.* [14], como la primera construcción de llaves basadas en la identidad; y recientemente se han propuesto diversos protocolos de intercambio de llaves basados en la identidad utilizando emparejamientos, entre los cuales se encuentran [85–89].

4.5. Ataques a protocolos de autenticación

En 1999 Halevi y Krawczyk [90] introdujeron la noción de seguridad para autenticación remota con contraseñas fáciles de recordar. Propusieron una lista de ataques básicos de los que un protocolo cliente servidor que utiliza contraseñas debe protegerse. En un trabajo más reciente Yongge Wang [91] presentó una lista con el mismo objetivo, pero enfocado a protocolos de autenticación basados en contraseñas utilizando *smart cards*.

Aunque estos ataques son importantes para la autenticación basada en contraseñas, no son suficientes para el caso en que la autenticación se lleva a cabo utilizando dispositivos móviles. En esta sección, se propone una lista de ataques de los que un protocolo de autenticación remota basado en contraseñas, utilizando dispositivos móviles debe protegerse. Un protocolo ideal de este tipo debería ser seguro contra tales ataques, por lo que en lo siguiente se abordará su seguridad siguiendo estos criterios y los presentados en la Sección 4.6.

- **Espionaje**: El atacante puede escuchar el canal de comunicación.
- **Repetición**: El atacante graba los mensajes (del canal de comunicación) que ha observado y después los reenvía.
- **Hombre de en medio**: El atacante intercepta los mensajes enviados entre las dos partes (el cliente y el servidor) y los reemplaza con sus propios mensajes. El atacante juega el papel de usuario en los mensajes que son enviados al servidor, y al mismo tiempo hace el papel de servidor en los mensajes que son enviados al usuario.

- **Adivinar contraseña**: También conocido como ataque de diccionario. En este ataque se supone que el atacante tiene acceso a un diccionario relativamente pequeño que contiene elecciones comunes de contraseñas. Existen dos formas en las que un atacante puede hacer uso del diccionario:

  a) **Fuera de línea**: El atacante graba comunicaciones pasadas, y entonces busca en el diccionario una contraseña que sea consistente con la comunicación grabada. Si la contraseña es encontrada, el atacante concluye que esa es la contraseña del usuario.

  b) **En línea**: El atacante repetidamente escoge una contraseña del diccionario y trata de usarla para suplantar al usuario. Si la suplantación falla, el atacante elimina esa contraseña del diccionario y prueba nuevamente usando una contraseña diferente.

- **Suplantación por llave comprometida**: El atacante obtiene los datos de la cuenta de un usuario y está en posición de autenticarse ante el servidor pretendiendo ser él, o puede pretender ser el servidor para el usuario, con el fin de obtener información útil. Por otro lado, también puede darse el caso de que el servidor quiera suplantar a un usuario haciendo uso de su conocimiento.

- **Dispositivo robado**: Este tipo de ataques tiene dos posibles escenarios, que dependen de la naturaleza del dispositivo en el cual se almacenan los datos sensibles.

  a) El dispositivo es a prueba de manipulaciones: El atacante no puede leer la información sensible almacenada en la memoria del dispositivo. Además, este escenario puede dividirse en dos casos en donde: el atacante puede hacer sólo un número limitado de consultas al dispositivo y en donde el atacante puede hacer un número ilimitado de consultas para revelar información útil. En el primer caso si el número de consultas rebasa el limite permitido el dispositivo es deshabilitado.

  b) El dispositivo no es a prueba de manipulaciones: El atacante con el dispositivo puede ser capaz de romper la protección del mismo y leer la información sensible almacenada en la memoria. En este caso el dispositivo se ve como una memoria que almacena la información sensible del usuario, protegida con una contraseña y para que el usuario pueda utilizar la información almacenada en el dispositivo, debe tener acceso a un equipo de confianza para llevar a cabo la autenticación.

- **Confidencialidad directa**: Este más que un ataque es una propiedad, la cual consiste en que si un atacante corrompe los secretos basados en la identidad de una o más de las entidades, la confidencialidad de las llaves de sesión establecidas previamente no deberá ser afectada. Esta propiedad puede ser vista de tres formas distintas [85]:

a) Un sistema ofrece confidencialidad directa parcial, si los secretos basados en la identidad de una o más de las entidades, pero no todas, pueden ser corrompidos sin afectar la confidencialidad de las llaves de sesión establecidas previamente.

b) Un sistema ofrece confidencialidad directa perfecta, si los secretos basados en la identidad de todas las entidades involucradas, pueden ser corrompidas sin afectar la confidencialidad de cualquier llave de sesión previamente establecida por las entidades.

c) Un sistema ofrece confidencialidad directa AC, si el secreto maestro de la AC puede ser corrompido y este hecho no compromete la confidencialidad de las llaves de sesión establecidas previamente por los usuarios. Ciertamente esta idea implica confidencialidad directa perfecta.

4.6. Características deseables en un protocolo de autenticación

En los artículos mencionados en la Sección 4.2en la página 48 podemos suponer que existe una computadora cliente, que es considerada confiable, en donde el usuario introduce su contraseña. Sin embargo, esta suposición no es verdadera en un sistema de autenticación basado en *smart cards* o dispositivos móviles, debido a que el lector de *smart cards* podría ser malicioso y puede interceptar las contraseñas de entrada del usuario, además que el *smart card* o el dispositivo móvil podrían ser robados y el atacante puede aplicar ataques de diccionario fuera de línea contra ellos.

Muchos autores han propuesto listas útiles de características deseables en los protocolos de autenticación. Tsai *et al.* [92] proveen una lista de nueve requerimientos de seguridad y diez metas para este tipo de protocolos. Sin embargo, su revisión de los esquemas disponibles en ese momento revela que todos son decepcionantes. Liao *et al.* [93] propusieron una lista de diez propiedades, las cuales Yang *et al.* [94] redujeron a cinco. Posteriormente, en el 2012 Wang [91] realizó una lista de ocho ataques que deben resistir estos protocolos e identificó tres categorías de ataques potenciales. En ese mismo año Scott [17], después de hacer una revisión de los trabajos anteriores, propuso una lista de condiciones deseables que un esquema ideal debe satisfacer, las cuales se presentan a continuación:

1. El protocolo debe dar como resultado la mutua autenticación del cliente y el servidor y derivar una misma llave criptográfica, con la cual serán cifradas las comunicaciones subsecuentes.

2. No debe haber datos relacionados al PIN almacenados en el servidor. Sin embargo, el servidor debe ser capaz de ayudar al cliente a recuperar el PIN olvidado.

3. El subyacente intercambio de llaves autenticadas debe ser inmune a la suplantación por llave comprometida.

4. El cliente deberá ser capaz de cambiar su PIN localmente sin involucrar al servidor.

5. Un atacante que gana la posesión del secreto del servidor de autenticación sólo debería ser capaz de:

   a) Establecer un servidor falso

   b) Dado el token de un cliente determinar su PIN

   Teniendo en cuenta que esto es básicamente lo mejor que se puede esperar, para cualquier protocolo de autenticación.

6. El servidor deberá ser capaz de identificar el grado de cualquier error pequeño ϵ en el secreto del cliente, con el fin de facilitar la inclusión de medidas biométricas imprecisas como un tercer factor.

7. El protocolo debe mantener la propiedad de confidencialidad directa.

8. El protocolo debe ser verdaderamente “multifactor”, es decir, si hay n factores involucrados, la pérdida de $n-1$ factores no debería ser suficiente para permitir que el último factor sea encontrado.

9. Ataques de adivinación de contraseña pueden ser llevados a cabo sólo en línea y por lo tanto pueden ser monitoreados y prevenidos por el servidor.

10. El sistema completo no deberá tener ningún punto de fallo.

4.7. Revisión del esquema “Software-only two-factor authentication”

En la siguiente sección se presenta la revisión del protocolo propuesto por Michael Scott “Software-only two-factor authentication” [17], la cual se encuentra dividida en las siguientes partes: en la Sección 4.7.1 se describen las fases que componen el protocolo de autenticación, en la Sección 4.7.2 se presenta el análisis de correctitud correspondiente al protocolo y por último en la Sección 4.7.3 se presenta el análisis de seguridad del protocolo de acuerdo a lo establecido en la Sección 4.6.

4.7.1. Fases del protocolo

El esquema de Scott esta basado en el protocolo de acuerdo de llaves autenticadas basadas en la identidad propuesto por Wang [89], el cual fue presentado para un escenario de punto apunto y Scott adaptó al escenario cliente servidor, aplicándolo con ayuda de emparejamientos bilineales tipo 3. El protocolo original basa su seguridad en el problema de decisión bilineal de Diffie-Hellman y forma parte del estándar IEEE

P1363.3 [95], mientras que el esquema propuesto por Scott basa su seguridad en el problema XDH explicado en la Sección 2.4.

El esquema se limita al escenario práctico y de bajo costo en el cual se asume que: Se utiliza un Token estático clonable, la contraseña de baja entropía esta compuesta por un PIN de cuatro dígitos y en donde un factor biométrico puede ser incluido. Estas características le permiten utilizar una solución utilizando sólo software, sobre un canal de comunicación hostil como Internet.

4.7.1.1. Registro

En esta fase se supone la existencia de una autoridad certificadora (AC) independiente, que no se requiere en línea y la cual tiene su secreto maestro, esta autoridad es responsable del registro y la emisión, fuera de línea, de los secretos basados en la identidad para el cliente y el servidor.

Al inicio la AC provee los secretos basados en la identidad, estos secretos son calculados como $S_c = C \cdot s$ para el cliente y $S_s = S \cdot s$ para el servidor, en donde s representa el secreto maestro de la AC, mientras que C y S son obtenidos mediante la aplicación de H_1 a la identidad del cliente ID_c y H_2 a la identidad del servidor ID_s. Es por demás aclarar que el hecho de conocer C y S_c o S y S_s no implica revelar alguna información acerca de s, debido a que está completamente protegido por la dificultad del problema del logaritmo discreto en curvas elípticas.

En un protocolo de este tipo es importante que los secretos del cliente y el servidor sean diferentes. Una manera simple de lograrlo es explotar la estructura de los emparejamientos tipo 3, poniendo los secretos de los clientes en el grupo $\mathbb{G}_1$ y los correspondientes a los servidores en $\mathbb{G}_2$. En un emparejamiento tipo 3 se asume que no existe un isomorfismo computable entre estos grupos, a pesar de que ambos son del mismo orden.

4.7.1.2. Proceso de autenticación

El cliente y el servidor deben conocer de antemano algunos datos necesarios para el proceso de autenticación, estos datos se describen a continuación:

- Conocimiento del cliente:
  - **IDc:** corresponde a la identidad del cliente.
  - S_c**:** este es un dato importante que corresponde al secreto basado en la identidad del cliente, el cual es un punto en el grupo $\mathbb{G}_1$ y debe ser almacenado de forma segura.
  - **PIN**: es la contraseña de baja entropía de cuatro dígitos memorizado por el cliente.
  - **Token:** este dato es calculado como: $Token = S_c - PIN \cdot C$, donde $C = H_1(IDc)$. Este dato también debe ser almacenado de forma segura.

- Conocimiento del servidor:
  - **IDs:** corresponde a la identidad del servidor.
  - S_s**:** este es un dato importante que corresponde al secreto basado en la identidad del servidor, el cual es un punto en el grupo $\mathbb{G}_2$ y debe ser almacenado de forma segura.

Una vez que el cliente y el servidor pasaron la fase de registro, ya cuentan con el conocimiento necesario para poder llevar a cabo el proceso de autenticación, este proceso se realiza por medio del protocolo descrito en la Figura 4.1:

Cliente	**Servidor**
Genera aleatoriamente $x < r$	Genera aleatoriamente $y, w < r$
$IDc \rightarrow$	$\leftarrow IDs$
$S = H_2(IDs), C = H_1(IDc)$	$C = H_1(IDc), S = H_2(IDs)$
$P_c = x \cdot C \rightarrow$	$\leftarrow P_s = y \cdot S, P_g = w \cdot C$
$\pi_c = H_{r1}(P_c \mid P_s \mid P_g)$	$\pi_s = H_{r2}(P_s \mid P_c \mid P_g)$
$\pi_s = H_{r2}(P_s \mid P_c \mid P_g)$	$\pi_c = H_{r1}(P_c \mid P_s \mid P_g)$
$k = e((x + \pi_c)(Token + PIN \cdot C), \pi_s \cdot S + P_s)$	$k = e(\pi_c \cdot C + P_c, (y + \pi_s)S_s)$
$K = H_k(k \mid x \cdot P_g)$	$K = H_k(k \mid w \cdot P_c)$
$M = H_r(IDc \mid IDs \mid K)$	$N = H_r(IDc \mid IDs \mid K)$
$M \rightarrow$	si $M \neq N$, interrumpe la conexión

Figura 4.1: Protocolo de autenticación de dos factores "Software-only two-factor authentication"

En el protocolo de la Figura 4.1, r se define como el orden de los grupos involucrados en el emparejamiento, las funciones $H_1(\cdot)$ y $H_2(\cdot)$ representan la función picadillo a los grupos $\mathbb{G}_1$ y $\mathbb{G}_2$ respectivamente, $H_{r1}(\cdot)$ y $H_{r2}(\cdot)$ son funciones picadillo que realizan la proyección de la concatenación de los puntos en $\mathbb{G}_1$ y $\mathbb{G}_2$ a un número en el rango de 1 a $r - 1$, $H_k(\cdot)$ es una función picadillo que realiza la proyección de la concatenación del resultado del emparejamiento k y un punto en $\mathbb{G}_1$ a un número en el rango de 1 a $r - 1$, mientras que $H_r(\cdot)$ es una función picadillo que realiza la proyección de la concatenación de las identidades del cliente y el servidor con la llave criptográfica a un número en el rango de 1 a $r - 1$.

4.7.1.3. Cambio y recuperación del PIN

El proceso de cambio de PIN se lleva a cabo de manera transparente para el servidor, lo que es una de las características deseables en un protocolo de autenticación.

Este procedimiento es realizado de manera simple, cuando un cliente quiere cambiar su PIN actual α por uno nuevo β, se realizan los siguientes dos pasos:

Paso 1 Se calcula el valor de $\beta \cdot C$

Paso 2 Se substrae $\beta \cdot C$ del secreto del cliente S_c, para obtener un nuevo Token $(S_c - (\beta \cdot C))$ el cual debe ser almacenado de forma segura.

Es de hacer notar que el Token previo no puede ser utilizado para completar exitosamente el protocolo de autenticación, si se desconoce el PIN α.

Por otro lado, la recuperación de un PIN olvidado sí requiere de la participación del servidor. Para esto se crea deliberadamente una circunstancia en donde el servidor puede llevar a cabo un ataque de diccionario fuera de línea sobre el PIN. Entonces un cliente que quiere recuperar su PIN debe seguir los pasos descritos a continuación:

Paso 1 El cliente calcula y envía al servidor $X = H(e((s - \alpha)C + g \cdot C, S))$, donde $(s - \alpha)C$ corresponde a su Token y g es el PIN incorrecto.

Paso 2 Después, el cliente le prueba su identidad al servidor por medio de una pregunta, de la que sólo el cliente puede conocer la respuesta.

Paso 3 Una vez probada la identidad del cliente por el servidor, este va fuera de línea y calcula $Y = H(e(C, S_s - iS))$, para todas las posibles i, hasta que $X = Y$.

Paso 4 La diferencia i encontrada entre el PIN correcto α y el PIN incorrecto g, se envía al cliente por correo electrónico o por cualquier otro medio.

Paso 5 Entonces el cliente puede recuperar su PIN a partir de $\alpha = i + g$

Cabe señalar que el PIN α en ningún momento es revelado al servidor, debido a que éste no conoce el valor de g.

4.7.2. Análisis de correctitud

En la siguiente sección se presenta el análisis de correctitud del protocolo, con el fin de verificar que realmente las partes involucradas en él, generen la misma llave criptográfica de sesión. Además, se muestra que la fase de recuperación de PIN puede llevarse a cabo de manera exitosa.

Durante la ejecución del protocolo se intercambian valores relacionados con la identidad de los participantes ID_c e ID_s; además de los puntos $P_c = x \cdot C$, $P_g = w \cdot C$ y $P_s = y \cdot S$, donde x, y y w son valores aleatorios generados por ellos, C y S son el resultado de las funciones picadillo H_1 y H_2 aplicadas sobre ID_c e ID_s respectivamente. Estos valores intercambiados en el protocolo generan datos que son conocidos

por cada una de las partes involucradas, como: $S = H_2(ID_s),\ C = H_1(ID_c),\ \pi_c = H_{r1}(P_a|P_s|P_g)$ y $\pi_s = H_{r2}(P_s|P_a|P_g)$.

La parte más importante del protocolo se encuentra en el cálculo de los emparejamientos $k = e((x+\pi_c)(Token + PIN \cdot C), \pi_s \cdot S + P_s)$ y $k = e(\pi_c \cdot C + P_c, (y+\pi_s)S_s)$, realizados en el cliente y en el servidor respectivamente, ya que es en donde se espera obtener el mismo valor cuando se utilizan secretos basados en la identidad "legítimos". A continuación se verifica que esto realmente suceda, comprobando la igualdad de los emparejamientos:

$$e((x+\pi_c)(Token + PIN \cdot C), \pi_s \cdot S + P_s) \quad = \quad e(\pi_c \cdot C + P_c, (y+\pi_s)S_s)$$

Ahora, teniendo en cuenta el hecho de que $Token + PIN \cdot C$ es igual a S_c, sustituyendo P_s, P_c y aplicando la propiedad de bilinearidad de los emparejamientos, obtenemos:

$$\begin{aligned}
e((x+\pi_c)S_c, \pi_s \cdot S + P_s) &= e(\pi_c \cdot C + P_c, (y+\pi_s)S_s) \\
e((x+\pi_c)S_c, \pi_s \cdot S + y \cdot S) &= e(\pi_c \cdot C + x \cdot C, (y+\pi_s)S_s) \\
e(S_c, \pi_s \cdot S + y \cdot S)^{(x+\pi_c)} &= e(\pi_c \cdot C + x \cdot C, S_s)^{(y+\pi_s)} \\
e(S_c, (y+\pi_s)S)^{(x+\pi_c)} &= e((x+\pi_c)C, S_s)^{(y+\pi_s)} \\
e(S_c, S)^{(x+\pi_c)(y+\pi_s)} &= e(C, S_s)^{(x+\pi_c)(y+\pi_s)} \\
e(C, S)^{s(x+\pi_c)(y+\pi_s)} &= e(C, S)^{s(x+\pi_c)(y+\pi_s)}.
\end{aligned}$$

Otra parte significativa en el protocolo, es la correspondiente a la generación de la llave criptográfica mutua, la cual se realiza calculando $K = H_k(k \mid x \cdot P_g)$ en el cliente y $K = H_r(k \mid w \cdot P_c)$ en el servidor. La verificación de este paso se realiza a continuación, sustituyendo los valores de P_g y P_c:

$$\begin{aligned}
H_k(k \mid x \cdot P_g) &= H_k(k \mid w \cdot P_c) \\
H_k(k \mid x \cdot w \cdot C) &= H_k(k \mid w \cdot x \cdot C)
\end{aligned}$$

Teniendo en cuenta lo anterior podemos concluir que, si ambas partes cuentan con secretos basados en su identidad "legitimos", el protocolo realiza exitosamente la autenticación y deriva una llave criptográfica de sesión mutua.

Como se describió en la fase de recuperación de PIN, este proceso es llevado a cabo a través de la siguiente igualdad

$$H(e((s-\alpha)C + g \cdot C, S)) = H(e(C, S_s - i \cdot S)),$$

la cual prueba el servidor para todas las posibles i. Si se pasa por alto el cálculo de H y se toma en cuenta que los valores correspondientes a α y g son mucho muy pequeños comparados con s, se puede observar que la igualdad se satisface cuando:

$$e((s-\alpha)C + g \cdot C, S) = e(C, S_s - \overbrace{(\alpha - g)}^{i} \cdot S).$$

Una vez que el cliente recibe el valor de i, puede utilizar el conocimiento de g para recuperar su PIN α a través de la ecuación $\alpha = i + g$.

4.7.3. Análisis de seguridad

Como se mencionó en la Sección 4.6, muchos autores han propuesto listas útiles de características deseables en los protocolos de autenticación. En el 2012, Scott [17], después de hacer una revisión de los trabajos realizados, propuso una lista de condiciones deseables que un esquema ideal debe satisfacer. A continuación se presenta el análisis de seguridad del protocolo de acuerdo a los puntos establecidos por él y a los ataques descritos en la Sección 4.5.

1. El protocolo debe dar como resultado la mutua autenticación del cliente y el servidor, y derivar una misma llave criptográfica.

   La autenticación mutua es llevada a cabo gracias a que para lograr derivar la misma llave criptográfica K, las partes involucradas deben contar con secretos basados en su identidad "legítimos". Si alguno de los secretos basados en la identidad de los clientes no lo es, entonces el servidor deberá cortar las comunicaciones. En el caso de que el servidor sea falso, los usuarios pueden saber que están tratando con él, ya que las comunicaciones siguientes deberán estar cifradas con la llave derivada K, en este contexto la autenticación sólo podrá tener éxito si el atacante fue capaz de obtener el secreto del servidor.

2. No debe haber datos relacionados al PIN almacenados en el servidor. Sin embargo, el servidor debe ser capaz de ayudar al cliente a recuperar su PIN olvidado.

   Dado el hecho de que el PIN sólo es útil para el cliente y tomando en cuenta que la llave criptográfica de sesión K es calculada dentro del protocolo y está en constante cambio, el servidor no necesita almacenar información relacionada con el PIN o la llave criptográfica K. Por otra parte, como se observó en la Sección 4.7.1.3, el servidor tiene la capacidad de ayudar a un cliente a recuperar su PIN.

3. El subyacente intercambio de llaves autenticadas debe ser inmune a la suplantación por llave comprometida.

   La solución para bloquear el ataque de suplantación por llave comprometida es "sobrecargar" ambos lados del emparejamiento con otras operaciones necesarias.

   Como se observa en el protocolo, esto es realizado de la siguiente manera: El cliente calcula $k = e((x+\pi_c)(Token+PIN\cdot C), \pi_s \cdot S + P_s)$ y el servidor hace el cálculo de $k = e(\pi_c \cdot C + P_c, (y+\pi_s)S_s)$. Este tipo de ataque como se menciona en la Sección 4.5 tiene dos escenarios, los cuales se revisan a continuación:

   a) El servidor pretende ser un cliente aprovechando el conocimiento de S_s, ID_c y de $C = H_1(ID_c)$. En la ejecución del protocolo, se realizan con éxito los pasos de la Figura 4.2, después se debe realizar el cálculo de los emparejamientos y es aquí en donde el servidor (denotado como cliente) pretende aprovechar su conocimiento, con el fin de hacerse pasar por un cliente verdadero.

      El Servidor realiza el cálculo como: $k = e(\pi_c \cdot C + P_c, (y+\pi_s)S_s)$, mientras que el cliente desconoce el valor del Token y del PIN, necesarios para calcular

$k = e((x+\pi_c)(Token+PIN\cdot C), \pi_s \cdot S+P_s)$. El problema a resolver es explotar la bilinearidad y el conocimiento del cliente con la finalidad de mover el componente que involucra el secreto de la AC de un lado del emparejamiento al otro.

Cliente	**Servidor**
Genera aleatoriamente $x < r$	Genera aleatoriamente $y, w < r$
$IDc \rightarrow$	$\leftarrow IDs$
$S = H_2(IDs), C = H_1(IDc)$	$C = H_1(IDc), S = H_2(IDs)$
$P_c = x \cdot C \rightarrow$	$\leftarrow P_s = y \cdot S, P_g = w \cdot C$
$\pi_c = H_{r1}(P_c \mid P_s \mid P_g)$	$\pi_s = H_{r2}(P_s \mid P_c \mid P_g)$
$\pi_s = H_{r2}(P_s \mid P_c \mid P_g)$	$\pi_c = H_{r1}(P_c \mid P_s \mid P_g)$

Figura 4.2: Pasos realizados con exito en el protocolo en los ataques a) y b)

En este caso, el cliente deberá realizar el emparejamiento utilizando $(x+\pi_c)C$ en un lado y tratará de aprovechar su conocimiento para el lado faltante. Lo que el cliente espera es poder utilizar S_s para lograr su cometido, sin embargo, esto no es posible dado que significa que cuenta con la capacidad de calcular el valor de y teniendo $P_s = y \cdot S$, es decir, resolver el problema del logaritmo discreto. Si suponemos que el cliente de alguna forma obtuvo el valor de y, este puede tener éxito realizando:

$$\begin{aligned} k &= e((x+\pi_c)C, \pi_s \cdot S + P_s) \\ &= e((x+\pi_c)C, (\pi_s + y)S_s) \\ &= e(C, S)^{s(x+\pi_c)(\pi_s+y)} \end{aligned}$$

b) El cliente se hace pasar por servidor, con el fin de conseguir información útil. En este caso se tiene un atacante que obtuvo el Token y PIN de un cliente, y con esta información pretende hacerse pasar por el servidor. Una vez más se realizan con éxito los pasos de la Figura 4.2 y el punto crucial es el cálculo de los emparejamientos.

   En este caso el cliente realiza el cálculo del emparejamiento de manera normal, por otro lado el servidor no tiene un valor "legitimo" S_s. Por tanto, el servidor necesita aprovechar el conocimiento del Token y PIN del cliente, en el primer operando y utilizar como segundo operando $(y+\pi_s)S$, sin embargo, esto no es posible dado que el servidor debe conocer una forma de obtener x a partir de $P_c = x \cdot C$, lo cual implica resolver el problema del logaritmo discreto. Una vez más, si suponemos que el servidor puede obtener el valor de x y teniendo en cuenta que $S_c = Token + C \cdot PIN$, puede tener éxito de

la siguiente forma:

$$\begin{aligned} k &= e(\pi_c \cdot C + P_c, (y + \pi_s)S) \\ &= e((\pi_c + x)S_c, (y + \pi_s)S) \\ &= e(C, S)^{s(\pi_c + x)(y + \pi_s)} \end{aligned}$$

4. El cliente deberá ser capaz de cambiar su PIN localmente sin involucrar al servidor.

   En el protocolo, el PIN es útil sólo en la parte del cliente ya que es ahí donde se reconstruye su secreto basado en la identidad, a partir del Token y el PIN. El cambio de PIN puede ser realizado localmente gracias a la ecuación:

$$S_c = \underbrace{(s - PIN)C}_{Token} + PIN \cdot C,$$

   en donde si un incremento δ es sumado al PIN, el efecto puede ser contrarestado incrementando el PIN en el Token en la misma cantidad. Por ejemplo, si el PIN es incrementado por δ, se satisface la relación

$$S_c = (s - (PIN + \delta))C + (PIN + \delta)C,$$

   la cual no requiere participación del servidor.

5. Un atacante que gana la posesión del secreto S_s del servidor de autenticación sólo debería ser capaz de:

   a) Establecer un servidor falso

   b) Dado el Token de un cliente determinar su PIN

   De la comunicación, el atacante conoce IDs y S_s. Con este conocimiento, el atacante esta en posición de montar un servidor falso, el cual le permite establecer una comunicación con cualquier cliente que solicite ser autenticado. Sin embargo, esta comunicación no revela información relacionada con el Token y el PIN de los clientes y como se verificó en el punto 3 inciso a), el atacante no es capaz de suplantar a un cliente.

   Por otra parte, si el atacante además del ID_s y S_s obtiene el Token de un cliente, está en posición de recuperar el PIN relacionado, probando para toda i hasta que la siguiente relación se cumpla.

$$e(Token + i \cdot C, S) = e(C, S_s)$$

   Como se observa, en la relación es necesario el conocimiento de C, el cual puede obtenerse observando el canal de comunicación para conseguir ID_c y calcular $C = H_1(ID_c)$. El éxito del atacante se debe a que el tamaño del espacio muestral

del PIN es relativamente pequeño, y una vez que el atacante obtiene el PIN esta en posición de suplantar al cliente ante el servidor.

Por lo tanto se puede concluir que lo anterior es lo mejor que puede hacer el atacante que obtiene el secreto del servidor.

6. El servidor deberá ser capaz de identificar el grado de cualquier error pequeño ϵ en el secreto del cliente. Con el fin de facilitar la inclusión de medidas biométricas imprecisas como un tercer factor.

   En el protocolo, si el secreto reconstruido del cliente es incorrecto por una pequeña cantidad, el servidor aún puede derivar una llave mutua de sesión ya que es posible determinar el grado del error y compensarlo. Por ejemplo: si el cliente usa como su secreto $S_c + \epsilon \cdot C$, el servidor puede compensarlo utilizando $S_s + \epsilon \cdot S$ como su secreto.

   Teniendo en cuenta que el servidor puede tomarse un tiempo para buscar el error ϵ, el protocolo podría contar con la capacidad de corregir errores en el secreto del cliente y con esto permitir la biometría como un tercer factor.

7. El protocolo debe mantener la propiedad de confidencialidad directa.

   Como se observó en la Sección 4.5, la propiedad de confidencialidad directa puede ser vista de tres formas: confidencialidad directa parcial, confidencialidad directa perfecta y confidencialidad directa AC. Sin embargo, esta última implica las otras dos, por lo que este punto sera analizado a partir de ella.

   De acuerdo con Liqun Chen and Caroline Kudla [85], es deseable contar con un protocolo de intercambio de llaves, en el que los secretos basados en la identidad sean utilizados para la autenticación, pero las llaves de sesión se ocupen de forma desconocida por la AC o por cualquiera que sólo conoce estas llaves. El método para lograr este objetivo consiste en que los usuarios calculen una llave compartida a través de un Diffie-Hellman de sus contribuciones efímeras.

