Esta tesis doctoral contiene un índice que enlaza a cada uno de los capítulos de la misma.
Existen asimismo botones de retorno al índice al principio y final de cada uno de los capítulos.

Ir directamente al índice

Para una correcta visualización del texto es necesaria la versión de Adobe Acrobat Reader 7.0 o posteriores

Aquesta tesi doctoral conté un índex que enllaça a cadascun dels capítols. Existeixen així mateix botons de retorn a l'index al principi i final de cadascun deis capítols .

Anar directament a l'index

Per a una correcta visualització del text és necessària la versió d' Adobe Acrobat Reader 7.0 o posteriors.

Universidad de Alicante

Departamento de Física, Ingeniería
de Sistemas y Teoría de la Señal

Sistemas criptográficos de curva elíptica basados en matrices

Memoria presentada para optar al grado de
Doctor en Informática por

FRANCISCO ANTONIO FERRÁNDEZ AGULLÓ

Alicante, 17 de marzo de 2005

Don JOAN JOSEP CLIMENT COLOMA, Catedrático de Universidad del Departamento de Ciencia de la Computación e Inteligencia Artificial de la Universidad de Alicante,

CERTIFICA:

Que la presente memoria, *Sistemas criptográficos de curva elíptica basados en matrices*, ha sido realizada bajo su dirección en el Departamento de Física, Ingeniería de Sistemas y Teoría de la Señal de la Universidad de Alicante por el Ingeniero Don FRANCISCO ANTONIO FERRÁNDEZ AGULLÓ, y constituye su tesis para optar al grado de Doctor en Informática. Para que conste, en cumplimiento de la legislación vigente, autoriza la presentación de la referida tesis doctoral ante la comisión de Doctorado de la Universidad de Alicante, firmando el presente certificado.

Alicante, a 17 de marzo de 2005.

Don JOSÉ MIGUEL TORREJÓN VÁZQUEZ, Catedrático de Escuela Universitaria del Departamento de Física, Ingeniería de Sistemas y Teoría de la Señal de la Universidad de Alicante, y tutor del doctorando Don FRANCISCO ANTONIO FERRÁNDEZ AGULLÓ,

CERTIFICA:

Que la presente memoria, *Sistemas criptográficos de curva elíptica basados en matrices*, realizada bajo la dirección de Don JOAN JOSEP CLIMENT COLOMA en el Departamento de Física, Ingeniería de Sistemas y Teoría de la Señal de la Universidad de Alicante por el Ingeniero Don FRANCISCO ANTONIO FERRÁNDEZ AGULLÓ, constituye su tesis para optar al grado de Doctor en Informática. Para que conste, en cumplimiento de la legislación vigente, se ratifica en la autorización de la presentación de la referida tesis doctoral ante la comisión de Doctorado de la Universidad de Alicante, firmando el presente certificado.

Alicante, a 17 de marzo de 2005.

A mi familia

Quiero expresar mi más sincero agradecimiento al profesor Joan Josep Climent Coloma, por su esfuerzo y constancia en el trabajo, por su continua orientación para no perderme en el desconocimiento, y por su confianza, sin la cual esta tesis no habría sido posible.

También quiero agradecer el apoyo y ánimo recibidos de los compañeros del grupo de Criptología y Seguridad Computacional del Departamento de Ciencia de la Computación e Inteligencia Artificial, por haberme transmitido el ambiente agradable de trabajo que existe entre ellos.

Asimismo, mi agradecimiento al Departamento de Física, Ingeniería de Sistemas y Teoría de la Señal, donde he realizado la presente memoria de investigación, porque siempre he recibido la ayuda que he necesitado.

Tampoco quiero olvidarme de mis compañeros de trabajo en el Servicio de Informática, de mis amigos, y de aquellos que han pasado por mi vida y me han soportado en los momentos difíciles durante el desarrollo de esta tesis.

Índice general

Prólogo

Desde tiempos inmemoriales se han utilizado técnicas de ocultación de la información –los egipcios y otros pueblos antiguos ya empleaban estas técnicas–, sin embargo, la complejidad de los sistemas ha aumentado considerablemente desde que se usaban esclavos en este proceso: se les rapaba la cabeza y se les tatuaba la información en el cuero cabelludo, de forma que cuando les volvía a crecer el pelo ya podían ser enviados a transmitir la información sin peligro de ser descubiertos. ¿Sin peligro? El éxito de este sistema se basaba en no descubrir el truco, es decir la clave, que consistía en el hecho de saber que la información se encontraba en su "cabeza".

La ocultación de la información, es decir, la privacidad, más los nuevos conceptos como, por ejemplo, la autenticidad o la confianza, son los servicios que proporciona la criptografía. Esta ciencia, o arte como ha sido considerada[1], clásicamente ligada al mundo del poder, es decir al militar, se ha transformado en una herramienta indispensable en nuestros días. El crecimiento de la información en la última mitad del siglo XX y la demanda de confidencialidad de la misma ha generado la necesidad de seguridad y privacidad en las comunicaciones de cualquier índole, no sólo del militar. En los últimos años, además, han surgido otros servicios reclamados por la sociedad de la era digital al amparo de la criptografía, como por ejemplo la autentificación, la integridad, el control de accesos, el no rechazo o la disponibilidad.

La criptografía de clave pública es capaz de facilitar la mayor parte de los servicios antes mencionados. Estos servicios son proporcionados por técnicas como el cifrado, la firma digital o el intercambio de claves. La criptografía de clave privada suministra, entre otros, los sistemas más rápidos para proveer de privacidad y además con claves de menor

[1] Véase la definición de criptografía que aparece en el diccionario de la RAE: "Criptografía: arte de escribir con clave secreta o de un modo enigmático".

tamaño. La principal debilidad de ésta consiste en la necesidad de transmitir la clave previamente entre los usuarios que desean comunicarse. Aquí es donde se fusionan ambas técnicas, pues se acuerdan las claves mediante un sistema de clave pública y se cifra la información mediante un sistema de clave privada.

Toda información tiene fecha de caducidad. Difícilmente encontraremos algo que dentro de 100 años deba seguir oculto. Un sistema se considera seguro si protege la información durante su vida útil. Podemos conocer este tiempo, que puede abarcar desde apenas unos segundos hasta varios años, pero no podemos saber con certeza si el sistema mantendrá su seguridad a lo largo de ese período. Es preciso realizar estimaciones sobre el aumento de la potencia de cálculo de los ordenadores e incluso debemos considerar que se pueden descubrir nuevos y más potentes ataques contra los sistemas que utilizamos. Por todo ello es recomendable utilizar sistemas de dimensiones algo mayores de lo necesario, dando un margen aceptable de seguridad, pero manteniendo un equilibrio con el coste que esto supone en recursos económicos y temporales.

Los sistemas clásicos han dejado de ser útiles, ya que su seguridad se fundamenta en la dificultad de romperlos de forma "manual". La aparición de los ordenadores y el continuo aumento en la potencia de los mismos requiere de sistemas con claves cada vez mayores para mantener la seguridad.

Aumentar el tamaño de los problemas criptográficos de forma que se incremente la seguridad no siempre es una tarea trivial. En la mayoría de los casos requiere de un aumento considerable en los recursos, tanto en tiempo y espacio de almacenamiento como en velocidad de la comunicación. Puede llegar incluso a ser inviable para los recursos disponibles habitualmente.

La presente memoria muestra las investigaciones realizadas en esta vía, es decir, la búsqueda de soluciones al problema de aumentar la seguridad de los sistemas criptográficos con los mínimos requerimientos posibles. Para ello, se han diseñado tres nuevas funciones matemáticas, basadas en curvas elípticas y matrices, y se han aplicado a la criptografía de clave pública.

Hemos dividido la memoria en seis capítulos más un apéndice. Con objeto de hacerla lo más autocontenida posible, se han incluido los dos primeros capítulos. En el primero se realiza una introducción a la criptografía de clave pública así como a los esquemas criptográficos más comunes. Asimismo, se proporciona una introducción a la seguridad de los sistemas y formas de medirla. Por último, se hace un repaso a los esfuerzos previos realizados con sistemas criptográficos basados en matrices. El segundo capítulo se dedica íntegramente a las curvas elípticas ya que los tres sistemas propuestos se basan en ellas. Se muestran las características generales, propiedades, aritmética, ataques conocidos y sistemas criptográficos habituales basados en las mismas.

El tercer, cuarto y quinto capítulos se destinan a plantear los tres nuevos sistemas propuestos. En los tres casos se describen los problemas como funciones matemáticas, se analizan sus propiedades y posibles debilidades, y se aplican a los esquemas criptográficos usuales de clave pública basados en curvas elípticas.

El primero de ellos (capítulo 3) es un sistema lineal que combina matrices de escalares con matrices de puntos de una curva elíptica. Este problema matemático permite aumentar la seguridad de un sistema criptográfico de curva elíptica incrementando el número de instancias necesarias para resolverlo. Además, es posible definir con este sistema todos los esquemas criptográficos usuales basados en curvas elípticas.

El segundo sistema (capítulo 4) consiste en una función no lineal que utiliza ciertos elementos compuestos por escalares y puntos de una curva elíptica. Tras realizar un estudio sobre el mismo, se comprueba que, igual que el anterior sistema, posee las propiedades necesarias para su uso en criptografía de clave pública. A pesar de que el sistema funciona, se verá que no aporta ventajas significativas frente a los sistemas habituales de curva elíptica, por tanto sólo se puede plantear como alternativa. En cualquier caso, su estudio es necesario para abordar el siguiente sistema, que consiste en una ampliación del mismo consiguiendo las características requeridas para aumentar la seguridad.

Este tercer –y último– sistema (capítulo 5) combina bloques de matrices de escalares con bloques de matrices de puntos de una curva elíptica a modo de matrices triangulares por bloques. Consigue eliminar las deficiencias encontradas en el sistema anterior, aumentando el espacio de claves disponibles cuanto se quiera mediante sencillas operaciones, e incrementando correspondientemente la seguridad. Este sistema, además, se aplica a los esquemas comunes de criptografía de clave pública obteniendo sistemas funcionales en el caso del intercambio de claves y del cifrado, no así en el caso de los esquemas de firma digital.

Los problemas planteados en el primer y segundo sistema vienen avalados por la presentación de sendos artículos en la *WSEAS Conference on Mathematics and Computers in Business and Economics* que se celebró en Venecia del 15 al 17 de noviembre de 2004, [17] y [19] respectivamente, que además fueron seleccionados para su publicación en la revista *WSEAS Transactions on Business and Economics*, [16] y [18]. El estudio y adaptación del primer sistema a los esquemas criptográficos convencionales de curva elíptica, así como el tercer sistema, están en proceso de evaluación para su posible publicación ([20] y [21]).

En el sexto capítulo se extraen las conclusiones pertinentes y se establecen las líneas futuras de investigación relacionadas con los objetivos y el trabajo desarrollado en la presente memoria.

Por último, el apéndice presenta una serie de consideraciones necesarias para la implementación práctica de los tres sistemas expuestos. Tras ello, se exponen los resultados de las pruebas realizadas para verificar la aplicabilidad y funcionalidad de los sistemas.

Capítulo 1

Criptografía de clave pública

1.1 Introducción

Durante milenios, la criptografía de clave privada ha estado basada en las mismas técnicas, de sustitución y permutación, que se aplicaban de forma manual. El objetivo era, fundamentalmente, la ocultación de la información mediante el cifrado y su posterior recuperación mediante el descifrado de la misma. La invención de las máquinas semiautomáticas de cifrado supuso un gran avance que posibilitó la aparición de algoritmos de cifrado más complejos y seguros [98]. La llegada de los ordenadores ha aumentado más aún la potencia de la criptografía de clave privada, pero así y todo, estos sistemas siguen basándose en algoritmos de sustitución y permutación.

La aparición de la criptografía de clave pública supuso una auténtica revolución en seguridad. En primer lugar, los algoritmos de clave pública están basados en funciones matemáticas. En segundo lugar, esta criptografía es asimétrica, es decir, utiliza dos claves de forma separada. En general, todo sistema criptográfico o criptosistema consiste en dos algoritmos que dependen de una clave. Con uno se cifra la información y con el otro se descifra. Si utilizamos una misma clave para ambos algoritmos, estaremos hablando de criptografía de clave privada o simétrica. Si, en cambio, usamos una clave para cifrar y otra distinta para descifrar, entonces estaremos hablando de criptografía de clave pública o

asimétrica. En ambos casos, una clave diferente produce que el algoritmo actúe de forma distinta sobre la información.

La criptografía de clave pública se ideó con el objetivo de solucionar dos de los problemas más complejos asociados intrínsecamente a la criptografía de clave privada. El primero es la distribución y gestión de claves. Para que el emisor y el receptor de una comunicación puedan compartir una clave secreta, es necesario que ésta les haya sido suministrada de alguna forma secreta, o bien que se utilice un centro de distribución de claves [31]. Además, en una red de n usuarios, cada par de usuarios debe compartir una clave secreta, por lo que se precisan $n(n-1)/2$ claves distintas. Si n es grande, la gestión de las claves se vuelve intratable.

El segundo problema a solucionar es el de la firma digital. El uso de la criptografía empezaba a extenderse, saliendo del contexto militar, para pasar al mundo del comercio y de las comunicaciones privadas. Esto suponía la necesidad del equivalente a la firma manuscrita en los documentos en papel.

Ambos problemas fueron solucionados mediante el descubrimiento de la criptografía de clave pública en 1976 [98] (o al menos fecha en la que se hizo pública, pues el descubrimiento auténtico parece datar de mediados de los años 60 en entornos militares [97]).

1.1.1 Breve reseña histórica

El conocido sistema de intercambio de claves de Diffie y Hellman, publicado en mayo de 1976 [26], fue el primer sistema criptográfico de clave pública que se dio a conocer. Éste es un modelo abstracto de esquema criptográfico cuyo objetivo es el acuerdo de claves entre dos usuarios. El problema reside en averiguar la clave compartida por ambos y está basado en el problema matemático del logaritmo discreto (DLP).

Poco más de un año después, en 1978, apareció el primer sistema criptográfico de clave pública llevado a la práctica, el RSA [80]. Debe su nombre a sus creadores: Rivest, Shamir y Adleman, y aunque está basado en el modelo de Diffie-Hellman, utiliza en cambio el problema de la factorización de enteros (IFP) como base matemática. Hoy en día todavía persiste como el sistema más usado para cifrar y firmar en Internet, habiendo resistido multitud de ataques.

Otro conocido problema que utiliza el IFP es el criptosistema de Rabin [78].

En 1985 apareció una variante basada en el DLP: el sistema criptográfico de ElGamal [27]. Aunque parece que inicialmente fue diseñado para firmar, en la misma publicación se propuso una modificación para cifrar.

El NIST (*National Institut of Standards and Technology*) adoptó en 1994 el sistema DSA (*Digital Signature Algorithm*) como estándar para su uso por las organizaciones gubernamentales norteamericanas, integrándolo en el DSS (*Digital Signature Standard*) [68]. Este esquema de firma digital es una variante del sistema de ElGamal, y por tanto basado en el DLP. Hace uso de una función *hash* para reducir el tamaño del mensaje que se firma, con lo que se obtienen, en consecuencia, firmas más pequeñas, y precisamente ese pequeño tamaño es el argumento más usado por sus detractores.

Otros criptosistemas surgidos a partir del DLP son el esquema de firma de Schnorr [83] y el de Nyberg-Rueppel [70].

En 1985, Koblitz y Miller introdujeron, independientemente, el concepto de criptografía de curva elíptica (ECC) en [49] y [65] respectivamente. El problema del logaritmo discreto adaptado a las curvas elípticas (ECDLP) tiene el mismo fundamento que el DLP, pero en lugar de usar enteros como símbolos de la información a cifrar, utiliza los puntos de una curva elíptica. Debido a las ventajas ofrecidas por las curvas elípticas, el crecimiento en investigación sobre la ECC ha desbordado las previsiones más optimistas. Como muestra del auge que ha tomado en los últimos años, cabe destacar que el ECDSA (*Elliptic Curve* DSA) ha sido aceptado como estándar en 1998 por el ISO [39], [40], en 1999 por el ANSI [1], [2], y en 2000 por el IEEE [38] y el NIST [69].

1.1.2 Sistemas criptográficos de clave pública

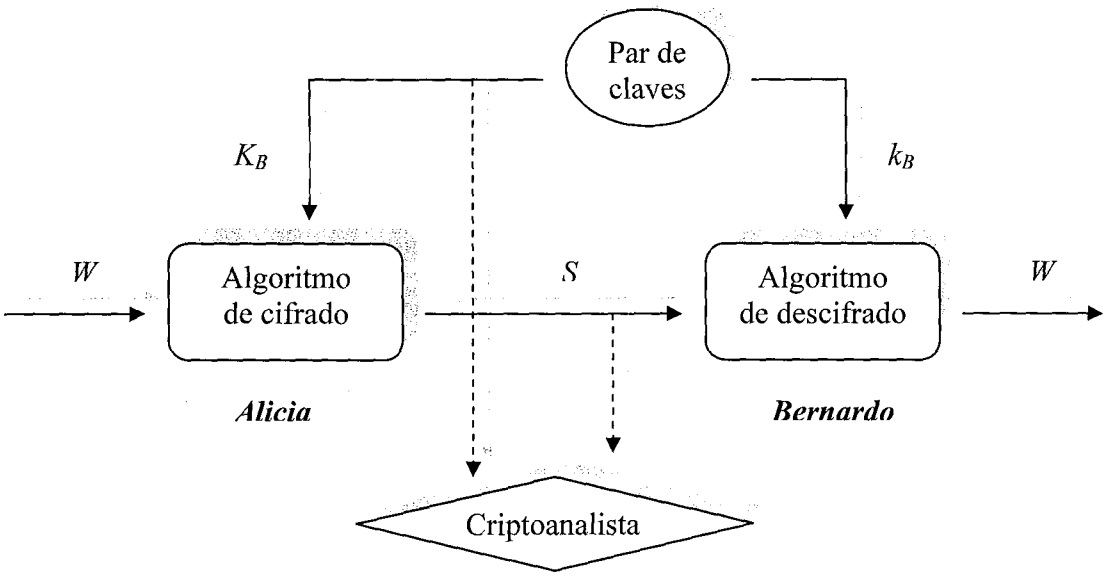
Los esquemas siguientes representan dos criptosistemas de clave pública. La figura 1.1 muestra un esquema de cifrado, mientras que la figura 1.2 muestra un esquema de autentificación mediante firma digital. Para denotar a los usuarios se utilizan los nombres usuales de Alicia y Bernardo. Los elementos básicos de ambos esquemas son:

- **Texto en claro**: Es el mensaje o información legible W. Está formado por una u-tupla $W=[W_1,W_2,\ldots,W_u]$ de elementos que pertenecen a un determinado alfabeto finito.
- **Texto cifrado**: Es el mensaje transformado de forma ilegible S. Está formado por una v-tupla $S=[S_1,S_2,\ldots,S_v]$ de elementos que pertenecen a un determinado alfabeto finito. Depende del texto en claro, del algoritmo de cifrado que se utilice y de la clave que se aplique al algoritmo.
- **Algoritmo de cifrado**: Algoritmo E_K que somete al mensaje en claro W a varias transformaciones que dependen de la clave utilizada K, para transformarlo en el mensaje S.

- **Clave pública y clave privada**: Par de claves mediante las cuales se aplica el algoritmo de cifrado y de descifrado. Si se utiliza la clave pública K para cifrar, la privada k se usará para descifrar, y viceversa. La transformación producida en el mensaje en claro es diferente para cada clave que se use.
- **Algoritmo de descifrado**: Este algoritmo D_k es capaz de obtener el texto en claro W a partir del texto cifrado S y de la clave de descifrado k correspondiente a la clave K que se utilizó en el cifrado.
- **Criptoanalista**: Es el interceptor de la transmisión. Puede obtener la información cifrada S, y la clave pública K, pero a partir de éstas no debe poder obtener el mensaje en claro W ni la clave privada k.

Para poner en marcha un criptosistema de clave pública, cada usuario genera un par de claves para su uso en el cifrado y descifrado de mensajes. Asimismo, cada usuario publica una de sus claves para que sea accesible por el resto de usuarios. La clave asociada a ésta se mantendrá en secreto como clave privada. Si Alicia pretende enviar un mensaje confidencial W a Bernardo, toma la clave pública K_B de éste y la utiliza para cifrar el mensaje $S = E_{K_B}(W)$, como puede verse en la figura 1.1.

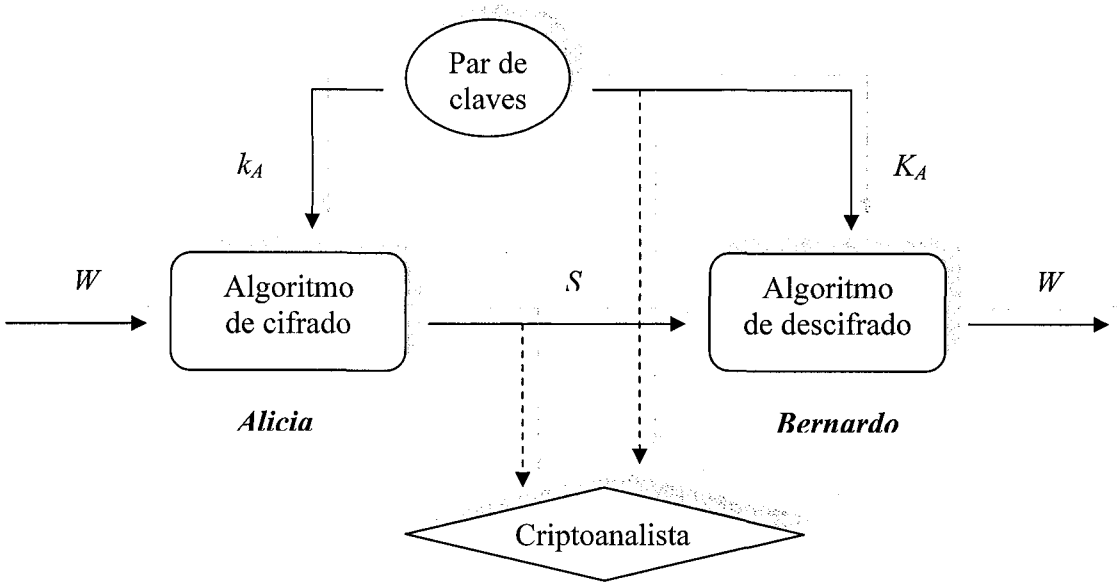
Figura 1.1: Esquema de cifrado



Cuando Bernardo reciba el mensaje, usará su clave privada k_B para descifrarlo $W = D_{k_B}(S)$. Como nadie más posee la clave privada que hace pareja con la clave pública que se usó en el cifrado nadie, salvo Bernardo, puede descifrar el mensaje.

En el caso de la firma digital (figura 1.2), Alicia pretende identificarse (autentificarse) y asegurar que el mensaje que le envía a Bernardo es suyo. Para ello, Alicia toma su clave privada k_A, cifra el mensaje $S = E_{k_A}(W)$, y se lo envía a Bernardo. Éste, por su parte, utiliza la clave pública K_A de Alicia para descifrar el mensaje $W = D_{K_A}(S)$. En este caso cualquiera puede leer el mensaje (no se proporciona confidencialidad), pero Bernardo puede asegurar que proviene de Alicia (que no puede negar que lo ha enviado) y que el mensaje no ha sido modificado.

Figura 1.2: Esquema de firma digital



No es difícil crear esquemas conjuntos que provean de confidencialidad y autentificación simultáneamente. Tan sólo sería necesario usar dos claves para el cifrado y las correspondientes para el descifrado. Si Alicia desea enviar un mensaje secreto W a Bernardo, asegurando además que es suyo, debe tomar su clave privada k_A para cifrar el mensaje, $S = E_{k_A}(W)$, y seguidamente tomar la clave pública de Bernardo K_B para cifrar de nuevo el mensaje, $Z = E_{K_B}(S)$. Bernardo, por su parte, usaría su clave privada k_B para descifrar el mensaje, $S = D_{k_B}(Z)$, con lo que estaría seguro de su confidencialidad. Tras ello, usaría la clave pública de Alicia K_A para descifrar nuevamente el mensaje, $W = D_{K_A}(S)$, cerciorándose de que proviene de Alicia ya que sólo ella ha podido cifrarlo con su clave privada.

Con estos sistemas, todas las partes tienen acceso a las claves públicas, y las claves privadas nunca son distribuidas, permaneciendo en secreto. Además, en cualquier

momento es posible cambiar las claves privadas publicando las nuevas claves públicas asociadas.

Diffie y Hellman, en su publicación inicial sobre criptografía de clave pública, postularon las características que debía cumplir un criptosistema de este tipo [26]:

- Debe ser sencillo y eficiente, para un usuario, generar un par de claves complementarias, una privada k y otra pública K.
- A partir de una clave pública K y de un mensaje W, debe ser sencillo y eficiente cifrar W obteniendo el correspondiente mensaje cifrado $S = E_K(W)$.
- El descifrado de S debe poder hacerse eficientemente a partir de la clave privada correspondiente k, recuperando el mensaje original, $W = D_k[S]$.
- Debe ser razonablemente imposible, para un criptoanalista, determinar la clave privada k a partir de la clave pública K.
- Igualmente, debe ser razonablemente imposible, para un criptoanalista, recuperar el mensaje en claro W a partir del mensaje cifrado S, y de la clave pública K.

Aunque no es imprescindible en un criptosistema de clave pública, Stallings añade un requerimiento adicional [103]:

- El cifrado y el descifrado debe poder aplicarse en cualquier orden:

$$W = E_K[D_k(W)] = D_K[E_k(W)].$$

La criptografía de clave pública precisa de claves más grandes que la de clave simétrica; ello supone un coste adicional en el tiempo que se necesita para cifrar y descifrar o para aplicar el resto de técnicas. Este coste viene justificado en aquellas aplicaciones que proveen de servicios que no se pueden obtener con la criptografía de clave privada. Sin embargo, los algoritmos de cifrado y descifrado son mucho más eficientes con la criptografía de clave simétrica. Esto supone que ambas técnicas, lejos de rivalizar, se complementan, de forma que es posible utilizar los servicios que proporciona la criptografía de clave pública y, entre ellos, el de obtener claves secretas que permitan el uso con seguridad de algoritmos de cifrado de clave privada.

1.1.3 Funciones *hash*

En el esquema de firma digital es necesario cifrar (firmar) el mensaje completo. Además, para el uso práctico del mensaje, debe existir una copia del mismo en claro y, para mantener la certificación de que el mensaje está autentificado, también es necesario mantener una copia cifrada del mismo. Esto requiere una gran cantidad de espacio de

almacenamiento en la práctica. Una forma mucho más eficiente de realizar la tarea de autentificación, sin excesivos requerimientos de espacio, consiste en cifrar un bloque pequeño de datos que es el resultado de una función aplicada previamente sobre el mensaje. Este tipo de funciones se conocen como *hash*.

Un valor *hash* h es generado por una función H como $h = H(W)$, donde W es un mensaje de tamaño variable. Para que una función *hash* H sea útil para la autentificación de mensajes, debe cumplir unos requerimientos básicos [96]:

- H debe poder aplicarse a bloques de datos de cualquier tamaño.
- El resultado de la función debe ser un bloque de tamaño fijo.
- $H(W)$ debe ser relativamente fácil de calcular, ya sea mediante *software* o *hardware*.
- Para un valor dado h, debe ser imposible, en un tiempo razonable, encontrar W tal que $H(W) = h$ (función de una vía).
- Para un mensaje dado W, debe ser inviable, en un tiempo razonable, encontrar un mensaje $W' \neq W$ tal que $H(W') = H(W)$.
- Debe ser inviable, en términos razonables, encontrar un par (W, W') tal que $H(W) = H(W')$.

Las funciones *hash* más importantes actualmente son MD5, SHA-1 y RIPEMD-160. Damos una breve referencia de estos algoritmos:

- **MD5** (*Message-Digest algorithm*) [79]: Fue desarrollado por Rivest en 1992 y ha sido el más usado en los últimos años. Toma una entrada de longitud arbitraria y produce una salida de 128 bits. La entrada se procesa en bloques de 512 bits.
- **SHA** (*Secure Hash Algorithm*): Es una creación del NIST, cuya versión revisada, el SHA-1, fue publicada en 1995 como FIPS 180-1 [67]. Es una evolución del MD4 (versión anterior del MD5). Toma una entrada en bloques de 512 bits y produce una salida de 160 bits.
- **RIPEMD-160** [10]: Fue desarrollado bajo el RIPE (*European RACE Integrity Primitives Evaluation*) y publicado en 1997. Toma una entrada de longitud arbitraria en bloques de 512 bits y produce una salida de 160 bits.

1.1.4 Funciones unidireccionales con trampa

Para establecer un criptosistema de clave pública es necesario utilizar un problema matemático que posea determinadas características, llamado función de una sola vía (o

unidireccional) con trampa (en el sentido de puerta falsa). El cálculo directo de una función de este tipo debe ser sencillo. Además, el resultado de este cálculo debe depender de una clave escogida previamente y aplicada a la función, de forma que se obtengan resultados distintos para claves distintas. El resultado de la función sería la clave pública. Por otra parte, la función debe ser invertible, obteniendo un único resultado posible, sin embargo esta inversión debe ser imposible de realizar en un tiempo computacionalmente aceptable, a no ser que conozcamos la trampa o clave privada para invertir la función de forma sencilla y rápida.

Otra característica fundamental de estos criptosistemas es que determinar la clave privada a partir del algoritmo criptográfico usado y de la clave pública, debe ser una tarea imposible en un tiempo computacionalmente aceptable.

Actualmente existen tres problemas matemáticos usados mayoritariamente en criptografía de clave pública: los ya mencionados IFP y DLP, y el DLP sobre curvas elípticas (ECDLP):

- **IFP**: Consiste en, dado un número compuesto n formado por dos números primos grandes p y q, encontrar p y q. Encontrar números primos grandes para establecer el sistema es una tarea relativamente sencilla, mientras que la factorización del producto de tales primos es computacionalmente intratable.

  El IFP es el problema matemático subyacente del sistema criptográfico más utilizado actualmente, el RSA.

- **DLP**: Se basa en averiguar el logaritmo discreto de un número grande en una base dada. Se define de la siguiente forma: para un número primo p, se considera el grupo multiplicativo $\mathbb{Z}_p^*$ cuyo orden es $p-1$; dados un generador α de $\mathbb{Z}_p^*$ y un elemento cualquiera β de $\mathbb{Z}_p^*$, el problema del logaritmo discreto $\log_\alpha \beta$ consiste en encontrar el único entero x entre 0 y $p-2$ tal que $\beta = \alpha^x$.

  El DSA, establecido como estándar norteamericano, está basado en el DLP.

- **ECDLP**: Tiene el mismo fundamento que el DLP, pero usa el grupo de puntos de una curva elíptica. Se define como sigue: dado el grupo de puntos G determinado por una curva elíptica, un punto P generador de G y Q un punto cualquiera de G, el logaritmo discreto elíptico consiste en averiguar el único escalar x, entre 0 y $o(G)$, tal que $xP = Q$. En el siguiente capítulo se profundizará en el álgebra de las curvas elípticas.

  La gran ventaja del ECDLP es que permite cifrar la información de forma que, ante el mismo tamaño del problema en los sistemas criptográficos convencionales, podamos conservar oculta la información durante mucho más tiempo. Consecuentemente, con claves menores obtendremos niveles de seguridad

similares a los de los otros sistemas con claves bastante más grandes. Esto supone un considerable ahorro de espacio de almacenamiento así como de tiempo para procesar y transmitir las claves. Por ello la primera utilidad que se ha buscado desde su nacimiento a la criptografía de curva elíptica ha sido su uso en entornos de recursos reducidos. Este es el caso de los entornos móviles, como los teléfonos móviles, los PDAs o los ordenadores portátiles, donde se debe tener en cuenta el ancho de banda disponible, la memoria o la carga de la batería. Otra de las ventajas que presenta es su similitud con el DLP, de forma que los criptosistemas desarrollados para éste son fácilmente adaptables al ECDLP.

1.2 Sistemas criptográficos basados en el DLP

Existen tres aplicaciones básicas de los sistemas criptográficos asimétricos según el uso que se haga de las claves privada y pública:

- **Cifrado / descifrado**: El remitente cifra el mensaje con la clave pública del receptor. Provee de confidencialidad, ocultando la información a posibles atacantes.
- **Firma digital**: El remitente firma digitalmente el mensaje con su clave privada. Provee de autentificación, es decir, básicamente asegura que el remitente es quien dice ser, pero también permite otros servicios como el no repudio por parte del remitente o la integridad de los datos.
- **Intercambio de claves**: Permite que dos elementos de la comunicación intercambien una clave de sesión de forma secreta para su uso posterior con un sistema criptográfico de clave privada o cualquier otro objetivo.

Según el algoritmo criptográfico que se use, se obtendrá una o varias de estas aplicaciones. Por poner algunos ejemplos, tenemos que el sistema RSA es capaz de realizar el cifrado, la firma digital y el intercambio de claves; la criptografía de curva elíptica también proporciona estas tres aplicaciones; el conocido sistema de intercambio de claves de Diffie-Hellman sólo permite la aplicación que su propio nombre indica; y el DSS del gobierno americano provee sólo de firma digital.

Dado que los sistemas criptográficos propuestos están relacionados con las curvas elípticas, y por tanto con el ECDLP, se hará la descripción de aquellos sistemas y protocolos que utilizan el DLP como problema matemático subyacente.

En los sistemas de cifrado, el mensaje a cifrar se supone ya codificado, es decir, consiste en un elemento perteneciente al mismo grupo en el que están definidas las claves (normalmente enteros).

1.2.1 Protocolo de intercambio de claves de Diffie-Hellman

En este sistema [26] dos usuarios, Alicia y Bernardo, desean compartir una clave secreta para su uso posterior en el cifrado de mensajes. Se precisa un grupo finito de elementos, para el que suele utilizarse el grupo multiplicativo $\mathbb{Z}_p^*$, siendo p un número primo y g un generador del grupo. Ambos elementos, p y g, deben hacerse públicos.

1. Alicia toma de forma aleatoria una clave privada $1 \le k_A \le p-1$, calcula su clave pública correspondiente $K_A = g^{k_A} \bmod p$ y se la envía a Bernardo.
2. Éste, por su parte, toma una clave privada $1 \le k_B \le p-1$ de forma aleatoria, calcula su clave pública correspondiente $K_B = g^{k_B} \bmod p$ y se la transmite a Alicia.
3. Alicia recibe entonces K_B y calcula $Z_1 = K_B{}^{k_A} \bmod p$.
4. Bernardo hace lo propio con K_A, calculando $Z_2 = K_A{}^{k_B} \bmod p$.

El protocolo funciona ya que

$$Z_1 = K_B{}^{k_A} = (g^{k_B})^{k_A} = (g^{k_A})^{k_B} = K_A{}^{k_B} = Z_2,$$

donde todas las operaciones se realizan módulo p; por tanto, Alicia y Bernardo toman Z_1 y Z_2, respectivamente, como secreto compartido.

Aunque están directamente relacionados, el problema del logaritmo discreto y el llamado problema de Diffie-Hellman (DHP) no son exactamente lo mismo. El DLP consiste en encontrar un entero k a partir de g^k. El DHP consiste en encontrar $g^{k_A k_B}$ a partir de g^{k_A} y g^{k_B}. Si bien está comúnmente aceptado que ambos problemas son equivalentes, hasta la fecha no se ha conseguido demostrar.

1.2.2 Esquema de cifrado de ElGamal

ElGamal desarrolló en 1985 un sistema de cifrado [27] basado en la intratabilidad del logaritmo discreto a partir del esquema de intercambio de claves de Diffie-Hellman.

Para poner en funcionamiento este esquema, deben hacerse públicos un número primo grande p y una raíz primitiva g de 1 módulo p, es decir, $g^{p-1} \equiv 1 \pmod{p}$ y $g^{d-1} \not\equiv 1 \pmod{p}$ para todo d tal que $1 < d < p$.

Alicia desea enviar un mensaje W codificado como entero, con $1 \le W \le p-1$, a Bernardo, asegurando la confidencialidad del mismo. Para ello:

1. Bernardo escoge de forma aleatoria un entero k_B, con $1 \le k_B \le p-1$, que constituirá su clave privada, y calcula la clave pública correspondiente, $K_B = g^{k_B} \bmod p$, que enviará a Alicia.
2. Alicia toma entonces un entero k_A, con $1 \le k_A \le p-1$, como su clave privada, y calcula la clave de cifrado $Z_1 = K_B^{k_A} \bmod p$, que es secreta.
3. Para cifrar el mensaje, Alicia calcula la expresión $S = Z_1 W \bmod p$, y transmite el mensaje cifrado S a Bernardo junto con su clave pública $K_A = g^{k_A} \bmod p$.
4. Bernardo es capaz de obtener la clave de descifrado, que coincide con la de cifrado, haciendo $Z_2 = K_A^{k_B} \bmod p$.
5. A continuación debe obtener el inverso de Z_2 módulo p, para recuperar el mensaje como $W' = Z_2^{-1} S \bmod p$.

Se comprueba que

$$Z_1 = K_B^{k_A} = (g^{k_B})^{k_A} = (g^{k_A})^{k_B} = K_A^{k_B} = Z_2$$

y por tanto

$$W' = Z_2^{-1} S = Z_1^{-1} S = Z_1^{-1} Z_1 W = W,$$

donde todas las operaciones se realizan módulo p.

Como se puede observar, el esquema de cifrado de ElGamal es una adaptación del esquema de Diffie-Hellman en el que se obtiene la clave secreta compartida de la misma forma, pero además se cifra y descifra el mensaje usando dicha clave.

1.2.3 Esquema de cifrado Massey-Omura

El esquema de cifrado Massey-Omura [56], llamado también de "doble cierre", se utiliza rara vez en la práctica ya que requiere de un paso más de comunicación entre Alicia y Bernardo; a cambio consigue cifrar el mensaje sin necesidad de utilizar una clave compartida entre ambos.

Consideremos el grupo multiplicativo $\mathbb{Z}_p^*$, siendo p un número primo. Alicia desea enviar a Bernardo un mensaje W, con $1 \le W \le p-1$, de forma confidencial. Para ello:

1. Alicia obtiene de forma aleatoria un entero k_A tal que $\text{mcd}(k_A, p-1) = 1$ y le envía a Bernardo $K_A = W^{k_A} \bmod p$.
2. Bernardo obtiene de forma similar un entero k_B y calcula ${K_A}^{k_B} = W^{k_A k_B} \bmod p$, enviando el resultado de nuevo a Alicia.
3. Ahora Alicia obtiene el inverso de su clave $k_A^{-1} \bmod (p-1)$ y le envía a Bernardo $Y = \left({K_A}^{k_B}\right)^{k_A^{-1}} \bmod p$.
4. Por último, Bernardo calcula el inverso de su clave $k_B^{-1} \bmod (p-1)$ y calcula $Z = Y^{k_B^{-1}} \bmod p$.

El esquema funciona ya que

$$Z = Y^{k_B^{-1}} = \left(\left({K_A}^{k_B}\right)^{k_A^{-1}}\right)^{k_B^{-1}} = W^{k_A k_B k_A^{-1} k_B^{-1}} = W,$$

donde las operaciones se realizan módulo p. Notar que los inversos se calculan módulo $p-1$ ya que actúan como exponentes.

1.2.4 Firma digital de ElGamal

En el mismo trabajo [27] que publicó ElGamal en 1985, propuso además un esquema de firma digital basado también en el problema del logaritmo discreto.

En este esquema es necesario transmitir el mensaje junto con la firma para poder verificarla, con lo que no se provee de privacidad. La firma por tanto estará en todo caso compuesta por los elementos resultantes de la firma en sí, más el mensaje. Los restantes esquemas de firma digital siguen la misma filosofía.

Partiendo de los elementos públicos g y p definidos según el esquema anterior, Alicia desea enviar la firma de un mensaje W, con $1 \le W < p-1$.

1. Alicia toma una clave privada k_A en la forma anterior, con su clave pública correspondiente $K_A = g^{k_A} \bmod p$.
2. Además, escoge un entero r, con $1 < r < p-1$, que cumpla la propiedad $\text{mcd}(r, p-1) = 1$, y calcula $R = g^r \bmod p$.
3. Ahora debe obtener un entero s tal que

$$W = (sr + k_A R) \mod (p-1).$$

La firma de Alicia para el mensaje W quedará determinada por el par (R,s) junto con W, que es necesario en la verificación.