   En el protocolo las contribuciones efímeras son los números aleatorios x y w, los cuales generan P_c del lado del cliente y P_g del lado del servidor. El intercambio Diffie-Hellman se lleva a cabo al calcular $K = H_k(k \mid x \cdot P_g)$ y $K = H_r(k \mid w \cdot P_c)$ en el cliente y el servidor, respectivamente. Obteniendo en ambas partes el valor correspondiente a $K = H_r(k \mid (w \cdot x)C)$.

8. El protocolo debe ser verdaderamente "multifactor", es decir, si hay n factores involucrados, la pérdida de $n-1$ factores no deberá ser suficiente para permitir que el último factor sea encontrado.

   En el análisis de este punto se considera que el protocolo es de dos factores, dejando de lado la posibilidad del factor biométrico. Los factores considerados en el protocolo son el PIN que es memorizado por el cliente y el Token, el cual es un valor almacenado en un dispositivo. Los dos factores involucrados en el protocolo obedecen la relación $S_c = (s - PIN)C + PIN \cdot C$, por lo que si se

supone la pérdida de alguno de los dos, no significa la posibilidad de recuperar el factor faltante. Esto es debido a que se desconoce el valor de S_c y cualquier valor es posible para $(s - PIN)C + C \cdot PIN$, además de que es necesaria la colaboración del servidor para comprobar el éxito. Sin embargo, si se cuenta con el Token y S_c, es fácil encontrar el valor del PIN considerando que el espacio muestral de PIN es relativamente pequeño.

9. Ataques de adivinación de contraseña pueden ser llevados a cabo sólo en línea y por lo tanto pueden ser monitoreados y prevenidos por el servidor.

   Dado que la única manera de saber si un PIN es correcto es completando exitosamente el protocolo, entonces sólo se pueden aplicar ataques de adivinación de contraseña en línea, por lo tanto el servidor puede monitorear estos ataques y tomar las medidas pertinentes.

10. El sistema completo no deberá tener ningún punto de fallo.

    El único punto vulnerable en todo el sistema es el secreto maestro de la AC. Sin embargo, este secreto maestro puede ser fácilmente un secreto compartido entre una serie de AC independientes [16]. Por ejemplo, si se considera el caso en donde se cuenta con dos AC, estas autoridades independientemente generan y emiten al cliente, los secretos basados en la identidad s_1C y s_2C. Los cuales pueden ser sumados por el cliente para derivar $S_c = s_1C + s_2C$.

En la Sección 4.5 se consideran diferentes tipos de ataques, algunos de los cuales fueron abordados en párrafos anteriores. Sin embargo, cuatro de ellos no fueron considerados y serán analizados a continuación:

- Los ataques de espionaje y repetición no se pueden evitar ya que el canal de comunicación es Internet y es considerado inseguro. Entonces, un atacante puede utilizar uno de los mensajes interceptados para aplicar un ataque de adivinación de contraseña y obtener la llave K, la cual fue utilizada para cifrar el mensaje interceptado. Sin embargo, el conocimiento de K es útil sólo para la sesión en la cual se interceptó el mensaje.

- El ataque del hombre de en medio no es posible debido a que en el paso que corresponde al cálculo del emparejamiento, el atacante necesita conocer de alguna manera el secreto de la AC. Gracias a que en el protocolo sólo se comunican valores efímeros, relacionados con las identidades del cliente y el servidor, y que en ningún momento se comunican mensajes relacionados con sus secretos basados en la identidad, el ataque no puede ser llevado a cabo exitosamente.

- El ataque de dispositivo robado está estrechamente relacionado con el tipo de dispositivo que se utiliza en el proceso de autenticación. En el protocolo se sugiere la utilización de un Token estático, el cual puede ser clonado. Considerando las características del protocolo el dispositivo debe tener almacenada la

información correspondiente al Token y al secreto del cliente, en este escenario un atacante puede tener éxito de la siguiente manera:

Una vez que un atacante obtiene el dispositivo y el ID_c esta en posición de utilizarlo para recuperar el PIN del cliente. Para realizar esta operación el atacante no requiere la intervención del servidor, si suponemos que los datos fueron cifrados y almacenados en el dispositivo, utilizando como llave de cifrado el PIN. Entonces el atacante quien se supone cuenta con un lector vulnerado, puede obtener los valores del Token y el secreto del cliente, mediante un ataque de diccionario sobre el espacio muestral del PIN, comprobando que el descifrado de los datos satisfaga la igualdad $S_c = Token + PIN \cdot C$. Sin embargo, se puede dar el caso en que más de un intento la igualdad se cumpla, lo que sólo reduce el tamaño del diccionario, dando la oportunidad de tener más probabilidad de éxito al probar los valores de PIN obtenidos ante el servidor. El ataque descrito puede frustrarse, si se utiliza un dispositivo que límite el número de intentos de lectura de datos.

4.8. Revisión del esquema “M-PIN technology”

Este esquema al contrario del revisado en la Sección 4.7, no se limita a un escenario en particular, el autor lo presentó para realizar la autenticación a través de Internet, utilizando una página web como la interfaz de usuario (cliente) para llevarlo a cabo. El protocolo fue propuesto por Michael Scott [18], él propone dos versiones del protocolo la cuales se describen a continuación:

a) M-PIN HS: Este protocolo sólo remplaza el mecanismo de usuario y contraseña en el proceso de autenticación, y utiliza el protocolo SSL para autenticar el servidor ante algún cliente.

b) M-PIN: Además de brindar el mecanismo de usuario y contraseña, proporciona la funcionalidad del protocolo SSL.

En esta tesis se considera sólo la versión “M-PIN”, dado que se intenta hacer una justa comparación con el protocolo de la Sección 4.7.

4.8.1. Fases del protocolo

El esquema puede ser considerado como una simplificación del descrito en la Sección 4.7 y basa su seguridad en la suposición de que el problema de decisión de Diffie-Hellman es difícil de resolver en el grupo $\mathbb{G}_1$ de un emparejamiento tipo 3. Este problema es conocido como XDH y fue abordado en la Sección 2.4.

En las siguientes subsecciones se presentan las fases involucradas en el protocolo de autenticación de dos factores, correspondientes al registro de usuarios, el proceso de autenticación, el cambio y la recuperación del PIN olvidado.

4.8.1.1. Registro

Tanto en el protocolo "M-PIN" como en el analizado en la Sección 4.7, se intenta separar la operación de autenticación del proceso de registro, por lo que se supone la existencia de una autoridad certificadora (AC) independiente, que no se requiere en línea y que cuenta con un secreto maestro. La cual es responsable del registro y la emisión, fuera de línea, de los secretos basados en la identidad para el cliente y el servidor.

Al inicio, la AC provee los secretos basados en la identidad, que son calculados como $S_c = C \cdot s$ para el cliente y $S_s = S \cdot s$ para el servidor, en donde s representa el secreto maestro de la AC, mientras que C y S son obtenidos mediante la aplicación de H_1 a la identidad del cliente ID_c y H_2 a la identidad del servidor ID_s. Nuevamente, es importante que los secretos del cliente y el servidor se mantengan distintos, por lo que los secretos de los clientes se colocan en el grupo $\mathbb{G}_1$ y los correspondientes a los servidores en el grupo $\mathbb{G}_2$, del emparejamiento.

4.8.1.2. Proceso de autenticación

Para llevar a cabo la autenticación el cliente y el servidor deben conocer de antemano algunos datos necesarios, estos datos se describen a continuación:

- Conocimiento del cliente:
  - **IDc:** corresponde a la identidad del cliente.
  - S_c**:** este es un dato importante que corresponde al secreto basado en la identidad del cliente, el cual es un punto en el grupo $\mathbb{G}_1$ y debe ser almacenado de forma segura.
  - **PIN**: es la contraseña de baja entropía de cuatro dígitos memorizado por el cliente.
  - **Token:** este dato es calculado como: $Token = S_c - PIN \cdot C$, donde C es calculado como en la fase de registro. Este dato también debe ser almacenado de forma segura.
- Conocimiento del servidor:
  - **IDs:** corresponde a la identidad del servidor.
  - S_s**:** este es un dato importante que corresponde al secreto basado en la identidad del servidor, el cual es un punto en el grupo $\mathbb{G}_2$ y debe ser almacenado de forma segura.

Una vez que el cliente y el servidor pasaron la fase de registro ya cuentan con el conocimiento necesario para poder llevar a cabo el proceso de autenticación, este proceso se realiza por medio del protocolo descrito en la Figura 4.3, en ella, r denota el orden de los grupos involucrados en el emparejamiento, las funciones $H_1(\cdot)$ y

$H_2(\cdot)$ representan la función picadillo a los grupos $\mathbb{G}_1$ y $\mathbb{G}_2$ respectivamente, mientras que $H_g(\cdot)$ es una función picadillo que realiza la proyección del resultado del emparejamiento k a un número en el rango de 1 a $r-1$.

Cliente	**Servidor**
Genera aleatoriamente $x, m < r$	Genera aleatoriamente $y, n < r$
$C = H_1(IDc)$	
$U = x \cdot C$	
$Z = m(Token + PIN \cdot C)$	
$IDc,\ U,\ Z \quad \rightarrow$	
	$S = H_2(IDs), \quad C = H_1(IDc)$
	$t = e(n \cdot Z, S)$
	$\leftarrow \quad y,\ t$
$V = -(x + y)(Token + PIN \cdot C) \quad \rightarrow$	
	$w = e(n \cdot V, S)$
$k = t^{(x+y)/m}$	$k = e(n(U + y \cdot C), S_s)$ Si $w \cdot y \neq 1$, interrumpe la conexión, si no
$K = H_g(k)$	$K = H_g(k)$

Figura 4.3: Protocolo de autenticación de dos factores "M-PIN"

El protocolo descrito, fue planteado para poder autenticar clientes que cuenten con bajo poder de cómputo ante servidores con un alto poder de procesamiento. Este protocolo a diferencia del descrito en la Sección 4.7 esta desbalanceado, es decir, no se realizan los mismos cálculos en el cliente y en el servidor. Como se puede observar en la Figura 4.3, la parte del cliente en su mayoría utiliza operaciones en el grupo $\mathbb{G}_1$ y una exponenciación en el grupo $\mathbb{G}_T$, mientras que del lado del servidor utiliza operaciones en $\mathbb{G}_1$ y el cálculo de emparejamientos, los cuales tienen un alto costo computacional.

Para lograr desbalacear el protocolo, el autor opto por delegar el cálculo del emparejamiento del lado del cliente al servidor, con base en el trabajo de Chevalier-Mames *et al.* [96]. Considerando el cálculo del emparejamiento $C = e(P, Q)$, la idea es que P y Q sean pasados al servidor para que realice el cálculo y regrese el resultado C al cliente. Sin embargo, esto trae consigo ciertas implicaciones de seguridad, ya que al menos uno de los parámetros del emparejamiento es secreto y se desea evitar pasar su valor a una entidad que probablemente no sea confiable.

Si suponemos que P es el parámetro secreto, el procedimiento se puede lograr de manera simple, utilizando las propiedades del emparejamiento para enmascarar P, de tal forma que se pueda desenmascarar el resultado del emparejamiento sin un

costo computacional elevado, cuando éste es regresado al cliente. Entonces, el cliente genera un valor aleatorio m y enmascara P como $P' = m \cdot P$ y envía al servidor P' y Q. El servidor hace el cálculo del emparejamiento y regresa $D = e(P', Q)$, una vez que el valor D es recibido por el cliente, este puede recuperar el valor correcto a través del siguiente cálculo $C = D^{1/m}$, gracias a la propiedad de bilinearidad de los emparejamientos.

4.8.1.3. Cambio y recuperación del PIN

El proceso de cambio y recuperación de PIN, no se contempla en el trabajo de Scott [18]. Sin embargo, el cambio de PIN puede ser realizado de manera simple, como en el protocolo descrito en la Sección 4.7. Cuando un cliente quiere cambiar su PIN actual α por uno nuevo β, se realiza lo siguiente:

Paso 1 Se calcula el valor de $\beta \cdot C$

Paso 2 Se substrae $\beta \cdot C$ del secreto del cliente S_c, para obtener un nuevo Token $(S_c - (\beta \cdot C))$ el cual debe ser almacenado de forma segura.

Por otra parte el proceso de recuperación de PIN, podría realizarse como se propone a continuación:

Paso 1 El cliente genera aleatoriamente el valor x y envía al servidor los valores IDc, $C = H_1(IDc)$ y $U = x \cdot C$.

Paso 2 Después, el cliente le prueba su identidad al servidor por medio de una pregunta, de la que sólo él puede conocer la respuesta.

Paso 3 Una vez probada la identidad del cliente por el servidor, este genera y envía el valor aleatorio y al cliente y calcula $C = H_1(IDc)$.

Paso 4 El cliente calcula $V = -(x + y)(Token + g \cdot C)$, en donde g es un indicio del PIN, y lo envía al servidor.

Paso 5 El servidor, fuera de línea, calcula $e(U + (y \cdot C), S_s) \cdot e(V + (i \cdot C), S) = 1$, para todas las posibles i, hasta que la igualdad se cumpla.

Paso 6 La diferencia i encontrada entre el PIN correcto α y el PIN incorrecto g, se envía al cliente por correo electrónico o por cualquier otro medio.

Paso 7 Entonces el cliente puede recuperar su PIN a partir de $\alpha = i + g$.

Cabe señalar que el PIN α en ningún momento es revelado al servidor, ya que él no conoce el valor de g utilizado.

4.8.2. Análisis de correctitud

En la siguiente sección se presenta el análisis de correctitud del protocolo, con el fin de verificar que realmente las partes involucradas en él, generen la misma llave criptográfica de sesión.

La parte más importante del protocolo se encuentra en el cálculo del emparejamiento $k = e(n(U+y\cdot C, S_s))$ del lado del servidor y en la exponenciación $k = t^{(x+y)/m}$ en el cliente, ya que es en donde se espera obtener el mismo valor cuando se utilizan secretos basados en la identidad "legítimos", para que ambas partes obtengan una misma llave criptográfica. Considerando que:

$$U = x \cdot C, \qquad Z = m(\underbrace{Token + (PIN \cdot C)}_{S_c}) \qquad y \qquad V = -(x+y)S_c.$$

El servidor calcula $t = e(n \cdot Z, S) = e(n \cdot m \cdot S_c, S)$ y lo envía al cliente junto con el valor aleatorio y generado por el servidor. Una vez que el cliente obtiene t puede calcular el valor k, del cual se deriva la llave con la que se cifran las comunicaciones subsecuentes, de la siguiente manera:

$$\begin{aligned} k &= t^{(x+y)/m} \\ &= e(n \cdot m \cdot S_c, S)^{(x+y)/m} \\ &= e(C, S)^{\frac{n \cdot m \cdot s \cdot (x+y)}{m}} \\ &= e(C, S)^{n \cdot s \cdot (x+y)}. \end{aligned}$$

Por su parte el servidor genera el valor de k como se muestra a continuación:

$$\begin{aligned} k &= e(n(U + (y \cdot C)), S_s) \\ &= e(n((x \cdot C) + (y \cdot C)), S_s) \\ &= e((x \cdot C) + (y \cdot C), S)^{n \cdot s} \\ &= e(C, S)^{n \cdot s \cdot (x+y)}. \end{aligned}$$

Dado que el servidor sólo mantiene la comunicación si $w \cdot k = 1$, se debe comprobar que esto suceda. Tomando en consideración que

$$w = e(n \cdot V, S) \qquad y \qquad k = e(C, S)^{n \cdot s \cdot (x+y)},$$

la comprobación se desarrolla de la siguiente manera:

$$\begin{aligned} w \cdot k &= e(n \cdot V, S) \cdot e(C, S)^{n \cdot s \cdot (x+y)} \\ &= e(-n \cdot (x+y) \cdot S_c, S) \cdot e(C, S)^{n \cdot s \cdot (x+y)} \\ &= e(C, S)^{-n \cdot s \cdot (x+y)} \cdot e(C, S)^{n \cdot s \cdot (x+y)} \\ &= e(C, S)^0 = 1. \end{aligned}$$

Por lo tanto el protocolo tiene éxito si y sólo si las partes involucradas en él cuentan con secretos basados en la identidad "legítimos".

Otro aspecto importante en el protocolo, es la fase de recuperación del PIN que es llevada a cabo a través de la siguiente igualdad

$$e(V + (i \cdot C), S) \cdot e(U + (y \cdot C), S_s) = 1,$$

la cual prueba el servidor para todos los valores posibles de i. Se puede observar que la igualdad se satisface cuando $i = \alpha - g$, como se muestra a continuación:

$$\begin{aligned} w \cdot k &= e(V + (i \cdot C), S) \cdot e(U + (g \cdot C), S_s) \\ &= e(C, S)^{-(s-\alpha+g+\overbrace{\alpha - g}^{i})\cdot(x+y)} \cdot e(C, S)^{s\cdot(x+y)} \\ &= e(C, S)^{-s\cdot(x+y)} \cdot e(C, S)^{s\cdot(x+y)} \\ &= e(C, S)^0 = 1. \end{aligned}$$

Una vez que el cliente recibe el valor de i, puede utilizar el conocimiento de g para recuperar su PIN α, a través de la ecuación $\alpha = i + g$.

4.8.3. Análisis de seguridad

Como se mencionó en la Sección 4.6, muchos autores han propuesto listas útiles de características deseables en los protocolos de autenticación. A continuación se presenta el análisis de seguridad del protocolo de acuerdo a los puntos establecidos por Scott [17] y a los ataques presentados en la Sección 4.5.

Como se puede observar en las Figuras 4.1 y 4.3, este protocolo se puede considerar como una simplificación del llamado "Software only two-factor authentication", por tanto el análisis de algunas características son similares a las descritas en la Sección 4.8.3, entonces a continuación se presentan las características que son específicas para el protocolo "M-PIN".

1. El protocolo debe dar como resultado la mutua autenticación del cliente y el servidor y derivar una misma llave criptográfica.

   La autenticación mutua es realizada, dado que, para lograr derivar la misma llave criptográfica K, las partes involucradas deben contar con secretos basados en su identidad "legítimos". Si alguno de los secretos basados en la identidad de los clientes no lo es, entonces el servidor puede identificarlo con ayuda de la igualdad $w \cdot y \neq 1$, que en caso de cumplirse deberá cortar las comunicaciones. En el caso de que el servidor sea falso, los usuarios pueden saber que están tratando con él dado que las comunicaciones siguientes deberán estar cifradas con la llave derivada K.

2. No debe haber datos relacionados al PIN almacenados en el servidor. Sin embargo, el servidor debe ser capaz de ayudar al cliente a recuperar su PIN olvidado.

   Dado el hecho de que el PIN sólo es útil para el cliente y tomando en cuenta que la llave criptográfica K es calculada dentro del protocolo y está en constante cambio, el servidor no necesita almacenar información relacionada con

en PIN o la llave criptográfica mutua. Por otra parte, como se observó en la sección 4.8.1.3, el servidor tiene la capacidad de ayudar a un cliente a recuperar su PIN olvidado.

3. El subyacente intercambio de llaves autenticadas debe ser inmune a la suplantación por llave comprometida.

   La solución para bloquear el ataque de suplantación por llave comprometida es "sobrecargar" ambos lados del emparejamiento con otras operaciones necesarias. Esto es realizado sólo en un lado del emparejamiento, por ejemplo: $t = e(n \cdot Z, S)$, $w = e(n \cdot V, Q)$ y $k = e(n(U + (y \cdot C)), S_s)$. Sin embargo, como se menciona en la Sección 4.5 este ataque tiene dos escenarios, los cuales se revisan a continuación para verificar que realmente se bloquea:

   a) El servidor pretende ser un cliente aprovechando el conocimiento de S_s, ID_c y de $C = H_1(ID_c)$. El delegar el cálculo del emparejamiento al servidor provoca que el ataque no tenga éxito de manera inmediata, es decir, el protocolo fracasa al momento de calcular las variables $Z = m \cdot S_c$ y $V = -(x+y) \cdot S_c$, ya que el servidor debe conocer el secreto del cliente S_c para poder suplantarlo o de alguna manera obtener el secreto de la AC.

   b) El cliente se hace pasar por servidor, con el fin de conseguir información útil. En este caso se tiene un atacante que obtuvo el Token y PIN de un cliente, y con esta información pretende hacerse pasar por el servidor. El punto crucial en el ataque es el cálculo del emparejamiento $k = e(n(U + (y \cdot C)), S_s)$.

      En este caso el cliente real lleva a cabo el protocolo de manera normal, por otro lado el servidor no tiene un valor "legitimo" de S_s, por lo tanto necesita aprovechar el conocimiento del Token y el PIN del cliente para lograr tener éxito.

      En el caso particular, en donde el ataque se centra en el cliente del cual se obtuvo el Token y PIN, el atacante puede realizar de manera satisfactoria casi todo el protocolo, utilizando $S = H_2(IDc)$. Aunque al llegar al cálculo de k en el servidor, el atacante espera poder utilizar su conocimiento para desplazar el secreto de la AC, de un lado del emparejamiento al otro. Lo cual no es posible debido a que desconoce el valor aleatorio x generado por el cliente. Si suponemos que el atacante de alguna manera obtiene el valor de x, puede tener éxito como se muestra a continuación:

      Sabiendo que $U = x \cdot C$, el atacante puede remplazarlo de manera sencilla por $U = x \cdot S_c$ en el lado del servidor, dejando el cálculo de k de la siguiente forma:

$$\begin{aligned} k &= e(n(U + (y \cdot S_c)), S) \\ &= e(n((x \cdot S_c) + (y \cdot S_c)), S) \\ &= e(n(x + y) \cdot S_c, S) \\ &= e(C, S)^{n \cdot s \cdot (x+y)}. \end{aligned}$$

Lo cual produce un valor de k completamente válido. Sin embargo, el valor de x esta protegido por la dificultad del problema del logaritmo discreto y obtener x significaría haber solucionado el problema, por lo tanto es imposible llevar a cabo el procedimiento anterior.

7. El protocolo debe mantener la propiedad de confidencialidad directa.

   Como se observo en la Sección 4.5, la propiedad de confidencialidad directa puede ser vista de tres formas: confidencialidad directa parcial, confidencialidad directa perfecta y confidencialidad directa AC. Sin embargo, esta última implica las otras dos, por lo que este punto sera analizado a partir de ella.

   El método para lograr este objetivo consiste en que los usuarios calculen una llave de sesión compartida a través de un Diffie-Hellman de sus contribuciones efímeras.

   En el protocolo las contribuciones efímeras son los números aleatorios y, x y m generados por el servidor y el cliente, respectivamente. Una especie de Diffie-Hellman se lleva a cabo al calcular k en el cliente y el servidor, obteniendo en ambas partes el valor correspondiente a $K = H_g(k)$.

En la Sección 4.5 se consideran diferentes tipos de ataques, algunos de los cuales fueron abordados en párrafos anteriores. Sin embargo, cuatro de ellos no fueron considerados y serán analizados a continuación:

- El ataque de dispositivo robado esta estrechamente relacionado con el tipo de dispositivo que se utiliza en el proceso de autenticación. El protocolo se utiliza en un ambiente web, por lo que el token y el secreto del cliente deben de estar almacenados en un archivo, debidamente cifrado. Si como en el análisis de este punto en el protocolo de la Sección 4.7, el archivo es cifrado utilizando como llave el PIN, entonces se puede llevar a cabo el mismo ataque. Una forma de prevenir este ataque es mantener los datos almacenados sin formato, es decir, almacenar la información como una sola cadena.

5

Implementación de protocolos de autenticación de dos factores en dispositivos móviles

"La inteligencia consiste no sólo en el conocimiento,
sino también en la destreza de aplicar
los conocimientos en la práctica."
∼Aristóteles (384-322 a.c.) ∼

En este capítulo se presentan los detalles de la implementación de las bibliotecas criptográficas para el cálculo eficiente de emparejamientos bilineales y curvas elípticas. Así como su uso en la implementación de protocolos de autenticación de dos factores. En la Sección 5.1 se presenta la arquitectura utilizada para la implementación de los protocolos, además de la manera en que fueron realizadas las bibliotecas correspondientes; en la Sección 5.2 se presenta la implementación del protocolo "Software-only two-factor authentication" y por último en la Sección 5.3 se presenta la implementación del protocolo "M-PIN technology".

5.1. Implementación de las primitivas criptográficas

Uno de los objetivos de esta tesis es la implementación de protocolos de autenticación de dos factores en un dispositivo móvil, en particular los analizados en el Capítulo 4. Con la finalidad de lograr este objetivo, es necesario implementar de manera eficiente las primitivas criptográficas involucradas en ellos, tales como la multiplicación escalar de puntos en una curva elíptica, el emparejamiento bilineal y las funciones picadillo hacia los grupos involucrados en el emparejamiento. Debido a que estas primitivas están implementadas sobre campos finitos, es necesario que la aritmética involucrada en ellos también sea implementada eficientemente.

Como se puede observar en el Capítulo 4, los protocolos de autenticación "Software-only two-factor authentication" y "M-PIN technology" comparten la misma arquitectura, es decir, ambos protocolos necesitan de una autoridad certificadora, un servidor y un cliente. Como se ha mencionado en párrafos anteriores, el objetivo se centra en realizar la implementación de dichos protocolos utilizando como clientes dispositivos móviles, los cuales cuentan con recursos limitados, pero nada despreciables, en comparación con una computadora existente hoy en día. En la Figura 5.1 se muestra la arquitectura utilizada en la implementación de ambos protocolos.

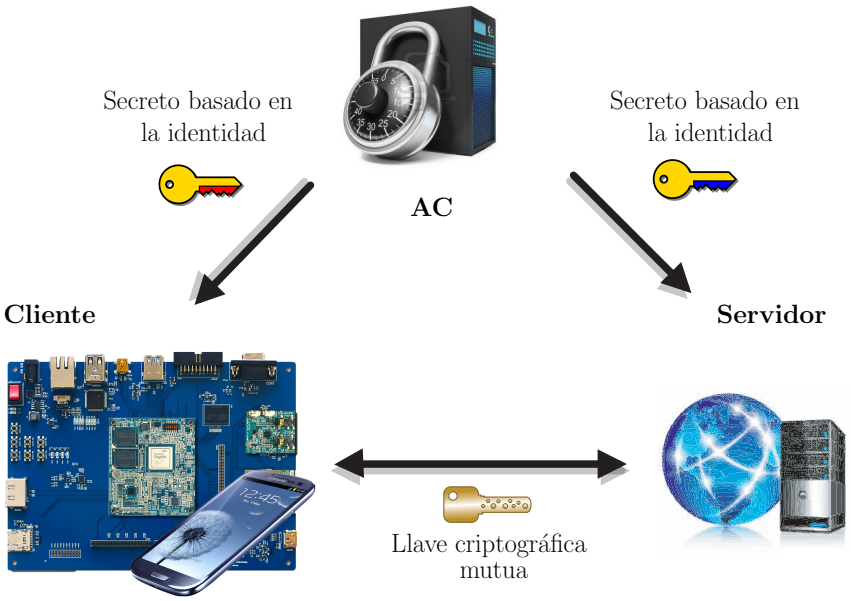


Figura 5.1: Arquitectura utilizada para la implementación de protocolos de autenticación de dos factores.

Es necesario tomar en cuenta las características de cada una de las entidades involucradas en los protocolos, por que de esta manera es posible aprovecharlas al máximo con la finalidad de lograr implementaciones eficientes. Para el caso de la arquitectura mostrada en la Figura 5.1, las características de cada entidad se presentan a continuación:

- **AC:** Las características de la autoridad certificadora no fueron tomadas en cuenta, debido a que las implementaciones están centradas en la comunicación entre los clientes y el servidor. Ademas de que se da por hecho su existencia y se supone que la fase de registro ya fue realizada.
- **Cliente:** Para el cliente se utilizó la tarjeta de desarrollo Arndale Samsung Exynos 5 Dual, que cuenta con un procesador Cortex-A15[1] de doble núcleo a 1.7 GHz., el cual implementa la arquitectura ARMv7-A utilizada en los dispositivos

[1] http://www.arm.com/products/processors/cortex-a/cortex-a15.php (7-10-2013)

móviles actuales. Uno de los puntos importantes de esta arquitectura es que el procesador trabaja con palabras de 32 bits y que además permite el uso de instrucciones NEON[2].

- **Servidor:** Para el lado del servidor se utilizó una computadora con un procesador Intel core i7-2630QM de 4 núcleos, el cual en modo normal llega a los 2.0 GHz. pudiendo alcanzar los 2.9 GHz en modo turbo, cuando sólo uno de los núcleos está activo. El tamaño de palabra con la que trabaja el procesador es de 64 bits.

En las siguientes secciones se detalla la construcción de las primitivas criptográficas y la implementación de las bibliotecas para el cliente y el servidor, las cuales toman en consideración las particularidades antes mencionadas.

5.1.1. Costo de la aritmética de la torre de campos

La implementación eficiente de la aritmética de campos finitos es fundamental para el desarrollo de esquemas basados en curvas elípticas y emparejamientos, debido a que cada una de las operaciones involucradas en la implementación de ellos está conformada por operaciones aritméticas definidas sobre un campo finito. A continuación se presentan los costos asociados a la construcción de la torre de campos, de acuerdo a los algoritmos presentados en el Apéndice B.