Bernardo comprueba entonces la firma digital de Alicia sobre W de la siguiente forma:

1. Calcula R^s, $K_A{}^R$ y realiza el producto de ambos resultados $V_1 = R^s K_A{}^R \mod p$.
2. Calcula, por otra parte, $V_2 = g^W \mod p$.
3. Sólo resta comprobar que ambos valores coinciden: $V_1 \equiv V_2 \pmod p$.

El esquema funciona correctamente ya que

$$V_1 = R^s K_A{}^R = g^{sr} g^{k_A R} = g^{sr+k_A R} = g^W = V_2.$$

Se debe hacer notar que las operaciones sobre los exponentes se realizan módulo $p-1$ y las operaciones sobre las bases se calculan módulo p.

El inconveniente de la firma de ElGamal es que duplica la longitud del mensaje que se firma. En el resto de sistemas de firma digital se utiliza una función *hash* que reduce el tamaño de la firma.

1.2.5 Firma digital de Schnorr

El esquema de firma de Schnorr [83], aunque menos usado, está patentado en Estados Unidos (expira en 2008) y en muchos otros países. Similar al esquema de firma digital de ElGamal, consigue en la práctica unos tiempos de proceso más reducidos y obtiene firmas más cortas que las de aquél.

Se precisan dos primos, p y q, tales que q sea un factor primo de $p-1$. Después se elige un entero g distinto de 1, tal que $g^q \equiv 1 \pmod p$. Todos estos elementos son públicos y pueden ser usados por un grupo de usuarios.

Alicia pretende firmar un mensaje W. Para ello:

1. Toma una clave privada $1 \le k_A < q$ y calcula la correspondiente clave pública, que en este caso es $K_{-A} = g^{-k_A} \mod p$.
2. Después toma un entero $1 \le r < q$, y calcula $R = g^r \mod p$.
3. Posteriormente, Alicia concatena W y R, y al resultado le aplica una función *hash* H obteniendo:

$$h = H(W,R).$$

4. Ahora calcula

$$s = (r + k_A h) \bmod q,$$

y transmite a Bernardo la firma, consistente en el par (h,s) junto con el mensaje W.

Para realizar la comprobación de la firma:

1. Bernardo calcula el producto $R' = g^s K_{-A}{}^h \bmod p$.
2. Tras ello concatena el resultado con el mensaje y le aplica la función *hash* H obteniendo:

$$h' = H(W, R').$$

3. Por último, confirma que $h' = h$.

El esquema funciona correctamente ya que

$$R' = g^s K_{-A}{}^h = g^{r+k_A h} g^{-k_A h} = g^r = R$$

y por tanto

$$h' = H(W, R') = H(W, R) = h.$$

1.2.6 DSA

El NIST propuso en 1991 el DSS (*Digital Signature Standard*) mediante el FIPS 186 (*Federal Information Processing Standard*) [68]. El DSS fue adoptado en 1994 tras muchos debates y críticas por la oposición frente al estándar "de facto", el RSA.

El DSS es un esquema de firma digital que utiliza el algoritmo DSA (*Digital Signature Algorithm*), cuyo fundamento se encuentra en el esquema de firma de ElGamal [27] y de Schnorr [83], corrigiendo alguno de los inconvenientes de éstos.

Para plantear el sistema se necesitan elementos similares al esquema de Schnorr, pero en este caso, al tratarse de un estándar, vienen más especificados. Se precisa de un primo p de L bits, con $512 \le L \le 1024$ y múltiplo de 64; de un primo q de 160 bits, tal que $q \mid p-1$; y de un entero h, con $1 < h < p-1$, para calcular otro entero $g = h^{(p-1)/q} \bmod p$, con $g > 1$, que se usará como base para obtener las claves. Estos elementos son públicos y pueden utilizarse para un grupo de usuarios. El algoritmo requiere, además, de una función *hash*. El estándar especifica para ello el algoritmo SHA-1, que denotamos mediante la función H.

Alicia desea firmar un mensaje W.

1. Lo primero que debe hacer es obtener su clave privada $1 \le k_A < q$ y su correspondiente clave pública $K_A = g^{k_A} \bmod p$.
2. Tras ello, genera un número aleatorio $1 \le r < q$ y calcula $R = (g^r \bmod p) \bmod q$.
3. Por último calcula

$$s = r^{-1}\left(H(W) + k_A R\right) \bmod q .$$

Ahora envía a Bernardo los parámetros R y s, que forman su firma sobre W.

Para verificar la firma:

1. Bernardo debe calcular el inverso de s módulo q, para obtener $v_1 = H(W)s^{-1} \bmod q$ y $v_2 = Rs^{-1} \bmod q$.
2. Después calcula $R' = \left(g^{v_1} K_A^{\,v_2} \bmod p\right) \bmod q$.
3. Bernardo verifica la firma si se cumple $R' = R$.

El esquema funciona ya que

$$\begin{aligned} R' &= \left(g^{v_1} K_A^{\,v_2} \bmod p\right) \bmod q \\ &= \left((g^{H(W)s^{-1} \bmod q} g^{k_A R s^{-1} \bmod q}) \bmod p\right) \bmod q \\ &= \left(g^{s^{-1}(H(W)+k_A R) \bmod q} \bmod p\right) \bmod q \\ &= \left(g^r \bmod p\right) \bmod q \\ &= R . \end{aligned}$$

1.3 Seguridad de los sistemas criptográficos

1.3.1 Complejidad y potencia de computación

La teoría de la complejidad computacional estudia, básicamente, el orden del número de iteraciones necesario para la ejecución de los algoritmos. Esto se traduce en necesidades de tiempo o potencia de cálculo y, algunas veces, de espacio para almacenamiento. Lo que se pretende es clasificar los algoritmos, independientemente de su índole, según el tiempo necesario para que terminen y devuelvan un resultado. Esta clasificación se realiza formulando el número de iteraciones o pasos en función del

tamaño de la entrada en bits. Por simplicidad sólo usaremos la notación asintótica O, y como referencia podemos ver las funciones de complejidad más frecuentes, donde n representa el tamaño de la entrada, y a y b sendas constantes positivas:

$$O(1) < O(\log n) < O(n) < O(n^a) < O(b^n),$$

que son denominadas, según ese mismo orden, complejidad constante, logarítmica, lineal, polinómica y exponencial. Esta notación admite además cualquier mezcla de las anteriores. A veces, para indicar el tamaño n de la entrada se utiliza la expresión "tamaño del problema", que suele ser aproximadamente el tamaño de las claves usadas. También suele ocurrir que se indica la complejidad en función de la entrada en lugar de su tamaño, así $O(q)$, siendo $q = 2^m$ la entrada, representa una complejidad exponencial.

En la literatura sobre criptografía se suele utilizar el término fácil o calculable en un tiempo aceptable, o difícil o imposible de calcular en un tiempo razonable, términos aplicados a un algoritmo o función. Fácil indica que tiene una complejidad polinómica, es decir, el algoritmo se ejecuta en un tiempo que es función polinómica de la entrada. Difícil, aunque es un término aplicado con menos rigor, suele indicar que la complejidad para ejecutar el algoritmo es superior a la polinómica, sin precisar en muchos casos si llega a ser exponencial o no.

Otro término que suele aparecer frecuentemente es "complejidad de orden subexponencial", utilizado para indicar aquella complejidad que se encuentra entre la polinómica y la exponencial, sin alcanzar enteramente a ninguna de éstas.

Se suele hablar de complejidad indistintamente cuando se hace referencia a los problemas matemáticos como a los algoritmos que los atacan o resuelven. Estrictamente se debe hablar de complejidad de estos algoritmos y de dureza o algún término similar cuando se habla de problema matemático.

Para medir la potencia computacional que es capaz de desarrollar cualquier medio informático, ya sea un ordenador, un cluster, un desarrollo hardware específico o un conjunto de ordenadores distribuidos, se suele utilizar la unidad de medida MIPS, que indica un millón de instrucciones por segundo. Sirva como referencia decir que un procesador Intel Pentium IV a 3 GHz, capaz de ejecutar hasta 6 instrucciones por ciclo de reloj, posee una potencia de unos pocos de miles de MIPS.

Otra medida muy usada también para medir el rendimiento de los ordenadores es la de MFLOPS, que indica un millón de instrucciones en coma flotante por segundo. Esta unidad se suele emplear en máquinas grandes, como los supercomputadores o los ordenadores con varios procesadores. Por citar algún ejemplo, el BlueGene/L de IBM, basado en 32.768 procesadores PowerPC, es capaz de desarrollar una potencia de 70.720.000 MFLOPS; o el MareNostrum eServer BladeCenter JS20/Myrinet del Centro

Nacional de Super-computación, ubicado en la Universidad Politécnica de Cataluña y basado en 3.564 PowerPC, que es capaz de procesar 20.530.000 MFLOPS.

Dado que las fuentes son muy diversas y que estas medidas, MIPS y MFLOPS, dependen de muchos factores intrínsecos a la máquina –como el tamaño de la palabra considerada o el hardware que acompaña a la máquina–, así como extrínsecos a la misma –como el programa de medición utilizado–, se ha optado por utilizar el término MIPS de forma genérica (término más usado en criptografía), y en los pocos casos en los que no se dispone de una adaptación aproximada entre MIPS y MFLOPS, usar la relación 2:1 entre ambas.

Asimismo, como medida de la potencia-tiempo (o, equivalentemente, número de instrucciones) necesaria para romper los sistemas, se suele utilizar la de MIPS-años, que sería el equivalente a la cantidad de instrucciones que es capaz de ejecutar una máquina con una potencia de 1 MIPS en funcionamiento ininterrumpido durante un año, o de forma correspondiente, cantidad de instrucciones capaz de ejecutar una máquina de 31.536.000 MIPS funcionando durante un segundo.

1.3.2 Algoritmos de ataque contra los sistemas

Existen básicamente dos tipos de algoritmos cuyo objetivo es el de romper los sistemas criptográficos: los de propósito particular y los de propósito general. Los primeros intentan utilizar las características específicas de los números que se emplean en el sistema, bien el número a factorizar, el número del que queremos hallar el logaritmo, o los distintos números que entran en juego en la ECC. Los de propósito general, en cambio, se aplican independientemente de los parámetros escogidos para establecer el sistema. Los algoritmos de propósito particular suelen ser útiles desde el descubrimiento de una debilidad en determinado tipo de números hasta que se prueba su eficacia. En ese momento lo normal es evitar la característica de los números que se considera débil, con lo que el algoritmo de ataque deja de ser útil. Nos centraremos por tanto en los algoritmos de propósito general.

Los dos algoritmos de factorización más usados actualmente son el de la criba cuadrática (*Quadratic Sieve*, QS) y el de la criba del cuerpo de números (*Number Field Sieve*, NFS). El QS es un algoritmo de factorización de propósito general ya que su complejidad sólo depende del tamaño de la entrada. El NFS, aunque inicialmente se creó para un tipo determinado de enteros, fue adaptado más tarde para trabajar con cualquier tipo, pasando a denominarse NFS generalizado [62]. En la práctica, este algoritmo de factorización es el más rápido de todos los existentes de propósito general. Ambos algoritmos son, además, fácilmente paralelizables y de orden subexponencial. La tabla 1.1

(véase [13]) muestra el progreso en los ataques contra el RSA y, especialmente, la gran ventaja que se obtiene con el NFS generalizado. En la última fila de la tabla, la potencia de 8.000 MIPS-años la consiguió un grupo de investigadores internacionales usando una gran cantidad de ordenadores durante poco más de 3 meses y medio.

Tabla 1.1: Ataques contra el RSA

Número de bits	Año	MIPS-años	Algoritmo
236	1984	0,1	QS
352	1988	140	QS
399	1993	825	QS
429	1994	5.000	QS
395	1995	250	NFS generalizado
432	1996	750	NFS generalizado
466	1999	2.000	NFS generalizado
512	1999	8.000	NFS generalizado

En 1999, Shamir (uno de los inventores del RSA) propuso una máquina denominada "TWINKLE" (*The Weizmann Institute Key Locating Engine*) capaz de acelerar el proceso de criba en dos o tres órdenes de magnitud [94]. Aunque el concepto es teórico, si esta máquina se llevara a la práctica no costaría mucho más del precio de dos o tres ordenadores convencionales [90].

Como vemos, es cuestión de tiempo que aparezcan algoritmos cada vez más rápidos y eficientes, capaces de disminuir drásticamente la potencia necesaria para romper los sistemas, tanto los basados en el problema de la factorización de enteros como el resto.

El algoritmo más potente que se conoce para resolver el DLP es el *index-calculus*. Es un algoritmo de tipo probabilístico aplicable sólo a cuerpos finitos, que en aplicaciones prácticas suelen ser $\mathbb{F}_p$, donde p es un número primo impar, y $\mathbb{F}_{2^m}$, con $m \geq 1$. Actualmente es el único algoritmo de orden subexponencial capaz de atacar el DLP, y por tanto el más rápido. La forma de trabajar es muy similar a la del QS y NFS para el IFP. Además es fácilmente paralelizable.

Existen, por otra parte, algoritmos para calcular logaritmos discretos que atacan el grupo donde están definidos; éstos son los llamados de raíz cuadrada, pero son de orden exponencial y por tanto su eficiencia no es comparable a la del *index-calculus*.

Se considera que el DLP bajo un cuerpo primo es más difícil de resolver en la práctica que calcular logaritmos en cuerpos de característica 2. Como ejemplo tenemos que en 2001 se resolvió un DLP establecido sobre un cuerpo $\mathbb{F}_p$ de característica prima, donde p era de 398 bits, usando una variante del *index-calculus* llamada criba del cuerpo de

números (similar al algoritmo NFS del IFP). En contraste, poco después, en 2002, se resolvió un logaritmo de característica 2 en el cuerpo $\mathbb{F}_{2^{607}}$. En este caso se usó otra variante del *index-calculus* llamada el "algoritmo de Coppersmith" [44].

Aunque de momento no se ha podido demostrar, se considera que tanto el IFP como el DLP tienen la misma complejidad, que es de orden subexponencial. En general, los algoritmos que se usan para factorizar se pueden adaptar para calcular logaritmos discretos. De hecho, suele ocurrir que cuando se encuentra una mejora en los algoritmos usados contra uno de los problemas, la adaptación al otro suele aparecer poco tiempo después.

La mayoría de los ataques contra el ECDLP provienen o se inventaron para atacar el DLP. Son, en general, fácilmente adaptables ya que el fundamento de ambos sistemas es el mismo. Sin embargo, la característica fundamental que distingue a ambos sistemas es que para el DLP se suele usar el grupo multiplicativo $\mathbb{Z}_p^*$, cuyo orden es $p-1$, donde existen realmente dos operaciones. Para el ECDLP, en cambio, no se ha podido definir otra operación además de la propia del grupo de puntos de la curva elíptica. Esta ventaja inhibe la aplicabilidad del *index-calculus* al grupo de puntos de la curva [65], [93]. Los únicos algoritmos que se pueden aplicar son los de raíz cuadrada, ya que éstos siempre son aplicables a cualquier grupo, pero, como se ha comentado, son de orden exponencial.

Actualmente el mejor algoritmo que se conoce para atacar el ECDLP es el Pollard-ρ [77], el cual, con algunas mejoras, alcanza un tiempo esperado de ejecución de $O\left(\sqrt{\pi n}/2\right)$ [108]. Además, este algoritmo es paralelizable. En la tabla 1.2 (véase [14]) podemos observar el progreso en los esfuerzos por romper el ECDLP.

Tabla 1.2: Ataques contra el ECDLP

Característica del cuerpo	Bits del cuerpo	Año	MIPS-años
2	79	1997	0,05
2	89	1998	1
2	97	1999	20
2	109	2004	2.000
p	79	1997	0,02
p	89	1998	0,4
p	97	1998	8
p	109	2002	8.000

No existe evidencia matemática de que el problema del logaritmo discreto con curvas elípticas en cuerpos de característica 2 sea más difícil de resolver que en cuerpos de característica prima, ni a la inversa. Por otra parte, debemos tener en cuenta que el

aumento en el tamaño del cuerpo supone que las iteraciones necesarias aumentan de forma cuadrática. Si suponemos que las máquinas trabajan con tamaños de palabra de 32 bits, un cuerpo cuyo orden es de 109 bits requiere del uso de 4 palabras, mientras que un cuerpo cuyo orden es de 89 bits requiere sólo 3. Esto supone un incremento de $(4/3)^2$ en el número de iteraciones necesario [14].

En el capítulo 2 se expondrán más ampliamente los diferentes ataques al ECDLP.

1.3.3 Estimaciones de cómputo necesario para romper los sistemas

Para seleccionar el tamaño adecuado de clave (o tamaño del cuerpo) de un sistema criptográfico, hemos de considerar la dureza intrínseca del sistema empleado. La tabla 1.3 muestra la potencia-tiempo (número de instrucciones) que sería necesaria para factorizar un entero de determinado tamaño, es decir, para romper el RSA, usando el algoritmo más eficiente conocido [72].

Tabla 1.3: Potencia-tiempo necesaria para atacar el RSA

Tamaño en bits	MIPS-años
512	$3 \cdot 10^4$
768	$2 \cdot 10^8$
1.024	$3 \cdot 10^{11}$
1.280	$1 \cdot 10^{14}$
1.536	$3 \cdot 10^{16}$
2.048	$3 \cdot 10^{20}$

Las potencias correspondientes para el DLP son similares. Para hacernos una idea de la magnitud de estos números, notar que la edad del Universo es de, aproximadamente, $1{,}37 \cdot 10^{10}$ años.

Como referencia debemos conocer la potencia que determinadas entidades pueden alcanzar. Como es obvio, no tiene los mismos recursos un *hacker* que una empresa, una organización gubernamental o un equipo distribuido en Internet con miles de máquinas a su disposición. La tabla 1.4 proporciona una estimación (actualizada a finales de 2004) de la relación entre potencia y usuario posible de la misma. Los datos están estimados según información obtenida de diversas fuentes en Internet; por ejemplo, el esfuerzo masivo conjunto se ha aproximado al esfuerzo que se está realizando en el proyecto SETI [89] para búsqueda de vida extraterrestre. En cuanto al futuro inmediato, aunque en gran

medida es impredecible, debemos considerar la ley de Moore, que se ajusta bastante a la realidad, y que predice que cada 18 meses se duplica la potencia de los ordenadores.

Tabla 1.4: Potencia que pueden alcanzar determinadas entidades

Entidad	Potencia (MIPS)
Hacker (ordenador doméstico)	$5 \cdot 10^3$
Grupos de investigación (redes de ordenadores / clusters)	$5 \cdot 10^5$
Grandes empresas (supercomputadores / grandes redes)	$1 \cdot 10^7$
Gobiernos (supercomputadores / grandes redes)	$5 \cdot 10^7$
Esfuerzo masivo conjunto (Internet)	$2 \cdot 10^8$

Relacionando la tabla 1.3 con la 1.4 podemos realizar la estimación de quién puede romper qué clave y en cuánto tiempo. Por ejemplo, un posible *hacker* podría factorizar un entero (romper el RSA) de 512 bits en 6 años; o mediante un esfuerzo conjunto de cientos de miles de ordenadores en Internet, se podría factorizar un número de 1.024 bits en 1.500 años (con los algoritmos actuales), de ahí que 1.024 bits se considere un tamaño de clave seguro para el RSA.

Odlyzko [72] realizó una estimación sobre la potencia de cálculo disponible: si el 0,1% de la potencia de todos los ordenadores a nivel mundial estuviera disponible durante un año para, trabajando en un esfuerzo conjunto, romper un sistema criptográfico, entonces la potencia de cómputo total sería de 10^8 MIPS-años en 2004, y de 10^{10} a 10^{11} MIPS-años en 2014.

Otro punto de referencia debe ser la durabilidad de los datos. La tabla 1.5 (extraída de [82]) muestra una estimación de la vida útil de la información.

Tabla 1.5: Estimación de la durabilidad de la información

Información	Tiempo de vida
Tácticas militares	minutos / horas
Anuncio de productos, fusiones, valoraciones importantes	días / semanas
Planes de negocio a largo plazo	años
Secretos comerciales (ej: composición de la Coca-Cola)	décadas
Secretos armamentísticos (creación de bombas)	> 40 años
Identidad de espías	> 50 años
Asuntos personales secretos	> 50 años
Secretos diplomáticos embarazosos	> 65 años
Información censurada por gobiernos	100 años

Se debe cuantificar el esfuerzo en seguridad dependiendo del tiempo que deba permanecer oculta la información. El uso de claves mayores de lo necesario para almacenar de forma segura determinados datos repercute en los recursos necesarios y puede ser del todo inútil. Así, existe información que debe ocultarse durante décadas y otra que puede tener validez durante apenas unos minutos.

En cuanto a los criptosistemas de curva elíptica, como hemos podido ver, son muy superiores en términos de dureza intrínseca frente al IFP y al DLP. La tabla 1.6 (extraída de [50]) proporciona una estimación de la potencia de cálculo necesaria para romper curvas genéricas bajo cuerpos de los tamaños indicados.

Tabla 1.6: Potencia necesaria contra el ECDLP

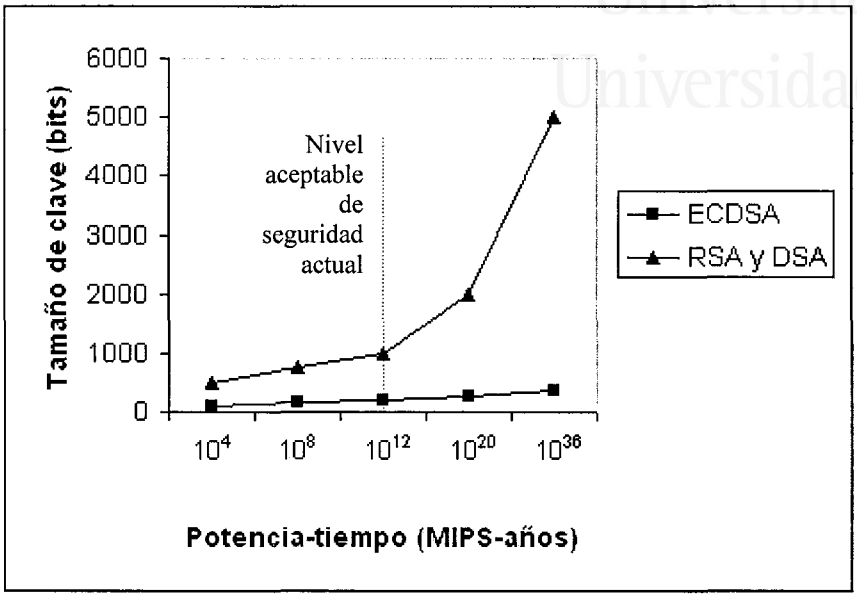
Tamaño en bits	MIPS-años
160	$8{,}5 \cdot 10^{11}$
186	$7{,}0 \cdot 10^{15}$
234	$1{,}2 \cdot 10^{23}$
354	$1{,}3 \cdot 10^{41}$
426	$9{,}2 \cdot 10^{51}$

Como ejemplo, si se dispone de 10.000 ordenadores con una potencia de 1.000 MIPS cada uno, entonces para resolver un logaritmo discreto de curva elíptica con un tamaño de entrada de 160 bits se precisarían 85.000 años.

Tanto el IFP, como el DLP, como las curvas elípticas, han sido estudiados durante mucho tiempo (más de 100 años) como entidades matemáticas. Sin embargo, como entidades criptográficas son relativamente recientes (el DLP y el IFP de mediados de los años 70 y el ECDLP del año 1985). Esto supone que, aunque existe un vasto estudio matemático sobre todos ellos, sólo recientemente se han conseguido verdaderas mejoras para la resolución de dichos sistemas. Es muy difícil que lleguemos a ver una demostración de la dureza intrínseca de los mismos, en cambio sí es factible obtener cada vez mejores algoritmos capaces de atacarlos. En cualquier caso, los sistemas criptográficos de curva elíptica se muestran como los claros vencedores en comparación con el resto ya que no se conoce ningún algoritmo de orden subexponencial capaz de resolverlos, de modo que la potencia necesaria para romper estos sistemas aumenta de forma exponencial respecto al tamaño de clave frente al crecimiento lineal que muestran el RSA y el DSA.

La figura 1.3 (extraída de [62]) muestra un gráfico (en escala logarítmica) de comparación entre los niveles de seguridad que proporcionan estas tecnologías.

Figura 1.3: Comparación entre la seguridad ofrecida por el RSA-DSA y el ECDSA



Según lo visto, se considera un nivel de seguridad aceptable el uso de 160 bits para curvas elípticas o, correspondientemente, el de 1.024 bits para el RSA o el DSA. A muy corto plazo puede ser válido el uso de 768 bits en el RSA o 131 bits en el ECDSA, pero han dejado de ser recomendables. Para información de alto valor o perdurabilidad, se recomienda el uso de claves de 2.048 bits en el RSA o el DSA y de 233 bits en el ECDSA.

Es de destacar que el nivel de seguridad aceptable en el cifrado simétrico ronda los 80 bits, nivel que coincide con el número de iteraciones necesario para atacar una curva elíptica de 160 bits con un algoritmo de raíz cuadrada.

1.4 Sistemas criptográficos basados en matrices

Desde que apareció la criptografía de clave pública, uno de los objetivos ha sido encontrar nuevas funciones de una única vía con trampa, o nuevos sistemas basados en las funciones existentes, que consiguieran aumentar la seguridad de las funciones y sistemas criptográficos conocidos. Entre los posibles planteamientos, encontramos el uso de matrices para definir funciones unidireccionales con trampa; sin embargo, a pesar de las expectativas, los sistemas definidos no han aportado mejoras frente a los existentes.

En 1984 apareció uno de los principales métodos relacionado con el uso de matrices [73]. Este sistema establece el problema del logaritmo discreto empleando matrices como elementos del grupo utilizado y, en concreto, establece el protocolo de intercambio de claves de Diffie-Hellman. Veamos la descripción del método.

Si p es un número primo, $\mathbb{F}_p$ denota el cuerpo finito de p elementos. El grupo lineal general, denotado por $GL(n,\mathbb{F}_p)$, es el conjunto de todas las matrices invertibles de tamaño $n\times n$ con elementos en $\mathbb{F}_p$ y con la multiplicación de matrices. El orden del grupo lineal general es $\prod_{i=0}^{n-1}(p^n-p^i)$. Para formar un sistema de intercambio de claves, se precisa escoger el número primo p de gran tamaño y un elemento $A\in GL(n,\mathbb{F}_p)$, tal que $A^r\equiv I \pmod p$, siendo r el orden de A.

El número primo p, la matriz A y su orden r deben hacerse públicos. Cada usuario escoge aleatoriamente un número entero positivo x_i, menor que r, y genera la matriz pública C_i, donde $C_i = A^{x_i} \bmod p$. Si Alicia quiere compartir una clave con Bernardo, entonces coge la matriz pública C_2 calculada por Bernardo, y calcula $C_2^{x_1} \bmod p$. Bernardo, correspondientemente, coge la matriz C_1 calculada previamente por Alicia, y obtiene $C_1^{x_2} \bmod p$. Se puede comprobar que ambas claves son la misma, es decir,

$$C_2^{x_1} \bmod p = A^{x_1x_2} \bmod p = C_1^{x_2} \bmod p .$$

Con el sistema de intercambio de claves usual de Diffie-Hellman, el máximo número de claves secretas está limitado a $p-1$. Sin embargo, con este sistema extendido, el máximo número depende del orden de la matriz pública usada.

Para diseñar este sistema se debe construir una matriz base $A\in GL(n,\mathbb{F}_p)$ y determinar su orden r. Consideremos un polinomio primitivo $f(x)$ de grado m:

$$f(x) = x^m + a_{m-1}x^{m-1} + \cdots + a_2x^2 + a_1x + a_0 \text{, con } a_i \in \mathbb{F}_p$$

La matriz asociada a $f(x)$ es la matriz B de tamaño $n\times n$ dada por:

$$B = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{m-2} & -a_{m-1} \end{pmatrix}$$

Como B es una raíz de $f(x)$, tenemos que $f(B)=0$ y el orden de B es por tanto p^m-1, es decir $B^{p^m-1} \equiv I \pmod p$ [55].

La forma adecuada de diseñar el sistema consiste en seleccionar polinomios primitivos $f_1, f_2, \ldots, f_s$ de grados $m_1, m_2, \ldots, m_s$, respectivamente, sobre $\mathbb{F}_p$ y componer la matriz por bloques:

$$\overline{B} = \begin{pmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_s \end{pmatrix},$$

donde el orden de B_i es igual a $p^{m_i} - 1$, para $1 \le i \le s$. El orden de la matriz $\overline{B}$ viene dado entonces por:

$$\operatorname{mcm}\left(p^{m_1} - 1, p^{m_2} - 1, \ldots, p^{m_s} - 1\right)$$

Resulta interesante "camuflar" la matriz $\overline{B}$ para que no se puedan obtener directamente las submatrices o bloques B_i. Para ello, se toma una matriz invertible $Y \in GL(n, \mathbb{F}_p)$, con $n = \sum_{i=1}^{s} m_i$, y se calcula $A = Y\overline{B}Y^{-1}$, tomando A como la matriz base del sistema criptográfico. De esta forma el orden de A es el mismo que el de $\overline{B}$.

Aunque este problema se planteó inicialmente sobre $\mathbb{F}_p$, la extensión a $\mathbb{F}_q$, siendo q una potencia de p, es inmediata. De hecho, las referencias posteriores a este artículo ya tratan el problema sobre $\mathbb{F}_q$.

Poco después, en 1985, se sugirió que este problema podía reducirse al problema de calcular logaritmos discretos en los cuerpos $\mathbb{F}_{q^{m_i}}$ [71]. En 1992, se publicó un artículo que demostraba cómo reducir el problema del logaritmo discreto en ciertos subgrupos cíclicos de $GL(n, \mathbb{F}_q)$ al problema del logaritmo discreto en algunas extensiones de $\mathbb{F}_q$ [63]. Estos resultados denotan la importancia de escoger adecuadamente el grupo finito a usar en el logaritmo discreto.

Consideremos el problema del logaritmo discreto según el sistema anterior, que consiste en encontrar el escalar x tal que $A^x = C$. El polinomio característico f de $\overline{B}$, y por tanto de A, está formado por los polinomios primitivos f_i escogidos para componer la matriz $\overline{B}$, esto es, $f = f_1 f_2 \cdots f_s$. Cada polinomio f_i, con $1 \le i \le s$, se escinde en $\mathbb{F}_{q^{m_i}}$. Por tanto, el cuerpo de escisión de f sobre $\mathbb{F}_q$ es $\mathbb{F}_{q^k}$, donde $k = \operatorname{mcm}(m_1, m_2, \ldots, m_s)$. Como las raíces de los f_i son distintas los valores propios de A también son distintos, y por tanto A es diagonalizable sobre $\mathbb{F}_{q^k}$. En consecuencia, existe una matriz invertible Q tal que $D = Q^{-1}AQ$, donde D es la matriz diagonal cuyos elementos son las raíces de f. Tenemos entonces que $D^x = Q^{-1}A^xQ = Q^{-1}CQ$, y por tanto podemos determinar D^x.

Denotamos mediante α_{ij}, con $1 \le j \le m_i$, a las raíces de f_i en $\mathbb{F}_{q^k}$. Para cada i, sea β_i la entrada en D^x que se encuentra en la misma posición que α_{i1} en D, entonces tenemos que $\beta_i = \alpha_{i1}^x$. Esta expresión consiste en una instancia del problema del logaritmo discreto en $\mathbb{F}_{q^k}$ y resolviéndola se determina x módulo $(q^{m_i}-1)$. Una vez obtenidas las s expresiones podemos usar el teorema chino del resto [86] para averiguar x módulo el orden de A.

Se comprueba en dicha publicación, además, que es factible realizar esta reducción mediante un algoritmo de complejidad polinómica. Por otra parte, la operación de grupo realizada con los elementos formados con la matriz base A resulta más costosa de realizar que la operación de grupo en el cuerpo $\mathbb{F}_{q^m}$. La conclusión obtenida es que el grupo formado mediante estas matrices no ofrece ninguna ventaja sobre el uso del logaritmo discreto sobre cuerpos finitos para la implementación de protocolos criptográficos cuya seguridad está basada en el cálculo de logaritmos discretos en un grupo.

En 1998, apareció un artículo que eliminó definitivamente las posibilidades de este sistema, pues extendía los resultados del anterior al encontrar una forma de reducir el problema del logaritmo discreto en $GL(n,\mathbb{F}_q)$ al mismo problema en pequeñas extensiones de $\mathbb{F}_q$, realizando esta reducción con un coste polinómico [64]. Con esta reducción se prueba que el uso del logaritmo discreto con matrices no supone ventaja alguna frente al logaritmo discreto tradicional.

Capítulo 2

Criptografía de curva elíptica

En este capítulo realizamos una introducción general a las curvas elípticas y a sus propiedades esenciales, necesarias para diseñar en la práctica sistemas criptográficos basados en las mismas. Aunque casi todo el texto desarrollado aquí es imprescindible para la implementación de estos sistemas, la parte fundamental consiste en la definición del grupo de puntos racionales de la curva elíptica y las operaciones entre los mismos.

Para más información sobre la criptografía de curvas elípticas en general, se puede ver [5], [12], [14], [37], [43], [49], [50], [60], [91] y [106].

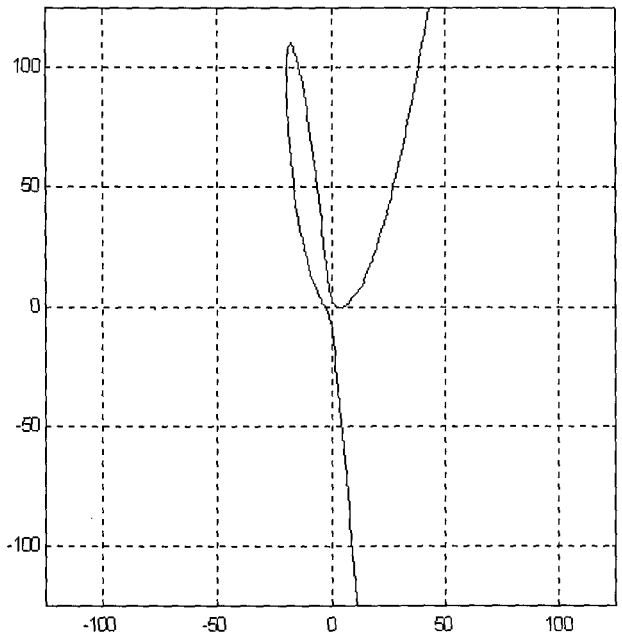
2.1 Introducción a las curvas elípticas

Una curva elíptica E con coeficientes en un cuerpo $\mathbb{K}$, que denotamos por $E/\mathbb{K}$, se define como el conjunto de soluciones de una ecuación de Weierstrass en coordenadas afines de la forma

$$E/\mathbb{K}: \quad y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \ , \quad \text{con } a_i \in \mathbb{K}, \text{ para } i = 1,2,3,4,6 .$$

Aunque en la mayoría de aplicaciones las curvas elípticas se definen con coeficientes en un cuerpo, también pueden definirse con coeficientes en un anillo. Para la descripción inicial de las curvas elípticas, usaremos el cuerpo $\mathbb{R}$ de los número reales, esto es, los puntos de la curva $(x, y) \in \mathbb{R}^2$ y los coeficientes $a_i \in \mathbb{R}$. Así, una curva elíptica tendría un aspecto como el siguiente:

Figura 2.1: Curva elíptica sobre $\mathbb{R}$



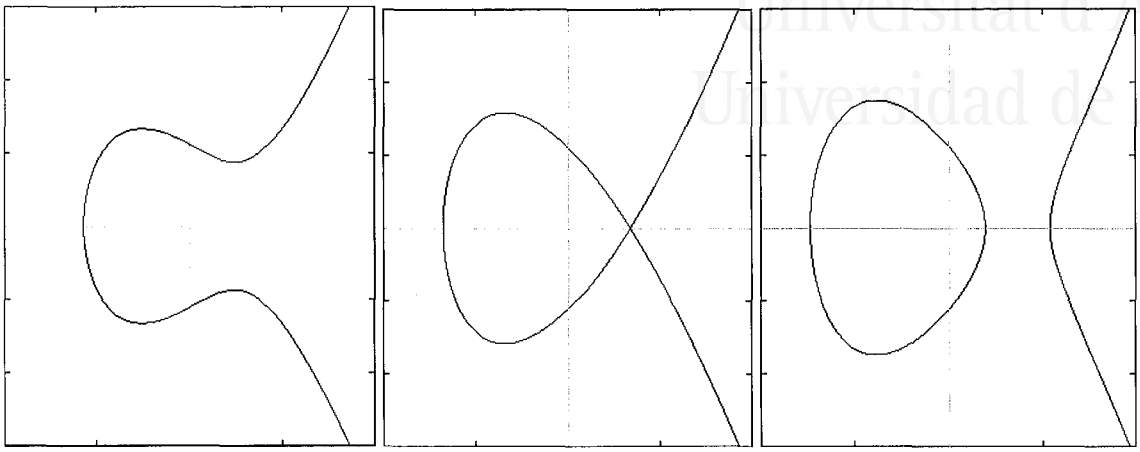
La forma de la ecuación antes presentada se conoce como forma normal de Weierstrass de una curva cúbica, ya que es la mejor reducción que se puede obtener de una curva cúbica general. Ahora bien, si el cuerpo donde se define esta ecuación es de característica diferente de 2 ó 3, como es el caso de $\mathbb{R}$, entonces esta forma se puede reducir mediante un cambio de variable. Así la ecuación reducida quedaría:

$$y^2 = x^3 + ax + b.$$

Esta forma define curvas visualmente más sencillas, que poseen el rasgo fundamental de ser simétricas respecto al eje de abcisas.

Basándonos en esta última ecuación, distinguimos 3 tipos de curva elíptica atendiendo a la cantidad de puntos que corten al eje de abcisas:

Figura 2.2: Curvas elípticas según el número de puntos de corte con el eje de abcisas



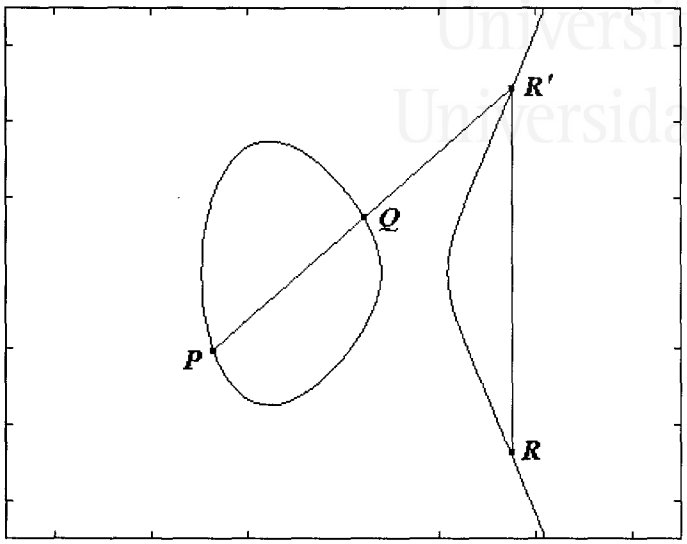
Estos puntos de corte corresponderán a los puntos cuya componente en el eje de ordenadas es 0, y podemos obtenerlos hallando las tres raíces del polinomio definido por $x^3+ax+b=0$.

La figura intermedia corresponde al caso en el que el polinomio x^3+ax+b posee factores repetidos, es decir, cuando $4a^3+27b^2=0$. En este caso se dice que la curva elíptica posee un punto singular en la coordenada donde existen dos raíces. El caso particular en el que $a=0$ y $b=0$, nos daría una raíz triple en $x=0$, también correspondiente a un punto singular.