La construcción de la torre de campos se realizó de acuerdo a lo propuesto por Jean-Luc Beuchat *et al.* [97], para una curva elíptica BN, la cual tiene un grado de encajamiento $k = 12$, lo que permite utilizar la siguiente torre de campos, para el cálculo del emparejamiento bilineal:

$$\begin{aligned} \mathbb{F}_{p^2} &= \mathbb{F}_p[u]/(u^2-\beta), \text{ en donde } \beta = -1, \\ \mathbb{F}_{p^4} &= \mathbb{F}_{p^2}[V]/(V^2-\xi), \\ \mathbb{F}_{p^6} &= \mathbb{F}_{p^2}[V]/(V^3-\xi), \text{ en donde } \xi = u+1, \\ \mathbb{F}_{p^{12}} &= \mathbb{F}_{p^6}[W]/(W^2-\gamma), \text{ en donde } \gamma = V, \end{aligned} \tag{5.1}$$

Como se puede observar, la aritmética en $\mathbb{F}_p$ es la base de la torre de campos y por lo tanto es la parte con mayor importancia, dado que una buena implementación de la aritmética en el campo base se ve reflejada en las operaciones que se realicen sobre las capas superiores de la torre, esto se debe a que la aritmética en el campo base está directamente relacionada con la arquitectura del procesador para el cual se va a realizar la implementación, en especial con el tamaño de palabra, y con el tamaño en bits del primo p.

En particular, la multiplicación es la operación más relevante en la implementación de los emparejamientos bilineales y sirve de base para comparar costos en cada uno de

[2]http://www.arm.com/products/processors/technologies/neon.php (7-10-2013)

los campos. Esta operación es la más utilizada, por lo que el costo de las operaciones aritméticas en las capas superiores depende directamente de su eficiencia. Dado que la suma y la resta son operaciones muy baratas, por lo general no suelen ser tomadas en cuenta, por otro lado, el inverso multiplicativo y la raíz cuadrada son operaciones que no se ejecutan frecuentemente por lo que sus costos tampoco son representativos.

A continuación se presentan las características de cada una de las extensiones de campo que compone la torre, considerando que (a, m, s, i), $(\tilde{a}, \tilde{m}, \tilde{s}, \tilde{i})$ y (A, M, S, I) representan el costo de una suma, multiplicación, elevación al cuadrado e inversión en el campo $\mathbb{F}_p$, $\mathbb{F}_{p^2}$ y $\mathbb{F}_{p^6}$, respectivamente:

- $\mathbb{F}_{p^2}$: En este campo se utilizó para la multiplicación el método de Karatsuba y para la elevación al cuadrado el método complejo, a un costo de 3 y 2 multiplicaciones en $\mathbb{F}_p$, respectivamente. El cálculo del inverso de un elemento $a = a_0 + a_1 u \in \mathbb{F}_{p^2}$, puede ser realizado por medio de la identidad $(a_0 + a_1 u)^{-1} = (a_0 - a_1 u)/(a_0^2 + \beta a_1^2)$.

- $\mathbb{F}_{p^4}$: El objetivo de la construcción de este campo es el de apoyar en el calculo de la elevación al cuadrado en el grupo ciclotómico. La operación considerada en el campo $\mathbb{F}_{p^4}$ es la elevación al cuadrado utilizando Karatsuba la cual tiene un costo de $3\tilde{s}$.

- $\mathbb{F}_{p^6}$: Usando nuevamente Karatsuba para la multiplicación en $\mathbb{F}_{p^6}$, ésta puede ser calculada con un costo de $6\tilde{m}$ más algunas operaciones de adición. La operación de elevar al cuadrado puede ser calculada mediante la formula descrita en [98] a un costo de $2\tilde{m} + 3\tilde{s}$ más algunas operaciones de adición. Mientras que la inversión puede ser calculada con un costo de $9\tilde{m} + 3\tilde{s} + 4m_\xi + 5\tilde{a} + \tilde{i}$.

- $\mathbb{F}_{p^{12}}$: Dado que este campo fue construido como una extension cuadrática de $\mathbb{F}_{p^6}$, los costos de la aritmética son parecidos a los del campo $\mathbb{F}_{p^2}$. Por lo que, una multiplicación, elevada al cuadrado e inversión en $\mathbb{F}_{p^{12}}$ tienen un costo de: $3M+5A$, $2M+5A$ y $2M+2S+2A+I$, respectivamente. Sin embargo, si $f \in \mathbb{F}_{p^{12}}$ pertenece al grupo ciclotómico $\mathbb{G}_{\phi_6}(\mathbb{F}_{p^2})$, la operación de elevar al cuadrado f^2 puede reducirse a tres elevadas al cuadrado en $\mathbb{F}_{p^4}$.

Algunas de las operaciones mencionadas requieren de la multiplicación, en el campo base, por el coeficiente constante $\beta \in \mathbb{F}_p$ del polinomio irreducible $u^2 - \beta$, denotada como m_β. Otras requieren calcular la multiplicación de un elemento arbitrario en $\mathbb{F}_{p^2}$ por la constante $\xi = u + 1 \in \mathbb{F}_{p^2}$, lo cual tiene un costo de una multiplicación por la constante β y se denota como m_ξ. Sin embargo, el costo de m_ξ es esencialmente el de m_β.

En la Tabla 5.1 se presentan los costos computacionales de la aritmética de la torre de campos en términos de las operaciones aritméticas en $\mathbb{F}_{p^2}$, denotadas como $(\tilde{a}, \tilde{m}, \tilde{s}, \tilde{i})$.

Campo	Adición/ Sustracción	Multiplicación	Cuadrado	Inversión
$\mathbb{F}_{p^2}$	$\tilde{a} = 2a$	$\tilde{m} = 3m + 5a + m_\beta$	$\tilde{s} = 2m + 3a + m_\beta$	$\tilde{i} = 4m + m_\beta + 2a + i$
$\mathbb{F}_{p^4}$	$2\tilde{a}$		$3\tilde{s} + 1m_\xi + 4\tilde{a}$	
$\mathbb{F}_{p^6}$	$3\tilde{a}$	$6\tilde{m} + 2m_\xi + 15\tilde{a}$	$2\tilde{m} + 3\tilde{s} + 2m_\xi + 9\tilde{a}$	$9\tilde{m} + 3\tilde{s} + 4m_\xi + 5\tilde{a} + \tilde{i}$
$\mathbb{F}_{p^{12}}$	$6\tilde{a}$	$18\tilde{m} + 6m_\xi + 60\tilde{a} + m_\gamma$	$12\tilde{m} + 4m_\xi + 45\tilde{a} + 2m_\gamma$	$25\tilde{m} + 9\tilde{s} + 12m_\xi + 61\tilde{a} + \tilde{i} + m_\gamma$
$\mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$			$9\tilde{s} + 4m_\xi + 30\tilde{a}$	Conjugación

Tabla 5.1: Resumen de costos de la aritmética de torre de campos.

5.1.2. Costo de la aritmética de curvas elípticas

Los grupos de orden primo r necesarios para el cálculo del emparejamiento bilineal en la familia de curvas BN pueden ser definidos de la siguiente manera:

Sea $E : y^2 = x^3 + b$ una curva elíptica definida sobre un campo finito $\mathbb{F}_q$, con característica $q = p^n$, en donde p es un número primo, $n \in \mathbb{Z}^+$ y $b \in \mathbb{F}_q$. Los grupos $\mathbb{G}_1$, $\mathbb{G}_2$ y $\mathbb{G}_T$ se definen como:

$$\begin{aligned} \mathbb{G}_1 &= E(\mathbb{F}_p)[r] \\ \mathbb{G}_2 &= E(\mathbb{F}_{p^2})[r] \\ \mathbb{G}_T &= \mathbb{F}^*_{p^{12}} \end{aligned}$$

Por el momento utilizaremos los grupos aditivos $\mathbb{G}_1$ y $\mathbb{G}_2$, las operaciones en el grupo $\mathbb{G}_t$ serán abordadas más adelante. Para el caso de las operaciones de suma y doblado de puntos, los costos asociados a ellas se presentan en la Tabla 5.2, mientras que los costos aproximados para el cálculo de la multiplicación escalar utilizando los métodos ω-NAF, GLV y GS para un escalar 256 bits se presentan en la Tabla 5.3, en donde A, D e I, representan la suma, el doblado y la Inversión de puntos, respectivamente.

Los costos presentados en las tablas se establecieron de acuerdo a los algoritmos explicados en el Apéndice B en la Sección B.2.

5.1.3. Costo del cálculo del emparejamiento óptimo ate

Para calcular el costo del emparejamiento, se consideran los costos asociados a las funciones que lo componen, es decir, el ciclo de Miller y la exponenciación final. Para estimar los costos del ciclo de Miller, de acuerdo al Algoritmo B.35, se deben obtener los costos de las operaciones involucradas en él, las cuales se presentan en la Tabla 5.4.

Grupo	Doblado de Puntos	Suma de Puntos	Inverso Aditivo
$\mathbb{G}_1$	$7m + 10a$	$11m + 7a$	a
$\mathbb{G}_2$	$3\tilde{m} + 4\tilde{s} + 10\tilde{a}$	$5\tilde{m} + 6\tilde{s} + 7\tilde{a}$	$\tilde{a}$

Tabla 5.2: Resumen de costos de las operaciones de suma y doblado de puntos en los grupos $\mathbb{G}_1$ y $\mathbb{G}_2$

Grupo	Método	Costo Funcional	Costo Operacional
$\mathbb{G}_1$	ω-NAF ($\omega = 3$)	$65A + 257D$	$2514m + 3025a$
	GLV ($\omega = 3$)	$66A + 130D$	$1636m + 1762a$
$\mathbb{G}_2$	ω-NAF ($\omega = 3$)	$65A + 257D$	$1096\tilde{m} + 1418\tilde{s} + 3025\tilde{a}$
	GS ($\omega = 3$)	$68A + 68D$	$544\tilde{m} + 680\tilde{s} + 1156\tilde{a}$

Tabla 5.3: Resumen de costos de la operación de multiplicación escalar en los grupos $\mathbb{G}_1$ y $\mathbb{G}_2$

Operación	Notación	Costo
Doblado de punto y evaluación de la línea tangente	$2T$ y $\ell_{T,T}(P)$	$3\tilde{m} + 6\tilde{s} + 17\tilde{a} + 4m$
Suma de punto y evaluación de la línea secante	$T + Q$ y $\ell_{T,Q}(P)$	$11\tilde{m} + 2\tilde{s} + 19\tilde{a} + 4m$
Multiplicación dispersa	$f \cdot \ell_{\_,\_}(P)$	$13\tilde{m} + 29\tilde{a} + 3m_\xi$
Operador de Frobenius en Q	$\pi(Q)$ o $\pi^2(Q)$	$5\tilde{m}$

Tabla 5.4: Resumen de costos de las operaciones del ciclo de Miller

El valor de z utilizado para obtener los parámetros p, r y t de la curva elíptica BN, de acuerdo a la Sección 2.3.3.1, es $z = -2^{62} - 2^{55} - 1$. Esto provoca que en la línea 1 del Algoritmo B.35, el valor de $s = 6z+2$, este dado por $s = -(2^{64}+2^{63}+2^{57}+2^{56}+2^2)$ que es un número negativo de 65 bits y cuyo peso de Hamming es de 5 bits. Con el fin de evitar el signo de s en el cálculo del ciclo de Miller, se realiza el procedimiento sin tomar en cuenta el signo y al final se computa el conjugado del resultado obtenido. A continuación se muestra el costo del ciclo de Miller, con base en el número de operaciones utilizadas.

$$
\begin{aligned}
\text{Ciclo de Miller} \quad = 63 \quad & (\text{Elevaciones al cuadrado en } \mathbb{F}_{p^{12}} + \\
& \text{Multiplicaciones dispersas} + \\
& \text{Evaluaciones de línea y doblado de punto}) + \\
5 \quad & (\text{Multiplicaciones dispersas} + \\
& \text{Evaluaciones de línea y suma de puntos}) + \\
2 \quad & (\text{Operadores Frobenius} + \text{Multiplicaciones dispersas} + \\
& \text{Evaluaciones de línea y suma de puntos })+ \\
1 \quad & \text{Conjugado en } \mathbb{F}_{p^{12}}.
\end{aligned}
$$

A partir de la Tabla 5.4 y la Tabla 5.1 es posible obtener el costo total del ciclo de Miller en términos de las operaciones en $\mathbb{F}_{p^2}$:

$$\textbf{Ciclo de Miller} \quad = 1942\tilde{m} + 392\tilde{s} + 6072\tilde{a} + 280m + 462m_{\xi} + 126m_{\gamma}.$$

Si se conoce el punto en el grupo $\mathbb{G}_2$, la evaluación de las líneas se realizan con $4m$ y no es necesario el cálculo operadores Frobenius, quedando el costo del ciclo de Miller como $1666\tilde{m} + 4868\tilde{a} + 280m + 462m_{\xi} + 126m_{\gamma}$.

Para obtener el costo de la exponenciación final, se sigue la metodología anterior planteando el número de operaciones utilizadas en el Algoritmo B.38, para posteriormente hacer el cálculo de las operaciones necesarias en el campo $\mathbb{F}_{p^2}$. La exponenciación final requiere la siguiente cantidad de operaciones:

$$
\begin{aligned}
\text{Exponenciación Final} \quad = \quad & 3 \text{ exponenciaciones por } z \text{ en } \mathbb{F}_{p^{12}} + \\
& 12 \text{ multiplicaciones en } \mathbb{F}_{p^{12}} + \\
& 3 \text{ elevaciones al cuadrado en } \mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2}) + \\
& 4 \text{ operadores de Frobenius} + \\
& 1 \text{ inverso en } \mathbb{F}_{p^{12}} + 3 \text{ conjugados en } \mathbb{F}_{p^{12}}.
\end{aligned}
$$

El cálculo de las exponenciaciones por $z = -(2^{62} + 2^{55} + 1)$ fue realizado mediante el Algoritmo B.42. Dado que z tiene un valor negativo, el cálculo se realiza tomando el valor absoluto de z y aplicando un conjugado al final. En la Tabla 5.5 se muestran los costos de las operaciones necesarias para el calculo de cuadrados comprimidos (Sección B.1.6), necesarios para la exponenciación por z.

Operación	**Notación**	**Costo**
Cuadrado Comprimido	$\mathcal{C}(\cdot)$	$6\tilde{s} + 24\tilde{a} + 3m_{\xi}$
Descompresión	$\mathcal{D}(\cdot)$	$3\tilde{m} + 3\tilde{s} + 12\tilde{a} + \tilde{i} + 2m_{\xi}$

Tabla 5.5: Resumen de costos de las operaciones de la exponenciación por z

La línea 15 del Algoritmo B.42, requiere dos descompresiones $\mathcal{D}(\cdot)$, sin embargo, utilizando el truco de Montgomery para inversiones simultáneas [99], ambas descompresiones tienen un costo de $9\tilde{m}+6\tilde{s}+24\tilde{a}+\tilde{i}+4m_{\xi}$. Finalmente, el algoritmo requiere dos multiplicaciones en el campo $\mathbb{F}_{p^{12}}$, por lo que el costo de una exponenciación por z queda como se muestra a continuación:

$$\begin{aligned}\text{Exponenciación por } z &= 62\ (6\tilde{s} + 24\tilde{a} + 3m_\xi) + 2(18\tilde{m} + 60\tilde{a} + 6m_\xi + m_\gamma) + \\ &\quad (9\tilde{m} + 6\tilde{s} + 24\tilde{a} + \tilde{i} + 4m_\xi) \\ &= 45\tilde{m} + 378\tilde{s} + 1632\tilde{a} + 202m_\xi + 2m_\gamma + \tilde{i}.\end{aligned}$$

En base a los Algoritmos B.39, B.40 y B.41 se puede decir que el costo computacional de aplicar los operadores de Frobenius es de $5\tilde{m}$, sin embargo, de acuerdo a los parámetros seleccionados el costo real es de $5\tilde{m} + \tilde{a}$, $8m + \tilde{a}$ y $3\tilde{m} + 4\tilde{a}$, respectivamente. A continuación se presenta el costo de la exponenciación final, a partir de la Tabla 5.4 y la Tabla 5.1:

$$\textbf{Exponenciación Final } = 384\tilde{m} + 1170\tilde{s} + 5776\tilde{a} + 4\tilde{i} + 8m + 702m_\xi + 19m_\gamma.$$

Ahora ya se cuenta con la información necesaria para calcular el costo del emparejamiento, con base en los resultados obtenidos del ciclo de Miller y la exponenciación final, el cual es de:

$$\hat{e} = 2326\tilde{m} + 1562\tilde{s} + 11848\tilde{a} + 4\tilde{i} + 288m + 1164m_x i + 145m_\gamma.$$

Por otro lado, cuando se conoce un punto en $\mathbb{G}_2$ el costo del emparejamiento se reduce a $2050\tilde{m} + 1170\tilde{s} + 10644\tilde{a} + 288m + 1164m_\xi + 145m_\gamma$.

5.1.4. Biblioteca criptográfica para el servidor

La implementación de la biblioteca criptográfica, utilizada en el servidor, fue desarrollada tomando como base la biblioteca realizada por Mitsunari Shigeo y Teruya Tadanori[3] basada en el trabajo de Beuchat *et al.* [97]. Cabe mencionar que de esta biblioteca se utilizaron sólo las operaciones relacionadas con la torre de campos. La biblioteca se escribió en el lenguaje de programación C++, para el conjunto de instrucciones x86-64 y con la finalidad de mejorar su rendimiento los autores hicieron uso de Xbyak[4], un ensamblador x86/x64 *just-in-time* para el lenguaje C++.

Como se ha mencionado, la implementación eficiente de la aritmética en el campo base de la torre de campos en especial el multiplicador, es fundamental para obtener una implementación eficiente de los algoritmos que se construyen sobre ella. Es por eso que acontinuación se muestran algunos detalles de su implementación:

El conjunto de instrucciones x86-64 cuenta con la operación *mul*, la cual multiplica dos enteros sin signo produciendo un entero sin signo de 128 bits, esta operación toma cerca de 3 ciclos en un procesador Intel Core i7. La brecha entre el costo de una multiplicación y el costo de una adición o sustracción, en términos de ciclos es mucho muy pequeña, en comparación con arquitecturas previas. Por lo que se debe tener cuidado al elegir los algoritmos que implementan la aritmética en $\mathbb{F}_p$, por ejemplo: En el caso de operandos de 256 bits, la multiplicación realizada utilizando el método clásico es más rápida que utilizando el método de Karatsuba.

[3] https://github.com/herumi/ate-pairing
[4] http://homepage1.nifty.com/herumi/soft/xbyak_e.html

Como se muestra en el Apéndice B.1, un elemento $x \in \mathbb{F}_p$ se representa como $x = (x_0, x_1, x_2, x_3)$, en donde x_i, $0 \leq i \leq 3$ son enteros de 64 bits. La adición y sustracción en este campo son realizadas de manera sencilla, se suman los operandos y al final se realiza la reducción al campo $\mathbb{F}_p$, por otra parte las operaciones de multiplicación e inversión son realizadas de acuerdo a los Algoritmos B.3 y B.5, respectivamente.

Los autores de esta biblioteca mejoraron algunas de las operaciones de la aritmética en $\mathbb{F}_{p^2}$, aprovechando las características del procesador, por ejemplo:

- **Multiplicación:** Fue implementada usando la multiplicación de Montgomery dividida en dos pasos: la multiplicación de dos enteros de 256 bits, produciendo un resultado de 512 bits, denotada como *mul256*; y la reducción de Montgomery del entero de 512 bits a uno de 256 bits, denotada como *mod512*. De acuerdo al Algoritmo B.9, se deben calcular tres multiplicaciones sobre $\mathbb{F}_p$ con sus respectivas reducciones. Sin embargo, se pueden mantener los resultados de las líneas 1, 2 y 4 en tres valores temporales de 512 bits. Estos valores podemos sumarlos o restarlos sin reducción, y sólo aplicar la operación *mod512* para obtener los valores de c_0 y $c_1 \in \mathbb{F}_p$, como muestra el Algoritmo 5.1. Las funciones *addNC/subNC*, de adición y sustracción, aplicadas a enteros de 256 o 512 bits, no realizan la comprobación del acarreo de salida.

Algoritmo 5.1 Multiplicación en $\mathbb{F}_{p^2}$ optimizada

Entrada: $A = a_0 + a_1 u, \quad B = b_0 + b_1 u,\ A, B \in \mathbb{F}_{p^2}$
Salida: $C = A \cdot B,\ C \in \mathbb{F}_{p^2}$
1: $s \leftarrow addNC(a_0, a_1)$
2: $t \leftarrow addNC(b_0, b_1)$
3: $d_0 \leftarrow mul256(s, t)$
4: $d_1 \leftarrow mul256(a_0, b_0)$
5: $d_2 \leftarrow mul256(a_1, b_1)$
6: $d_0 \leftarrow subNC(d_0, d_1)$
7: $d_0 \leftarrow subNC(d_0, d_2)$
8: $c_1 \leftarrow mod512(d_0)$
9: $d_1 \leftarrow d_1 - d_2$
10: $c_0 \leftarrow mod512(d_1)$
11: **return** $C = c_0 + c_1 u$

La operación de adición y sustracción de los elementos $x, y \in \mathbb{F}_p$, incluye la comprobación de $x + y \geq p$ o $x < y$, la cual es costosa y es conveniente evitarla cuanto sea posible. Afortunadamente, la selección del primo p satisface $6p < N$, en donde $N = 2^{256}$, y la función *mod512* puede reducir el operando x, cuando $x < pN$. Esto implica que se puede sumar seis veces sin comprobar el acarreo de salida, por ejemplo: en la línea 8 del algoritmo, d_0 es igual a $(a_0 + a_1)(b_0 + b_1) - a_0 b_0 - a_1 b_1 = a_0 b_1 + a_1 b_0 < 2p^2 < pN$.

En la línea 9, la operación $d_1 = a_0b_0 - a_1b_1$ es realizada como una sustracción de enteros de 512 bits con acarreo, considerando $x = a_0b_0 - a_1b_1$ y un entero t de 512 y 256 bits respectivamente, la operación antes mencionada se puede llevar a cabo de la siguiente manera: si $x < 0$ entonces $t = p$, otra cosa $t = 0$, entonces $d_1 = x + tN$, en donde la operación de adición sólo utiliza los 256 bits de más significativos de x.

- **Elevación al cuadrado:** El Algoritmo 5.2 muestra la optimización realizada, en base a las ideas planteadas en el párrafo anterior. Las operaciones en las líneas 2, 3 y 4 son realizadas en el campo $\mathbb{F}_p$.

Algoritmo 5.2 Elevación al Cuadrado en $\mathbb{F}_{p^2}$ optimizada

Entrada: $A = a_0 + a_1u$, $A \in \mathbb{F}_{p^2}$
Salida: $C = A^2$, $C \in \mathbb{F}_{p^2}$
1: $t_1 \leftarrow addNC(a_1, a_1)$
2: $c_1 \leftarrow t_1 * a_0$
3: $t_1 \leftarrow a_0 - a_1$
4: $t_2 \leftarrow addNC(a_0, a_1)$
5: $c_0 \leftarrow t_1 * t_2$
6: **return** $C = c_0 + c_1u$

En la Tabla 5.6 se presentan los costos asociados a la aritmética de torre de campos en la biblioteca utilizada por el servidor, de acuerdo con los algoritmos de la Sección B.1 y las optimizaciones antes mencionadas.

La implementación de las operaciones involucradas en la aritmética de curvas elípticas y en el cálculo del emparejamiento bilineal fueron realizadas a partir de los algoritmos descritos en la Sección B.2 y en la Sección B.3, respectivamente. En las Tablas 5.7 y 5.8 se muestran los costos asociados a estas operaciones.

5.1.5. Biblioteca criptográfica para el cliente

La implementación de la biblioteca criptográfica, utilizada en el cliente, fue hecha tomando en su mayoría la biblioteca realizada por Ana Helena Sánchez [100]. A esta biblioteca se agregaron las operaciones necesarias para el cálculo de las funciones picadillo deterministas (Capítulo 3) y las operaciones en el campo $\mathbb{F}_r$, en donde r corresponde al orden de la curva elíptica utilizada. La biblioteca fue escrita en el lenguaje de programación C, para el conjunto de instrucciones del procesador ARM Cortex A-15 y con la finalidad de mejorar su rendimiento hace uso de la tecnología NEON.

La tecnología NEON es una extensión para la serie de procesadores ARM Cortex™-A, con una arquitectura de 128 bits que trabaja sobre el modelo SIMD (Single Instruction, Multiple Data). Esta arquitectura consta de 32 registros de 64 bits (*doubleword*), los cuales pueden verse como 16 registros de 128 bits (*quadword*). Las instrucciones

Campo	Operación	Ciclos de reloj ($\times 10^3$)
$\mathbb{F}_{p^2}$	adición	0.021
	sustracción	0.020
	multiplicación	0.252
	cuadrado	0.206
	inversión	3.656
$\mathbb{F}_{p^6}$	adición	0.055
	sustracción	0.057
	multiplicación	1.561
	cuadrado	1.427
	inversión	7.302
$\mathbb{F}_{p^{12}}$	adición	0.108
	sustracción	0.106
	multiplicación	4.733
	cuadrado	3.457
	inversión	13.41
$\mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$	cuadrado	2.403
	inversión	0.086

Tabla 5.6: Costos de la aritmética de torre de campos de la biblioteca implementada en el procesador Intel core i7-2630QM a 2.0 GHz.

Grupo	Operación	Método	Ciclos de reloj ($\times 10^3$)
$\mathbb{G}_1$	Doblado de puntos	Jacobianas	0.868
	Suma de puntos	Mixtas	1.115
	Multiplicación Escalar	ω-NAF	286
		GLV	195
$\mathbb{G}_2$	Doblado de puntos	Jacobianas	1.912
	Suma de puntos	Mixtas	2.965
	Multiplicación Escalar	ω-NAF	639
		GLS	387
$\mathbb{G}_T$	Exponenciación	GS	651

Tabla 5.7: Costos de la aritmética de curvas elípticas de la biblioteca implementada en el procesador Intel core i7-2630QM a 2.0 GHz.

NEON realizan un procesamiento SIMD comprimido, de manera que: los registros son vistos como vectores, cuyos elementos son de un mismo tipo de dato; Los tipos de datos son de 8, 16, 32 y 64 bits, con signo o sin signo; las instrucciones ejecutan la misma operación sobre cada uno de los elementos de los vectores al mismo tiempo.

Las principales operaciones utilizadas en la biblioteca son las correspondientes a las instrucciones *vmull_u32* y *vmlal_u32*, las cuales permiten realizar dos multiplica-

Operación	**Ciclos de reloj** ($\times 10^3$)
Ciclo de Miller	715
Exponenciación Final	434
Emparejamiento	1132

Tabla 5.8: Costos del emparejamiento óptimo ate implementado en el procesador Intel core i7-2630QM a 2.0 GHz.

ciones enteras de 32 bits almacenando el resultado en registros de 64 bits, esta última instrucción, además, permite sumar un entero de 64 bits a cada multiplicación. Cabe señalar, que la carga y el almacenamiento de las variables suele ser muy costoso, dado que implica pasar de la arquitectura NEON a la arquitectura ARM y para realizar esta operación debe buscarse un conjunto de registros consecutivos de 32 bits que puedan almacenar los 64/128 bits de los registros de la arquitectura NEON. Por lo que para aprovechar realmente las instrucciones NEON es conveniente evitar el uso compartido de variables entre ambas arquitecturas.

Como se muestra en el Apéndice B.1 en la página 113, un elemento $x \in \mathbb{F}_p$, en una arquitectura de 32 bits, se representa como $x = (x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7)$, en donde x_i, $0 \leq i \leq 7$ son enteros de 32 bits. La adición y sustracción en este campo son realizadas de manera sencilla, se suman los operandos y al final se realiza la reducción al campo $\mathbb{F}_p$, por otra parte la operación inversión es realizada de acuerdo al Algoritmo B.5, mientras que la multiplicación es realizada de la siguiente manera:

- mul_{NEON}: Esta función realiza dos multiplicaciones independientes en el campo $\mathbb{F}_p$ utilizando el método CIOS [101]. Recibe como entrada 4 elementos y entrega como salida 2 elementos en $\mathbb{F}_p$.
- $mule_{NEON}$: Realiza dos multiplicaciones enteras, recibe como entrada 4 elementos de 256 bits y obtiene como salida dos elementos de 512 bits.
- red_{NEON}: Calcula la reducción módulo p de acuerdo al método SOS [101]. La función recibe como entrada 2 argumentos de 512 bits y regresa como salida 2 elementos en el campo $\mathbb{F}_p$.

Estas funciones fueron utilizadas en los algoritmos de la aritmética para $\mathbb{F}_{p^2}$, al igual que la técnica descrita en la multiplicación en la sección anterior. En el Algoritmo 5.3 y en el Algoritmo 5.4 se muestran las operaciones de multiplicación y elevación al cuadrado en $\mathbb{F}_{p^2}$, utilizando las funciones descritas. También fueron utilizadas en estas mismas operaciones sobre las distintas extensiones de campo y en el cálculo de la suma y doblado de puntos en $\mathbb{G}_1$.

En la Tabla 5.9 se presentan los costos asociados a la aritmética de torre de campos en la biblioteca utilizada por el cliente, de acuerdo con los algoritmos de la Sección

Algoritmo 5.3 Multiplicación optimizada en $\mathbb{F}_{p^2}$

Entrada: $A = a_0 + a_1 u, \quad B = b_0 + b_1 u \in \mathbb{F}_{p^2}$
Salida: $C = A \cdot B \in \mathbb{F}_{p^2}$
1: $s \leftarrow a_0 + a_1$
2: $t \leftarrow b_0 + b_1$
3: $(d_0, d_1) \leftarrow mule_{NEON}(s, t, a_0, b_0)$
4: $d_2 \leftarrow mul256(a_1, b_1)$
5: $d_0 \leftarrow d_0 - d_1 - d_2$
6: $d_1 \leftarrow d_1 - d_2$
7: $(c_1, c_0) \leftarrow red_{NEON}(d_0, d_1)$
8: **return** $C = c_0 + c_1 u$

Algoritmo 5.4 Elevación al cuadrado en $\mathbb{F}_{p^2}$

Entrada: $A = a_0 + a_1 u \in \mathbb{F}_{p^2}$
Salida: $C = A^2 \in \mathbb{F}_{p^2}$
1: $c_0 \leftarrow a_0 - a_1$
2: $c_2 \leftarrow a_0 + a_1$
3: $(c_1, c_0) \leftarrow mul_{NEON}(a_0, a_1, c_0, c_2)$
4: $c_1 \leftarrow 2c_1$
5: $c_0 \leftarrow c_0 - c_1$
6: **return** $C = c_0 + c_1 u$

B.1en la página 113 y las optimizaciones antes mencionadas. En las Tablas 5.10 y 5.11 se muestran los costos asociados a las operaciones involucradas en la aritmética de curvas elípticas y en el cálculo del emparejamiento bilineal.

5.2. Costo de la implementación del protocolo "Software-only two-factor authentication"

En esta sección se presentan los costos asociados a las fases involucradas en el protocolo de autenticación "Software-only two-factor authentication", correspondientes al proceso de autenticación, el cambio de PIN y la recuperación del PIN olvidado. La fase de registro de usuarios no es tomada en cuenta, dado que se realiza fuera de línea.

En la Tabla 5.12 se presenta el resumen de costos, de acuerdo a las operaciones realizadas en cada fase, tanto para el cliente como para el servidor, en donde:

- H_1 y H_2: Representan las funciones picadillo deterministas a $\mathbb{G}_1$ y $\mathbb{G}_2$, respectivamente.
- M_{G1}, M_{G2} y M_{PIN}: Las dos primeras son las operaciones de multiplicación escalar en los grupos $\mathbb{G}_1$ y $\mathbb{G}_2$, respectivamente. Mientras que la última representa

Campo	Operación	Ciclos de reloj ($\times 10^3$)
$\mathbb{F}_{p^2}$	adición	0.011
	sustracción	0.013
	multiplicación	1.342
	cuadrado	0.862
	inversión	28.91
$\mathbb{F}_{p^6}$	adición	0.039
	sustracción	0.037
	multiplicación	8.324
	cuadrado	6.530
	inversión	37.072
$\mathbb{F}_{p^{12}}$	adición	0.92
	sustracción	0.87
	multiplicación	30.847
	cuadrado	16.805
	inversión	70.95
$\mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$	cuadrado	8.086
	inversión	0.046

Tabla 5.9: Costos de la aritmética de torre de campos de la biblioteca implementada en el procesador Cortex-A15 a 1.7 GHz.

Grupo	Operación	Método	Ciclos de reloj ($\times 10^3$)
$\mathbb{G}_1$	Doblado de puntos	Jacobianas	3.34
	Suma de puntos	Mixtas	4.33
	Multiplicación Escalar	ω-NAF	1,234
		GLV	814
$\mathbb{G}_2$	Doblado de puntos	Jacobianas	8.65
	Suma de puntos	Mixtas	13.29
	Multiplicación Escalar	ω-NAF	3,255
		GLS	1,675
$\mathbb{G}_T$	Exponenciación	GS	2,742

Tabla 5.10: Costos de la aritmética de curvas elípticas de la biblioteca implementada en el procesador Cortex-A15 a 1.7 GHz.

la multiplicación escalar por el PIN, esta operación es mucho menos costosa que M_{G1} y M_{G2}, dado el tamaño en bits del PIN. Estas operaciones fueron implementadas de acuerdo a los métodos GLV, GS y ω-NAF, respectivamente.

- A_{G1} y A_{G2}: Representan las operaciones de suma de puntos en los grupos $\mathbb{G}_1$ y $\mathbb{G}_2$, respectivamente. Ambas operaciones fueron realizadas en coordenadas mixtas.

Operación	**Ciclos de reloj** ($\times 10^3$)
Ciclo de Miller	3,728
Exponenciación Final	2,469
Emparejamiento	6,261

Tabla 5.11: Costos del emparejamiento óptimo ate implementado en el procesador Cortex-A15 a 1.7 GHz.

- e: Se refiere a la operación de emparejamiento bilineal.
- H_x: Representa las funciones picadillo: $H_{r1}(\cdot)$ y $H_{r2}(\cdot)$ que realizan la proyección de la concatenación de los puntos en $\mathbb{G}_1$ y $\mathbb{G}_2$ a un número en el rango de 1 a $r-1$; $H_k(\cdot)$ que realiza la proyección de la concatenación del resultado del emparejamiento y un punto en $\mathbb{G}_1$ a un número en el rango de 1 a $r-1$; y $H_r(\cdot)$ que realiza la proyección de la concatenación de las identidades del cliente y el servidor con la llave criptográfica a un número en el rango de 1 a $r-1$.

  Para la implementación de H_x se realizó la concatenación de la representación hexadecimal de los operandos, a este resultado se le aplicó el algoritmo SHA256 y finalmente se realizó la reducción al rango $[1, r-1]$.

Fase	**Cliente**	**Servidor**
Autenticación	$H_1 + H_2 + 3M_{G1} + M_{G2} +$ $A_{G1} + A_{G2} + M_{PIN} + e + 4H_x$	$H_1 + H_2 + 3M_{G1} + 2M_{G2} +$ $A_{G1} + e + 4H_x$
Cambio de PIN	$H_1 + 2M_{PIN} + 2A_{G1}$	-
Recuperación de PIN	$e + H_1 + H_2 + 2M_{PIN} +$ $2A_{G1} + H_x$	$H_1 + H_2 +$ $i(e + M_{PIN} + A_{G2} + H_x)$

Tabla 5.12: Costos de la implementación del protocolo "Software-only two-factor authentication".

En la tabla anterior el costo del cliente correspondiente al cambio de PIN, difiere de las operaciones presentadas en la Sección 4.7.1.3. Esto es por la manera en que se realizó la implementación, por que se consideró necesario verificar que el usuario es realmente quien dice ser, comprobando que se cumpliera la igualdad $sC = Token + PIN \cdot C$ la cual tiene un costo de $H_1 + A_{G1} + M_{PIN}$.

En la Tabla 5.13 se presentan los costos del protocolo, en ella sólo se considera el costo del proceso de autenticación, dado que es el único que se se realiza en línea. La fase de recuperación de PIN no se considera en línea, debido a que la respuesta del servidor al cliente se realiza por un medio diferente, por ejemplo: correo electrónico. Además de que en la literatura sólo se encontraron costos de la fase de autenticación.

Trabajo	Cliente		Servidor
Michael Scott [17]	Intel i5-520M a 2.4 GHz.		
	4.10 ms		4.48 ms
Este trabajo	Intel i7	Cortex-A15	Intel i7
	1.521 ms	7.10 ms	1.519 ms

Tabla 5.13: Resumen de tiempos de la implementación del protocolo "Software-only two-factor authentication".

En la tabla se puede apreciar una comparación de los resultados obtenidos en [17] y los alcanzados en este trabajo de tesis, la cual muestra que la implementación en el procesador Intel i7 es un tanto más eficiente. Además se puede observar que el cliente en el procesador Cortex A-15 toma casi el doble que lo reportado en [17], considerando las características de ambos procesadores se puede concluir que la implementación de este tipo de protocolos es una opción viable.

5.3. Costo de la implementación del protocolo "M-PIN"

En esta sección se presentan los costos asociados a las fases involucradas en el protocolo de autenticación "M-PIN", correspondientes al proceso de autenticación, el cambio de PIN y la recuperación del PIN olvidado. Nuevamente la fase de registro de usuarios no es tomada en cuenta, dado que se realiza fuera de línea.

En la Tabla 5.14 se presenta el resumen de costos, de acuerdo a las operaciones realizadas en cada fase, tanto para el cliente como para el servidor. La notación utilizada en la tabla se explica a continuación:

- H_1, H_2, M_{G1}, M_{G2}, M_{PIN}, A_{G1}, A_{G2} y e: representan las mismas operaciones que en la sección anterior.
- M_{GT}: Representa la multiplicación en el campo $\mathbb{F}_{p^{12}}$, esta operación se realizó de acuerdo al algoritmo mostrado en la Sección B.1.
- E_{GT}: Denota la operación de exponenciación en el campo $\mathbb{F}_{p^{12}}$, la cual fue realizada utilizando el método GS (Sección B.2).
- H_x: Representa la funciones picadillo: $H_g(\cdot)$ que realiza la proyección del resultado del emparejamiento a un número en el rango de 1 a $r-1$. Esta operación fue realizada de la misma forma que en la sección anterior.

En laTabla 5.14 el costo del cliente correspondiente al cambio de PIN, difiere de las operaciones presentadas en la Sección 4.7.1.3. Esto es por la manera en que se realizó la

Fase	Cliente	Servidor
Autenticación	$H_1 + 3M_{G1} +$ $A_{G1} + M_{PIN} + E_{GT} + H_x$	$H_1 + H_2 + 4M_{G1} +$ $A_{G1} + 3e + M_{GT} + H_x$
Cambio de PIN	$H_1 + 2M_{PIN} + 2A_{G1}$	-
Recuperación de PIN	$H_1 + 2M_{G1} + M_{PIN} +$ A_{G1}	$H_1 + M_{G1} + 2A_{G1} +$ $iM_{PIN} + (i+1)e + iM_{GT}$

Tabla 5.14: Costos de la implementación del protocolo "M-PIN".

implementación, por que se consideró necesario verificar que el usuario es realmente quien dice ser, comprobando que se cumpliera la igualdad $sC = Token + PIN \cdot C$ la cual tiene un costo de $H_1 + A_{G1} + M_{PIN}$.

En la Tabla 5.15 se presentan los costos del protocolo, en ella sólo se considera el costo del proceso de autenticación, dado que es el único que se se realiza en línea. La fase de recuperación de PIN no se considera en línea, debido a que la respuesta del servidor al cliente se realiza por un medio diferente, por ejemplo: correo electrónico. Además de que en la literatura sólo se encontraron costos de la fase de autenticación.

Trabajo	Cliente		Servidor
Michael Scott [18]	3 seg.		"Pocos ms"
Este trabajo	Intel i7	Cortex-A15	Intel i7
	2.013 ms	3.781 ms	1.714 ms

Tabla 5.15: Resumen de tiempos de la implementación del protocolo "M-PIN".

En la tabla se puede apreciar una comparación entre la implementación realizada en [18] (Tomando su mejor tiempo) y la realizada en este trabajo. Sin embargo, hay que tomar en cuenta que la implementación del cliente en [18] fue realizada utilizando javascript, con la finalidad de utilizarla sobre un navegador web, esta característica es una ventaja ya que permite utilizar el protocolo en cualquier dispositivo. Por otro lado, se puede ver que la implementación realizada en esta tesis, aprovechando las características del procesador, produce mejores resultados, probando así, que es posible aplicar de manera eficiente un protocolo de autenticación de dos factores utilizando dispositivos móviles.

5.4. Aplicación en el dispositivo móvil

Como se ha venido mencionando, uno de los objetivos de esta tesis es la implementación de los protocolos de autenticación de dos factores análizados en el Capítulo 4. Por lo que para realizar la prueba de concepto se realizó una modesta aplicación,

en la cual es posible autenticarse ante el servidor, realizar el cambio de PIN y llevar a cabo la recuperación del PIN.

La aplicación en el dispositivo móvil tiene las siguiente estructura: La implementación de la biblioteca realizada en el lenguaje de programación C, fue compilada con la herramienta NDK, una vez hecha la compilación se utilizó la interfaz nativa de Java (JNI, por sus siglas en ingles) como intermediario entre las funciones compiladas por el NDK y la aplicación desarrollada con SDK. Por otro lado, la biblioteca desarrollada en C++ para el servidor fue compilada con la herramienta *gcc* versión 4.6.

Uno de los principales problemas encontrados al desarrollar la aplicación, fue el almacenamiento de los datos sensibles, es decir, el token y el secreto basado en la identidad del cliente. Para dar solución a este problema se opto por aplicar el siguiente procedimiento:

- Los datos sensibles se almacenan sin ningún tipo de formato en un archivo en la memoria del dispositivo.
- Posteriormente se calcula el picadillo del PIN utilizando una función picadillo que produzca un digesto de 128 bits.
- El resultado del picadillo es utilizado como llave para cifrar el archivo con los datos sensibles, haciendo uso del modo de operación CBC del algoritmo de cifrado AES.

Esto implica que para utilizar el token o el secreto basado en la identidad del cliente en alguna de las fases de los protocolos, es necesario descifrarlos lo cual se realiza de manera similar al procedimiento anterior, sólo que con el algoritmo de descifrado AES. Básicamente, este procedimiento es realizado al inicio de todas las fases.

En el Apéndice C se encuentran las instrucciones para la instalación del cliente y el servidor, además de las instrucciones para utilizar la aplicación.

6 Conclusiones

"El éxito se mide no tanto por la posición que uno ha alcanzado en la vida, si no por los obstáculos que ha tenido que vencer en el camino hacia el éxito."
∼Booker T. Washington (1856-1915)∼

En esta tesis se realizó el análisis de uno de los algoritmos fundamentales en los esquemas basados en la identidad, el conocido como función picadillo, en particular el correspondiente a la proyección a grupos definidos por curvas elípticas, utilizados para el cálculo de emparejamientos bilineales. Como se mencionó en el Capítulo 3 en la página 27, este trabajo se enfocó en obtener la función picadillo hacia el grupo $\mathbb{G}_2$ de manera determinista. Además, se realizó el análisis e implementación de dos protocolos de autenticación, que utilizan dos factores para llevar a cabo el proceso de autenticación utilizando dispositivos móviles y los cuales además proporcionan una solución para el uso de contraseñas de baja entropía, en este trabajo denominadas PIN. Cabe señalar que la función propuesta en este trabajo es una parte fundamental en dichos protocolos.

6.1. Resumen de resultados

Las funciones picadillo a los grupo $\mathbb{G}_1$ y $\mathbb{G}_2$ son una parte importante en los protocolos basados en la identidad, un ejemplo de ello se observa en el Capítulo 4 en la página 47, en donde los protocolos analizados utilizan estas funciones para colocar los clientes en $\mathbb{G}_1$ y los servidores en $\mathbb{G}_2$, con la finalidad de mantener sus secretos basados en la identidad distintos. Cabe mencionar que es fundamental que las funciones puedan ser calculadas de manera eficiente y que este cálculo no permita a un atacante llevar a cabo un ataque de análisis de tiempo, por esta razón se propuso hacer la función de manera determinista. Aprovechando el trabajo realizado por Pierre-Alain Fouque y Mehdi Tibouchi [64], en el cual mostraron como calcular la función picadillo a $\mathbb{G}_1$ de manera determinista cuando la característica del campo es $q \equiv 7 \bmod 12$, en la familia de curvas BN. En este trabajo de tesis se realizó la modificación de

este enfoque para su utilización en el grupo $\mathbb{G}_2$, tomando ventaja de que el grupo está definido sobre el campo $\mathbb{F}_{q^2}$, lo que implica que $q^2 \equiv 1 \bmod 12$, considerando que $q \equiv 3 \bmod 4$.

Se realizó la implementación de la función picadillo a $\mathbb{G}_2$ de manera determinista y no determinista, la cual muestra que la propuesta realizada en esta tesis es un 3.15 % más eficiente computacionalmente hablando que la no determinista, por lo que podemos concluir que nuestra propuesta es una mejor opción dado que además nos permite suprimir los ataques de análisis de tiempo. También es importante mencionar, que el cálculo de puntos en $E'(\mathbb{F}_{q^2})$ de manera determinista (sin considerar la multiplicación por el cofactor) es un 13.3 % más eficiente que el obtenido mediante el algoritmo no determinista.

Se desarrollaron dos bibliotecas criptográficas (cliente ARM y servidor x86-64), para las cuales se consideró una curva BN con 127 bits de seguridad, cuyo parámetro z utilizado para la obtención del primo p, el cual proporciona la característica del campo; y el primo r, el cual representa el orden de las curvas elípticas, se tomó como $z = -2^{62} - 2^{55} - 1$. Esta selección es importante ya que afecta directamente la torre de campos, principalmente en la selección de los polinomios irreducibles y está directamente relacionado con el número de operaciones de la exponenciación final. La principal ventaja de esta selección es que las operaciones con las variables β y ξ, pueden realizarse con simples sumas.

Se analizaron dos protocolos de autenticación de dos factores utilizando algo que el usuario sabe y algo que el usuario tiene y que realizan el proceso de autenticación permitiendo al usuario sólo memorizar una contraseña de cuatro dígitos. Para el análisis, se revisó la literatura existente y se obtuvo una lista de ataques, los cuales consideramos que un protocolo de este tipo debe cumplir para ser considerado seguro.

El análisis del protocolo "Software-only two-factor authentication", mostró que la identidad de los participantes no se oculta, y es uno de los puntos que deben tomarse en cuenta, sin embargo, este hecho no vulnera el protocolo. Este protocolo fue planteado para utilizar un token estático clonable como uno de los factores, lo que permite realizar un ataque para obtener el PIN (Sección 4.7.3 en la página 60), rompiendo así la condición de ser multifactor.

El análisis del protocolo M-PIN, mostró que dependiendo la manera de almacenar la información sensible en el dispositivo, se puede realizar un ataque para obtener el PIN (Sección 4.8.3 en la página 70), rompiendo así la condición de ser multifactor. Además, este protocolo es vulnerable a ataques de confinamiento de grupo, lo que se puede bloquear si se utiliza una curva en la que el cofactor sea primo y mucho más grande que la característica del campo base (es decir, una curva BN "GT-Strong").

Se realizó la implementación de los protocolos antes mencionados. Las implementaciones muestran que su utilización en dispositivos móviles es una opción viable, dado el poder de cómputo que tienen los dispositivos actuales. Se puede resaltar que ambos protocolos toman sólo unos cuantos milisegundos para su ejecución. Sin embargo, podemos decir que el protocolo M-PIN es mucho más eficiente ya que realiza el proceso de autenticación en menos de la mitad de tiempo que el protocolo Software-only two-factor authentication y en comparación con los resultados reportados en [18] nuestra implementación es mucho más rápida considerando que el mejor tiempo reportado en [18] es de 3,000 milisegundos.

Se desarrolló una pequeña aplicación Android, para mostrar el funcionamiento de los protocolos anteriores, la cual permite autenticarse ante un servidor y realizar el cambio y recuperación del PIN.

6.2. Trabajo futuro

Como trabajo futuro podemos considerar los siguientes puntos: Aplicar el enfoque utilizado para el cálculo de la función picadillo al grupo $\mathbb{G}_2$ a otras familias de curvas elípticas como: BW (Brezing-Weng) [33], KSS (Kachisa-Schaefer-Scott) [34] y BLS (Barreto-Lynn-Scott) [35].

Mejorar la biblioteca del cliente utilizando técnicas de paralelización, para aprovechar las nuevas arquitecturas multinúcleo para dispositivos móviles. Además de permitir la opción de utilizar diferentes curvas y distintos parámetros.

Realizar la implementación del protocolo “M-PIN” para una curva GT-Strong, con el fin de eliminar la posibilidad de ataques de confinamiento de grupo.

Bibliografía

[1] Pei Zheng and Lionel Ni. *Smart Phone and Next Generation Mobile Computing*. Morgan Kaufmann Series in Networking. Elsevier Science, 2010.

[2] Wade Trappe and Lawrence C. Washington. *Introduction to Cryptography: With Coding Theory*. Pearson Prentice Hall, second edition, 2006.

[3] Lawrence C. Washington. *Elliptic Curves: Number Theory and Cryptography*. Chapman & Hall/CRC, second edition, 2008.

[4] National Bureau of Standard. *Data Encryption Standard*. In FIPS PUB 46, Federal Information Processing Standards Publication. U.S. Department of Commerce, Washington D.C., USA, 1977.

[5] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, Leuven, Belgium, first edition, 2002.

[6] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM*, 21:120–126, 1978.

[7] Taher Elgamal. *A public key cryptosystem and a signature scheme based on discrete logarithms. Information Theory, IEEE Transactions on*, 31(4):469–472, 1985.

[8] Whitfield Diffie and Martin E. Hellman. *New directions in cryptography. IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[9] Victor S. Miller. *Use of elliptic curves in cryptography. Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO'85*, pages 417–426, 1986.

[10] Neal Koblitz. *Elliptic Curve Cryptosystems. Mathematics of Computation*, 48(177):203–209, 1987.

[11] Alfred Menezes, Scott Vanstone, and Tatsuaki Okamoto. *Reducing elliptic curve logarithms to logarithms in a finite field. Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 80–89, 1991.

[12] Antoine Joux. *A One Round Protocol for Tripartite Diffie-Hellman. Proceedings of the 4th International Symposium on Algorithmic Number Theory*, pages 385–394, 2000.

[13] Shigeo Mitsunari, Ryuichi Sakai, and Masao Kasahara. *A new traitor tracing. IEICE transactions on fundamentals of electronics, communications and computer sciences*, 85(2):481–484, 2002.

[14] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. *Cryptosystems based on pairing. 2000 Symposium on Cryptography and Information Security (SCIS2000)*, pages 26–28, 2000.

[15] Adi Shamir. Identity-based cryptosystems and signature schemes. *Proceedings of CRYPTO'84 on Advances in cryptology*, pages 47–53, 1984.

[16] Dan Boneh and Matt Franklin. *Identity-Based Encryption from the Weil Pairing. Advances in Cryptology CRYPTO 2001*, 2139:213–229, 2001.

[17] Michael Scott. *Replacing Username/Password with Software-Only Two-Factor Authentication. IACR Eprint archive*, 2012.

[18] Michael Scott. *M-Pin Technology (Version 3.0). Certivox Labs*, 2013.

[19] Laura Fuentes-Castañeda. *Estudio y Análisis de Emparejamientos Bilineales Definidos sobre Curvas Ordinarias con Alto Nivel de Seguridad.* Master's thesis, CINVESTAV-IPN, 2011.

[20] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography.* Professional Computing. Springer, 2004.

[21] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography.* CRC Press, Inc., Boca Raton, Florida, USA, 1st edition, 1996.

[22] Robert P. Gallant, Robert J. Lambert, and Scott A. Vanstone. *Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms. Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference*, 2139:190–200, 2001.

[23] Steven D. Galbraith, Xibin Lin, and Michael Scott. Endomorphisms for faster elliptic curve cryptography on a large class of curves. *Lecture Notes in Computer Science, In EUROCRYPT*, 5479:518–535, 2009.

[24] Florian Hess, Nigel Smart, and Frederik Vercauteren. *The Eta Pairing Revisited. IEEE Transactions on Information Theory*, 52:4595–4602, 2006.

[25] Paulo Barreto, Ben Lynn, and Michael Scott. *On the Selection of Pairing-Friendly Groups. Selected Areas in Cryptography*, 3006:17–25, 2004.

[26] Paul C. Van Oorschot and Michael J. Wiener. *Parallel Collision Search with Cryptanalytic Applications. Journal of Cryptology*, 12:1–28, 1996.

[27] John Pollard. *Monte Carlo methods for Index Computation (mod p). Mathematics of Computation*, 32:918–924, 1978.

[28] Leonard M. Adleman and Ming-Deh A. Huang. *Function Field Sieve Method for Discrete Logarithms over Finite Fields. Inf. Comput.*, 151:5–16, May 1999.

[29] David Freeman, Michael Scott, and Edlyn Teske. *A Taxonomy of Pairing-Friendly Elliptic Curves. Journal of Cryptology*, 23:224–280, 2010.

[30] Shafi Goldwasser and Joe Kilian. *Primality Testing Using Elliptic Curves. J. ACM*, 46:450–472, 1999.

[31] Naomi Benger and Michael Scott. *Constructing Tower Extensions of Finite Fields for Implementation of Pairing-Based Cryptography. Arithmetic of Finite Fields*, 6087:180–195, 2010.

[32] Paulo Barreto and Michael Naehrig. *Pairing-friendly elliptic curves of prime order. Selected Areas in Cryptography – SAC 2005*, 3897:319–331, 2006.

[33] Friederike Brezing and Annegret Weng. *Elliptic Curves Suitable for Pairing Based Cryptography. Designs, Codes and Cryptography*, 37:133–141, 2005.

[34] Ezekiel Kachisa, Edward Schaefer, and Michael Scott. *Constructing Brezing-Weng Pairing-Friendly Elliptic Curves Using Elements in the Cyclotomic Field. Pairing-Based Cryptography - Pairing 2008*, 5209:126–135, 2008.

[35] Paulo Barreto, Ben Lynn, and Michael Scott. *Constructing Elliptic Curves with Prescribed Embedding Degrees. Security in Communication Networks*, 2576:257–267, 2003.

[36] Victor Miller. *The Weil Pairing, and Its Efficient Calculation. Journal of Cryptology*, 17:235–261, 2004.

[37] Frederik Vercauteren. *Optimal Pairings. IEEE Trans. Inf. Theor.*, 56:455–461, January 2010.

[38] Craig Costello, Tanja Lange, and Michael Naehrig. *Faster Pairing Computations on Curves with High-Degree Twists. Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography*, 6056:224–242, May 26-28, 2010.

[39] Diego F. Aranha, Koray Karabina, Patrick Longa, CatherineH. Gebotys, and Julio López. *Faster Explicit Formulas for Computing Pairings over Ordinary Curves. Advances in Cryptology EUROCRYPT 2011*, 6632:48–68, 2011.

[40] Laura Fuentes-Castañeda, Edward Knapp, and Francisco Rodríguez-Henríquez. *Faster hashing to $\mathbb{G}_2$. Proceedings of the 18th international conference on Selected Areas in Cryptography*, pages 412–430, 2012.

[41] Arjen K. Lenstra, Hendrik W. Lenstra, and László Lovász. *Factoring Polynomials with Rational Coefficients. Mathematische Annalen*, 261:515–534, 1982.

[42] Joonsang Baek and Yuliang Zheng. *Identity-Based Threshold Decryption. Public Key Cryptography PKC 2004*, 2947:262–276, 2004.

[43] Craig Gentry and Alice Silverberg. *Hierarchical ID-Based Cryptography. Advances in Cryptology ASIACRYPT 2002*, 2501:548–566, 2002.

[44] Jeremy Horwitz and Ben Lynn. *Toward Hierarchical Identity-Based Encryption. Advances in Cryptology EUROCRYPT 2002*, 2332:466–481, 2002.

[45] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. *Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. Advances in Cryptology EUROCRYPT 2003*, 2656:416–432, 2003.

[46] Dan Boneh, Ben Lynn, and Hovav Shacham. *Short Signatures from the Weil Pairing. Advances in Cryptology ASIACRYPT 2001*, 2248:514–532, 2001.

[47] JaeCha Choon and Jung Hee Cheon. *An Identity-Based Signature from Gap Diffie-Hellman Groups. Public Key Cryptography PKC 2003*, 2567:18–30, 2002.

[48] Xavier Boyen. *Multipurpose Identity-Based Signcryption. Advances in Cryptology CRYPTO 2003*, 2729:383–399, 2003.

[49] David P. Jablon. *Strong password-only authenticated key exchange. SIGCOMM Comput. Commun. Rev.*, 26(5):5–26, October 1996.

[50] Victor Boyko, Philip MacKenzie, and Sarvar Patel. *Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. Advances in Cryptology EUROCRYPT 2000*, 1807:156–171, 2000.

[51] Mihir Bellare and Phillip Rogaway. *Random oracles are practical: a paradigm for designing efficient protocols. Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73, 1993.

[52] Mehdi Tibouchi. *A Note on Hashing to BN Curves. SCIS 2012, IEICE*, 2012.

[53] Ueli Maurer, Renato Renner, and Clemens Holenstein. *Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. Theory of cryptography*, pages 21–39, 2004.

[54] David P. Jablon. *Strong password-only authenticated key exchange. SIGCOMM Comput. Commun. Rev.*, 26(5):5–26, October 1996.

[55] René Schoof. *Elliptic curves over finite fields and the computation of square roots mod p. Mathematics of computation*, 44(170):483–494, 1985.

[56] Andrejz Schinzel and Skałba Mariusz. *On equations $y^2 = x^n + k$ in a finite field. Bull. Pol. Acad. Sci. Math.*, 52(3):223–226, 2004.

[57] Andrew Shallue and Christiaan E. van de Woestijne. *Construction of rational points on elliptic curves over finite fields. Proceedings of the 7th international conference on Algorithmic Number Theory*, pages 510–524, 2006.

[58] Thomas Icart and Sagem Sécurité. *How to hash into elliptic curves. of Lecture Notes in Computer Science*, pages 303–316, 2009.

[59] Maciej Ulas. *Rational points on certain hyperelliptic curves over finite fields. Bull. Polish Acad. Sci. Math.*, 55(2):97–104, 2007.

[60] Eric Brier, Jean-Sébastien Coron, Thomas Icart, David Madore, Hugues Randriam, and Mehdi Tibouchi. *Efficient Indifferentiable Hashing into Ordinary Elliptic Curves. Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference*, 6223:237–254, 2010.

[61] Reza Rezaeian Farashahi. *Hashing into Hessian Curves. Progress in Cryptology AFRICACRYPT 2011*, 6737:278–289, 2011.

[62] Jean-Gabriel Kammerer, Reynald Lercier, and Guénaël Renault. *Encoding points on hyperelliptic curves over finite fields in deterministic polynomial time. Pairing-Based Cryptography-Pairing 2010*, pages 278–297, 2010.

[63] Jean-Marc Couveignes and Jean-Gabriel Kammerer. *The Geometry of Flex Tangents to a Cubic Curve and its Parameterizations. Cryptology ePrint Archive, Report 2011/033*, 2011. http://eprint.iacr.org/.