Se sabe que si el polinomio x^3+ax+b no tiene factores repetidos, o lo que es lo mismo, si $4a^3+27b^2\neq 0$ (con lo que evitamos curvas con puntos singulares), entonces el conjunto de puntos de dichas curvas constituyen un grupo con la operación suma definida de la siguiente manera:

1. Dados dos puntos P y Q pertenecientes a la curva, trazamos la recta que pasa por ambos y obtenemos otro punto de corte que denotamos por R'. Ahora trazamos la vertical desde éste y obtenemos el punto R. Podemos verlo gráficamente en la figura 2.3 utilizando como ejemplo la curva $y^2=x^3-7x+5$:

Figura 2.3: Suma de puntos

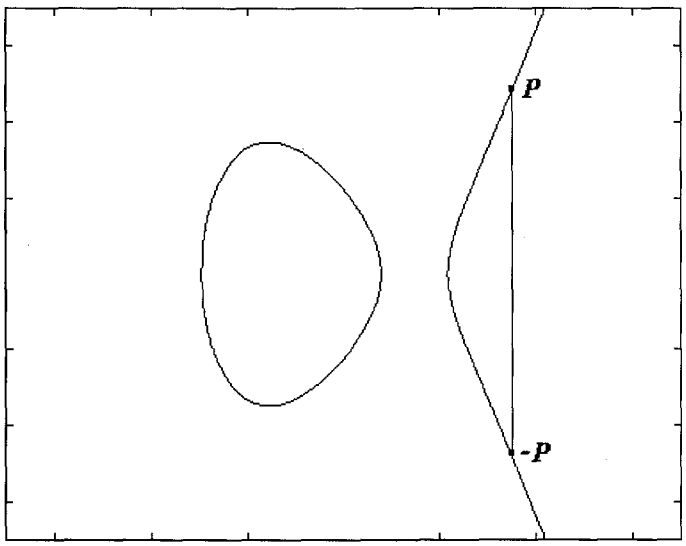


Así, definimos la suma de dos puntos como:

$$P+Q=R.$$

2. Dado un punto cualquiera P, definimos su opuesto, que denotamos por $-P$, como el punto de la curva que obtenemos al trazar la vertical desde el punto P. Lo podemos ver gráficamente usando la misma curva:

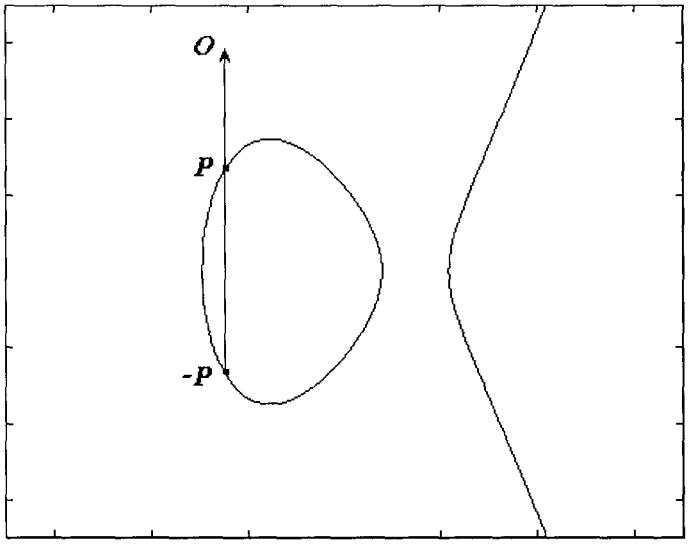
Figura 2.4: Opuesto de un punto



Como vemos en este caso, tan sólo varía la ordenada del punto P.

3. El elemento neutro se obtiene de la suma de un punto con su opuesto. La línea que pasa por ambos puntos se pierde en el infinito, de forma que hemos de agregar este punto a la curva para obtener el neutro, al que llamamos punto en el infinito o punto cero, y denotamos por O:

Figura 2.5: Punto en el infinito

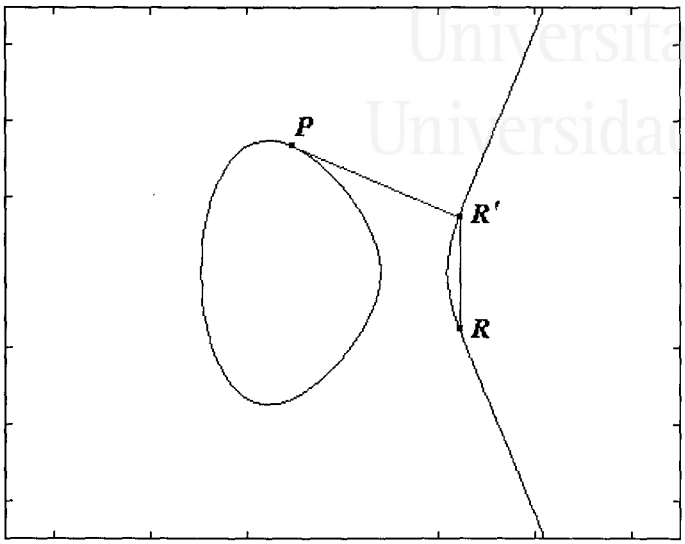


De esta forma tenemos que

$$P+(-P)=O.$$

4. En el caso particular de que los sumandos sean el mismo punto, es decir, pretendamos sumar $P+P$, entonces la recta que debemos trazar es la tangente a P:

Figura 2.6: Duplicación de un punto

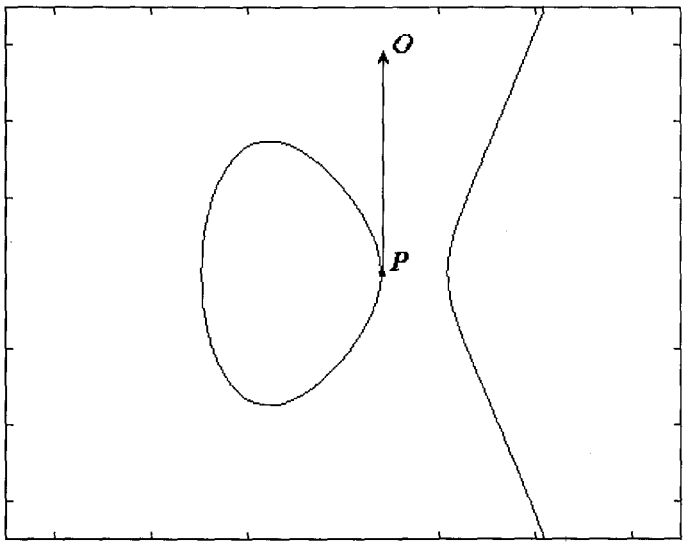


Con lo que

$$P + P = R\,.$$

5. Otro caso particular es cuando queremos calcular $P+P$ y la tangente a la curva es una recta vertical. Esto ocurrirá cuando la ordenada de P sea 0. En este caso la recta se pierde en el infinito, con lo que $P+P=O$.

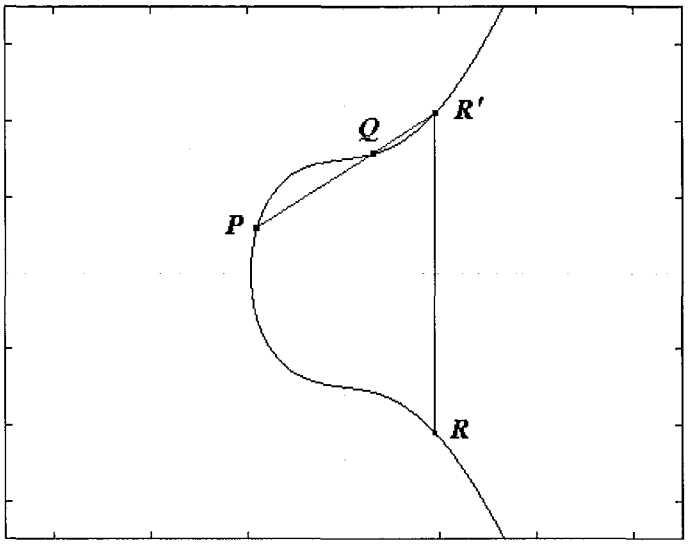
Figura 2.7: Duplicación de un punto que es su opuesto



Podemos aplicar este proceso geométrico a cualquiera de los dos tipos de curva elíptica posibles atendiendo al número de puntos de corte con el eje de abcisas (recordemos que

pueden ser uno o tres puntos de corte). Veamos un ejemplo de la suma de dos puntos en una curva con un único punto de corte con el eje de abcisas:

Figura 2.8: Suma de puntos en una curva con un punto de corte



La definición de suma, hecha de forma informal y lo más sencilla posible, sería: si tres puntos de una curva elíptica se pueden unir por una línea recta, su suma es O. Además siempre encontraremos 3 puntos de corte (repetidos o no) ya que la ecuación que define la curva es de grado 3.

Hemos de observar que el grupo es abeliano, ya que la recta que pasa por dos puntos es la misma, independientemente del orden en el que aparezcan los sumandos.

Una vez expuesta la suma de forma geométrica sobre $\mathbb{R}$, la describimos analíticamente, pero generalizando para cuerpos de cualquier característica.

1. **Adición y duplicación de puntos.**

   Sean los puntos $P=(x_1,y_1)$ y $Q=(x_2,y_2)$. Hemos de hallar el resultado de $P+Q=(x_1,y_1)+(x_2,y_2)=(x_3,y_3)$, que llamaremos suma de puntos si $P\neq Q$ o duplicación de puntos si $P=Q$.

   Consideremos la recta l que pasa por P y Q siendo $P\neq Q$, o bien la recta tangente a la curva en P siendo $P=Q$. La pendiente de esta recta será:

$$\lambda=\begin{cases}\dfrac{y_2-y_1}{x_2-x_1}, & \text{si } P\neq Q\\[2ex] \dfrac{3x_1^2+2a_2x_1+a_4-a_1y_1}{2y_1+a_1x_1+a_3}, & \text{si } P=Q\end{cases}$$

El término independiente de l es $\beta = y_1 - \lambda x_1$, con lo que la ecuación que define la recta es $l : y = \lambda x + \beta$. Para encontrar el tercer punto de intersección, sustituimos la ecuación de la recta en la ecuación de la curva, obteniendo:

$$x^3 + a_2 x^2 + a_4 x + a_6 - (\lambda x + \beta)^2 - a_1 x(\lambda x + \beta) - a_3(\lambda x + \beta) = 0$$

Las raíces de esta ecuación son x_1, x_2 y x_3, con lo que podemos escribirla como:

$$(x - x_1)(x - x_2)(x - x_3) = 0$$

Igualando los coeficientes de x^2 en estas dos últimas ecuaciones, obtenemos:

$$-(x_1 + x_2 + x_3) = a_2 - \lambda^2 - a_1 \lambda$$

Por último, despejamos x_3 y después calculamos y_3:

$$x_3 = \lambda^2 + a_1 \lambda - a_2 - x_1 - x_2$$

$$y_3 = -(\lambda + a_1) x_3 - \beta - a_3$$

2. **Elemento neutro.**

   O denota el elemento neutro, de forma que $P + O = O + P = P$.

3. **Opuesto.**

   Sea $P = (x, y)$, entonces, como se deriva de la ecuación de Weierstrass en coordenadas afines, $-P = (x, -y - a_1 x - a_3)$, punto que también pertenece a la curva, de forma que $(x, y) + (x, -y - a_1 x - a_3) = O$.

Es posible demostrar todas las propiedades de la suma para concluir que el conjunto de puntos de la curva con la suma así definida es un grupo abeliano [91].

En criptografía se utilizan preferentemente cuerpos finitos de característica 2 o de característica un número primo impar. Los de característica 2 están formados por 2^m elementos para algún entero positivo m, y poseen la ventaja de poder representarse como m-tuplas, cadenas de m elementos de 0 y 1. Esto proporciona una indudable ventaja en las implementaciones *hardware*, ya que los ordenadores trabajan de forma natural con estos elementos. Los de característica impar a veces proporcionan mejores resultados en implementaciones *software*.

Por otra parte, tanto la ecuación que define la curva como varias de las propiedades que poseen, varían según sea la característica. En particular se suele distinguir entre curvas de característica 2, de característica 3 y de característica mayor que 3. Se tratarán el primer y el último caso de forma conjunta cuando sea posible, e independientemente cuando no lo sea, mientras que se omitirá el caso de característica 3 ya que suele ser menos eficiente en sistemas criptográficos y prácticamente no se utiliza.

Tras la introducción, que pretendía dar una visión general e intuitiva de las curvas elípticas y su aritmética, pasamos a la descripción formal.

2.2 Definición de curva elíptica

Consideremos un cuerpo $\mathbb{K}$ finito o infinito, denotamos su clausura algebraica por $\overline{\mathbb{K}}$. En el caso particular de que $\mathbb{K}=\mathbb{F}_q$, con $q=p^m$, entonces $\overline{\mathbb{K}}=\bigcup_{n\geq 1}\mathbb{F}_{q^n}$.

El plano proyectivo $\mathbb{P}^2(\mathbb{K})$ es el conjunto de clases de equivalencia de la relación $\sim$ sobre $\mathbb{K}^3\setminus\{(0,0,0)\}$, donde

$$(x_1,y_1,z_1)\sim(x_2,y_2,z_2)$$

si y sólo si existe un elemento $u\in\mathbb{K}^*$ tal que

$$x_1=ux_2,\ y_1=uy_2,\ z_1=uz_2.$$

La clase de equivalencia que contiene a un elemento (x,y,z) se denota mediante $(x:y:z)$.

La ecuación de Weierstrass es una ecuación homogénea (en coordenadas proyectivas) de grado 3 de la forma

$$Y^2Z+a_1XYZ+a_3YZ^2=X^3+a_2X^2Z+a_4XZ^2+a_6Z^3,$$

donde $a_1,a_2,a_3,a_4,a_6\in\overline{\mathbb{K}}$. La ecuación de Weierstrass es singular si tiene puntos singulares, es decir, si de todos los puntos proyectivos $P=(X:Y:Z)\in\mathbb{P}^2(\overline{\mathbb{K}})$ que satisfacen

$$F(X,Y,Z)=Y^2Z+a_1XYZ+a_3YZ^2-X^3-a_2X^2Z-a_4XZ^2-a_6Z^3=0,$$

existe al menos uno tal que las tres derivadas parciales cumplen $\frac{\partial F}{\partial X}=\frac{\partial F}{\partial Y}=\frac{\partial F}{\partial Z}=0$.

Una curva elíptica E (o curva algebraica de género 1) es el conjunto de todas las soluciones en $\mathbb{P}^2(\overline{\mathbb{K}})$ de una ecuación de Weierstrass no singular. Existe exactamente un punto en E cuya coordenada Z es 0, que será el punto $(0:1:0)$. Lo llamamos punto en el infinito y lo denotamos por O.

Por simplicidad reescribimos la ecuación de Weierstrass para una curva elíptica usando las coordenadas afines (ecuación no homogénea) $x=X/Z$ e $y=Y/Z$ como

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 .$$

Una curva elíptica E es el conjunto de soluciones de la ecuación anterior en el plano afín $\mathbb{A}^2(\overline{\mathbb{K}}) = \overline{\mathbb{K}} \times \overline{\mathbb{K}}$, junto con el punto en el infinito O. Si $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$ diremos que E está definida sobre $\mathbb{K}$, y la denotaremos por $E/\mathbb{K}$.

Si E está definida sobre $\mathbb{K}$, el conjunto de puntos cuyas coordenadas pertenecen a $\mathbb{K}$ y que cumplen la ecuación, junto con el punto en el infinito O, forman el conjunto de puntos racionales de E sobre $\mathbb{K}$, que denotaremos por $E(\mathbb{K})$.

2.3 Adición de puntos

Los puntos racionales de una curva elíptica, $E(\mathbb{K})$, junto con una operación llamada suma de puntos, forman un grupo abeliano, $(E(\mathbb{K}),+)$, definido analíticamente como sigue.

Para todo $P, Q \in E(\mathbb{K})$:

1. **Adición y duplicación de puntos.**

   Sean $P = (x_1, y_1)$ y $Q = (x_2, y_2)$. Entonces $P + Q = (x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, teniendo que:

$$\lambda = \begin{cases} (y_2 - y_1)(x_2 - x_1)^{-1}, & \text{si } P \neq Q \\ (3x_1^2 + 2a_2x_1 + a_4 - a_1y_1)(2y_1 + a_1x_1 + a_3)^{-1}, & \text{si } P = Q \end{cases}$$

$$x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2$$

$$y_3 = -(\lambda + a_1)x_3 - \beta - a_3$$

2. **Elemento neutro.**

   O es el elemento neutro, por tanto $P + O = O + P = P$.

3. **Opuesto.**

   Sea $P = (x, y)$, entonces $-P = (x, -y - a_1x - a_3)$, punto que también pertenece a la curva, de forma que $(x, y) + (x, -y - a_1x - a_3) = O$.

Consideremos un entero $k \in \mathbb{N}$, llamamos múltiplo de un punto a la operación de sumar un punto P consigo mismo k veces, de forma que $kP = \sum_{i=1}^{k} P$. La extensión para

los enteros $k \leq 0$ se obtiene de forma inmediata con $0P = O$ y $-kP = \sum_{i=1}^{k}(-P)$. Claramente el resultado de estas operaciones son puntos que pertenecen al grupo de puntos racionales de la curva elíptica. El múltiplo de un punto es la operación más importante del álgebra de las curvas elípticas ya que permite definir sistemas criptográficos.

2.4 El discriminante y el *j*-invariante

Dos curvas elípticas $E_1/\mathbb{K}$ y $E_2/\mathbb{K}$ son isomorfas, y escribimos $E_1/\mathbb{K} \cong E_2/\mathbb{K}$, si son esencialmente la misma, es decir, si podemos establecer un cambio de variables biyectivo que nos transforme una curva elíptica en la otra y viceversa. En general decimos que un cambio de variables es "admisible" si da lugar a una curva isomorfa. La relación de isomorfía entre curvas establecida por un cambio de variables admisible es una relación de equivalencia.

Si dos curvas son isomorfas, $E_1/\mathbb{K} \cong E_2/\mathbb{K}$, entonces los grupos abelianos $E_1(\mathbb{K})$ y $E_2(\mathbb{K})$ son isomorfos. Esto es debido a que el cambio de variables establecido permite definir un isomorfismo entre ambos grupos. Sin embargo la implicación contraria no es cierta en general.

Formalmente, dos curvas elípticas

$$E_1/\mathbb{K}: y^2 + a_1x + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

$$E_2/\mathbb{K}: y^2 + c_1x + c_3y = x^3 + c_2x^2 + c_4x + c_6$$

son isomorfas sobre $\mathbb{K}$ si y sólo si existen elementos $u, r, s, t \in \mathbb{K}$, con $u \neq 0$, tales que el cambio admisible de variables

$$(x, y) \mapsto (u^2x + r, u^3y + u^2sx + t)$$

transforma la ecuación de la curva $E_1/\mathbb{K}$ en la ecuación de la curva $E_2/\mathbb{K}$. Asimismo, el cambio de variables inverso

$$(x, y) \mapsto (u^{-2}(x - r), u^{-3}(y - sx - t + rs))$$

transforma la ecuación de la curva $E_2/\mathbb{K}$ en la ecuación de la curva $E_1/\mathbb{K}$.

Los cambios anteriores conducen al conjunto de ecuaciones:

$$
\begin{aligned}
uc_1 &= a_1 + 2s \\
u^2c_2 &= a_2 - sa_1 + 3r - s^2 \\
u^3c_3 &= a_3 + ra_1 + 2t \\
u^4c_4 &= a_4 - sa_3 + 2ra_2 - (t+rs)a_1 + 3r^2 - 2st \\
u^6c_6 &= a_6 + ra_4 + r^2a_2 + r^3 - ta_3 - t^2 - rta_1
\end{aligned}
$$

Veamos ahora dos expresiones que facilitan el cálculo de curvas elípticas singulares y de isomorfismos entre curvas. Dada una curva elíptica $E/\mathbb{K}$ en la forma vista, precisamos de las expresiones siguientes:

$$
\begin{aligned}
d_2 &= a_1^2 + 4a_2 \\
d_4 &= a_1a_3 + 2a_4 \\
d_6 &= a_3^2 + 4a_6 \\
d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \\
e_4 &= d_2^2 - 24d_4
\end{aligned}
$$

para definir el discriminante Δ de la ecuación de Weierstrass:

$$\Delta = -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6$$

y, cuando $\Delta \neq 0$, el *j*-invariante de la curva, que denotamos por $j(E)$, como:

$$j(E) = \frac{e_4^3}{\Delta}$$

Los siguientes teoremas muestran la utilidad de estas expresiones, cuya demostración puede hallarse en [91].

Teorema 2.1 (Proposición 1.4 (a) de [91]): *Una curva elíptica es singular si y sólo si* $\Delta = 0$.

Teorema 2.2 (Proposición 1.4 (b) de [91]): *Si dos curvas elípticas son isomorfas sobre* $\mathbb{K}$, $E_1/\mathbb{K} \cong E_2/\mathbb{K}$, *entonces* $j(E_1) = j(E_2)$. *La implicación contraria es cierta si* $\mathbb{K}$ *es un cuerpo algebraicamente cerrado.*

Por tanto, si dos curvas tienen el mismo *j*-invariante, entonces son isomorfas en $\mathbb{K}$ o en alguna extensión de $\mathbb{K}$.