[64] Pierre-Alain Fouque and Mehdi Tibouchi. *Indifferentiable Hashing to Barreto-Naehrig Curves. LATINCRYPT*, 7533:1–17, 2012.

[65] Gora Adj and Francisco Rodríguez-Henríquez. *Square root computation over even extension fields. Cryptology ePrint Archive, Report 2012/685*, 2012.

[66] Eric Bach and Klaus Huber. *Note on Taking Square-Roots Modulo N. IEEE Transactions on Information Theory*, 45(2):807–809, 1999.

[67] Michael Scott. *Implementing Cryptographic Pairings over Barreto-Naehrig Curves. Pairing-Based Cryptography – Pairing 2007, First International Conference*, 4575:177–196, 2007.

[68] Li Gong, T. Mark A. Lomas, Roger M. Needham, and Jerome H. Saltzer. *Protecting Poorly Chosen Secrets from Guessing Attacks. IEEE Journal on Selected Areas in Communications*, 11:648–656, 1993.

[69] Steven M. Bellovin and Michael Merritt. *Encrypted key exchange: password-based protocols secure against dictionary attacks. Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on*, pages 72–84, 1992.

[70] Audun Jøsang and Gunnar Sanderud. *Security in mobile communications: challenges and opportunities. Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003 - Volume 21*, pages 43–48, 2003.

[71] Hyun-Sung Kim, Sung-Woon Lee, and Kee-Young Yoo. *ID-based password authentication scheme using smart cards and fingerprints. SIGOPS Oper. Syst. Rev.*, 37(4):32–41, October 2003.

[72] Michael Scott. *Cryptanalysis of an ID-based password authentication scheme using smart cards and fingerprints. SIGOPS Oper. Syst. Rev.*, 38(2):73–75, April 2004.

[73] Rafael Martínez-Peláez and Francisco Rico-Novella. *Cryptanalysis of Sood et al.'s Authentication Scheme using Smart Cards. Cryptology ePrint Archive, Report 2012/386*, 2012. http://eprint.iacr.org/.

[74] Sandeep K. Sood, Anil K. Sarje, and Kuldeep Singh. *An improvement of Liao et al.'s authentication scheme using smart cards. Advance Computing Conference (IACC), 2010 IEEE 2nd International*, pages 240–245, 2010.

[75] Feng Hao and Dylan Clarke. *Security Analysis of a Multi-factor Authenticated Key Exchange Protocol. Applied Cryptography and Network Security*, 7341:1–11, 2012.

[76] Feng Hao. *On robust key agreement based on public key authentication. Security and Communication Networks*, pages n/a–n/a, 2012.

[77] David Pointcheval and Sébastien Zimmer. *Multi-factor Authenticated Key Exchange. Applied Cryptography and Network Security*, 5037:277–295, 2008.

[78] Whitfield Diffie and Martin Hellman. *New directions in cryptography. Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

[79] Alfred Menezes, Minghua Qu, and Scott A. Vanstone. *Some new key agreement protocols providing mutual implicit authentication. Selected Areas in Cryptography*, 1995.

[80] Burton S. Kaliski. *An unknown key-share attack on the MQV key agreement protocol. ACM Trans. Inf. Syst. Secur.*, 4(3):275–288, August 2001.

[81] Eiji Okamoto. *Proposal for identity-based key distribution systems. Electronics Letters*, 22(24):1283–1284, 1986.

[82] Kazue Tanaka and Eiji Okamoto. *Key distribution system for mail systems using ID-related information directory. Computers and Security*, 10(1):25 – 33, 1991.

[83] M. Girault and Paillès. J.C. *An identity-based scheme providing zero-knowledge authentication and authenticated key exchange. ESORICS '90*, pages 173–184, 1990.

[84] ISO/IEC 11770-3. *Information technology - Security Techniques - Key management – Part 3: Mechanisms using asymmetric techniques. International Organization for Standardization*, 1999.

[85] Liqun Chen and Caroline Kudla. *Identity based authenticated key agreement protocols from pairings. Computer Security Foundations Workshop, 2003. Proceedings. 16th IEEE*, pages 219–233, 2003.

[86] Noel McCullagh and Paulo S.L.M. Barreto. *A New Two-Party Identity-Based Authenticated Key Agreement. Topics in Cryptology – CT-RSA 2005*, 3376:262–274, 2005.

[87] Kyungah Shim. *Efficient ID-based authenticated key agreement protocol based on Weil pairing. Electronics Letters*, 39(8):653–654, 2003.

[88] Nigel P. Smart. *Identity-based authenticated key agreement protocol based on Weil pairing. Electronics Letters*, 38(13):630–632, 2002.

[89] Yongge Wang. *Efficient Identity-Based and Authenticated Key Agreement Protocol. Transactions on Computational Science XVII*, 7420:172–197, 2013.

[90] Shai Halevi and Hugo Krawczyk. *Public-key cryptography and password protocols. ACM Trans. Inf. Syst. Secur.*, 2(3):230–268, August 1999.

[91] Yongge Wang. *Password Protected Smart Card and Memory Stick Authentication against Off-Line Dictionary Attacks. Information Security and Privacy Research*, 376:489–500, 2012.

[92] Chwei shyong Tsai, Cheng chi Lee, and Min shiang Hwang. *Password Authentication Schemes: Current Status and Key Issues. International Journal of Network Security*, 3(2):101–115, 2006.

[93] I-En Liao, Cheng-Chi Lee, and Min-Shiang Hwang. *A password authentication scheme over insecure networks. Journal of Computer and System Sciences*, 72(4):727 – 740, 2006.

[94] Guomin Yang, DuncanS. Wong, Huaxiong Wang, and Xiaotie Deng. *Formal Analysis and Systematic Construction of Two-Factor Authentication Scheme. Information and Communications Security*, 4307:82–91, 2006.

[95] IEEE P1363. *Standard Specifications For Public-Key Cryptography.* http://grouper.ieee.org/groups/1363/.

[96] Benoît Chevallier-Mames, Jean-Sébastien Coron, Noel McCullagh, David Naccache, and Michael Scott. *Secure delegation of elliptic-curve pairing. Smart Card Research and Advanced Application*, pages 24–35, 2010.

[97] Jean-Luc Beuchat, Jorge Enrique González-Díaz, Shigeo Mitsunari, Eiji Okamoto, Francisco Rodríguez-Henríquez, and Tadanori Teruya. *High-Speed Software Implementation of the Optimal Ate Pairing over Barreto-Naehrig Curves. Pairing-Based Cryptography - Pairing 2010 - 4th International Conference*, 6487:21–39, December 13-15, 2010.

[98] Jaewook Chung and M. Anwar Hasan. *Asymmetric Squaring Formulas. Proceedings of the 18th IEEE Symposium on Computer Arithmetic*, pages 113–122, June 25-27, 2007.

[99] Peter L. Montgomery. *Speeding the Pollard and Elliptic Curve Methods of Factorization. Mathematics of Computation*, 48(177):243–264, January, 1987.

[100] Ana Helena Sánchez Ramárez. *Implementación de la Criptografía Basada en Atributos en un Dispositivo Móvil. CINVESTAV-IPN*, 2012.

[101] Cetin Kaya Koc, Tolga Acar, and Burton S. Kaliski Jr. *Analyzing and Comparing Montgomery Multiplication Algorithms. IEEE Micro*, 16(3):26–33, June, 1996.

[102] Victor Shoup. *A computational introduction to number theory and algebra.* Cambridge University Press, New York, New York, USA, 2 edition, 2005.

[103] Selcuk Baktir and Berk Sunar. *Optimal Tower Fields. IEEE Trans. Comput.*, 53:1231–1243, October 2004.

[104] Robert Granger and Michael Scott. *Faster Squaring in the Cyclotomic Subgroup of Sixth Degree Extensions. Public Key Cryptography - PKC 2010*, 6056:209–223, 2010.

[105] Rudolf Lidl and Harald Niederreiter. *Finite Fields (Encyclopedia of Mathematics and its Applications)*. Cambridge University Press, October 1996.

[106] Jeffrey Hoffstein, Jill Pipher, and J.H. Silverman. *An Introduction to Mathematical Cryptography*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[107] Peter L. Montgomery. *Modular Multiplication without Trial Division. Mathematics of Computation*, 44(170):519–521, April, 1985.

[108] Robert Granger and Michael Scott. *Faster Squaring in the Cyclotomic Subgroup of Sixth Degree Extensions. In PKC'10 Proceedings of the 13th international conference on Practice and Theory in Public Key Cryptography*, 6056:209–223, May 26-28, 2010.

[109] Koray Karabina. *Squaring in cyclotomic subgroups. IACR Cryptology ePrint Archive*, 2010:542, 2010.

A

Teoría de números

En este capítulo se introducen los conceptos que fueron utilizados a lo largo de esta tesis. Se presentan las definiciones y propiedades de los conceptos de grupo en la Sección A.1, anillo en la Sección A.2, campos en la Sección A.3, grupo ciclotómico en la Sección A.4, *lattice* en la Sección A.5 y la descripción de los diferentes morfismos en la Sección A.6.

A.1. Grupo

Un grupo es un objeto matemático abstracto denotado por $\mathbb{G} = (\mathbb{G}, \star, e)$, conformado por un conjunto $\mathbb{G}$, un elemento identidad $e \in \mathbb{G}$ y una operación binaria $\star : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$. En donde el conjunto $(\mathbb{G}, \star, e)$ cumple las siguientes propiedades:

I) La operación $\star$ es cerrada sobre los elementos del conjunto $\mathbb{G}$, es decir, $\forall a, b \in \mathbb{G},\ a \star b \in \mathbb{G}$.

II) El elemento identidad e es único y para todo $a \in \mathbb{G}$, se cumple que $a \star e = e \star a = a$.

III) La operación $\star$ es asociativa sobre los elementos de $\mathbb{G}$, es decir, dados $a,\ b,\ c \in \mathbb{G}$, entonces $a \star (b \star c) = (a \star b) \star c$.

IV) Para todo $a \in \mathbb{G}$ existe un único elemento $\bar{a} \in \mathbb{G}$, llamado el inverso de a, tal que $a \star \bar{a} = \bar{a} \star a = e$.

Además el grupo $\mathbb{G}$ se denomina grupo abeliano (o grupo conmutativo) si cumple una propiedad más:

V) La operación $\star$ es conmutativa, es decir, $\forall\ a, b \in \mathbb{G}$, se satisface la igualdad $a \star b = b \star a$.

El grupo $\mathbb{G}$, implica que $\mathbb{G}$ es un conjunto de elementos que forman un grupo bajo la operación $\star$ y cuyo elemento identidad es e. Además, dado un elemento $g \in \mathbb{G}$ y un elemento $m \in \mathbb{Z}^+$, la aplicación de m veces el operador $\star$ sobre el elemento g se denota como $\star^m(g)$.

Existen algunos conceptos importantes relacionados con grupos, los cuales se definen a continuación:

Definición A.1. *(Orden del grupo). El orden de un grupo $(\mathbb{G}, \star, e)$ está definido como el número de elementos del conjunto $\mathbb{G}$ y se denota como ord($\mathbb{G}$). Si $\mathbb{G}$ es infinito se dice que el ord($\mathbb{G}$) = ∞.*

Definición A.2. *(Orden de un elemento del grupo). El orden de un elemento $g \in \mathbb{G}$, denotado por ord(g), es el menor entero positivo r tal que $\star^r(g) = e$. Si dicho r no existe se dice que el ord(g) = ∞.*

Lema A.1. *(ord(g) | ord($\mathbb{G}$)). Sea $\mathbb{G}$ un grupo de orden finito $n \in \mathbb{Z}^+$, $\forall\ g \in \mathbb{G}$ el orden de g divide al orden del grupo, lo cual implica que $\star^n(g) = e$.*

Definición A.3. *(Generador del grupo). Dado el grupo $\mathbb{G}$, se dice que un elemento $g \in \mathbb{G}$ es un generador del grupo, si para cada elemento $h \in \mathbb{G}$ existe $i \in \mathbb{Z}^+$, tal que $h = \star^i(g)$. Al grupo generado por g se le denota como $\mathbb{G} = \langle g \rangle$.*

Definición A.4. *(Grupo Cíclico). Un grupo $\mathbb{G} = (\mathbb{G}, \star, e)$ es cíclico si $\mathbb{G} = \langle g \rangle$ para algún generador $g \in \mathbb{G}$.*

El número de elementos generadores en un grupo cíclico $\mathbb{G}$ de orden $n \in \mathbb{Z}^+$, está definido por la función indicatriz de Euler $\varphi(n)$, la cual dado el entero positivo n calcula el número de enteros positivos menores o iguales que son primos relativos con él. En el caso en que $\mathbb{G}$ tiene orden primo p, el número de generadores de $\mathbb{G}$ esta dado por $\varphi(p) = p - 1$.

A.1.1. Notación

Por lo general en lugar de utilizar un símbolo especial, como $\star$, para denotar la operación binaria de un grupo se puede utilizar el operador aditivo ("+") o el operador multiplicativo ("$\cdot$").

Si un grupo $\mathbb{G}$ es escrito de manera aditiva entonces el elemento identidad se denota como "0", de manera que $\mathbb{G} = (\mathbb{G}, +, 0)$. En esta notación el inverso de un elemento $a \in \mathbb{G}$ es denotado por $-a$ y para $a, b \in \mathbb{G}$, la operación $a-b$ denota $a+(-b)$. Además, la aplicación de m veces el operador "+" sobre a es denotada como $m \cdot a$, en donde $m \in \mathbb{Z}^+$.

Por otra parte si un grupo $\mathbb{G}$ es escrito de manera multiplicativa entonces el elemento identidad se denota como "$1_{\mathbb{G}}$", de manera que $\mathbb{G} = (\mathbb{G}, \cdot, 1_{\mathbb{G}})$. En esta notación el inverso de un elemento $a \in \mathbb{G}$ es denotado por a^{-1} o $1/a$ y para $a, b \in \mathbb{G}$, la operación a/b denota $a \cdot b^{-}1$. Además, la aplicación de m veces el operador "$\cdot$" sobre a es denotada como a^m, en donde $m \in \mathbb{Z}^+$.

A.1.2. Subgrupos

Dado el grupo $(\mathbb{G}, \star, e)$ y sea $\mathbb{H}$ un subconjunto de $\mathbb{G}$, si $\mathbb{H}$ forma un grupo bajo la operación $\star$ con "e" como elemento identidad, entonces se dice que $(\mathbb{H}, \star, e)$ es un subgrupo de $(\mathbb{G}, \star, e)$. Uno de los teoremas importantes en la teoría de grupos es el de Lagrange, el cual se enuncia a continuación:

Teorema A.1. *(Teorema de Lagrange) [102] . Sea $\mathbb{G} = (\mathbb{G}, \star, e)$ un grupo abeliano finito y sea $\mathbb{H} = (\mathbb{H}, \star, e)$ un subgrupo de $\mathbb{G}$, entonces el orden de $\mathbb{H}$ divide al orden de $\mathbb{G}$.*

Para un grupo abeliano $(\mathbb{G}, \star, e)$, los subconjuntos $\mathbb{G}$ y $\{e\}$ son subgrupos. Sin embargo estos subgrupos no son muy interesantes. Una forma fácil de encontrar subgrupos, más interesantes, dentro del grupo abeliano es usando los siguientes dos teoremas:

Teorema A.2. *[102, Teorema 8.22]. Sea $\mathbb{G} = (\mathbb{G}, \star, e)$ un grupo abeliano y sea m un número entero, el conjunto*

$$\mathbb{G}\{m\} = \{a \in \mathbb{G} \mid \star^m (a) = e\} = \{a \in \mathbb{G} \mid ord(a) = m\},$$

forma un subgrupo de $\mathbb{G}$, definido como $(\mathbb{G}\{m\}, \star, e)$.

Teorema A.3. *[102, Teorema 8.21]. Sea $\mathbb{G} = (\mathbb{G}, \star, e)$ un grupo abeliano y sea m un número entero tal que*

$$\star^m(\mathbb{G}) = \{\star^m(a) \mid a \in \mathbb{G}\},$$

entonces $\star^m(\mathbb{G}) = (\star^m(\mathbb{G}), \star, e)$ es un subgrupo de $\mathbb{G}$.

A.1.3. Clase lateral

Sea $\mathbb{H} = (\mathbb{H}, \star, e)$ un subgrupo de $\mathbb{G} = (\mathbb{G}, \star, e)$, para todo $a, b \in \mathbb{G}$ se escribe $a \equiv b \pmod{\mathbb{H}}$, si $a \star \bar{b} \in \mathbb{H}$, donde $\bar{b}$ es el inverso de b. A la expresión " $\equiv \pmod{\mathbb{H}}$ " se le conoce como *relación de equivalencia* y divide al grupo $\mathbb{G}$ en *clases de equivalencia*.

Dado $a \in \mathbb{G}$, a la clase de equivalencia que contiene al elemento "a" se le denota como $[a]_{\mathbb{H}}$, la cual está definida como:

$$[a]_{\mathbb{H}} = a \star \mathbb{H} = \{a \star h \mid h \in \mathbb{H}\},$$

es decir, $x \in [a]_{\mathbb{H}} \iff x \equiv a \pmod{\mathbb{H}}$.

Las clases de equivalencia son llamadas *clases laterales de $\mathbb{H}$ en $\mathbb{G}$*, en donde un elemento de una clase es llamado el *representativo de la clase lateral*. El conjunto de todas las clases laterales está denotado como $\mathbb{G}/\mathbb{H}$ y forma un grupo $(\mathbb{G}/\mathbb{H}, \star, [e]_{\mathbb{H}})$, donde

$$[a]_{\mathbb{H}} \star [b]_{\mathbb{H}} = [a \star b]_{\mathbb{H}},$$

este grupo es llamado *grupo cociente de $\mathbb{G}$ módulo $\mathbb{H}$*.

A.2. Anillo

Un anillo conmutativo $R = (R, +, \cdot, 0, 1_R)$ con unidad, está conformado por un conjunto R que tiene definidas las operaciones binarias de adición ("+") y multiplicación ("·") sobre R y el cual cumple las siguientes propiedades:

I) El conjunto R bajo la adición forma un grupo abeliano $(R, +, 0)$, en el que 0 denota la identidad aditiva.

II) La multiplicación es asociativa, es decir, $\forall a, b, c \in R$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.

III) La multiplicación es distributiva sobre la adición, se cumple que, $\forall a, b, c \in R$, $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ y $(b + c) \cdot a = (b \cdot a) + (c \cdot a)$.

IV) Existe una identidad multiplicativa única, es decir, existe el elemento 1_R, tal que, $1_R \cdot a = a = a \cdot 1_R$ para todo elemento $a \in R$.

V) La multiplicación es conmutativa, $\forall a, b \in R$ se cumple $a \cdot b = b \cdot a$.

Sea R un anillo, se dice que $u \in R$ es una unidad si existe un elemento $u^{-1} \in R$, el cual es el inverso multiplicativo de u, es decir, $u \cdot u^{-1} = 1_R$. El conjunto de unidades se denota por R^* y es cerrado bajo la multiplicación, por lo que R^* es un grupo abeliano, llamado *grupo multiplicativo de las unidades de R.*

A.3. Campo

Un campo $\mathbb{F}$ es un anillo conmutativo $(\mathbb{F}, +, \cdot, 0, 1_{\mathbb{F}})$ en el cual cada elemento diferente de cero es una unidad. Un campo cumple las siguientes propiedades:

I) $\mathbb{F}^+ = (\mathbb{F}, +, 0)$ es un grupo abeliano con el "0" como unidad aditiva.

II) $\mathbb{F}^* = (\mathbb{F} - \{0\}, \cdot, 1_{\mathbb{F}})$ es un grupo abeliano con el "$1_{\mathbb{F}}$" como unidad multiplicativa.

III) La multiplicación se distribuye a ambos lados de la adición.

Definición A.5. *(Característica de un campo). Dado el campo $\mathbb{F}$, sea $n \in \mathbb{Z}^+$. Se dice que n es la característica de $\mathbb{F}$, si n es el menor entero positivo tal que $n \cdot 1 = \sum_{i=0}^{n-1} 1 = 0$. En caso de que no exista tal entero n, se dice que $\mathbb{F}$ es de característica 0.*

Se dice que $\mathbb{F}$ forma un campo finito si su característica es diferente de 0, en caso contrario se dice que el campo es infinito. Un campo finito $\mathbb{F}_q$ existe si y sólo si, su característica $q = p^m$, tal que p es un número primo y el entero $m \geq 1$.

Un campo finito $\mathbb{F}$ cuya característica es un número primo p suele abreviarse como $\mathbb{F}_p$. Por ejemplo, el conjunto $\mathbb{F}_p = \{0, 1, 2, ..., p-2, p-1\}$ define un campo finito $(\mathbb{F}_p, +, \cdot, 0, 1)$ bajo las operaciones de adición y multiplicación módulo p.

A.3.1. Extensión de un campo finito

El conjunto de polinomios en la variable z con coeficientes en el campo $\mathbb{F}_p$, está denotado por $\mathbb{F}_p[z]$. Dado un número entero n, el conjunto finito compuesto por los polinomios en $\mathbb{F}_p[z]$ de grado menor a $n-1$, es decir,

$$\{a_{n-1}z^{n-1} + a_{n-2}z^{n-2} + \cdots + a_2z^2 + a_1z + a_0 \ \mid \ a_i \in \mathbb{F}_p \text{ con, } 0 \leq i \leq n-1\},$$

forma un "campo finito de característica p", denotado por $\mathbb{F}_{p^n}$, bajo las operaciones de adición y multiplicación módulo $f(z)$, donde $f(z)$ es un polinomio irreducible de grado n [20]. De esta manera, el campo $\mathbb{F}_{p^n}$ es de orden p^n y puede ser representado como:

$$\mathbb{F}_{p^n} = \mathbb{F}_p[z]/f(z) \cong \text{ los polinomios en } \mathbb{F}_p[z] \text{ mod } f(z).$$

Definición A.6. *(Polinomio irreducible). Dado un polinomio $f(z)$ no constante de grado $n \geq 2$, se dice que $f(z)$ es irreducible, si no puede factorizarse como el producto de polinomios de grado menor a n.*

Considerando los campos $\mathbb{F}$ y $\mathbb{K}$ con $\mathbb{F} \subseteq \mathbb{K}$, al campo $\mathbb{K}$ se le conoce como una extension de $\mathbb{F}$, si existe un morfismo inyectivo $\rho : \mathbb{F} \to \mathbb{K}$. Una extensión del campo $\mathbb{F}_q$ de grado m, es de la forma $\mathbb{F}_{q^m} \cong \mathbb{F}_q[z]/f(z)$, en donde $f(z)$ es un polinomio irreducible de grado m sobre $\mathbb{F}_q$.

Dado un elemento $a \in \mathbb{K}$ se dice que a es un *elemento algebraico* sobre $\mathbb{F}$ si $f(a) = 0$ para algún polinomio $f \in \mathbb{F}[z]$ distinto de cero, por otro lado a es considerado un *elemento trascendente* sobre $\mathbb{F}$ si $f(a) \neq 0$ para todo polinomio no constante $f \in \mathbb{F}[z]$. Una extensión $\mathbb{K}$ del campo $\mathbb{F}$ es una *extensión algebraica* sobre $\mathbb{F}$ si todo elemento en $\mathbb{K}$ es algebraico sobre $\mathbb{F}$ y se dice que un campo $\mathbb{K}$ es *algebraicamente cerrado* si $\mathbb{K}$ no tiene extensiones algebraicas propias o si toda extensión algebraica $\mathbb{L}$ de $\mathbb{K}$ satisface que $\mathbb{L} = \mathbb{K}$.

Dos conceptos importantes relacionados con campos finitos son la *cerradura algebraica* y la *norma de un elemento*:

Definición A.7. *(Cerradura algebraica de un campo finito $\mathbb{F}_p$). Sea p un número primo, la cerradura algebraica del campo finito $\mathbb{F}_p$, denotada por $\bar{\mathbb{F}}_p$, es el conjunto infinito de todas sus extensiones, es decir,*

$$\bar{\mathbb{F}}_p = \bigcup_{m \geq 1} \mathbb{F}_{p^m}.$$

Definición A.8. *(Norma). Sea p un número primo y sea $n \in \mathbb{N}$, los conjugados de $a \in \mathbb{F}_{p^n}$ son los elementos a^{p^i}, donde $0 \leq i \leq n-1$. La norma de a, denotada por $|a|$, es el producto de todos los conjugados de a, es decir,*

$$|a| = \prod_{i=0}^{n-1} a^{p^i}.$$

A.3.2. Torre de campo

Se le conoce como torre de campos a la construcción formada por campos finitos, donde cada campo finito representa un nivel de la torre y el cual corresponde a una extensión de los campos de niveles inferiores. Con el objetivo de hacer más eficiente la aritmética en $\mathbb{F}_{p^n}$, Baktir y Sunar [103] propusieron expresar $\mathbb{F}_{p^n} = \mathbb{F}_p[z]/f(z)$ como una extensión del campo finito $\mathbb{F}_q$, en donde $q = p^m$, tal que $m|n$:

$$\begin{aligned} \mathbb{F}_{p^n} &= \mathbb{F}_q[v]/h(v) \quad \text{, donde } h(v) \in \mathbb{F}_q[v] \text{ es de grado } n/m \\ \mathbb{F}_q &= \mathbb{F}_p[u]/g(u) \quad \text{, donde } g(u) \in \mathbb{F}_p[u] \text{ es de grado } m. \end{aligned}$$

Como se puede observar, la estructura de la torre de campos depende directamente del valor de n. Particularmente, dados los enteros positivos "a" y "b", se dice que un campo finito $\mathbb{F}_{p^n}$ es "amable" con los emparejamientos si $n = 2^a3^b$, tal que $\mathbb{F}_{p^n}$ se puede representar a través de a extensiones cuadráticas y b extensiones cúbicas del campo base [104].

Además, para todo $n = 2^a3^b$, si $4 \nmid n$ entonces la torre de campos se puede construir mediante binomios irreducibles. Por el contrario, si $n \equiv 0 \mod 4$, se requiere que $p^n \equiv 1 \mod 4$ para utilizar esta misma representación [105, Teorema 3.75].

A.4. Grupo ciclotómico

Definición A.9. *(Raíces de la unidad). Dado $n \in \mathbb{N}$, las raíces n-ésimas de la unidad son las n soluciones del polinomio $z^n - 1 = 0$, las cuales están denotadas por z_j.*

$$z^n - 1 = \prod_{j=0}^{n-1}(z - z_j).$$

El conjunto de las raíces n-ésimas de la unidad denotado como μ_n, forma un grupo cíclico $\mu_n = (\mu_n, \cdot, 1)$.

Definición A.10. *(Raíces primitivas de la unidad). Sea $z^* \in \mu_n$, se dice que z^* es una raíz "primitiva" de la unidad, si y sólo si $\mu_n = \langle z^* \rangle$. Sea $\varphi(\cdot)$ la función indicatriz de Euler, μ_n tiene $\varphi(n)$ raíces primitivas.*

Definición A.11. *(Polinomio ciclotómico). El n-ésimo polinomio ciclotómico $\Phi_n(z)$ tiene grado $\varphi(n)$ y está definido como se muestra a continuación:*

$$\Phi_n(z) = \prod_{l=0}^{\varphi(n)-1} (z - z_l^*)$$

donde z_l^* son las raíces n-ésimas primitivas de la unidad.

Dado que las raíces de $\Phi_n(z)$ forman un subconjunto de μ_n, donde μ_n es el conjunto de raíces del polinomio $z^n - 1$, entonces $\Phi_n(z)|z^n - 1$.

Definición A.12. *(Grupo ciclotómico). Sea p un número primo y sea $\mathbb{F}_{p^n}^*$ el grupo multiplicativo de un campo finito de característica p, el n-ésimo grupo ciclotómico $\mathbb{G}_{\Phi_n(p)}$ es un subgrugo de $\mathbb{F}_{p^n}^*$, definido por:*

$$\mathbb{G}_{\Phi_n(p)} = (\{\alpha \in \mathbb{F}_{p^n}^* | \alpha^{\Phi_n(p)} = 1\}, \ \cdot \ , 1)$$

A.5. Rejilla (*Lattice*)

En esta sección se definen de manera general los conceptos principales para el estudio de las rejillas [41, 106], las cuales son empleadas para el cálculo rápido de la función picadillo hacia $\mathbb{G}_2$ [40].

Definición A.13. *(Espacio Vectorial). Un espacio vectorial V es un subconjunto de $\mathbb{R}^m$, el cual es cerrado bajo la operación de suma y bajo la multiplicación escalar por elementos en $\mathbb{R}$, es decir:*

$$\forall \ w_1, w_2 \in V \ \ y \ \ \forall \ \alpha_1, \alpha_2 \in \mathbb{R}, \text{ se cumple que: } \ \alpha_1 w_1 + \alpha_2 w_2 \in V$$

El conjunto de vectores $w_1, w_2, \ldots, w_n \in V$ es *linealmente independiente*, si la única forma en que se satisfaga igualdad

$$\alpha_1 w_1 + \alpha_2 w_2 + \cdots + \alpha_n w_n = 0,$$

es si y sólo si $\alpha_1 = \alpha_2 = \cdots = \alpha_n = 0$, en donde $\alpha_i \in \mathbb{R}$, para $1 \leq i \leq n$. .

Definición A.14. *(Base de un espacio vectorial). La base de V es el conjunto de vectores linealmente independientes $v_1, \ldots, v_n$, tales que, todo vector $w \in V$ puede ser representado como una combinación lineal de v_i, es decir:*

$$w = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n,$$

donde $\alpha_i \in \mathbb{R}$, para $1 \leq i \leq n$.

Definición A.15. *(Rejilla). Sea $v_1, \ldots, v_n \in \mathbb{R}^m$ la base del espacio vectorial V, la rejilla L generada por dicha base, es el conjunto de vectores formados por la combinación lineal de $v_1, ..., v_n$ con coeficientes en $\mathbb{Z}$, es decir $L \subset V$.*

Uno de los problemas fundamentales en rejillas, consiste en encontrar el vector $w \in L$ con la menor norma euclidiana $\|w\|$. Dada la base $v_1, ..., v_n$ de L, el algoritmo LLL de Lenstra, Lenstra y Lovasz [41], obtiene n vectores $w_i \in L$ de m dimensiones, tales que:

$$\|w_i\| = \left(\sum_{j=0}^{m-1} |w_{ij}|^2 \right)^{1/2}$$

es mínima.

A.6. Morfismos

Un morfismo (también conocido como homomorfismo) es una proyección entre dos estructuras matemáticas. Existen diferentes tipos de morfismos:

- Monomorfismo. Un monomorfismo de X a Y está denotado como $f : X \to Y$. Para todos los morfismos $g_1, g_2 : Z \to X$, se cumple que $f \circ g_1 = f \circ g_2$.
- Isomorfismo. $f : X \to Y$ es un isomorfismo, si existe un morfismo $g : Y \to X$.
- Endomorfismo. Es un morfismo de un objeto matemático a sí mismo.
- Automorfismo. Es un endomorfismo invertible, es decir, es un isomorfismo a sí mismo.

B

Algoritmos para el cálculo eficiente de primitivas criptográficas

En este capítulo se presentan los algoritmos utilizados para la construcción de la torre de campos en la Sección B.1, de acuerdo a la Ecuación 5.1 presentada en el Capítulo 5. También se presentan los algoritmos necesarios para el cálculo de la aritmética de curvas elípticas Sección B.2 y por último se muestran los algoritmos para el cálculo del emparejamiento Sección B.3.

B.1. Aritmética de campos finitos

En esta sección se presentan los algoritmos utilizados para el cálculo de la aritmética en los campos: $\mathbb{F}_p$ Sección B.1.1, $\mathbb{F}_{p^2}$ Sección B.1.2, $\mathbb{F}_{p^4}$ Sección B.1.3, $\mathbb{F}_{p^6}$ Sección B.1.4, $\mathbb{F}_{p^{12}}$ Sección B.1.5 y $\mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$ Sección B.1.6.

B.1.1. Aritmética en $\mathbb{F}_p$

En general, un número entero de 256 bits no puede ser definido a través de los tipos de datos primitivos del lenguaje de programación C, es por ello que se utilizan otras técnicas para su uso y representación. Considerando un elemento $a \in \mathbb{F}_p$ con $|a| \simeq |p|$, donde $|a|$ el tamaño en bits de a. Entonces, a puede ser representado como un arreglo de n elementos:

$$a = (a_0, a_1, \ \ldots\ , a_{n-1}),$$

en donde $a = a_0 + a_1 2^w + a_2 2^{w \cdot 2} + ... + a_{n-1} 2^{w \cdot n-1}$, $n = [(\lfloor log_2\ p \rfloor + 1)/w]$ y $|a_i| = w$ y w que representa el tamaño de palabra, que puede ser de 32 o 64 bits dependiendo de la arquitectura del procesador. En particular, para $|a| = 256$ y una arquitectura

de 64 bits, a puede ser representado por medio de un arreglo de 4 palabras,

$$a = (a_0, a_1, a_2, a_3).$$

Adición y sustracción

Considerando los elementos $a, b \in \mathbb{F}_p$, la operación adición y sustracción en el campo finito $\mathbb{F}_p$ es $(a \pm b) \bmod p$. La forma de realizar estas operaciones utilizando la representación anterior, se muestra en el Algoritmo B.1 y en el Algoritmo B.2.

Algoritmo B.1 Adición en $\mathbb{F}_p$

Entrada: Primo p y $a, b \in [0, p-1]$, donde $a = (a_0, a_1, \ldots, a_{n-1})$ y $b = (b_0, b_1, \ldots, b_{n-1})$
Salida: $c = (a + b) \bmod p$.
1: $(c_0, \text{acarreo}) \leftarrow \text{Suma}(a_0, b_0)$
2: **for** $i = 1 \rightarrow n - 1$ **do**
3: $\quad (c_i, \text{acarreo}) \leftarrow \text{SumaAcarreo}(a_i, b_i, \text{acarreo})$
4: **end for**
5: $c \leftarrow (c_0, c_1, \ldots, c_{n-1})$
6: **if** existe acarreo o $c > p$ **then**
7: $\quad c \leftarrow c - p$ (Algoritmo B.2)
8: **end if**
9: **return** c

Algoritmo B.2 Sustracción en $\mathbb{F}_p$

Entrada: Primo p, $a, b \in [0, p-1]$, donde $a = (a_0, a_1, \ldots, a_{n-1})$ y $b = (b_0, b_1, \ldots, b_{n-1})$
Salida: $c = (a - b) \bmod p$.
1: $(c_0, \text{préstamo}) \leftarrow \text{Resta}(a_0, b_0)$
2: **for** $i = 1 \rightarrow n - 1$ **do**
3: $\quad (c_i, \text{préstamo}) \leftarrow \text{RestaPréstamo}(a_i, b_i, \text{préstamo})$
4: **end for**
5: $c \leftarrow (c_0, c_1, \ldots, c_{n-1})$
6: **if** préstamo **then**
7: $\quad c \leftarrow c + p$ (Algoritmo B.1)
8: **end if**
9: **return** c

Multiplicación

El multiplicador de Montgomery [107] es el método más eficiente conocido para realizar la multiplicación modular. El método reemplaza la operación de "dividir por p", en la reducción modulo p, por "divisiones entre r", en donde $r = 2^k$ con $k - 1 < |p| < k$. Para ello se realiza una proyección de los valores $a \in \mathbb{F}_p$ a valores $\tilde{a} \in \mathbb{F}_p$ utilizando la ecuación: $\tilde{a} = a \cdot r \bmod p$. Al valor $\tilde{a}$ se le denomina p-residuo de a y se dice que el valor $\tilde{a}$ se encuentra en el dominio de Montgomery. El producto de Montgomery de dos p-residuos se define como:

$$\text{MontPr}(\tilde{a}, \tilde{b}) = \tilde{a} \cdot \tilde{b} \cdot r^{-1} \bmod p.$$

Encontrar a dado su p-residuo $\tilde{a}$ puede realizarse a través del producto de Montgomery por el entero unidad:

$$\text{MontPr}(\tilde{a}, 1) = \tilde{a} \cdot 1 \cdot r^{-1} \bmod p = a \bmod p.$$

El multiplicador de Montgomery se muestra en el Algoritmo B.3 y para su implementación es necesario pre-calcular p' tal que $r \cdot r^{-1} - p \cdot p' = 1$, el cual puede ser

obtenido a través del algoritmo extendido de Euclides. El punto principal de este algoritmo radica en la obtención de $u = (t + (t \cdot p' \bmod r) \cdot p)/r$. Para entender su funcionamiento considere los siguientes aspectos:

1. Suponga que $m = t \cdot p' \pmod r$. Entonces $m \cdot p \equiv t \cdot p' \cdot p \pmod r$. Ahora bien, tenemos que $p' \equiv -p^{-1} \pmod r$, por lo que $t \cdot p' \cdot p \equiv -t \pmod r$. Por tanto $(t + m \cdot p) \equiv 0 \pmod r$, lo que significa que $(t + m \cdot p)$ es divisible entre r y que u es un número entero.

2. Además $u \cdot r = (t + m \cdot p) \equiv t \bmod p$, entonces $u \equiv t \cdot r^{-1} \pmod p$. Dado que $t = \tilde{a} \cdot \tilde{b}$, encontramos el valor buscado $u = \tilde{a} \cdot \tilde{b} \cdot r^{-1} \pmod p$.

3. Suponiendo que $0 \leq t < p \cdot r$, entonces $u \leq 2p$ (ya que $m < r$), por lo que la salida siempre es menor que p.

En este método, tanto la multiplicación como la división por r son operaciones rápidas, ésto se debe a que r es una potencia de 2. Sin embargo, requiere del cálculo de los p-residuos y de la proyección del producto de Montgomery a los enteros, por lo que este método usualmente se utiliza cuando se van a realizar varias multiplicaciones en sucesión.

Algoritmo B.3 Multiplicador de Montgomery

Entrada: Primo p, p', $r = 2^k$ y dos p-residuos $\tilde{a}$, $\tilde{b}$
Salida: MontPr($\tilde{a}, \tilde{b}$)
1: $t \leftarrow \tilde{a} \cdot \tilde{b}$
2: $u \leftarrow (t + (t \cdot p' \bmod r) \cdot p)/r$
3: **if** $u > p$ **then**
4: **return** $u - p$
5: **else**
6: **return** u
7: **end if**
8: **return** u

Inverso multiplicativo

Esta operación definida como $a^{-1} = 1/a \bmod p$, con $a \in \mathbb{F}_p$, consiste en encontrar $x \in \mathbb{F}_p$, tal que $a \cdot x = 1 \bmod p$, y puede realizarse con el método conocido como inverso multiplicativo de Montgomery, en el cual se selecciona un valor $r \geq 2^n$, en donde $n = \lfloor log_2\ p \rfloor$ y considerando $\tilde{a} = a \cdot r \bmod p$, el inverso multiplicativo de $\tilde{a}$ se define como $\tilde{a}^{-1} = a^{-1} \cdot r \bmod p$. El inverso multiplicativo de Montgomery [20] se realiza a través de dos pasos, el primero consiste en encontrar una inversión parcial $a^{-1}2^k$ con $k \in [n, 2n]$, tal y como se muestra en el Algoritmo B.4, el cual es una modificación del algoritmo extendido de Euclides. Mientras que el segundo paso consiste en encontrar el inverso multiplicativo de Montgomery, como se muestra en el Algoritmo B.5.

Exponenciación

Considerando los elementos $a \in \mathbb{F}_p$ y $k \in \mathbb{Z}$, la exponenciación modular se define como $a^k \bmod p$. Realizar esta operación requiere de multiplicaciones consecutivas, por lo que puede ser calculado utilizando el multiplicador de Montgomery como muestra el Algoritmo B.6.

Algoritmo B.4 Inversión parcial de Montgomery

Entrada: $p > 0$, $a \in [1, p-1]$ y $n = \lfloor log_2\ p \rfloor$
Salida: (x, k) tal que $n \leq k \leq 2n$ y $x = a^{-1}2^k \bmod p$
1: $u \leftarrow a$, $v \leftarrow p$, $x_1 \leftarrow 1$, $x_2 \leftarrow 0$, $k \leftarrow 0$
2: **while** $v > 0$ **do**
3: **if** v es par **then**
4: $v \leftarrow v/2$, $x_1 \leftarrow 2x_1$
5: **else if** u es par **then**
6: $u \leftarrow u/2$, $x_2 \leftarrow 2x_2$
7: **else if** $v \geq u$ **then**
8: $v \leftarrow (v-u)/2$, $x_2 \leftarrow x_2 + x_1$, $x_1 \leftarrow 2x_1$
9: **else**
10: $u \leftarrow (u-v)/2$, $x_1 \leftarrow x_2 + x_1$, $x_2 \leftarrow 2x_2$
11: **end if**
12: $k \leftarrow k + 1$
13: **end while**
14: **if** $x_1 > p$ **then**
15: **return** $x_1 \leftarrow x_1 - p$
16: **end if**
17: **return** (x_1, k)

Algoritmo B.5 Inverso multiplicativo de Montgomery

Entrada: Un primo $p > 2$, $n = \lceil \log_2(p) \rceil$, $\tilde{a} = a \cdot r \bmod p$, donde $r = 2^n$
Salida: $a^{-1}r \bmod p$
1: Encontrar (x, k), tal que $x = \tilde{a}^{-1}2^k \bmod p$, $n \leq k \leq 2n$ (Algoritmo B.4)
2: **if** $k < n$ **then**
3: $x \leftarrow \text{MontPr}(x, r^2)$
4: $k \leftarrow x + n$
5: **end if**
6: $x \leftarrow \text{MontPr}(x, r^2)$
7: $x \leftarrow \text{MontPr}(x, 2^{2n-k})$
8: **return** x.

Algoritmo B.6 Exponenciación de Montgomery

Entrada: $p > 0$, $r = 2^n$, $x \in [1, p-1]$, $e = (e_l, ..., e_0)_2$
Salida: $x^e \bmod p$
1: $\bar{x} \leftarrow x \cdot r \bmod p$
2: $A \leftarrow r \bmod p$
3: **for** $i = l \rightarrow 0$ **do**
4: $A \leftarrow \text{MontPr}(A, A)$
5: **if** $e_i == 1$ **then**
6: $A \leftarrow \text{MontPr}(A, \bar{x})$
7: **end if**
8: **end for**
9: **return** $A \leftarrow \text{MontPr}(A, 1)$

B.1.2. Aritmética en $\mathbb{F}_{p^2}$

La extensión cuadrática se define como: $\mathbb{F}_{p^2} = \mathbb{F}_p[u]/(u^2 - \beta)$, donde $\beta = -1$. Un elemento $A \in \mathbb{F}_{p^2}$ puede ser visto como $A = a_0 + a_1 u$, donde $a_0, a_1 \in \mathbb{F}_p$. De acuerdo a esta representación se definen las siguientes operaciones:

Adición y sustracción

Dados los elementos A y $B \in \mathbb{F}_{p^2}$, el cálculo de $C = A \pm B$ requiere de dos operaciones de adición o sustracción en el campo $\mathbb{F}_p$, como se muestra en el Algoritmo B.7 y en el Algoritmo B.8.

Algoritmo B.7 Adición en $\mathbb{F}_{p^2}$

Entrada: $A = a_0 + a_1 u$, $B = b_0 + b_1 u$, $A, B \in \mathbb{F}_{p^2}$
Salida: $C = A + B$, $C \in \mathbb{F}_{p^2}$
1: $c_0 \leftarrow a_0 + b_0$
2: $c_1 \leftarrow a_1 + b_1$
3: **return** $C = c_0 + c_1 u$

Algoritmo B.8 Sustracción en $\mathbb{F}_{p^2}$

Entrada: $A = a_0 + a_1 u,\ \ B = b_0 + b_1 u,\ A, B \in \mathbb{F}_{p^2}$
Salida: $C = A - B,\ C \in \mathbb{F}_{p^2}$
1: $c_0 \leftarrow a_0 - b_0$
2: $c_1 \leftarrow a_1 - b_1$
3: **return** $C = c_0 + c_1 u$

Multiplicación

La multiplicación de dos elementos $A, B \in \mathbb{F}_{p^2}$, se define como:

$$(a_0 + a_1 u) \cdot (b_0 + b_1 u) = a_0 b_0 + a_1 b_1 \beta + (a_0 b_1 + a_1 b_0) u,$$

usando el método de Karatsuba, tenemos que:

$$(a_0 b_1 + a_1 b_0) = (a_0 + a_1) \cdot (b_0 + b_1) - a_0 b_0 - a_1 b_1.$$

El Algoritmo B.9 muestra la multiplicación utilizando esta observación.

Algoritmo B.9 Multiplicación en $\mathbb{F}_{p^2}$

Entrada: $A = a_0 + a_1 u,\ \ B = b_0 + b_1 u,\ A, B \in \mathbb{F}_{p^2}$
Salida: $C = A \cdot B,\ C \in \mathbb{F}_{p^2}$
1: $t_0 \leftarrow a_0 \cdot b_0$
2: $t_1 \leftarrow a_1 \cdot b_1$
3: $c_0 \leftarrow t_0 + t_1 \beta$
4: $c_1 \leftarrow (a_0 + a_1) \cdot (b_0 + b_1) - t_0 - t_1$
5: **return** $C = c_0 + c_1 u$

Algunas veces es necesario realizar la multiplicación $A \cdot B$ con $B = b_0 + 0u$. El uso del algoritmo anterior provocaría el cálculo innecesario de multiplicaciones, en el Algoritmo B.10 se describe esta operación aprovechando el conocimiento de que el segundo término de B es cero.

Algoritmo B.10 Multiplicación por $B = b_0 + 0u \in \mathbb{F}_{p^2}$

Entrada: $A = a_0 + a_1 u,\ B = b_0 + 0u$ con $A, B \in \mathbb{F}_{p^2}$
Salida: $C = A \cdot B,\ C \in \mathbb{F}_{p^2}$
1: $c_0 \leftarrow a_0 \cdot b_0$
2: $c_1 \leftarrow a_1 \cdot b_0$
3: **return** $C = c_0 + c_1 u$

Elevación al cuadrado

El cálculo de $C = A^2$, donde $A, C \in \mathbb{F}_{p^2}$, es calculada de forma análoga al método complejo de elevación al cuadrado, a través de la siguiente identidad:

$$(a_0 + a_1 u)^2 = (a_0 - \beta a_1) \cdot (a_0 - a_1) + (\beta + 1) a_0 a_1 + 2 a_0 a_1 u.$$

En el caso que $\beta = -1$ se tiene que:

$$(a_0 + a_1 u)^2 = (a_0 + a_1) \cdot (a_0 - a_1) + 2 a_0 a_1 u.$$

Por lo que la elevación al cuadrado puede ser calculada con un costo de 2 multiplicaciones en el campo base, como se muestra en el Algoritmo B.11.

Algoritmo B.11 Elevación al cuadrado en $\mathbb{F}_{p^2}$

Entrada: $A = a_0 + a_1 u$, $A \in \mathbb{F}_{p^2}$
Salida: $C = A^2$, $C \in \mathbb{F}_{p^2}$
1: $c_0 \leftarrow a_0 - a_1$
2: $c_2 \leftarrow a_0 + a_1$
3: $c_1 \leftarrow 2a_0 a_1$
4: $c_0 \leftarrow c_0 \cdot c_2$
5: **return** $C = c_0 + c_1 u$

Inverso multiplicativo

El inverso de un elemento en el grupo multiplicativo del campo finito $\mathbb{F}_{p^2}$, es calculado mediante la siguiente identidad:

$$(a_0 + a_1 u)^{-1} = \frac{a_0 - a_1 u}{(a_0 - a_1 z) \cdot (a_0 + a_1 z)} = \frac{a_0}{a_0^2 - a_1^2 \beta} - \frac{a_1 u}{a_0^2 - a_1^2 \beta}.$$

Esta operación tiene un costo de 4 multiplicaciones y una inversión en el campo $\mathbb{F}_p$ de acuerdo con el Algoritmo B.12.

Algoritmo B.12 Inverso multiplicativo en $\mathbb{F}_{p^2}$

Entrada: $A = a_0 + a_1 u$, $A \in \mathbb{F}_{p^2}$
Salida: $C = A^{-1}$, $C \in \mathbb{F}_{p^2}$
1: $t_0 \leftarrow a_0^2$
2: $t_1 \leftarrow a_1^2$
3: $t_0 \leftarrow t_0 - \beta t_1$
4: $t_1 \leftarrow t_0^{-1}$
5: $c_0 \leftarrow a_0 \cdot t_1$
6: $c_1 \leftarrow -a_1 \cdot t_1$
7: **return** $C = c_0 + c_1 u$

B.1.3. Aritmética en $\mathbb{F}_{p^4}$

La extensión cuadrática del campo $\mathbb{F}_{p^2}$ se define como como $\mathbb{F}_{p^4} = \mathbb{F}_{p^2}[V]/(V^2 - \xi)$, en donde $\xi = u + 1$. Un elemento $A \in \mathbb{F}_{p^4}$ puede ser visto como $A = a_0 + a_1 V$ donde $a_0, a_1 \in \mathbb{F}_{p^2}$. La operación de elevación al cuadrado se muestra en el Algoritmo B.13, ocupando el método de Karatsuba.

Algoritmo B.13 Elevación al cuadrado en $\mathbb{F}_{p^4}$

Entrada: $A = a_0 + a_1 u$, $A \in \mathbb{F}_{p^4}$
Salida: $C = A^2$, $C \in \mathbb{F}_{p^4}$
1: $t_0 \leftarrow a_0^2$
2: $t_1 \leftarrow a_1^2$
3: $c_0 \leftarrow t_0 + t_1 \xi$
4: $c_1 \leftarrow a_0 + a_1$
5: $c_1 \leftarrow c_1^2 - t_0 - t_1$
6: **return** $C = c_0 + c_1 u$

B.1.4. Aritmética en $\mathbb{F}_{p^6}$

La extensión cúbica del campo $\mathbb{F}_{p^2}$ se define como $\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[V]/(V^3 - \xi)$, en donde $\xi = u + 1$. En esta extensión un elemento $A \in \mathbb{F}_{p^6}$ puede ser visto como

$A = a_0 + a_1V + a_2V^2$, en donde $a_0, a_1, a_2 \in \mathbb{F}_{p^2}$. De acuerdo a esta representación se definen las siguientes operaciones:

Adición y sustracción

De la misma forma que en la aritmética sobre $\mathbb{F}_{p^2}$, las operaciones de adición y sustracción en $\mathbb{F}_{p^6}$ se realizan elemento a elemento como se muestra en el Algoritmo B.14 y en el Algoritmo B.15.

Algoritmo B.14 Adición en $\mathbb{F}_{p^6}$

Entrada: $A = a_0 + a_1V + a_2V^2$, $B = b_0 + b_1V + b_2V^2$, $A, B \in \mathbb{F}_{p^6}$
Salida: $C = A + B$, $C \in \mathbb{F}_{p^6}$
1: $c_0 \leftarrow a_0 + b_0$
2: $c_1 \leftarrow a_1 + b_1$
3: $c_2 \leftarrow a_2 + b_2$
4: **return** $C = c_0 + c_1V + c_2V^2$

Algoritmo B.15 Sustracción en $\mathbb{F}_{p^6}$

Entrada: $A = a_0 + a_1V + a_2V^2$, $B = b_0 + b_1V + b_2V^2$, $A, B \in \mathbb{F}_{p^6}$
Salida: $C = A - B$, $C \in \mathbb{F}_{p^6}$
1: $c_0 \leftarrow a_0 - b_0$
2: $c_1 \leftarrow a_1 - b_1$
3: $c_2 \leftarrow a_2 - b_2$
4: **return** $C = c_0 + c_1V + c_2V^2$

Multiplicación

El producto de los elementos $A = a_0 + a_1V + a_2V^2$ y $B = b_0 + b_1V + b_2V^2 \in \mathbb{F}_{p^6}$ se calcula a través del método de Karatsuba, con un costo de 6 multiplicaciones en el campo $\mathbb{F}_{p^2}$ como se observa en el Algoritmo B.16.

Algoritmo B.16 Multiplicación en $\mathbb{F}_{p^6}$

Entrada: $A = a_0 + a_1V + a_2V^2$, $B = b_0 + b_1V + b_2V^2$, $A, B \in \mathbb{F}_{p^6}$
Salida: $C = A \cdot B$, $C \in \mathbb{F}_{p^6}$
1: $v_0 \leftarrow a_0 \cdot b_0$
2: $v_1 \leftarrow a_1 \cdot b_1$
3: $v_2 \leftarrow a_2 \cdot b_2$
4: $c_0 \leftarrow ((a_1 + a_2) \cdot (b_1 + b_2) - v_1 - v_2) \cdot \xi + v_0$
5: $c_1 \leftarrow (a_0 + a_1) \cdot (b_0 + b_1) - v_0 - v_1 + \xi \cdot v_2$
6: $c_2 \leftarrow (a_0 + a_2) \cdot (b_0 + b_2) - v_0 - v_2 + v_1$
7: **return** $C = c_0 + c_1V + c_2V^2$

En el caso en que $B = b_0 + 0V + 0V^2$ esta operación se realiza de acuerdo al Algoritmo B.17. Mientras que cuando $B = b_0 + b_1V + 0V^2$ se utiliza el Algoritmo B.18.

Elevación al cuadrado

El Algoritmo B.19 describe el método basado en [98], para la obtención del cuadrado en extensiones cúbicas. Este algoritmo tiene un costo de 2 multiplicaciones y 3 cuadrados en el campo $\mathbb{F}_{p^2}$.

Algoritmo B.17 Multiplicación por $B = b_0 + 0V + 0V^2 \in \mathbb{F}_{p^6}$

Entrada: $A = a_0 + a_1V + a_2V^2$, $B = b_0 + 0V + 0V^2$ con $A, B \in \mathbb{F}_{p^6}$
Salida: $C = A \cdot B$, $C \in \mathbb{F}_{p^6}$
1: $c_0 \leftarrow a_0 \cdot b_0$
2: $c_1 \leftarrow a_1 \cdot b_0$
3: $c_2 \leftarrow a_2 \cdot b_0$
4: **return** $C = c_0 + c_1V + c_2V^2$

Algoritmo B.18 Multiplicación por $B = b_0 + b_1V + 0V^2 \in \mathbb{F}_{p^6}$

Entrada: $A = a_0 + a_1V + a_2V^2$, $B = b_0 + b_1V + 0V^2$ con $A, B \in \mathbb{F}_{p^6}$
Salida: $C = A \cdot B$, $C \in \mathbb{F}_{p^6}$
1: $t_0 \leftarrow a_0 \cdot b_0$
2: $t_1 \leftarrow a_1 \cdot b_1$
3: $c_0 \leftarrow ((a_1 + a_2) \cdot b_1 - t_1) \cdot \xi + t_0$
4: $c_1 \leftarrow (a_0 + a_1) \cdot (b_0 + b_1) - t_0 - t_1$
5: $c_2 \leftarrow a_2 \cdot b_0 + t_1$
6: **return** $C = c_0 + c_1V + c_2V^2$

Inverso multiplicativo

Considerando $A = a_0 + a_1V + a_2V^2$, $A \in \mathbb{F}_{p^6}$, la operación de inverso multiplicativo $C = A^{-1}$ se define como:

$$C = A^{-1} = (a_0 + a_1V + a_2V^2)^{-1} = (A + BV + CV^2)/F,$$

en donde:

$$A = a_0^2 - \xi a_1 a_2, \qquad B = \xi a_2^2 - a_0 a_1,$$
$$C = a_1^2 - a_0 a_2 \qquad y \qquad F = \xi a_1 C + a_0 A + \xi a_2 B.$$

Esta operación es calculada por medio del Algoritmo B.20 a un costo de 9 multiplicaciones, 3 cuadrados y un inverso en el campo $\mathbb{F}_{p^2}$.

Algoritmo B.19 Elevación al cuadrado en $\mathbb{F}_{p^6}$

Entrada: $A = a_0 + a_1V + a_2V^2$, $A \in \mathbb{F}_{p^6}$
Salida: $C = A^2$, $C \in \mathbb{F}_{p^6}$
1: $v_4 \leftarrow 2(a_0 \cdot a_1)$
2: $v_5 \leftarrow a_2^2$
3: $c_1 \leftarrow \xi \cdot v_5 + v_4$
4: $v_2 \leftarrow v_4 - v_5$
5: $v_3 \leftarrow a_0^2$
6: $v_4 \leftarrow a_0 - a_1 + a_2$
7: $v_5 \leftarrow 2(a_1 + a_2)$
8: $v_4 \leftarrow v_4^2$
9: $c_0 \leftarrow \xi \cdot v_5 + v_3$
10: $c_2 \leftarrow v_2 + v_4 + v_5 - v_3$
11: **return** $C = c_0 + c_1w + c_2w^2$

B.1.5. Aritmética en $\mathbb{F}_{p^{12}}$

La extensión cuadrática del campo $\mathbb{F}_{p^6}$ se define como $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^6}[W]/(W^2 - \gamma)$, en donde $\gamma = V$. Un elemento $A \in \mathbb{F}_{p^{12}}$ puede ser visto como $A = a_0 + a_1W$ donde $a_0, a_1 \in \mathbb{F}_{p^6}$. De acuerdo a esta representación se definen las siguientes operaciones:

Algoritmo B.20 Inverso multiplicativo en $\mathbb{F}_{p^6}$

Entrada: $A = a_0 + a_1 V + a_2 V^2$, $A \in \mathbb{F}_{p^6}$
Salida: $C = A^{-1}$, $C \in \mathbb{F}_{p^6}$
1: $t_0 \leftarrow a_0^2$
2: $t_1 \leftarrow a_1^2$
3: $t_2 \leftarrow a_2^2$
4: $t_3 \leftarrow a_0 \cdot a_1$
5: $t_4 \leftarrow a_0 \cdot a_2$
6: $t_5 \leftarrow a_1 \cdot a_2$
7: $c_0 \leftarrow t_0 - \xi \cdot t_5$
8: $c_1 \leftarrow \xi t_2 - t_3$
9: $c_2 \leftarrow t_1 - t_4$
10: $t_6 \leftarrow a_0 \cdot c_0$
11: $t_6 \leftarrow t_6 + \xi \cdot a_2 \cdot c_1$
12: $t_6 \leftarrow t_6 + \xi \cdot a_1 \cdot c_2$
13: $t_6 \leftarrow t_6^{-1}$
14: $c_0 \leftarrow c_0 \cdot t_6$
15: $c_1 \leftarrow c_1 \cdot t_6$
16: $c_2 \leftarrow c_2 \cdot t_6$
17: **return** $C = c_0 + c_1 V + c_2 V^2$

Adición y sustracción

Dados los elementos A, B y $C \in \mathbb{F}_{p^{12}}$, el cálculo de $C = A \pm B$ requiere de dos sumas o restas, segun sea el caso, en el campo $\mathbb{F}_{p^6}$ como muestra el Algoritmo B.21 y el Algoritmo B.22.