2.5 Curvas elípticas sobre cuerpos finitos

Según sea la característica del cuerpo sobre el que está definida la curva, podemos particularizar tanto la ecuación que define la curva, como las operaciones con los puntos de la misma, obteniendo notables simplificaciones. Como comentamos, se tratarán los cuerpos de característica mayor que 3, por una parte, y los de característica 2, por otra.

2.5.1 Sobre cuerpos finitos de característica mayor que 3

Los cuerpos finitos de característica distinta de 2 y 3 admiten una considerable reducción de la ecuación en la forma normal o "forma larga" de Weierstrass. Es indiferente que el orden del cuerpo sea primo o potencia de primo.

Consideremos una curva $E/\mathbb{K}$ definida por la ecuación de Weierstrass. Si la característica del cuerpo es distinta de 2, podemos aplicar el cambio admisible de variables:

$$(x,y)\mapsto\left(x,y-\frac{a_1}{2}x-\frac{a_3}{2}\right)$$

para obtener la curva

$$E'/\mathbb{K}: y^2=x^3+b_2x^2+b_4x+b_6$$

que nos proporciona el isomorfismo $E/\mathbb{K}\cong E'/\mathbb{K}$ ya que el cambio de variables es biyectivo. Si la característica del cuerpo también es distinta de 3, podemos aplicar el cambio admisible de variables:

$$(x,y)\mapsto\left(\frac{x-3b_2}{36},\frac{y}{216}\right)$$

con el que obtenemos la curva

$$E''/\mathbb{K}: y^2=x^3+ax+b$$

que es isomorfa a la anterior, $E'/\mathbb{K}\cong E''/\mathbb{K}$, ya que el cambio de variables también es biyectivo. Es obvio que $E/\mathbb{K}\cong E''/\mathbb{K}$. A partir de ahora consideramos que esta última ecuación, que se dice está en la "forma corta" de Weierstrass, permite describir cualquier curva elíptica definida sobre un cuerpo finito de característica mayor que 3.

Podemos particularizar el discriminante y el j-invariante para estas curvas, que se reducen considerablemente (notar que $1.728=2^6\cdot 3^3$):

$$\Delta = -16(4a^3 + 27b^2)$$

$$j(E) = \frac{-1.728(4a)^3}{\Delta}$$

Definimos ahora la adición de puntos de la curva para estos cuerpos. Sea una curva elíptica $E/\mathbb{K}: y^2 = x^3 + ax + b$, con la característica de $\mathbb{K}$ mayor que 3 y $\Delta \neq 0$, es decir, $4a^3 + 27b^2 \neq 0$. Para todo $P, Q \in E(\mathbb{K})$, la operación del grupo $(E(\mathbb{K}), +)$ se define de la siguiente forma:

1. **Adición y duplicación de puntos.**

   Sean $P = (x_1, y_1)$ y $Q = (x_2, y_2)$. Entonces $P + Q = (x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, siendo:

$$\lambda = \begin{cases} (y_2 - y_1)(x_2 - x_1)^{-1} & , \text{ si } P \neq Q \\ (3x_1^2 + a)(2y_1)^{-1} & , \text{ si } P = Q \end{cases}$$

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$

2. **Elemento neutro.**

   O es el elemento neutro: $P + O = O + P = P$.

3. **Opuesto.**

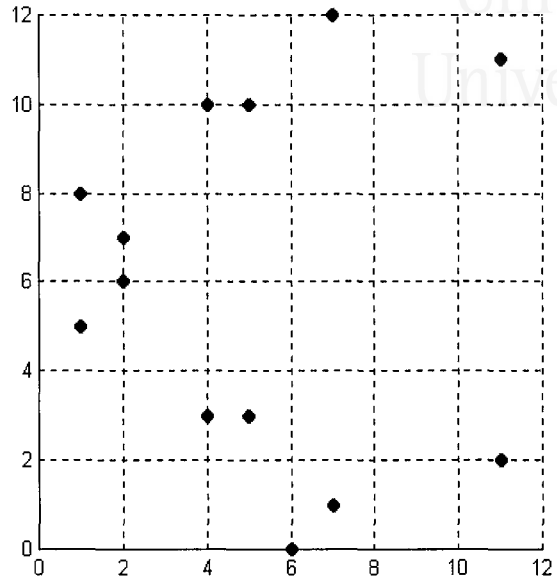
   Sea $P = (x, y)$, entonces $-P = (x, -y)$, que también pertenece a $E(\mathbb{K})$, de forma que $(x, y) + (x, -y) = O$.

Si comparamos una curva elíptica definida sobre un cuerpo finito con una curva definida sobre $\mathbb{R}$, veremos que los cambios no son aparentemente grandes, pero el hecho es que obtenemos una estructura que difiere considerablemente de la percepción visual que teníamos de una curva, y tan sólo mantiene algo de simetría. Como ejemplo podemos ver la curva $E(\mathbb{Z}_{13}): y^2 = x^3 + 4x + 7$ en la figura 2.9.

Para hacernos una idea de las posibles curvas existentes y de los puntos que contiene cada una de ellas, usamos el cuerpo $\mathbb{Z}_{13}$. El plano $\mathbb{Z}_{13} \times \mathbb{Z}_{13}$ consta de 169 puntos posibles, del $(0,0)$ al $(12,12)$. Variando los coeficientes a y b de la ecuación que define la curva, obtendríamos 169 posibilidades.

De las 169, tenemos que 156 cumplen la restricción $4a^3 + 27b^2 \neq 0$, es decir, 13 curvas son singulares.

Figura 2.9: Ejemplo de curva elíptica sobre $\mathbb{Z}_p$



Si analizamos las curvas de forma empírica, veremos que la cantidad de puntos posibles varían desde 6 (por ejemplo, $a=0$ y $b=7$) hasta 20 (por ejemplo, $a=0$ y $b=9$), a los que hemos de añadir el punto en el infinito, con lo que nos queda una variación del cardinal de los grupos posibles entre 7 y 21 puntos.

Un tamaño usual para p es de 163 bits, lo que equivale aproximadamente a un número de 49 cifras en decimal. Así podemos hacernos una idea de la gran cantidad de curvas disponibles y del elevado número de puntos de cada curva.

2.5.2 Sobre cuerpos finitos de característica 2

La ecuación que define una curva elíptica según la forma larga de Weierstrass también admite una reducción si está definida sobre un cuerpo de característica 2. Sin embargo, el cambio de variables que podemos aplicar depende de si el j-invariante es igual a 0 o no. En este caso resulta $j(E)=a_1^{12}/\Delta$, por tanto si $a_1 \neq 0$ entonces $j(E)\neq 0$. Posteriormente veremos que si $j(E)=0$ se trata de un tipo de curva elíptica llamada supersingular que no es útil en criptografía, por tanto no definiremos la adición de puntos en estas curvas.

Consideremos una curva elíptica $E/\mathbb{F}_{2^m}$ con $a_1 \neq 0$, y por tanto $j(E)\neq 0$. El cambio admisible de variables:

$$(x,y)\mapsto\left(a_1^2x+\frac{a_3}{a_1},a_1^3y+\frac{a_1^2a_4+a_3^2}{a_1^3}\right)$$

proporciona la curva:

$$E'/\mathbb{F}_{2^m} : y^2 + xy = x^3 + a_2x^2 + a_6,$$

siendo su discriminante Δ y su j-invariante:

$$\Delta = a_6 \quad \text{y} \quad j(E') = \frac{1}{a_6}$$

Sea una curva elíptica $E/\mathbb{F}_{2^m}$ con $j(E) \neq 0$, $\Delta \neq 0$, y por tanto $a_6 \neq 0$. Para todo $P, Q \in \mathbb{F}_{2^m}$, la operación del grupo $\left(E(\mathbb{F}_{2^m}), +\right)$ se define de la siguiente forma:

1. **Adición y duplicación de puntos.**

   Sean $P = (x_1, y_1)$ y $Q = (x_2, y_2)$. Entonces $P + Q = (x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, siendo:

$$\left.\begin{aligned} x_3 &= (y_1 + y_2)(x_1 + x_2)^{-1} + (y_1 + y_2)(x_1 + x_2)^{-1} + x_1 + x_2 + a_2 \\ y_3 &= (y_1 + y_2)(x_1 + x_2)^{-1}\left(x_1 + x_3\right) + x_3 + y_1 \end{aligned}\right\} \text{ si } P \neq Q$$

$$\left.\begin{aligned} x_3 &= x_1^2 + a_6 x_1^{-2} \\ y_3 &= x_1^2 + \left(x_1 + y_1 x_1^{-1}\right) x_3 + x_3 \end{aligned}\right\} \text{ si } P = Q.$$

2. **Elemento neutro.**

   $P + O = O + P = P$.

3. **Opuesto.**

   Sea $P = (x, y)$, entonces $-P = (x, y + x)$, que también pertenece a $E(\mathbb{F}_{2^m})$, de forma que $(x, y) + (x, y + x) = O$.

2.6 Estructura del grupo

2.6.1 El grupo de puntos racionales y el endomorfismo de Fröbenius

Dado que el cuerpo sobre el que tratamos la curva es finito, también lo es el número de puntos de la curva. Se define el orden del grupo de una curva $E(\mathbb{F}_q)$, que denotamos por $\#E(\mathbb{F}_q)$, como el número de puntos racionales que verifican la ecuación de dicha curva. Como veremos, en aplicaciones criptográficas es necesario que este cardinal sea primo o

al menos casi primo, es decir, divisible por un primo grande, para evitar determinados ataques.

Sabemos que, para cada coordenada x de un punto, podemos tener a lo sumo dos soluciones en la ecuación de la curva más el punto en el infinito, por tanto podemos asegurar que $\#E(\mathbb{F}_q) \leq 2q+1$. Si procedemos de forma heurística, podemos comprobar que para cada valor aleatorio de x la ecuación tiene una probabilidad de solución muy cercana a $1/2$. En consecuencia el orden de la curva tomará un valor aproximado a q, $\#E(\mathbb{F}_q) \approx q$.

Se define la traza t de Fröbenius según la siguiente expresión:

$$\#E(\mathbb{F}_q) = q+1-t.$$

La aplicación de Fröbenius de q-ésima potencia sobre una curva elíptica $E/\mathbb{F}_q$ se define como $\varphi : E(\overline{\mathbb{F}}_q) \to E(\overline{\mathbb{F}}_q)$ tal que:

$$\varphi(x,y) = \begin{cases} (x^q, y^q) & \text{si } (x,y) \neq O \\ O & \text{si } (x,y) = O \end{cases}$$

La aplicación φ se conoce comúnmente como endomorfismo de Fröbenius. Además, se encuentra ligado a la traza de Fröbenius mediante la ecuación:

$$\varphi^2 - t\varphi + q = 0,$$

es decir, para cualquier punto $P = (x,y)$ de la curva, tenemos que:

$$(x^{q^2}, y^{q^2}) - t \cdot (x^q, y^q) + q \cdot (x,y) = O,$$

donde la suma y la resta son operaciones sobre la curva. Esta expresión resulta fundamental en el cómputo del cardinal del grupo de puntos de una curva elíptica. El teorema de Hasse realiza una primera aproximación a dicha cantidad (véase [91]).

Teorema 2.3 (Teorema de Hasse): *La traza de Fröbenius t cumple que*

$$|t| \leq 2\sqrt{q},$$

y por tanto

$$\left| q+1-\#E(\mathbb{F}_q) \right| \leq 2\sqrt{q}.$$

En consecuencia, la probabilidad de obtener un punto de la curva a partir de una coordenada x, será al menos de $1/2 - 1/\sqrt{q}$. Con los valores usuales de q, que son muy grandes, la probabilidad se acercará mucho a $1/2$, lo cual indica lo sencillo que resulta encontrar un punto de la curva. Además, todos los puntos excepto a lo sumo 3, tendrán una coordenada x para dos coordenadas y correspondientes (serán los puntos de corte con el eje x). Estas excepciones son los puntos de orden 2, es decir, los puntos cuyas

coordenadas y son nulas (en la ecuación corta de Weierstrass para cuerpos de característica mayor que 3). Por tanto, el número esperado de puntos racionales es $q+1$ (considerando el punto en el infinito).

Como vemos, los puntos del grupo $E(\mathbb{F}_q)$ siguen una distribución casi uniforme. En el caso particular de curvas sobre $\mathbb{F}_p$, con p primo, existe una curva elíptica cuyo grupo de puntos racionales es de cualquier orden que se halle en el intervalo $\left(p+1-2\sqrt{p}, p+1+2\sqrt{p}\right)$ [58]. Sin embargo, estas propiedades no se cumplen para curvas sobre cuerpos de característica 2.

2.6.2 Curvas especiales

Existen dos tipos de curvas con características especiales que no son útiles en criptografía ya que se dispone de métodos que las hacen fácilmente atacables: las curvas supersingulares y las curvas anómalas. Por tanto, sólo es necesario conocer su definición para evitarlas en implementaciones prácticas.

Una curva $E/\mathbb{F}_q$ es supersingular si la característica p de $\mathbb{F}_q$ divide a la traza de Fröbenius t. De forma equivalente, se puede decir que una curva sobre un cuerpo finito con característica p es supersingular si y sólo si se cumple una de las dos condiciones siguientes:

1. $p=2$ ó 3, y $j(E)=0$.
2. $p \geq 5$ y $t=0$.

Una curva $E/\mathbb{F}_q$ es anómala cuando su traza de Fröbenius es 1, con lo que $\#E(\mathbb{F}_q)=q$. Estas curvas son criptográficamente débiles cuando $q=p$, la característica del cuerpo.

Las curvas de Koblitz [47], también llamadas curvas binarias anómalas, son curvas sobre el cuerpo finito $\mathbb{F}_{2^m}$ pero cuyas ecuaciones toman los coeficientes de $\mathbb{F}_2$. Para cada m, existen por tanto dos curvas de Koblitz sobre $\mathbb{F}_{2^m}$:

$$y^2 + xy = x^3 + 1$$

$$y^2 + xy = x^3 + x^2 + 1$$

El interés de las curvas de Koblitz en criptografía se debe a varios trabajos, [59], [100] y [101], en los que se muestra cómo calcular múltiplos de un punto de manera muy eficiente, con lo que en determinados entornos de recursos restringidos, estas curvas son una buena opción.

2.6.3 Estructura general del grupo

Mediante $\mathbb{Z}_n$ denotamos el grupo abeliano, cíclico y finito de n elementos con la adición módulo n. Este grupo es único. La operación $\oplus$ entre estos grupos denota la suma directa de los mismos. Todo grupo abeliano finito G puede descomponerse de forma única como suma directa de grupos cíclicos:

$$G = \mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2} \oplus \cdots \oplus \mathbb{Z}_{n_s}$$

donde $n_{i+1} \mid n_i$ para todo $i = 1,2,\ldots,s-1$. Decimos que G es un grupo abeliano de rango s y tipo $(n_1, n_2, \ldots, n_s)$.

$E(\mathbb{F}_q)$ es un grupo abeliano de tipo (n_1, n_2) y rango 1 ó 2. Esto significa que es isomorfo a $\mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}$, donde $n_2 \mid n_1$ y además $n_2 \mid q-1$.

Obviamente $\#E(\mathbb{F}_q) = n_1 n_2$. Si $n_2 = 1$, entonces $E(\mathbb{F}_q)$ es isomorfo a $\mathbb{Z}_{n_1}$, con $n_1 = \#E(\mathbb{F}_q)$, que como sabemos es un grupo cíclico, o sea que existe al menos un punto $P \in E(\mathbb{F}_q)$ que genera el grupo:

$$E(\mathbb{F}_q) = \langle P \rangle = \{kP : 0 \le k \le n_1 - 1\}$$

y cuyo orden es n_1. Si además este orden es primo, entonces todo punto de la curva (excepto O) será generador.

Si, en cambio, $n_2 \neq 1$, entonces el grupo de puntos no será cíclico ya que $n_2 \mid n_1$.

Definimos el orden de un punto $P \in E(\mathbb{F}_q)$, que denotamos por $o(P)$, como el menor entero positivo n tal que $nP = O$. A un punto de orden n también se le denomina punto de torsión n.

Se denota mediante $E(\mathbb{F}_q)[n]$ al subgrupo de puntos de orden n, llamado subgrupo de torsión n, definido por:

$$E(\mathbb{F}_q)[n] = \{P \in E(\overline{\mathbb{F}}_q) \mid nP = O\}.$$

2.7 Cálculo del orden del grupo

En aplicaciones criptográficas, es necesario calcular el número de puntos racionales del grupo de puntos de una curva elíptica, ya que se debe comprobar que este cardinal es

primo o al menos contiene un factor primo grande. Realizar este cómputo en un tiempo razonable no es un problema trivial y, aunque es un problema resuelto, requiere de bastantes recursos para que sea computacionalmente eficiente, recursos no siempre disponibles.

Existen tres formas básicas para obtener curvas elípticas apropiadas para su uso en criptografía atendiendo al número de puntos racionales que posean:

- Generar curvas de forma aleatoria y calcular el orden del grupo hasta encontrar una apropiada. El cálculo del orden se puede realizar directamente, o mediante métodos como el de Shanks y Mestre [22] o el de Schoof [84], [85].
- Generar curvas con el orden del grupo escogido previamente usando la teoría de multiplicación compleja. Sobre $\mathbb{F}_p$ puede verse el método de Morain [66], y sobre $\mathbb{F}_{2^m}$ puede verse el de Lay y Zimmer [53]. En [38] se puede ver una descripción detallada. Estos métodos requieren de complejas computaciones, como el cálculo de raíces de polinomios de grados elevados sobre cuerpos finitos grandes. Aún así, resultan computacionalmente más eficientes que el cálculo del número de puntos racionales. El inconveniente de estos métodos es que sólo se pueden generar curvas con determinadas características.
- Usar el grupo de puntos racionales $E(\mathbb{F}_{q^n})$ de una curva E definida sobre $\mathbb{F}_q$ para un q relativamente pequeño. Suelen denominarse "curvas de subcuerpo". En muchos casos se suelen utilizar estas curvas de subcuerpo en característica 2 (curvas de Koblitz, [48]) para aprovechar la mejora de eficiencia en la aritmética de los mismos. El problema aquí es que existen pocas curvas definidas sobre cuerpos finitos pequeños con el subgrupo requerido que posea un orden primo grande sobre el cuerpo extensión.

En general, el primer método es el más adecuado en implementaciones criptográficas, ya que mantiene la aleatoriedad pura deseable en criptografía, y por ello el resto de métodos son a veces descartados o menospreciados. En concreto, al existir muchas menos opciones a elegir en el caso de las curvas de subcuerpo, pueden ser mayormente caracterizadas y quizá se pueda encontrar alguna debilidad en ellas.

2.7.1 Comprobación del orden del grupo

En muchos de los métodos que se utilizan, no siempre se obtiene una única posibilidad como cardinal del grupo, es necesario por tanto comprobar cuál de los candidatos posibles

es el correcto, es decir, dado el grupo de puntos racionales $E(\mathbb{F}_q)$ de una curva elíptica, comprobar que un número m cumple que $\#E(\mathbb{F}_q) = m$.

Sabemos que el grupo de puntos es la suma directa de, como máximo, dos subgrupos cíclicos. Además, por el teorema de Hasse [91], sabemos que el número de puntos racionales de una curva elíptica sobre $\mathbb{F}_q$ cumple que:

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

Tras esta comprobación, la siguiente consistiría en obtener un punto P de la curva de forma aleatoria, y verificar que $mP = O$. Obviamente cuantos más puntos se comprueben, mayor será la probabilidad de que m sea efectivamente el orden de la curva.

Se debe tener en cuenta que el cardinal m debe ser de la forma $m = r \cdot s$, con r un primo grande y s un entero pequeño (típicamente entre 1 y 4). Así, es recomendable aplicar un test de primalidad al cardinal para descartar posibles valores inservibles de m.

Además de estas comprobaciones básicas, existen varios resultados que hacen esta búsqueda más eficiente, como los mostrados en [84].

2.7.2 Cálculo directo

Una forma directa para obtener el número de puntos, que es posible realizar para cuerpos de característica mayor que 3, consiste en evaluar la expresión:

$$\#E(\mathbb{F}_p) = p+1+\sum_{x=0}^{p-1}\left(\frac{x^3+ax+b}{p}\right),$$

donde $\left(\frac{\cdot}{p}\right)$ denota el símbolo de Legendre o del residuo cuadrático [86]. Para valores de p no demasiado grandes ($p < 10.000$, véase [22]) puede ser una alternativa sencilla y eficaz.