Algoritmo B.21 Adición en $\mathbb{F}_{p^{12}}$

Entrada: $A = a_0 + a_1 W$, $B = b_0 + b_1 W$, $A, B \in \mathbb{F}_{p^{12}}$
Salida: $C = A + B$, $C \in \mathbb{F}_{p^{12}}$
1: $c_0 \leftarrow a_0 + b_0$
2: $c_1 \leftarrow a_1 + b_1$
3: **return** $C = c_0 + c_1 W$

Algoritmo B.22 Sustracción en $\mathbb{F}_{p^{12}}$

Entrada: $A = a_0 + a_1 W$, $B = b_0 + b_1 W$, $A, B \in \mathbb{F}_{p^{12}}$
Salida: $C = A - B$, $C \in \mathbb{F}_{p^{12}}$
1: $c_0 \leftarrow a_0 - b_0$
2: $c_1 \leftarrow a_1 - b_1$
3: **return** $C = c_0 + c_1 W$

Multiplicación

Los algoritmos de multiplicación son muy similares a los de $\mathbb{F}_{p^2}$, dado que $\mathbb{F}_{p^{12}}$ también es una extensión cuadrática. Considerando los elementos $A, B \in \mathbb{F}_{p^{12}}$ la operación de multiplicación se realiza por medio del Algoritmo B.23.

Algoritmo B.23 Multiplicación en $\mathbb{F}_{p^{12}}$

Entrada: $A = a_0 + a_1 W$, $B = b_0 + b_1 W$, $A, B \in \mathbb{F}_{p^{12}}$
Salida: $C = A \cdot B$, $C \in \mathbb{F}_{p^{12}}$
1: $t_0 \leftarrow a_0 \cdot b_0$
2: $t_1 \leftarrow a_1 \cdot b_1$
3: $c_0 \leftarrow t_0 + t_1 \gamma$
4: $c_1 \leftarrow (a_0 + a_1) \cdot (b_0 + b_1) - t_0 - t_1$
5: **return** $C = c_0 + c_1 W$

Para el caso de la multiplicación por $B = b_0 + b_1 W$, en donde $b_0 \in \mathbb{F}_{p^2}$ y $b_1 = b_{10} + b_{11}V + 0V^2 \in \mathbb{F}_{p^6}$, se realiza mediante el Algoritmo B.24.

Algoritmo B.24 Multiplicación por $B = b_0 + b_1 W$ con $b_0 \in \mathbb{F}_{p^2}$ y $b_1 = b_{10} + b_{11}V + 0V^2$

Entrada: $A = a_0 + a_1 W$, $B = b_0 + b_1 W$, $A, B \in \mathbb{F}_{p^{12}}$
donde $b_0 = b_{00} + 0V + 0V^2$ y $b_1 = b_{10} + b_{11}V + 0V^2$
Salida: $C = A \cdot B$, $C \in \mathbb{F}_{p^{12}}$
1: $t_0 \leftarrow a_0 \cdot b_0$ Algoritmo B.17
2: $t_1 \leftarrow a_1 \cdot b_1$ Algoritmo B.18
3: $c_0 \leftarrow t_0 + t_1 \gamma$
4: $t_2 \leftarrow (b_{00} + b_{10}) + b_{11}V + 0V^2$
5: $c_1 \leftarrow (a_0 + a_1) \cdot t_2$ Algoritmo B.18
6: $c_1 \leftarrow c_1 - t_0 - t_1$
7: **return** $C = c_0 + c_1 W$

Elevación al cuadrado

La operación de elevar al cuadrado en $\mathbb{F}_{p^{12}}$ hace uso del método complejo, como muestra el Algoritmo B.25.

Algoritmo B.25 Elevación en $\mathbb{F}_{p^{12}}$

Entrada: $A = a_0 + a_1 W$, $A \in \mathbb{F}_{p^{12}}$
Salida: $C = A^2$, $C \in \mathbb{F}_{p^{12}}$
1: $c_0 \leftarrow a_0 - a_1$
2: $c_3 \leftarrow a_0 + \gamma \cdot a_1$
3: $c_2 \leftarrow a_0 \cdot a_1$
4: $c_0 \leftarrow c_0 \cdot c_3 + c_2$
5: $c_1 \leftarrow 2c_2$
6: $c_2 \leftarrow \gamma \cdot c_2$
7: $c_0 \leftarrow c_0 + c_2$
8: **return** $C = c_0 + c_1 W$

Inverso multiplicativo

El Algoritmo B.26 realiza la inversión de u elemento $A \in \mathbb{F}_{p^{12}}$.

Algoritmo B.26 Inverso multiplicativo en $\mathbb{F}_{p^{12}}$

Entrada: $A = a_0 + a_1 W$, $A \in \mathbb{F}_{p^{12}}$
Salida: $C = A^{-1}$, $C \in \mathbb{F}_{p^{12}}$
1: $t_0 \leftarrow a_0^2$
2: $t_1 \leftarrow a_1^2$
3: $t_0 \leftarrow t_0 - \gamma t_1$
4: $t_1 \leftarrow t_0^{-1}$
5: $c_0 \leftarrow a_0 \cdot t_1$
6: $c_1 \leftarrow -a_1 \cdot t_1$
7: **return** $C = c_0 + c_1 W$

B.1.6. Aritmética en el grupo ciclotómico $\mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$

En el cálculo de la exponenciación final (Sección 2.3.10) se puede observar que el elemento $f = g^{p^6-1} \in \mathbb{F}_{p^{12}}$ pertenece al grupo ciclotómico $\mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$. Este hecho nos genera una gran ventaja, ya que en el grupo ciclotómico las operaciones requieren un menor número de operaciones.

Inverso multiplicativo

Los elementos α del grupo ciclotómico satisfacen $\alpha^{p^6+1} = 1$, lo que quiere decir que $\alpha^{-1} = \alpha^{p^6} = \bar{\alpha}$, es decir, la inversión de elementos en $\mathbb{F}_{p^{12}}$, se puede realizar mediante una conjugación. Considerando un elemento $\alpha = a_0 + a_1 W \in \mathbb{F}_{p^{12}}$ el inverso $\alpha^{-1} = \bar{\alpha}$ se define como $\alpha^{-1} = a_0 - a_1 W$.

Elevación al cuadrado

El campo $\mathbb{F}_{p^{12}}$ puede ser definido como una extensión cúbica del campo $\mathbb{F}_{p^4}$, es decir, $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^4}[z]/(z^3 - \gamma)$, por lo que un elemento $\alpha \in \mathbb{F}_{p^{12}}$ puede representarse como: $\alpha = a + bz + cz^2$ con $a = a_0 + a_1 V$, $b = b_0 + b_1 V$, $c = c_0 + c_1 V$, en donde $a, b, c \in \mathbb{F}_{p^4}$ y $a_i, b_i, c_i \in \mathbb{F}_{p^2}$, con $i = 0, 1$. La obtención del cuadrado de un elemento $\alpha \in \mathbb{F}_{p^{12}}$ se realiza con 3 multiplicaciones y 3 cuadrados en $\mathbb{F}_{p^2}$ utilizando la forma clásica, como se ve a continuación:

$$\begin{aligned} \alpha^2 &= (a + bz + cz^2)^2 \\ &= (a^2 + 2bc\gamma) + (2ab + c^2\gamma)z + (2ac + b^2)z^2 \end{aligned} \tag{B.1}$$

Sin embargo, de acuerdo con los trabajos de Granger y Scott [108] y Karabina [109] se sabe que si α pertenece al grupo ciclotómico, la obtención del cuadrado α^2 puede realizarse con un menor número de operaciones.

Cuadrados de Granger y Scott

Sea $\alpha \in \mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$ un elemento del grupo ciclotómico de la forma $\alpha = a + bz + cz^2$, los elementos a, b, c satisfacen las siguientes identidades:

$$bc = a^2 - \bar{a}/\gamma, \qquad ab = c^2\gamma - \bar{b}, \qquad ac = b^2 - \bar{c},$$

en donde $\bar{a}, \bar{b}$ y $\bar{c}$ corresponden a los conjugados de a, b y c respectivamente.

Si se sustituyen estos en la Ecuación B.1, se tiene que la operación α^2 puede ser calculada con un costo de 3 cuadrados y 12 sumas en el campo $\mathbb{F}_{p^4}$, como se muestra a continuación (Algoritmo B.27):

$$\alpha^2 = (3a^2 - 2\bar{a}) + (3c^2\gamma + 2\bar{b})z + (3b^2 - 2\bar{c})z^2$$

Cuadrados de Karabina

El algoritmo de Karabina propone el uso de cuadrados comprimidos, para el cual, dado un α su cuadrado comprimido es de la forma $BV + CV^2$, en donde $\alpha^2 = (A + Bz + Cz^2)$. Este algoritmo se compone de tres bloques:

1. El elemento $\alpha = a + bz + cz^2$ es comprimido mediante la función $\mathcal{C}(\alpha) = (b_0 + b_1 V)z + (c_0 + c_1 V)z^2$

Algoritmo B.27 Elevación al cuadrado en $\mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$

Entrada: $A = g+hW$, $A \in \mathbb{F}_{p^{12}}$, con $g = g_0+g_1v+g_2v^2$ y $h = h_0 + h_1v + h_2v^2$.
Salida: $C = A^2$, $C \in \mathbb{F}_{p^{12}}$
1: $t_{0,0}, t_{1,1} \leftarrow (g_0 + h_1V)^2$
2: $t_{0,1}, t_{1,2} \leftarrow (h_0 + g_2V)^2$
3: $t_{0,2}, tmp \leftarrow (g_1 + h_2V)^2$
4: $t_{1,0} \leftarrow tmp \cdot \xi$
5: $c_{0,0} \leftarrow -2g_0 + 3t_{0,0}$
6: $c_{0,1} \leftarrow -2g_1 + 3t_{0,1}$
7: $c_{0,2} \leftarrow -2g_2 + 3t_{0,2}$
8: $c_{1,0} \leftarrow 2h_0 + 3t_{1,0}$
9: $c_{1,1} \leftarrow 2h_1 + 3t_{1,1}$
10: $c_{1,2} \leftarrow 2h_2 + 3t_{1,2}$
11: $c_0 \leftarrow c_{0,0} + c_{0,1}v + c_{0,2}v^2$
12: $c_1 \leftarrow c_{1,0} + c_{1,1}v + c_{1,2}v^2$
13: **return** $C = c_0 + c_1W$

2. Se calcula $\mathcal{C}(\alpha^2) = (B_0 + B_1V)z + (C_0 + C_1V)z^2$ de acuerdo a las siguientes ecuaciones:

$$B_0 = 2b_0 + 3((c_0 + c_1)^2 - c_0^2 - c_1^2), \qquad B_1 = 3(c_0^2 + c_1^2\xi) - 2b_1,$$

$$C_0 = 3(b_0^2 + b_1^2\xi) - 2c_0, \qquad C_1 = 2c_1 + 3((b_0 + b_1)^2 - b_0^2 - b_1^2),$$

a un costo de 2 cuadrados en $\mathbb{F}_{p^4}$ como se muestra en el Algoritmo B.28.

3. Para calcular $\alpha^2 = \mathcal{D}(\mathcal{C}(\alpha^2)) = (A_0 + A_1V) + (B_0 + B_1V)z + (C_0 + C_1V)z^2$ se obtienen los elementos A_0 y A_1 de la siguiente forma:

$$A_0 = \frac{C_1^2\xi + 3C_0^2 - 2B_1}{4B_0}, \quad A_1 = (2A_1^2 + B_0C_1 - 3B_1C_0)\xi + 1 \qquad \text{si } B_0 \neq 0 \text{ y}$$

$$A_0 = \frac{2C_0C_1}{B_1}, \qquad A_1 = (2A_1^2 - 3B_1C_0)\xi + 1 \qquad \text{si } B_0 = 0.$$

El método de descompresión tiene un costo 3 cuadrados, 3 multiplicaciones y 1 inverso.

Algoritmo B.28 Cuadrados comprimidos en $\mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$

Entrada: $A = g+hW$, $A \in \mathbb{F}_{p^{12}}$, con $g = g_0+g_1v+g_2v^2$ y $h = h_0 + h_1v + h_2v^2$.
Salida: $\mathcal{C}(A^2)$
1: $t_{0,0}, t_{1,1} \leftarrow (h_0 + g_2V)^2$
2: $t_{0,1}, tmp \leftarrow (g_1 + h_2V)^2$
3: $t_{1,0} \leftarrow tmp \cdot \xi$
4: $c_{0,1} \leftarrow -2g_1 + 3t_{0,0}$
5: $c_{0,2} \leftarrow -2g_2 + 3t_{0,2}$
6: $c_{1,0} \leftarrow 2h_0 + 3t_{1,0}$
7: $c_{1,2} \leftarrow 2h_2 + 3t_{1,1}$
8: $c_0 \leftarrow c_{0,1}v + c_{0,2}v^2$
9: $c_1 \leftarrow c_{1,0} + c_{1,2}v^2$
10: **return** $C = c_0 + c_1W$

Debido al costo computacional de la descompresión, este algoritmo sólo se utiliza cuando se calculan varios cuadrados en sucesión, por ejemplo, en el cálculo de la exponenciación final.

B.2. Aritmética de curvas elípticas

En esta sección se presentan los algoritmos para el calculo de las suma y doblado de puntos (Sección B.2.1), así como de la operación conocida como multiplicación escalar (Sección B.2.2).

B.2.1. Suma y doblado de puntos

Como se explico en la Sección 2.1.3.3, la forma más eficiente de realizar la operación de suma de puntos es a través de coordenadas mixtas, es decir, un punto en afines (con $Z = 1$) y otro en jacobianas. Mientras que la mejor forma de realizar el doblado de un punto es en coordenadas jacobianas. A continuación se presentan los algoritmos para realizar estas operaciones.

Algoritmo B.29 Suma de puntos en coordenadas mixtas

Entrada: $P = (X_1 : Y_1 : Z_1)$ y $Q = (X_2 : Y_2 : 1)$
Salida: $R = P + Q = (X_3 : Y_3 : Z_3)$
1: $t_1 \leftarrow Z_1^2$
2: $t_2 \leftarrow Z_1 \cdot t_1$
3: $t_3 \leftarrow X_2 \cdot t_1$
4: $t_4 \leftarrow Y_2 \cdot t_2$
5: $t_5 \leftarrow t_3 - X_1$
6: $t_6 \leftarrow t_4 - Y_1$
7: $t_7 \leftarrow t_5^2$
8: $t_8 \leftarrow t_7 \cdot t_5$
9: $t_9 \leftarrow X_1 \cdot t_7$
10: $X_3 \leftarrow t_6^2 - (t_8 + 2t_9)$
11: $Y_3 \leftarrow t_6 \cdot (t_9 - X_3) - Y_1 \cdot t_8$
12: $Z_3 \leftarrow Z_1 \cdot t_5$
13: **return** $(X_3 : Y_3 : Z_3)$

Algoritmo B.30 Doblado de punto en coordenadas jacobianas

Entrada: $P = (X_1 : Y_1 : Z_1)$
Salida: $2P = (X_2 : Y_2 : Z_2)$
1: $t_1 \leftarrow Y_1^2$
2: $t_2 \leftarrow 4X_1 \cdot t_1$
3: $t_3 \leftarrow 8t_1^2$
4: $t_4 \leftarrow 3X_1^2 + aZ_1^4$
5: $X_2 \leftarrow t_4^2 - 2t_2$
6: $Y_2 \leftarrow t_4 \cdot (t_2 - X_2) - t_3$
7: $Z_2 \leftarrow 2Y_1 \cdot Z_1$
8: **return** $(X_2 : Y_2 : Z_2)$

B.2.2. Multiplicación escalar

La multiplicación escalar consiste en calcular $Q = kP$, en donde k es un escalar y P, Q puntos en la curva elíptica definida sobre un campo finito (Sección 2.1.2). A continuación se presentan los métodos utilizados para realizar esta operación de manera eficiente.

Método de ventana ω-NAF

Definición B.1. *(Representación ω-NAF) [20]. Sea el entero positivo $\omega \geq 2$. La representación ω-NAF de un entero positivo k es una expresión $k = \sum_{i=0}^{l-1} k_i 2^i$, en*

donde cada coeficiente k_i diferente de cero es un número impar, $|k_i| \leq 2^{\omega-1}$, $k_{l-1} \neq 0$, y además se cumple que al menos uno de los ω coeficientes consecutivos es diferente de cero. El tamaño del ω-NAF es denotado como l.

Teorema B.1. *(Propiedades del ω-NAF) [20] Sea k un entero positivo,*

I) *k tiene una única representación ω-NAF denotada por $NAF_\omega(k)$,*

II) *La longitud de $NAF_\omega(k)$ es a lo más igual a la longitud de la representación binaria de k más un bit.*

III) *La densidad promedio de dígitos diferentes de cero con respecto a la longitud l de la representación ω-NAF es $1/(\omega+1)$.*

En el Algoritmo B.31 se presenta la manera de calcular $\text{NAF}_\omega(k)$. En él, k mods 2^ω denota el entero u cuyo valor satisface $u \equiv k \pmod{2^\omega}$ y donde $-2^{\omega-1} \leq u < 2^{\omega-1}$. Los dígitos del $\text{NAF}_\omega(k)$ son obtenidos dividiendo continuamente el entero k por 2, lo que permite obtener el residuo r en el intervalo $[-2^{\omega-1}, 2^{\omega-1}-1]$. Si k es impar y el residuo $r = k$ mods 2^ω es seleccionado, entonces $(k-r)/2$ será un número divisible por $2^{\omega-1}$, asegurando que los siguientes $\omega - 1$ dígitos sean cero.

Algoritmo B.31 Cálculo de la representación ω-NAF de un entero positivo

Entrada: Tamaño de la ventana ω, entero positivo k
Salida: $NAF_\omega(k)$
1: $i \leftarrow 0$
2: **while** $k \geq 1$ **do**
3: **if** k es impar **then**
4: $k_i \leftarrow k$ mods 2^ω
5: $k \leftarrow k - k_i$
6: **else**
7: $k_i \leftarrow 0$
8: **end if**
9: $k \leftarrow k/2$
10: $i \leftarrow i + 1$
11: **end while**
12: **return** $(k_{i-1}, k_{i-2}, \ldots, k_1, k_0)$

En el Algoritmo B.32 se realiza la multiplicación escalar por medio de una modificación del de Horner usando la representación $\text{NAF}_\omega(k)$ en vez de la representación binaria de k, a un costo de,

$$\left[1D + (2^{\omega-2} - 1)A\right] + \left[\frac{m}{\omega+1}A + mD\right], \tag{B.2}$$

donde $m = |k|$, D representa la operación doblado de puntos y A la operación suma de puntos. El primer término de la Ecuación B.2 corresponde al costo del precomputo de los puntos P_i en el Algoritmo B.32, mientras que el segundo término representa el número de operaciones dentro del ciclo de la línea 4. Como se observa, el costo para obtener los puntos P_i tiende a crecer exponencialmente, por lo que se debe seleccionar un valor ω que permita obtener un costo óptimo en las operaciones.

Algoritmo B.32 Multiplicación escalar: método ω-NAF

Entrada: Tamaño de representación ω, entero positivo k, $P \in E(\mathbb{F}_q)$
Salida: kP
1: Obtener $NAF_\omega(k) = \sum_{i=0}^{l-1} k_i 2^i$ (Algoritmo B.31)
2: Obtener $P_i = iP$ para $i \in [1, 3, 5, \ldots, 2^{\omega-1} - 1]$
3: $Q \leftarrow \mathcal{O}$
4: **for** $i = l - 1 \rightarrow 0$ **do**
5: $\quad Q \leftarrow 2Q$
6: $\quad$ **if** $k_i \neq 0$ **then**
7: $\quad\quad$ **if** $k_i > 0$ **then**
8: $\quad\quad\quad Q \leftarrow Q + P_{k_i}$
9: $\quad\quad$ **else**
10: $\quad\quad\quad Q \leftarrow Q - P_{-k_i}$
11: $\quad\quad$ **end if**
12: $\quad$ **end if**
13: **end for**
14: **return** Q

Método GLV

Gallant, Lambert and Vanstone [22] introdujeron un método (denominado GLV) para acelerar la multiplicación escalar kP en $E(\mathbb{F}_p)[r]$ aprovechando ciertas propiedades de la curva elíptica. Este método funciona si dado un punto P, puede tenerse conocimiento de un múltiplo no trivial de él. Esta información está disponible si existe un endomorfismo eficientemente computable ψ sobre $E/\mathbb{F}_p$, tal que $\psi(P) = \lambda P$. Si ψ existe, es posible obtener kP de manera eficiente representando $k \equiv k_0 + k_1\lambda \pmod r$ con $|k_i| < \sqrt{r}$ y realizando la doble multiplicación $k_0P + k_1\psi(P)$.

Este método requiere un cierto nivel de precómputo y almacenamiento, sin embargo, su eficiencia se debe a que únicamente emplea la mitad de doblados de punto, en comparación con el método de la regla de Horner. Por otra parte, este método puede ser combinado con el método ω-NAF para reducir el número de adiciones de punto como se muestra en el Algoritmo B.33, con un costo de:

$$\left[2D + (2^{\omega-1} - 2)A\right] + \left[\frac{m}{\omega + 1}A + \frac{m}{2}D\right],$$

en donde $m = |k|$. El primer término corresponde al cálculo de los puntos P_i y Q_i, mientras que el segundo término corresponde al costo del ciclo principal.

Algoritmo B.33 Multiplicación escalar: método GLV

Entrada: Tamaño de ventana ω, entero positivo k, $P \in E(\mathbb{F}_p)$, endomorfismo ψ sobre $E(\mathbb{F}_p)$.
Salida: kP
1: $Q \leftarrow \psi(P)$ $(= \lambda P)$
2: Descomponer $k = u + v\lambda$ donde $|u| = |v| = l$
3: Calcular $NAF_\omega(u) = \sum_{i=0}^{l-1} u_i 2^i$ y $NAF_\omega(v) = \sum_{i=0}^{l-1} v_i 2^i$ (Algoritmo B.31)
4: Obtener $P_i = iP$, $Q_i = iQ$ para $i \in [\pm 1, \pm 3, \pm 5, \ldots, \pm(2^{\omega-1} - 1)]$.
5: $R \leftarrow \mathcal{O}$
6: **for** $i = l - 1 \rightarrow 0$ **do**
7: $\quad R \leftarrow 2R$
8: $\quad$ **if** $u_i \neq 0$ **then**
9: $\quad\quad R \leftarrow R + P_{u_i}$
10: $\quad$ **end if**
11: $\quad$ **if** $v_i \neq 0$ **then**
12: $\quad\quad R \leftarrow R + Q_{v_i}$
13: $\quad$ **end if**
14: **end for**
15: **return** R

Un punto importante en este método, es la descomposición del escalar k, en los valores k_0, k_1, tales que $k \equiv k_0 + k_1\lambda \pmod r$ con $|k_i| < \sqrt{r}$.

El problema de descomponer k, puede ser solucionado al resolver el problema del vector más cercano en la rejilla (Sección A.5):

$$L = \left\{ x \in Z^m : \sum_{i=0}^{m-1} x_i \lambda^i \equiv 0 \pmod{r} \right\} \tag{B.3}$$

En la familia de curvas BN (Sección 2.3.3.1), se construye la curva elíptica $E_1 : y^2 = x^3 + b$ definida sobre $\mathbb{F}_p$ con $\#E(\mathbb{F}_p) = r$. Dado $\beta \in \mathbb{F}_p$, entonces la proyección $\psi : E_1 \to E_1$ definida por $(x, y) \to (\beta x, y)$ y $\mathcal{O} \to \mathcal{O}$ es un endomorfismo definido sobre $\mathbb{F}_p$. Si $P \in E(\mathbb{F}_p)$ es un punto de orden primo r, entonces ψ actúa sobre $\langle P \rangle$ como una multiplicación por λ donde $\lambda^2 + \lambda \equiv -1 \bmod r$.

Para la descomposición del escalar k, la base para la rejilla L de la ecuación (B.3) para $\lambda = -36z^4 + 1$ es:

$$\begin{pmatrix} 6z^2 + 2z & -2z - 1 \\ -2z - 1 & -6z^2 + 4z + 1 \end{pmatrix}.$$

Posteriormente es necesario encontrar el vector x cercano a $w = (n, 0)$ en la rejilla L, para ello, primero se obtiene el vector $v = wB^{-1}$:

$$wB^{-1} = \begin{pmatrix} \dfrac{k(6z^2 + 4z + 1)}{r} & \dfrac{-k(2z + 1)}{r} \end{pmatrix}.$$

Entonces, podemos obtener v realizando multiplicaciones enteras y divisiones por r. Finalmente obtenemos el vector $u = w - vB$ cuyas entradas son los coeficientes k_0, k_1 de la descomposición de k.

Una mejora a la descomposición del escalar puede hacerse en la obtención del vector v. Como se observó, su cálculo requiere de multiplicaciones y divisiones por r, una manera de evitar la división es a través de un método inspirado en la reducción de Barret, en el que utilizando cierto nivel de precómputo se realizan divisiones por 2^m en lugar de r, como se muestra a continuación:

$$v' = w'B^{-1} = (v'_0, v'_1) = \left(\left\lfloor \frac{2^m(6z^2 + 4z + 1)}{r} \right\rfloor \quad \left\lfloor \frac{-2^m(2z + 1)}{r} \right\rfloor \right)$$

Dado que v' puede ser obtenido previamente, el vector v puede entonces obtenerse a través de multiplicaciones escalares y divisiones por 2^m:

$$v = \left(\left\lfloor \frac{kv'_0}{2^m} \right\rfloor, \left\lfloor \frac{kv'_1}{2^m} \right\rfloor \right).$$

Método GLS

El método GLS [23] puede considerarse como una versión del método GLV, el cual aprovecha el endomorfismo de Frobenius.

En el caso de las curvas BN, se puede construir una curva elíptica $E : y^2 = x^3 + b$ sobre $\mathbb{F}_p$ de orden r. El grado de encajamiento de esta curva es $k = 12$, por lo que se tiene que la curva enlazada E' de E se encuentra definida sobre $\mathbb{F}_{p^2}$. Por lo que se define $\mathbb{G}_2$ como el subgrupo $E'(\mathbb{F}_{p^2})[r]$. Por definición se tiene que E y E' son isomorfas de manera que $\phi : E'(\mathbb{F}_{p^{k/d}}) \longrightarrow E(\mathbb{F}_{p^k})$. Dado π_p el operador de Frobenius en E, la función $\psi = \phi^{-1}\pi_p\phi$ es un endormorfismo de E' tal que $\psi : E'(\mathbb{F}_{p^{k/d}}) \to E'(\mathbb{F}_{p^{k/d}})$. Además, para $Q \in E'(\mathbb{F}_{p^{k/d}})$, se tiene que $\psi^k(Q) = Q$, $\psi(Q) = pQ$ y $\Phi_k(\psi)(Q) = \infty$, en donde Φ_k es el k-ésimo polinomio ciclotómico. Cabe señalar que $\psi = \phi^{-1}\pi_p\phi$ satisface $\psi^4 - \psi^2 + 1 = 0$.

De acuerdo con [23], la base reducida para la rejilla L de la ecuación (B.3) con $\lambda = t - 1 = 6z^2$ es:

$$B = \begin{pmatrix} z+1 & z & z & -2z \\ 2z+1 & -z & -(z+1) & -z \\ 2z & 2z+1 & 2z+1 & 2z+1 \\ z-1 & 4z+2 & -(2z-1) & z-1 \end{pmatrix}$$

en donde B se obtiene a través del algoritmo LLL y cumple que $Bv = 0$ para el vector columna $v = (1, \lambda, \lambda^2, \lambda^3)$. Se necesita encontrar el vector x más cercano a $w = (k, 0, 0, 0)$ en la rejilla L, como se muestra a continuación:

$$wB^{-1} = \begin{pmatrix} \dfrac{k(2z^2+3z+1)}{r} & \dfrac{k(12z^3+8z^2+z)}{r} & \dfrac{k(6z^3+4z^2+z)}{r} & \dfrac{k(-z^2-z)}{r} \end{pmatrix}$$

Finalmente se calcula el vector $u = w - vB$ cuyos elementos u_i corresponden a los coeficientes k_i de la descomposición de k. En este caso, k puede descomponerse en 4 coeficientes tales que $k \equiv \sum_{i=0}^{3} u_i\lambda^i$, donde, todas las entradas en el vector u satisfacen que $|u_i| < r^{1/4}$.

También se pueden evitar las divisiones por r en el cálculo de v, para ello se precomputa el vector $v' = (v'_0, v'_1, v'_2, v'_3)$ y posteriormente se realizan divisiones entre 2^m con $m \geq |r|$, como se observa a continuación:

$$v' = \begin{pmatrix} \dfrac{2^m(2z^2+3z+1)}{r} & \dfrac{2^m(12z^3+8z^2+z)}{r} & \dfrac{2^m(6z^3+4z^2+z)}{r} & \dfrac{2^m(-z^2-z)}{r} \end{pmatrix}$$

y

$$v = \left(\left\lfloor \frac{kv'_0}{2^m} \right\rfloor, \left\lfloor \frac{kv'_1}{2^m} \right\rfloor, \left\lfloor \frac{kv'_2}{2^m} \right\rfloor, \left\lfloor \frac{kv'_3}{2^m} \right\rfloor \right).$$

El Algoritmo B.34 describe el método GLS, el cual requiere únicamente de la cuarta parte de doblados de punto que en el algoritmo que utiliza la regla de Horner y hace uso del método de ventana ω-NAF para reducir el número de adiciones de punto. El costo del algoritmo es:

$$\left[4D + 4(2^{\omega-2} - 1)A\right] + \left[\frac{m}{\omega+1}A + \frac{m}{4}D\right]$$

en donde $m = |k|$, A es el número de adiciones de punto y D el número de doblados de punto.