2.7.3 El método de Shanks y Mestre

Existen métodos más eficientes que el cálculo directo, basados normalmente en técnicas de raíz cuadrada. Uno de los más usados es el de Shanks y Mestre [22], método que combina el BSGS (*Baby Step, Giant Step,* saltos enanos, saltos gigantes) de Shanks con el teorema de Hasse. Esta técnica da mejores resultados que el cálculo directo a partir de $p > 457$.

La idea consiste en determinar un punto $P \in E(\mathbb{F}_q)$ de forma aleatoria y calcular un entero m en el intervalo

$$q+1-2\sqrt{q} \le m \le q+1+2\sqrt{q}$$

tal que $mP = O$. Si m es el único punto en ese intervalo, entonces por el teorema de Hasse, $m = \#E(\mathbb{F}_q)$.

Para inicializar este método, primero se escoge el punto P. Sea m el entero buscado, sabemos que $mP = O$. Tras ello calculamos los saltos enanos:

$$0, \pm P, \pm 2P, \ldots, \pm sP, \text{ con } s \approx \sqrt[4]{q},$$

con lo que tenemos $2s+1$ puntos. Ahora se obtiene $Q = (2s+1)P$ y, usando la expansión binaria de $q+1$, calculamos $R = (q+1)P$. Los pasos gigantes quedan entonces:

$$R, R \pm Q, R \pm 2Q, \ldots, R \pm tQ, \text{ con } t = \frac{2\sqrt{q}}{2s+1},$$

que es aproximadamente igual a $\sqrt[4]{q}$.

Por el teorema de Hasse, deben coincidir en algún momento $R+iQ$ y jP, para algún i y j, con $-t \le i \le t$ y $-s \le j \le s$, de forma que:

$$m = q+1+(2s+1)i-j$$

En el caso de que existan dos enteros m y m' tales que $mP = m'P = O$, entonces el algoritmo falla. En este caso $o(P) = m-m'$, es decir, $(m-m')P = O$. Éste es un valor pequeño y rara vez ocurre. Para salvar este inconveniente suele ser suficiente ejecutar de nuevo el algoritmo con otro punto aleatorio.

La complejidad del método de Shanks y Mestre es de $O(q^{1/4+\varepsilon})$, tanto en tiempo como en espacio, donde ε es una constante positiva que puede hacerse arbitrariamente pequeña. Se pueden utilizar las mismas técnicas que en los algoritmos de raíz cuadrada (como el Pollard–ρ [77]) para reducir el espacio de almacenamiento necesario, sacrificando algo de la complejidad temporal de ejecución. En cualquier caso, para valores elevados de q, este método se hace rápidamente inviable.

2.7.4 El algoritmo de Schoof

El algoritmo presentado por Schoof en 1985 [85] para calcular el número de puntos racionales de una curva elíptica sobre un cuerpo finito, supuso una mejora fundamental en dicha labor, ya que redujo la complejidad exponencial de los algoritmos conocidos hasta la fecha a una complejidad polinómica. A pesar de ser una tarea bastante compleja, se

consiguió que el cálculo de puntos fuera computacionalmente viable en un tiempo razonable. Este tiempo de ejecución viene marcado por una complejidad de $O(\log^8 q)$.

Por el teorema de Hasse, $\#E(\mathbb{F}_q) = q+1-t$, con $|t| \leq 2\sqrt{q}$. El fundamento del algoritmo de Schoof consiste en determinar t módulo una serie de primos l, para $l \leq L$, donde L es el primo más pequeño tal que:

$$\prod_{\substack{2 \leq l \leq L \\ l \text{ primo}}} l > 4\sqrt{q}$$

De todos los resultados obtenidos se averigua cuál es el correcto mediante el teorema chino del resto.

Inicialmente este algoritmo no era muy eficiente en implementaciones prácticas; de hecho, cuando el número de puntos de la curva tomaba valores grandes para obtener sistemas seguros, el algoritmo se volvía inviable. Las numerosas mejoras producidas posteriormente lo han hecho suficientemente práctico, de forma que se ha logrado obtener el número de puntos racionales de curvas definidas sobre cuerpos con cardinales de más de 200 dígitos en binario en algunas horas [41]. Las marcas actuales se encuentran en algunos miles de dígitos en binario, cantidad más que suficiente para criptografía de curva elíptica. Las mejoras se deben fundamentalmente a Atkin y Elkies [28], [54], [84]. En cualquier caso, el algoritmo es complejo y requiere de recursos suficientes para su ejecución cuando se usan curvas de tamaños adecuados en criptografía.

2.8 Isomorfismos entre curvas elípticas

Veamos ahora cómo obtener curvas isomorfas a una dada y, cuando sea posible, el número de clases de isomorfía y el número de curvas isomorfas en cada clase.

2.8.1 Sobre cuerpos finitos de característica mayor que 3

Dos curvas elípticas definidas sobre un cuerpo de característica mayor que 3,

$$E_1 / \mathbb{F}_q : y^2 = x^3 + a_1 x + b_1 \quad \text{y} \quad E_2 / \mathbb{F}_q : y^2 = x^3 + a_2 x + b_2,$$

son isomorfas sobre $\mathbb{F}_q$ si y sólo si existe un elemento $u \in \mathbb{F}_q$, con $u \neq 0$, tal que $u^4 a_2 = a_1$ y $u^6 b_2 = b_1$.

Las aplicaciones que nos proporcionan el isomorfismo son

$$\phi(x,y) = (u^{-2}x, u^{-3}y),$$

que nos transforma la curva $E_1/\mathbb{K}$ en $E_2/\mathbb{K}$, obteniendo

$$E_2/\mathbb{F}_q : y^2 = x^3 + u^4 a_2 x + u^6 b_2;$$

y

$$\varphi(x,y) = (u^2 x, u^3 y),$$

que nos transforma $E_2/\mathbb{K}$ en $E_1/\mathbb{K}$, recuperando la ecuación inicial.

El isomorfismo entre curvas elípticas nos define una relación de equivalencia, de forma que una clase de isomorfía estará formada por aquellas curvas isomorfas entre sí.

Para averiguar el número de clases de isomorfía entre curvas, lo primero que debemos observar es que $a_1 = 0$ si y sólo si $a_2 = 0$, y $b_1 = 0$ si y sólo si $b_2 = 0$.

Consideremos ahora los casos posibles:

1. Si $a_1 = 0$ y $b_1 = 0$, entonces la curva es singular ya que $\Delta = 0$.

2. Si $a_1 \neq 0$ y $b_1 = 0$, entonces $j(E) = 1728$. En este caso $u^4 = \dfrac{a_1}{a_2}$, y puede ocurrir que $\mathbb{F}_q^*$ tenga un elemento α de orden 4, con lo que existirán cuatro soluciones: $\{u, \alpha u, \alpha^2 u, \alpha^3 u\}$. Si no existe dicho elemento, sólo hay dos soluciones: $\{u, -u\}$.

3. Si $a_1 = 0$ y $b_1 \neq 0$, entonces $j(E) = 0$. En este caso $u^6 = \dfrac{b_1}{b_2}$, y puede ocurrir que $\mathbb{F}_q^*$ tenga un elemento α de orden 3, con lo que existirán seis soluciones: $\{u, \alpha u, \alpha^2 u, -u, -\alpha u, -\alpha^2 u\}$. Si no existe dicho elemento, sólo hay dos soluciones: $\{u, -u\}$.

4. Si $a_1 \neq 0$ y $b_1 \neq 0$, entonces $j(E) \neq 0, 1728$. En este caso $u^2 = \dfrac{a_2 b_1}{a_1 b_2}$, con lo que sólo hay dos soluciones: $\{u, -u\}$.

En el caso general, tenemos el siguiente teorema.

Teorema 2.4 (Teorema 3.3 de [60]): *El número de clases de isomorfía entre curvas elípticas sobre un cuerpo finito* $\mathbb{F}_q$ *con característica mayor que 3, es* $2q+6$, $2q+2$, $2q+4$ *y* $2q$, *para* $q \equiv 1,5,7,11 \pmod{12}$ *respectivamente.*

Veamos un ejemplo. Denotamos mediante $E_{a,b}$ la curva dada por la ecuación $y^2 = x^3 + ax + b$. La tabla 2.1 muestra todas las curvas posibles sobre el cuerpo $\mathbb{F}_7$ agrupadas por clases de isomorfía.

Tabla 2.1: Clases de isomorfía entre curvas elípticas sobre $\mathbb{F}_7$

Clase de isomorfía	Nº de puntos	Grupo isomorfo	$j(E)$
$E_{1,1} \cong E_{4,1} \cong E_{2,1}$	5	$\mathbb{Z}_5$	1
$E_{1,3} \cong E_{4,3} \cong E_{2,3}$	6	$\mathbb{Z}_6$	5
$E_{1,4} \cong E_{4,4} \cong E_{2,4}$	10	$\mathbb{Z}_{10}$	5
$E_{1,6} \cong E_{4,6} \cong E_{2,6}$	11	$\mathbb{Z}_{11}$	1
$E_{3,1} \cong E_{5,1} \cong E_{6,1}$	12	$\mathbb{Z}_2 \oplus \mathbb{Z}_6$	2
$E_{3,2} \cong E_{5,2} \cong E_{6,2}$	9	$\mathbb{Z}_9$	3
$E_{3,3} \cong E_{5,3} \cong E_{6,3}$	6	$\mathbb{Z}_6$	4
$E_{3,4} \cong E_{5,4} \cong E_{6,4}$	10	$\mathbb{Z}_{10}$	4
$E_{3,5} \cong E_{6,5} \cong E_{5,5}$	7	$\mathbb{Z}_7$	3
$E_{3,6} \cong E_{5,6} \cong E_{6,6}$	4	$\mathbb{Z}_2 \oplus \mathbb{Z}_2$	2
$E_{1,0} \cong E_{2,0} \cong E_{4,0}$	8	$\mathbb{Z}_8$	6
$E_{3,0} \cong E_{5,0} \cong E_{6,0}$	8	$\mathbb{Z}_2 \oplus \mathbb{Z}_4$	6
$E_{0,1}$	12	$\mathbb{Z}_2 \oplus \mathbb{Z}_6$	0
$E_{0,2}$	9	$\mathbb{Z}_9$	0
$E_{0,3}$	13	$\mathbb{Z}_{13}$	0
$E_{0,4}$	3	$\mathbb{Z}_3$	0
$E_{0,5}$	7	$\mathbb{Z}_7$	0
$E_{0,6}$	4	$\mathbb{Z}_2 \oplus \mathbb{Z}_2$	0

Como vemos, efectivamente hay $2q + 4 = 18$ clases de isomorfía. Además, los grupos distintos isomorfos a las clases, son sólo 12, es decir, existen 12 grupos diferentes de puntos racionales $E(\mathbb{F}_7)$ asociados a las clases de isomorfía de las curvas $E/\mathbb{F}_7$.

Las curvas con $b = 0$ son las que contienen 8 puntos racionales, por tanto si $\#E(\mathbb{F}_q) = q + 1 - t$, entonces $t = 0$, lo que nos indica que son curvas supersingulares. Las curvas con $b = 5$ contienen 7 puntos racionales, que coincide con el tamaño del cuerpo sobre el que están definidas, por tanto son curvas anómalas.

2.8.2 Sobre cuerpos finitos de característica 2

Es preciso realizar algunas definiciones básicas sobre cuerpos finitos antes de proceder a la clasificación.

Se define la función traza Tr (que no debemos confundir con la traza de Fröbenius), como la función lineal $Tr:\mathbb{F}_{2^m}\to\mathbb{F}_2$ dada por:

$$Tr(\alpha)=\alpha+\alpha^2+\alpha^{2^2}+\cdots+\alpha^{2^{m-1}}.$$

Dado que las curvas elípticas supersingulares no son útiles en criptografía, caracterizamos el isomorfismo entre curvas para curvas no supersingulares, por ello $j(E)\neq 0$ y por tanto el término independiente de la ecuación en la forma corta de Weierstrass es no nulo.

Teorema 2.5 (Teorema 2.2 de [60]): *Dos curvas elípticas no supersingulares definidas sobre un cuerpo de característica 2,*

$$E_1/\mathbb{F}_{2^m}: y^2+xy=x^3+a_2x^2+a_6 \quad y \quad E_2/\mathbb{F}_{2^m}: y^2+xy=x^3+c_2x^2+c_6$$

con $a_6\neq 0$ *y* $c_6\neq 0$, *son isomorfas sobre* $\mathbb{F}_{2^m}$ *si y sólo si* $a_6=c_6$ *y existe un elemento* $s\in\mathbb{F}_{2^m}$, *con* $s\neq 0$, *tal que* $c_2=a_2+s+s^2$. *Esta última condición es equivalente a* $Tr(c_2)=Tr(a_2)$.

Del teorema anterior se derivan las posibles clases de isomorfía: si γ es un elemento tal que $Tr(\gamma)=1$ (si m es impar, podemos tomar $\gamma=1$), un representante de cada clase viene dado por:

$$\left\{y^2+xy=x^3+a_2x^2+a_6 \mid a_6\in\mathbb{F}_{2^m}^*, a_2\in\{0,\gamma\}\right\}.$$

Los cambios de variable que proporcionan el isomorfismo $E_1/\mathbb{F}_{2^m}\cong E_2/\mathbb{F}_{2^m}$ son los correspondientes a las aplicaciones:

$$\phi(x,y)=(x,y+sx)$$

para transformar $E_1/\mathbb{F}_{2^m}$ en $E_2/\mathbb{F}_{2^m}$, y la misma aplicación:

$$\varphi(x,y)=(x,y+sx)$$

para transformar $E_2/\mathbb{F}_{2^m}$ en $E_1/\mathbb{F}_{2^m}$ (no debemos olvidar que en los cuerpos de característica 2 cada elemento es su propio opuesto).

Dada una curva $E/\mathbb{F}_{2^m}: y^2+xy=x^3+a_2x^2+a_6$, se pretende contar las curvas isomorfas a ella. El elemento a_6 debe mantenerse, pero a_2 puede cambiarse por cualquier otro elemento que posea la misma traza. El total de elementos con la misma traza es $q/2$,

luego cada clase de isomorfía viene dada por $q/2$ curvas isomorfas entre ellas, es decir 2^{m-1}.

Ahora bien, para contar el número total de clases de isomorfía en $\mathbb{F}_{2^m}$, tenemos que a_6 puede ser cualquier elemento excepto el cero, es decir, $q-1$ posibilidades, y a_2 puede ser cualquier elemento siempre que tenga la misma traza. Como trazas posibles sólo hay 2, tenemos $2(q-1)$ clases de isomorfía.

2.9 Entrelazamiento de curvas elípticas

Veamos ahora una propiedad que nos relaciona las clases de isomorfía de curvas elípticas obteniendo algunas ventajas sobre el número de puntos.

2.9.1 Sobre cuerpos finitos de característica mayor que 3

Una curva en la forma corta de Weierstrass, $E_1/\mathbb{F}_q : y^2 = x^3 + a_1x + b_1$, está entrelazada a otra $E_2/\mathbb{F}_q : y^2 = x^3 + a_2x + b_2$ si $a_2 = u^2a_1$ y $b_2 = u^3b_1$ para algún residuo no cuadrático $u \in \mathbb{F}_q$.

El entrelazamiento de curvas nos define realmente un isomorfismo inviable de conseguir en $\mathbb{F}_q$, pero sí en $\overline{\mathbb{F}}_q$. Podemos calcular el *j*-invariante y comprobar que es el mismo en ambas curvas, pero éstas no pueden ser isomorfas porque entonces *u* tendría que ser residuo cuadrático en $\mathbb{F}_q$ y no lo es. De hecho ambas curvas son isomorfas sobre $\mathbb{F}_{q^2}$, donde *u* pasa a ser residuo cuadrático.

Teorema 2.6 (Apartado III.3.1 de [5]): *Sean* $E_1/\mathbb{F}_q$ *y* $E_2/\mathbb{F}_q$ *dos curvas elípticas entrelazadas. Los órdenes de los grupos de puntos racionales de ambas curvas satisfacen la relación:*

$$\#E_1(\mathbb{F}_q) + \#E_2(\mathbb{F}_q) = 2q + 2.$$

La ventaja de este teorema es obvia, conociendo el número de puntos de una curva, podemos conocer directamente el número de puntos de las curvas entrelazadas a la misma.

Se puede deducir fácilmente de la definición de entrelazamiento, que si dos curvas, $E_1/\mathbb{F}_q$ y $E_2/\mathbb{F}_q$, están entrelazadas a otra curva $E_3/\mathbb{F}_q$, entonces $E_1/\mathbb{F}_q$ y $E_2/\mathbb{F}_q$ son

isomorfas. Esto nos conduce a un entrelazamiento de clases de isomorfía, ya que si entre dos clases distintas existen dos curvas entrelazadas, una de cada clase, entonces cada curva de cada clase estará entrelazada con todas las curvas de la otra clase.

En la tabla 2.2, que muestra las curvas posibles sobre $\mathbb{F}_7$, se puede ver claramente la anterior aserción.

Tabla 2.2: Entrelazamiento de clases entre curvas elípticas sobre $\mathbb{F}_7$

Clases entrelazadas		$j(E)$	suma de puntos
$E_{1,1} \cong E_{4,1} \cong E_{2,1}$	$E_{1,6} \cong E_{4,6} \cong E_{2,6}$	1	$5+11=16$
$E_{1,3} \cong E_{4,3} \cong E_{2,3}$	$E_{1,4} \cong E_{4,4} \cong E_{2,4}$	5	$6+10=16$
$E_{3,1} \cong E_{5,1} \cong E_{6,1}$	$E_{3,6} \cong E_{5,6} \cong E_{6,6}$	2	$12+4=16$
$E_{3,2} \cong E_{5,2} \cong E_{6,2}$	$E_{3,5} \cong E_{6,5} \cong E_{5,5}$	3	$9+7=16$
$E_{3,3} \cong E_{5,3} \cong E_{6,3}$	$E_{3,4} \cong E_{5,4} \cong E_{6,4}$	4	$6+10=16$
$E_{1,0} \cong E_{2,0} \cong E_{4,0}$	$E_{3,0} \cong E_{5,0} \cong E_{6,0}$	6	$8+8=16$
$E_{0,1}$	$E_{0,6}$	0	$12+4=16$
$E_{0,2}$	$E_{0,5}$	0	$9+7=16$
$E_{0,3}$	$E_{0,4}$	0	$13+3=16$

Como vemos, si se toman dos curvas de una misma fila, si no son isomorfas, entonces estarán entrelazadas.

2.9.2 Sobre cuerpos finitos de característica 2

Las curvas $E_1/\mathbb{F}_{2^m} : y^2 + xy = x^3 + a_2x^2 + a_6$ y $E_2/\mathbb{F}_{2^m} : y^2 + xy = x^3 + c_2x^2 + c_6$, con $a_6 \neq 0$ y $c_6 \neq 0$, están entrelazadas si $a_6 = c_6$ y $Tr(a_2) \neq Tr(c_2)$.

Claramente el j-invariante es el mismo en ambas curvas ya que está en función únicamente del término a_6. Por otra parte, no son isomorfas ya que $Tr(a_2) \neq Tr(c_2)$.

De la misma forma que en el caso de característica mayor que 3, el entrelazamiento de curvas define un isomorfismo sobre alguna extensión de $\mathbb{F}_{2^m}$, en concreto sobre $\mathbb{F}_{2^{2m}}$, ya que en este cuerpo extensión el valor de la traza se invierte.

Teorema 2.7 (Apartado III.3.2 de [5]): *Dadas dos curvas entrelazadas* $E_1/\mathbb{F}_{2^m}$ *y* $E_2/\mathbb{F}_{2^m}$, *los órdenes de los grupos de puntos racionales de ambas curvas satisfacen la relación:*

$$\#E_1(\mathbb{F}_{2^m}) + \#E_2(\mathbb{F}_{2^m}) = 2^{m+1} + 2 .$$

Por tanto, si conocemos el cardinal del grupo de puntos de una curva, podemos conocer de forma inmediata el número de puntos de las curvas entrelazadas a la misma.

Esto, como en el apartado anterior, nos conduce a un entrelazamiento de clases, ya que si dos curvas, $E_1/\mathbb{F}_{2^m}$ y $E_2/\mathbb{F}_{2^m}$, están entrelazadas a otra curva $E_3/\mathbb{F}_{2^m}$, entonces $E_1/\mathbb{F}_{2^m}$ y $E_2/\mathbb{F}_{2^m}$ son isomorfas.