Algoritmo B.34 Multiplicación escalar: método GLS

Entrada: Tamaño de ventana ω, entero positivo k, $Q \in E(\mathbb{F}_{p^2})$, endomorfismo $\psi = \phi^{-1}\pi_p\phi$ sobre $E(\mathbb{F}_{p^2})$.
Salida: kQ
1: $R_0 \leftarrow Q$ $(= \lambda^0 Q)$
2: $R_1 \leftarrow \psi(Q)$ $(= \lambda^1 Q)$
3: $R_2 \leftarrow \psi^2(Q)$ $(= \lambda^2 Q)$
4: $R_3 \leftarrow \psi^3(Q)$ $(= \lambda^3 Q)$
5: Descomponen $k = k_0 + k_1\lambda + k_2\lambda^2 + k_3\lambda^3$ donde $|k_i| = l$
6: Calcular $NAF_\omega(k_i) = \sum_{j=0}^{l-1} k_{ij}2^j$ (Algoritmo B.31)
7: Obtener $R_i^j = iR_j$ para $i \in [\pm 1, \pm 3, \pm 5, \ldots, \pm(2^{\omega-1}-1)]$.
8: $R \leftarrow \mathcal{O}$
9: **for** $i = l-1 \rightarrow 0$ **do**
10: $R \leftarrow 2R$
11: **if** $k_{0i} \neq 0$ **then**
12: $R \leftarrow R + R^0_{k_{0i}}$
13: **end if**
14: **if** $k_{1i} \neq 0$ **then**
15: $R \leftarrow R + R^1_{k_{1i}}$
16: **end if**
17: **if** $k_{2i} \neq 0$ **then**
18: $R \leftarrow R + R^2_{k_{2i}}$
19: **end if**
20: **if** $k_{3i} \neq 0$ **then**
21: $R \leftarrow R + R^3_{k_{3i}}$
22: **end if**
23: **end for**
24: **return** R

B.3. Calculo del emparejamiento óptimo ate

El Algoritmo B.35 muestra el emparejamiento óptimo *ate* para curvas BN [97] de acuerdo a lo descrito en la Sección 2.3.8.1, en donde las operaciones de adición y sustracción de puntos son permitidas en el cálculo del emparejamiento.

Algoritmo B.35 Emparejamiento óptimo *ate* para curvas BN

Entrada: $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$
Salida: $\hat{a}_{opt}(Q, P)$
1: Escribir $s = 6z+2$ como $s = \sum_{i=0}^{l-1}$ con $s_i \in \{-1, 0, 1\}$
2: $T \leftarrow Q$, $f \leftarrow 1$
3: **for** $i = l-2 \rightarrow 0$ **do**
4: $f \leftarrow f^2 \cdot \ell_{T,T}(P)$, $T \leftarrow 2T$
5: **if** $s_i = 1$ **then**
6: $f \leftarrow f \cdot \ell_{T,Q}(P)$, $T \leftarrow T + Q$
7: **else if** $s_i = -1$ **then**
8: $f \leftarrow f \cdot \ell_{T,-Q}(P)$, $T \leftarrow T - Q$
9: **end if**
10: **end for**
11: $Q_1 \leftarrow \pi(Q)$
12: $Q_2 \leftarrow \pi^2(Q)$
13: $f \leftarrow f \cdot \ell_{T,Q_1}(P)$, $T \leftarrow T + Q_1$
14: $f \leftarrow f \cdot \ell_{T,-Q_2}(P)$, $T \leftarrow T - Q_2$
15: $g \leftarrow f^{(p^{12}-1)/r}$
16: **return** g

Tanto el doblado de punto $2T$ como la evaluación de la línea $\ell_{T,T}$ en la línea 4 del Algoritmo B.35, pueden realizarse simultáneamente como se muestra el Algoritmo B.36. También el cálculo de la suma de puntos $T+Q$ y la evaluación de la línea $\ell_{T,Q}$ de las líneas 6, 8, 13 y 14 puede hacerse de forma simultanea como muestra el Algortimo B.37.

La exponenciación de la línea 15 del Algoritmo B.35, es conocida como exponenciación final (Sección 2.3.10) y es calculada a través del Algoritmo B.38.

Algoritmo B.36 Doblado de punto y evaluación de la línea tangente

Entrada: $Q = (X_1, Y_1, Z_1)$, $P = (x_P, y_P)$ donde $Q \in E(\mathbb{F}_{p^2})$ y $P \in E(\mathbb{F}_p)$
Salida: $T = 2Q$ y $\ell_{Q,Q}(P)$
1: $A \leftarrow X_1 \cdot Y_1/2$
2: $B \leftarrow Y_1^2$
3: $C \leftarrow Z_1^2$
4: $E \leftarrow 3b'C$
5: $F \leftarrow 3E$
6: $G \leftarrow (B+F)/2$
7: $H \leftarrow (Y_1+Z_1)^2 - (B+C)$
8: $X_3 \leftarrow A \cdot (B-F)$
9: $Y_3 \leftarrow G^2 - 3E^2$
10: $Z_3 \leftarrow B \cdot H$
11: $l_0 \leftarrow -H \cdot y_p$ (Algoritmo B.10)
12: $l_1 \leftarrow 3X_1^2 \cdot x_p$ (Algoritmo B.10)
13: $l_2 \leftarrow E - B$
14: **return** $2Q = (X_3, Y_3, Z_3)$ y $\ell_{Q,Q}(P) = l_0 + l_1 w + l_2 w^3$

Algoritmo B.37 Adición de puntos y evaluación de la línea secante

Entrada: $T = (X_1, Y_1, Z_1), Q = (X_2, Y_2, 1)$, $P = (x_P, y_P)$ donde $T, Q \in E(\mathbb{F}_{p^2})$ y $P \in E(\mathbb{F}_p)$
Salida: $R = T + Q$ y $\ell_{T,Q}(P)$
1: $A \leftarrow Y_2 \cdot Z_1$
2: $B \leftarrow X_2 \cdot Z_1$
3: $\theta \leftarrow Y_1 - A$
4: $\lambda \leftarrow X_1 - B$
5: $C \leftarrow \theta^2$
6: $D \leftarrow \lambda^2$
7: $E \leftarrow D \cdot \lambda$
8: $F \leftarrow Z_1 \cdot C$
9: $G \leftarrow X_1 \cdot D$
10: $H \leftarrow E + F - 2G$
11: $I \leftarrow Y_1 \cdot E$
12: $J \leftarrow \theta \cdot X_2 - \lambda \cdot Y_2$
13: $X_3 \leftarrow \lambda \cdot H$
14: $Y_3 \leftarrow \theta \cdot (G - H) - I$
15: $Z_3 \leftarrow Z_1 \cdot E$
16: $l_0 \leftarrow \lambda \cdot y_p$ (Algoritmo B.10)
17: $l_1 \leftarrow -\theta \cdot x_p$ (Algoritmo B.10)
18: $l_2 \leftarrow J$
19: **return** $R = T + Q = (X_3, Y_3, Z_3)$ y $\ell_{T,Q}(P) = l_0 + l_1 w + l_2 w^3$

Algoritmo B.38 Exponenciación final

Entrada: $h \in \mathbb{F}_{p^{12}}$
Salida: $h^{(p^{12}-1)/r} \in \mathbb{F}_{p^{12}}$
1: $f_1 \leftarrow \bar{h}$
2: $f_2 \leftarrow h^{-1}$
3: $f \leftarrow f_1 \cdot f_2$
4: $f \leftarrow f^{p^2} \cdot f$ (Algoritmo B.40)
5: $f_c \leftarrow \bar{f}$
6: $t_1 \leftarrow f^z$
7: $t_2 \leftarrow t_1^2$ (Algoritmo B.27)
8: $t_3 \leftarrow t_1 \cdot t_2$
9: $t_4 \leftarrow t_3^2$ (Algoritmo B.27)
10: $t_5 \leftarrow t_4^z$
11: $t_6 \leftarrow t_5^z$
12: $t_7 \leftarrow t_6^2$ (Algoritmo B.27)
13: $a \leftarrow t_7 \cdot t_5 \cdot t_4$
14: $t_8 \leftarrow \bar{t_2}$
15: $t_9 \leftarrow (b \cdot f_c)$
16: $b \leftarrow a \cdot t_8$
17: $g \leftarrow a \cdot t_5 \cdot f$
18: $l_1 \leftarrow b^p$ (Algoritmo B.39)
19: $l_2 \leftarrow a^{p^2}$ (Algoritmo B.40)
20: $l_3 \leftarrow t_9^{p^3}$ (Algoritmo B.41)
21: $g \leftarrow g \cdot l_1 \cdot l_2 \cdot l_3$
22: **return** g

Los Algortimos B.39, B.40 y B.41 realizan el cálculo de f^p, f^{p^2} y f^{p^3} en la exponenciación final, respectivamente. Para cada caso, f es un elemento de la extensión $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^2}[u]/(W^6 - \xi)$.

Un punto importante en la exponenciación final es el cálculo de $f^z \in \mathbb{F}_{p^{12}}$. En general, z es seleccionada de manera que su peso de Hamming sea bajo, es decir, se busca que su representación binaria tenga pocos elementos de valor 1. Ésto representa una ventaja, ya que al ser f un elemento del grupo ciclotómico $\mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$, puede efectuarse esta exponenciación de manera eficiente utilizando los cuadrados de Karabina (Sección B.1.6). El Algoritmo B.42 realiza la exponenciación por z, en donde $\mathcal{C}(g)$

Algoritmo B.39 Operador de Frobenius para calcular f^p

Entrada: $f = g + hw$, $f \in \mathbb{F}_{p^{12}}$, con $g = g_0 + g_1 v + g_2 v^2$ y $h = h_0 + h_1 v + h_2 v^2$, $g, h \in \mathbb{F}_{p^6}$
Salida: $f^p \in \mathbb{F}_{p^{12}}$
1: Calcular $\gamma_{1,i} = \xi^{i(p-1)/6}$ para $i = 1, 2, 3, 4, 5$
2: $t_1 \leftarrow \bar{h_0} \cdot \gamma_{1,1}$
3: $t_2 \leftarrow \bar{g_1} \cdot \gamma_{1,2}$
4: $t_3 \leftarrow \bar{h_1} \cdot \gamma_{1,3}$
5: $t_4 \leftarrow \bar{g_2} \cdot \gamma_{1,4}$
6: $t_5 \leftarrow \bar{h_2} \cdot \gamma_{1,5}$
7: $c_0 \leftarrow \bar{g_0} + t_2 v + t_4 v^2$
8: $c_1 \leftarrow t_1 + t_3 v + t_5 v^2$
9: **return** $c \leftarrow c_0 + c_1 w$

Algoritmo B.40 Operador de Frobenius para calcular f^{p^2}

Entrada: $f = g + hw$, $f \in \mathbb{F}_{p^{12}}$, con $g = g_0 + g_1 v + g_2 v^2$ y $h = h_0 + h_1 v + h_2 v^2$, $g, h \in \mathbb{F}_{p^6}$
Salida: $f^{p^2} \in \mathbb{F}_{p^{12}}$
1: Calcular $\gamma_{2,i} = \gamma_{1,i} \cdot \gamma_{1,i}^p$ para $i = 1, 2, 3, 4, 5$
2: $t_1 \leftarrow h_0 \cdot \gamma_{2,1}$
3: $t_2 \leftarrow g_1 \cdot \gamma_{2,2}$
4: $t_3 \leftarrow h_1 \cdot \gamma_{2,3}$
5: $t_4 \leftarrow g_2 \cdot \gamma_{2,4}$
6: $t_5 \leftarrow h_2 \cdot \gamma_{2,5}$
7: $c_0 \leftarrow g_0 + t_2 v + t_4 v^2$
8: $c_1 \leftarrow t_1 + t_3 v + t_5 v^2$
9: **return** $c \leftarrow c_0 + c_1 w$

Algoritmo B.41 Operador de Frobenius para calcular f^{p^3}

Entrada: $f = g + hw$, $f \in \mathbb{F}_{p^{12}}$, con $g = g_0 + g_1 v + g_2 v^2$ y $h = h_0 + h_1 v + h_2 v^2$, $g, h \in \mathbb{F}_{p^6}$
Salida: $f^{p^3} \in \mathbb{F}_{p^{12}}$
1: Calcular $\gamma_{3,i} = \gamma_{1,i} \cdot \gamma_{2,i}$ para $i = 1, 2, 3, 4, 5$
2: $t_1 \leftarrow \bar{h_0} \cdot \gamma_{3,1}$
3: $t_2 \leftarrow \bar{g_1} \cdot \gamma_{3,2}$
4: $t_3 \leftarrow \bar{h_1} \cdot \gamma_{3,3}$
5: $t_4 \leftarrow \bar{g_2} \cdot \gamma_{3,4}$
6: $t_5 \leftarrow \bar{h_2} \cdot \gamma_{3,5}$
7: $c_0 \leftarrow \bar{g_0} + t_2 v + t_4 v^2$
8: $c_1 \leftarrow t_1 + t_3 v + t_5 v^2$
9: **return** $c \leftarrow c_0 + c_1 w$

calcula el cuadrado comprimido de $g \in \mathbb{F}_{p^{12}}$ mientras que $\mathcal{D}$ realiza la descompresión.

Algoritmo B.42 Exponenciación por z en $\mathbb{F}_{p^{12}}$

Entrada: $f \in \mathbb{F}_{p^{12}}$, y $z = \sum_{i=0}^{l-1} z_i 2^i$
Salida: $C = f^z \in \mathbb{F}_{p^{12}}$
1: $j \leftarrow 0$
2: $C \leftarrow 1$
3: **if** $z_0 = 1$ **then**
4: $\quad C \leftarrow f$
5: **end if**
6: $g \leftarrow f$
7: **for** $i = 1 \rightarrow l - 1$ **do**
8: $\quad g \leftarrow \mathcal{C}(g)$ (Algoritmo B.28)
9: $\quad$ **if** $z_i \neq 0$ **then**
10: $\quad\quad c_j \leftarrow g$
11: $\quad\quad j \leftarrow j + 1$
12: $\quad$ **end if**
13: **end for**
14: **for** $i = j - 1 \rightarrow 0$ **do**
15: $\quad C \leftarrow C \cdot \mathcal{D}(c_i)$
16: **end for**
17: **return** C

C

Instalación del entorno de trabajo y uso de la aplicación

En este apéndice se presenta el proceso de instalación del entorno de desarrollo utilizado durante la realización del trabajo de tesis. Además, se muestran las instrucciones necesarias para utilizar la aplicación desarrollada.

C.1. Instalación del entorno de trabajo

Como se menciono en el Capítulo 5 la aplicación cliente fue realizada en la tarjeta de desarrollo Arndale Samsung Exynos 5 Dual, la cual cuenta con el sistema operativo Android 4.1.1. Para el desarrollo en este sistema operativo se utilizaron dos herramientas el SDK (*Software Development Kit*) y el NDK (*Native Development Kit*), que pueden ser descargados directamente de la página de android (`http://developer.android.com/`) como se muestra en la Figura C.1. La herramienta SDK permite desarrollar aplicaciones sobre el sistema operativo, cuyo lenguaje de programación es Java, mientras que NDK permite desarrollar aplicaciones en el lenguaje C y C++.

Para poder utilizar el SDK es necesario contar con un IDE de desarrollo, para el caso de este trabajo se opto por Eclipse, una vez que se cuenta con un IDE se debe descarga el plugin correspondiente al ADT (*Android Developer Tools*). Este proceso es realizado seleccionando en el menú Help la opción *Install New Software* y añadiendo la dirección `https://dl-ssl.google.com/android/eclipse/` en el cuadro de texto *work with*, esto provocará que aparezca una lista de herramientas de desarrollo las cuales deberán ser descargadas e instaladas. Ya instalado el plugin se debe configurar indicando la ubicación de la herramienta SDK.

Por último, para trabajar sobre un dispositivo se debe seleccionar la opción *SDK Manager* del menú *Window*, lo cual muestra una lista de las versiones disponibles de

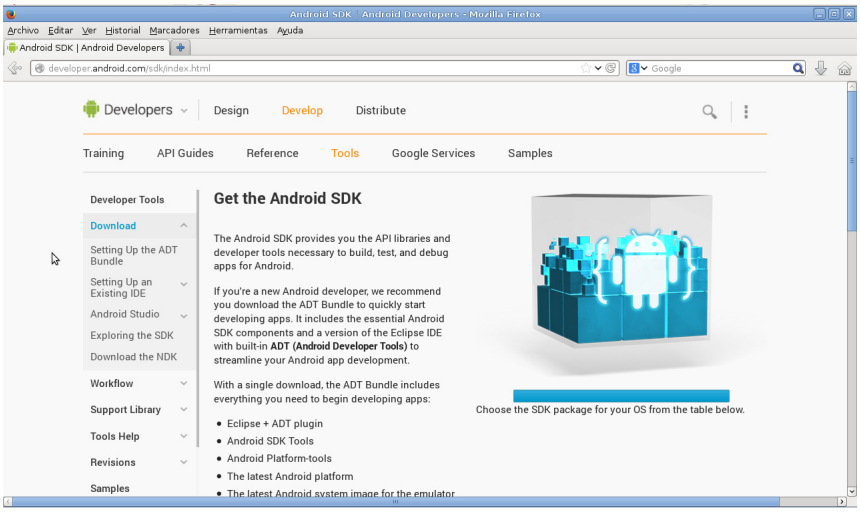


Figura C.1: Descarga de SDK y NDK.

Android y de las herramientas utilizadas en cada una (Figura C.2), aquí se eligen las herramientas que se desean instalar y la plataforma.

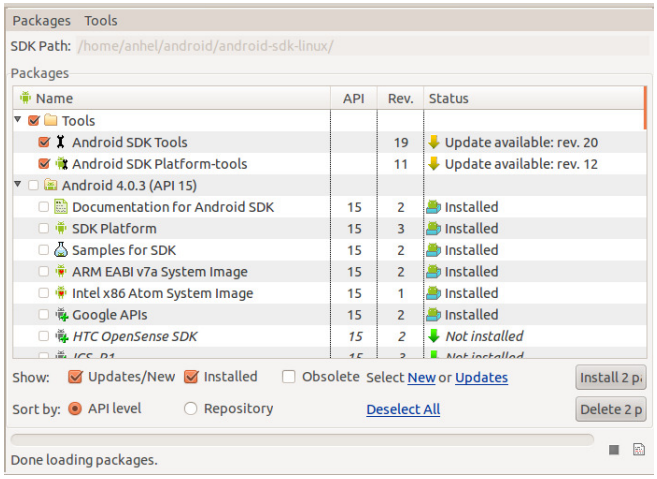


Figura C.2: Android SDK Manager.

Por otra parte para utilizar el NDK sólo se debe descomprimir y seguir los siguientes pasos: se crea un directorio con el nombre "jni" dentro del directorio raíz del proyecto, en esta carpeta se colocan las fuentes del programa en código C o C++. Para poder compilar el código es necesario agregar la ruta del NDK a la variable de entorno PATH y una vez situados el directorio del proyecto ejecutar desde una terminal la línea: *ndk-build* (eso es valido si se trabajo sobre el sistema operativo Linux). Otra forma de hacer este paso es agregando la ruta del NDK a Eclipse.

Con respecto a la utilización de la tarjeta de desarrollo Arndale Samsung Exynos 5 Dual, es necesario realizar la configuración que se describe en la página: `http://www.arndaleboard.org/wiki/index.php/WiKi#Host_Environment_for_Android`.

Para el programa del servidor es necesario contar con:

- Un procesador x64 Intel/AMD.
- Un sistema operativo de 64-bits, para este caso con el sistema operativo Linux.
- El compilador gcc 4.4.1 o superior.
- La herramienta Xbyak[1].

C.2. Manual de uso de la aplicación

Como se observo en el Capítulo 5, se realizó la implementación de dos protocolos de autenticación de dos factores, estos protocolos al tener el mismo objetivo comparten la misma estructura en la aplicación cliente, por lo tanto a continuación se explica como utilizar los programas realizados de manera genérica.

C.2.1. Servidor

Para iniciar el servidor, es necesario realizar la compilación del código asociado a él de la siguiente forma:

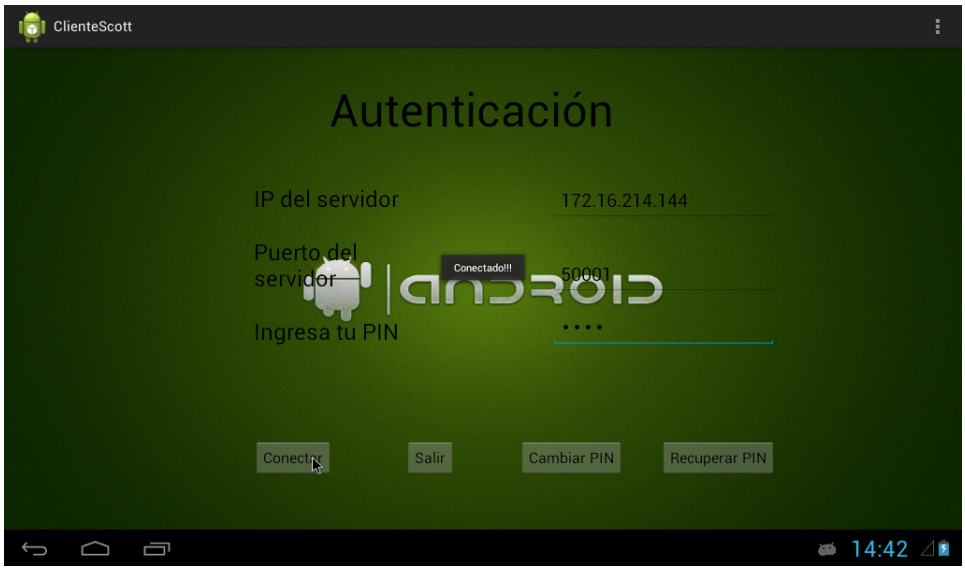
- En una terminal se ingresa al directorio del proyecto correspondiente al servidor,
- y se ejecuta la línea: $make
- Una vez compilado el código, para iniciar el servidor se ejecuta:
  $./servidor <IP> <Puerto>

C.2.2. Cliente

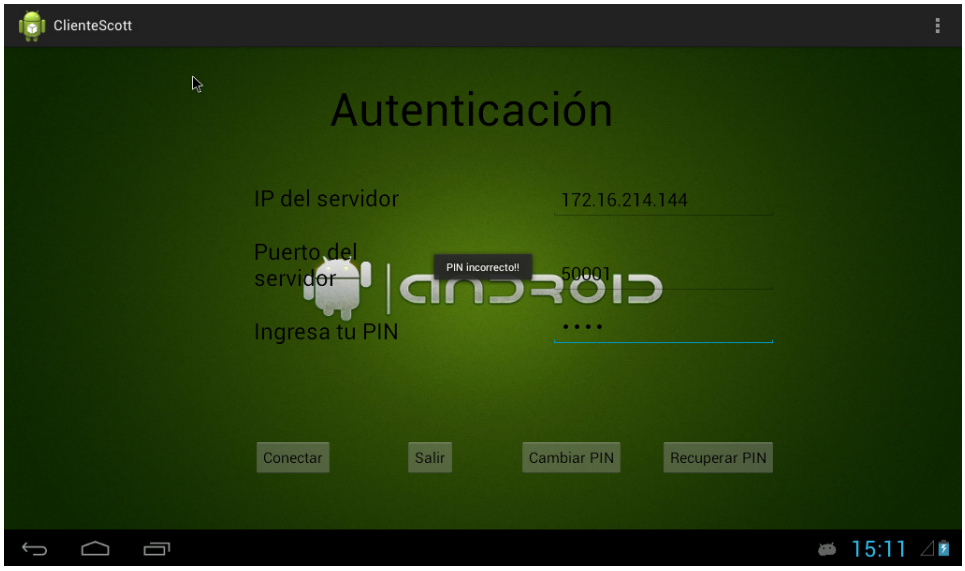
Para utilizar el cliente es necesario instalar la aplicación en el dispositivo. Esto puede ser realizado desde Eclipse al momento de la compilación de la aplicación o con el archivo .apk almacenado en la memoria del dispositivo. Una vez que se tiene instalada la aplicación, esta funciona de la siguiente manera:

1. **Fase de Autenticación:** En la interfaz se deben proporcionar ciertos datos para poder autenticarse ante el servidor, estos datos son: la dirección IP del servidor, el puerto en el cual escucha las peticiones y el PIN del usuario. Para esto se considera que el dispositivo cuenta con el archivo con los datos del cliente (Token y secreto). Una vez llenados los campos se selecciona la opción de conectar, esto puede dar como resultado: conexión fue exitosa o el pin es incorrecto, como se muestra en la Figura C.3.

[1]Xbyak puede ser descargada de la página http://homepage1.nifty.com/herumi/soft/xbyak_e.html.



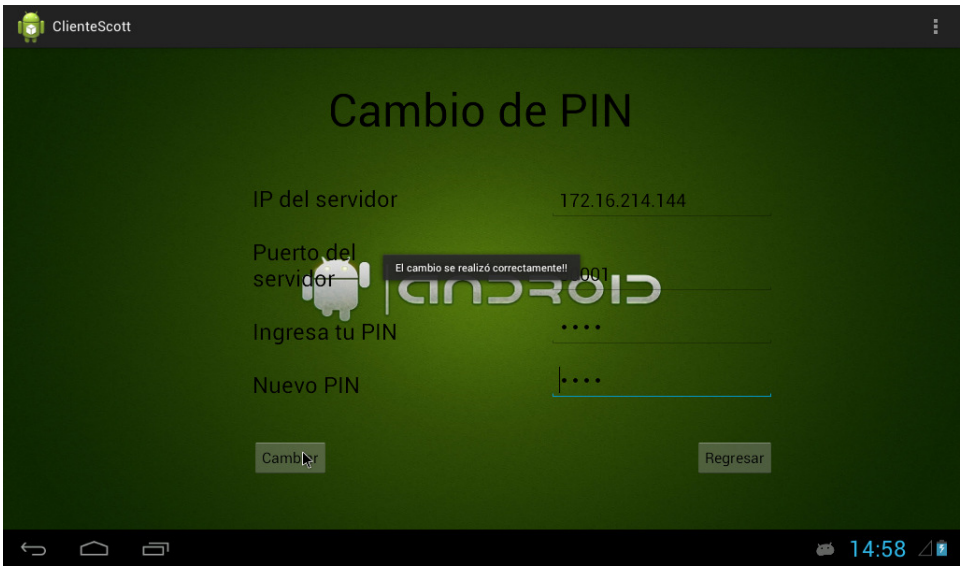
(a) Conexión exitosa.



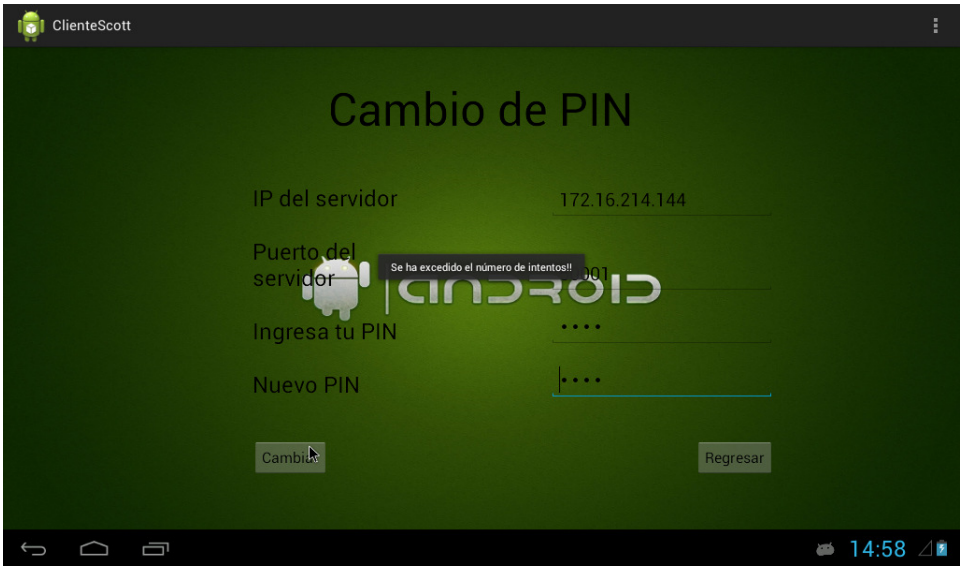
(b) PIN incorrecto.

Figura C.3: Aplicación cliente: fase de autenticación.

2. **Cambio de PIN:** En la interfaz se deben proporcionar los mismos datos que en la fase anterior, con la diferencia de que además se requiere el nuevo PIN. Una vez llenados los campos se selecciona la opción de cambiar, esto puede dar como resultado: el cambio se realizó correctamente, no se realizó el cambio o que se ha excedido del número de intentos, como se muestra en la Figura C.4. En el caso de que el usuario haya excedido el número de intentos la aplicación modifica el archivo con el token y el secreto del cliente, con el fin de que no pueda utilizarse más.



(a) Cambio correcto.



(b) Excedió número de intentos.

Figura C.4: Aplicación cliente: fase de cambio de PIN.

3. **Recuperación del PIN:** En la interfaz se deben proporcionar los mismos datos que en la fase anterior, con la diferencia de que se requiere un indicio del PIN, esto suponiendo que el usuario recuerda algunos de los dígitos. Una vez llenados los campos se selecciona la opción de enviar, lo que da como resultado que el PIN ha sido enviado al correo electrónico del usuario, como se muestra en la Figura C.5.

Figura C.5: Aplicación cliente: fase de recuperación del PIN.