2.10 Ataques conocidos contra las curvas elípticas

Presentamos en esta sección los ataques conocidos contra el ECDLP y, en los casos de viabilidad de éstos, cómo definir las curvas elípticas para evitarlos.

Para toda esta sección, el ECDLP se define como sigue: dada una curva elíptica $E/\mathbb{F}_q$, un punto $P \in E(\mathbb{F}_q)$ de orden n, y un punto Q tal que $Q = kP$, donde $0 \le k \le n-1$. El ECDLP consiste en encontrar k a partir de P y Q.

1. **Ataque por fuerza bruta**. Consiste en la búsqueda exhaustiva de k a partir de P, es decir, $P, 2P, 3P, \ldots$, hasta encontrar $Q = kP$. La complejidad es claramente exponencial, con un número de iteraciones esperado de $n/2$.

2. **El algoritmo de Pohlig-Hellman** [75]. Este algoritmo se aprovecha de la factorización de n, el orden del punto P. En este caso, la búsqueda de k se reduce a la búsqueda de k módulo cada uno de los factores primos de n. Posteriormente se recupera k usando el teorema chino del resto.

   Para evitar este algoritmo de ataque, el orden de la curva debe ser primo o casi primo, es decir, contener al menos un factor primo grande.

3. **Algoritmo BSGS** (*Baby Step, Giant Step*) [62]. Forma parte de los algoritmos de raíz cuadrada. Consiste en calcular puntos a partir de P, primero dando saltos pequeños, con una diferencia de un punto, y después, saltos grandes, con una diferencia de $\sqrt{n}$. De la coincidencia se obtiene k.

   Tiene una complejidad exponencial, $O\left(\sqrt{n}\right)$, tanto temporal como espacial.

   Este algoritmo, como todos los de raíz cuadrada, siempre son aplicables a cualquier grupo.

4. **Algoritmo Pollard-ρ** [77]. Este algoritmo está basado en el BSGS pero dando los saltos de forma aleatoria. Tiene aproximadamente la misma complejidad temporal que el BSGS, $O\left(\sqrt{\pi n/2}\right)$, pero sin apenas requerimientos de almacenamiento.

   Algunas mejoras posteriores, [33] y [108], consiguen alcanzar una complejidad de $O\left(\sqrt{\pi n}/2\right)$. Además, este algoritmo es paralelizable [74], mejorando la complejidad hasta $O\left(\sqrt{\pi n}/(2r)\right)$, donde r representa el número de procesadores funcionando en paralelo.

5. **Algoritmo Pollard-λ** [77]. El funcionamiento es similar al Pollard-ρ y también puede ser paralelizado de forma eficiente, aunque se muestra ligeramente más lento [74]. Este algoritmo es más rápido que el Pollard-ρ sólo cuando k se encuentra en un subintervalo $[0,b]$ donde $b < 0,39n$ [74].

6. **Múltiples instancias**. El algoritmo Pollard-ρ se encarga de almacenar tan sólo determinados puntos al hacer la búsqueda de k. Una vez resuelta una instancia del problema, es posible utilizar la información almacenada por este algoritmo para acelerar el cálculo de otra instancia del mismo problema [95]. En concreto, si el cálculo de la primera instancia supone un tiempo t, la segunda instancia debe suponer un tiempo de $\left(\sqrt{2}-1\right)t \approx 0,41t$. Tras haber resuelto estas dos instancias, la tercera debe tomar un tiempo de $\left(\sqrt{3}-\sqrt{2}\right)t \approx 0,32t$. De la misma forma, la cuarta instancia debe tomar un tiempo de $\left(\sqrt{4}-\sqrt{3}\right)t \approx 0,27t$, y así sucesivamente.

   Otra forma de ver esto es: resolver l instancias de un ECDLP (para el mismo punto P) supone realmente resolver $\sqrt{l}$ instancias, aunque sin considerar el espacio de almacenamiento requerido.

7. **Ataque MOV** (Menezes-Okamoto-Vanstone) [61]. En ciertas circunstancias, el ECDLP de una curva $E/\mathbb{F}_q$ puede reducirse al DLP en el grupo multiplicativo del cuerpo extensión $\mathbb{F}_{q^r}$ para algún $r \geq 1$, problema mucho más sencillo de resolver si r no es muy grande. Para asegurar que este algoritmo de reducción no es aplicable, hay que comprobar que n no divide a $q^r - 1$ para todo r pequeño para el que se pueda solucionar el DLP bajo $\mathbb{F}_{q^r}$. En la práctica, para valores usuales de $n \approx 2^{160}$, es suficiente comprobar los valores de r con $r \leq 20$ [1].

   Para la mayoría de las curvas, el r resultante será grande [3], con lo que no se podrá aplicar esta reducción. Sin embargo las curvas supersingulares cumplen que $r \leq 6$, con lo que se podría reducir el cálculo a tiempo subexponencial (criba del

cuerpo de números en el DLP). Por ello, estas curvas son excluidas explícitamente en los estándares.

Otro tipo de curvas para las que esta reducción es aplicable, es el de las curvas de traza 2, es decir, las curvas $E/\mathbb{F}_q$ para las que $\#E(\mathbb{F}_q) = q - 1$.

8. **Curvas anómalas de cuerpo primo**. Una curva $E/\mathbb{F}_p$ es anómala de cuerpo primo si $\#E(\mathbb{F}_p) = p$ (caso particular de las curvas anómalas). El ECDLP en estas curvas se puede resolver de forma eficiente [81], [88], [99]. Por tanto, se debe evitar que el cardinal del grupo de puntos racionales coincida con el cardinal del cuerpo sobre el que está definida la curva. Estas curvas son excluidas explícitamente en los estándares.
9. **Curvas definidas sobre cuerpos pequeños** [33], [108]. Supongamos que $E/\mathbb{F}_{2^e}$ es una curva elíptica; es posible acelerar el algoritmo Pollard-ρ por un factor de $\sqrt{d}$ cuando la curva está definida por $E/\mathbb{F}_{2^{ed}}$, por tanto, si $\mathbb{F}_{2^e}$ es un subcuerpo pequeño, la reducción es considerable.
10. **Curvas definidas sobre cuerpos compuestos**. En determinados casos, usando curvas $E/\mathbb{F}_{2^m}$, donde m es número compuesto, se podría usar el "descenso de Weil" para resolver el ECDLP de forma eficiente [32]. En concreto, si m tiene un divisor pequeño, el ECDLP se podría resolver de forma más rápida que con el Pollard-ρ [34].

    En general, es conveniente evitar las curvas sobre cuerpos compuestos. La mayoría de estándares las excluyen explícitamente para usos criptográficos.
11. **Algoritmo *Index-calculus*** [62]. Es el mejor algoritmo que se conoce capaz de atacar al DLP, sin embargo no se ha podido aplicar al ECDLP. De hecho existen trabajos que proporcionan argumentos más o menos sólidos que avalan la no aplicabilidad de este sistema al ECDLP [65], [93].
12. **Algoritmo *Xedni-calculus*** [92]. Una de las últimas formas de ataque ideadas contra el ECDLP; sin embargo el algoritmo falla en la práctica [42].

2.11 Codificación de datos en curvas elípticas

En muchos protocolos criptográficos es necesario transformar determinados datos, normalmente alfanuméricos ya codificados en $\mathbb{Z}_n$, en puntos de una curva elíptica $E/\mathbb{F}_q$. Para ello es preciso transformarlos de $\mathbb{Z}_n$ a coordenadas x e y de los puntos de la curva. Como cada coordenada x se corresponde como máximo a dos puntos de la curva, realmente sólo es necesario obtener la coordenada x y, como mucho, se utiliza un bit más para indicar cuál de las coordenadas y es la correspondiente.

Por una parte, no existe un algoritmo determinista capaz de encontrar una gran cantidad de puntos de una curva pero, como se vio en la sección 2.6 del grupo de la curva, la probabilidad de encontrar un punto de la misma es muy cercana a $1/2$. De hecho, existen algoritmos cuya probabilidad de fallo es muy baja.

Por otra parte, es preciso un método sistemático que permita transformar los elementos de $\mathbb{Z}_n$ en puntos, de forma que mantengan una relación simple y se pueda realizar el proceso inverso con eficacia. De los diversos métodos existentes, exponemos el de Koblitz, uno de los más eficientes y usados, de tipo probabilístico, claro está, pero donde se puede ajustar la probabilidad requerida [46].

Consideremos un alfabeto de N letras y fijemos una longitud l para los bloques de texto. Los caracteres del alfabeto se identifican con los enteros $0,1,\ldots,N-1$. Para establecer una biyección entre un bloque de texto $w=(a_0a_1\ldots a_{l-1})$, con los a_i pertenecientes al alfabeto, y un número $x_w \in \left[0, N^l\right]$, se considera la aplicación:

$$w=(a_0a_1\ldots a_{l-1}) \mapsto x_w = a_0N^{l-1} + a_1N^{l-2} + \cdots + a_{l-2}N + a_{l-1}$$

Si el texto o número a codificar ya se presenta como un único dato o entero, entonces no es necesario realizar esta transformación.

De cualquier manera, partimos de un entero $x_w \in \left[0, N^l\right]$ y debemos realizar la adaptación a un punto de la curva. Lo ideal sería que x_w fuera la coordenada x de un punto de la curva, pero como esto no es posible, se procede como sigue. Se pretende buscar el punto de la curva cuya componente x sea la más cercana a x_w siendo $x_w \le x$. La probabilidad de encontrar un punto de la curva para un x aleatorio es aproximadamente $1/2$, por tanto la probabilidad de encontrar un número en las $k = x - x_w$ primeras pruebas es de $(1-(1/2)^k)$. Tomando $k=50$ es razonablemente imposible no encontrar un punto de la curva.

Por otra parte, debe cumplirse que $kN^l < q$. Ahora tomamos el valor $kx_w + j$ sucesivamente para $j = 0,1,\ldots,k-1$ comprobando si corresponde a la abcisa de un punto de la curva. Cuando encontremos el punto

$$P_w = (kx_w + j, y) \in E(\mathbb{F}_q),$$

ya tendremos la adaptación hecha.

El proceso inverso de transformación del punto en un entero es bien sencillo, sólo es necesario realizar una división para obtener el entero original

$$x_w = \left\lfloor \frac{x}{k} \right\rfloor.$$

2.12 Sistemas criptográficos basados en el ECDLP

Veremos en esta sección los principales sistemas criptográficos que han sido adaptados al ECDLP. Estos sistemas son los expuestos en el capítulo 1 para el DLP, pero ajustados en mayor o menor medida para que funcionen con curvas elípticas. De la misma forma que aquéllos, los esquemas que exponemos en este capítulo proveen de algún servicio criptográfico, como el de intercambio de claves, el de cifrado, o el de firma digital.

Aunque no lo citemos expresamente en cada uno de los esquemas, se sobrentiende que las curvas utilizadas poseen las características necesarias para ser útiles en criptografía, como la no supersingularidad, el poseer un factor primo grande, etc., de forma que se eviten los ataques expuestos previamente.

En los sistemas en los que se cifra un mensaje w para su transmisión, normalmente debe ser embebido previamente en el grupo $E(\mathbb{F}_q)$, es decir, debe codificarse como un punto de la curva. Como todas las coordenadas x e y posibles no existen, es preciso utilizar alguna técnica de codificación como la vista en la sección anterior. En cualquier caso, suponemos, cuando se precise, que el mensaje w ha sido transformado de forma conveniente en la coordenada x_w de un punto W de la curva.

2.12.1 Protocolo de intercambio de claves de Diffie-Hellman con ECC

En este protocolo [106], dos usuarios (Alicia y Bernardo) quieren compartir una clave secreta. Utilizamos para ello el grupo finito de puntos racionales de una curva elíptica, $E(\mathbb{F}_q)$, cuyo cardinal es primo o casi primo, y un punto P de la curva que actúa como generador de un subgrupo de orden un número primo p de gran tamaño, y escribimos $o(P) = p$. Tanto los parámetros que definen la curva como los elementos P y p son públicamente conocidos.

1. Alicia toma una clave secreta $1 \le k_A \le p-1$, calcula su clave pública $K_A = k_A P$ y se la transmite a Bernardo.
2. Éste, de forma similar, toma una clave secreta $1 \le k_B \le p-1$, calcula su clave pública $K_B = k_B P$ y se la transmite a Alicia.
3. Alicia recibe K_B y calcula $Z_1 = k_A K_B$.
4. Bernardo hace lo propio con K_A, calculando $Z_2 = k_B K_A$.

La clave secreta compartida es $Z_1 = Z_2$.

El protocolo funciona correctamente ya que

$$Z_1 = k_A K_B = k_A k_B P = k_B k_A P = k_B K_A = Z_2 .$$

2.12.2 Esquema de cifrado de ElGamal con ECC

De forma similar al protocolo de intercambio de claves de Diffie-Hellman, en este esquema [106] se precisa el grupo de puntos de una curva elíptica, $E(\mathbb{F}_q)$, con cardinal primo o casi primo, y un punto P, con $o(P) = p$ un primo grande. Ambos parámetros y la curva son públicos.

Alicia desea enviar un mensaje $W \in E(\mathbb{F}_q)$ de forma privada a Bernardo. Para ello:

1. Bernardo escoge aleatoriamente un entero k_B, con $1 \le k_B \le p-1$, que constituirá su clave privada, y calcula su correspondiente clave pública, $K_B = k_B P$, que enviará a Alicia.
2. Alicia toma entonces un entero k_A, con $1 \le k_A \le p-1$, como su clave secreta, y calcula la clave de cifrado $Z_1 = k_A K_B$, que es secreta.
3. Para cifrar el mensaje, Alicia calcula el punto $S = Z_1 + W$, y se lo transmite a Bernardo junto con su clave pública $K_A = k_A P$.

4. Bernardo obtiene la clave de descifrado haciendo $Z_2 = k_B K_A$.
5. Por último, Bernardo utiliza el opuesto de Z_2, para recuperar el mensaje: $W' = S - Z_2$.

Igual que en el caso anterior, se cumple que $Z_1 = Z_2$ y, por tanto,

$$W' = S - Z_2 = S - Z_1 = S - (S - W) = W .$$

2.12.3 Esquema de cifrado Massey-Omura con ECC

Este esquema de cifrado [106] utiliza una curva elíptica $E(\mathbb{F}_q)$ con las mismas características del esquema anterior. Tanto la curva como su orden, es información pública.

Alicia quiere enviar a Bernardo un mensaje de forma confidencial codificado como punto de la curva elíptica, $W \in E(\mathbb{F}_q)$. Este punto debe ser de orden p.

1. Alicia toma un entero aleatorio k_A y calcula el punto $k_A W$, que envía a Bernardo.
2. Éste, por su parte, toma un entero aleatorio k_B, calcula $k_B k_A W$ y le devuelve el resultado a Alicia.
3. Alicia entonces obtiene el inverso del entero que había tomado previamente, $k_A^{-1} \bmod p$, calcula el punto $k_A^{-1} k_B k_A W$ y se lo devuelve a Bernardo.
4. Por último, Bernardo calcula el inverso de su clave privada, $k_B^{-1} \bmod p$, y obtiene $k_B^{-1} k_A^{-1} k_B k_A W$.

Como se comprueba fácilmente, $k_B^{-1} k_A^{-1} k_B k_A W = W$, con lo que Bernardo recupera el mensaje enviado por Alicia sin que nadie haya podido descubrirlo.

2.12.4 Firma digital de ElGamal con ECC

Para este esquema de firma [106] precisamos, igual que antes, un punto P generador de un subgrupo grande de $E(\mathbb{F}_q)$ de orden primo p, ambos datos públicos.

Alicia desea enviar la firma de un mensaje w, codificado como número entero y que cumple $1 \le w \le p-1$ (si fuera mayor se podría usar una función *hash* para reducirlo).

Se requiere una función

$$f : E(\mathbb{F}_q) \to \mathbb{Z}_p$$

que asigne a cada punto de la curva elíptica un entero. Esta función no necesita tener características especiales, es suficiente con que el entero resultante de aplicar la función sea grande y que muy pocos puntos se correspondan con el mismo entero. Por ejemplo, en el caso de que la curva esté definida sobre $\mathbb{Z}_p$, es suficiente tomar la coordenada x de un punto como su imagen, $f(x,y)=x$. De esta forma tan sólo dos puntos como máximo (un punto y su opuesto) se corresponderán con la misma imagen. Por simplicidad supondremos que podemos tomar directamente la coordenada x.

Para generar la firma, se sigue el procedimiento:

1. Alicia escoge aleatoriamente una clave privada k_A, con $1 \le k_A \le p-1$, con su clave pública correspondiente $K_A = k_A P$.
2. Alicia, además, escoge un entero r, con $1 \le r \le p-1$, que cumpla la propiedad $\text{mcd}(r,p)=1$, con lo que garantizamos la existencia de $r^{-1} \bmod p$. Como p es primo, cualquier r es válido. Tras ello, calcula $R = rP$.
3. Después obtiene un entero s como:

$$s = r^{-1}(w - k_A f(R)) \bmod p.$$

La firma de Alicia para el mensaje w quedará determinada por el par (R,s). Para que Bernardo pueda verificar la firma, debe recibir de Alicia tanto su firma (R,s) como el mensaje w. El proceso de verificación es el siguiente:

1. Bernardo calcula $V_1 = f(R)K_A + sR$.
2. Además, calcula $V_2 = wP$.
3. Si $V_1 = V_2$ la firma queda verificada.

El esquema de firma funciona ya que:

$$V_1 = f(R)K_A + sR = f(R)k_A P + srP = f(R)k_A P + (w - k_A f(R))P = wP = V_2$$

2.12.5 Firma digital de Schnorr con ECC

Como se vio en el capítulo 1, este esquema es similar al de ElGamal, aunque con firmas algo más cortas [104].

Se precisa como elemento público un punto P generador de un subgrupo de $E(\mathbb{F}_q)$ de orden primo p. Además, se requieren dos funciones: una debe transformar un punto de la curva elíptica en un entero; puede servir la función expuesta en el esquema de firma de ElGamal, $f: E(\mathbb{F}_q) \to \mathbb{Z}_p$. La otra debe ser una función *hash* H que actúe sobre dos enteros.

Alicia desea firmar un mensaje w codificado como número entero, con $1 \le w \le p-1$. Para ello:

1. Alicia toma una clave privada $1 \le k_A \le p-1$ y calcula su clave pública, que consiste en $K_A = -k_A P$.
2. Después toma un entero $1 \le r \le p-1$, y obtiene el punto $R = rP$.
3. Alicia utiliza las funciones f y H para calcular h como
$$h = H\left(w, f(R)\right).$$
4. Por último, calcula
$$s = (r + k_A h) \bmod p.$$

Alicia debe transmitir a Bernardo el mensaje w y su firma sobre el mismo, que consiste en el par (h, s). Para comprobar la firma, Bernardo procede de la siguiente forma:

1. Calcula $R' = sP + hK_A$.
2. Tras ello, le aplica la función f y utiliza la función *hash* H sobre este resultado y el mensaje a firmar w:
$$h' = H\left(w, f(R')\right).$$
3. La firma queda verificada si $h' = h$.

El esquema es válido ya que

$$R' = sP + hK_A = (r + k_A h)P + h(-k_A P) = rP = R$$

y, por tanto, se cumple que

$$h' = H\left(w, f(R')\right) = H\left(w, f(R)\right) = h.$$

2.12.6 ECDSA

Como se vio en el capítulo 1, el ECDSA (*Elliptic Curve* DSA) [43] ha sido aceptado como estándar por varias organizaciones internacionales. Básicamente se trata de una variante del algoritmo de firma de ElGamal sobre curvas elípticas, pero con algunas modificaciones que corrigen determinadas deficiencias de éste.

Consideremos un punto $P \in E(\mathbb{F}_q)$ generador de un subgrupo grande con $o(P) = p$ primo, datos conocidos públicamente. Alicia desea firmar un mensaje w, codificado como entero.

1. Lo primero que hace Alicia es obtener su clave privada $1 \le k_A \le p-1$ y su correspondiente clave pública $K_A = k_A P$.
2. Después escoge un entero aleatorio r, con $1 \le r \le p-1$ y calcula $R = rP$.
3. Tras ello obtiene un entero s como

$$s = r^{-1}\left(H(w) + f(R)k_A\right) \bmod p,$$

donde f es una función que a un punto de la curva le hace corresponder su coordenada x, y H es una función *hash* (los estándares especifican la función SHA–1) que devuelve un entero $H(w) < p$.

Los parámetros de la firma de Alicia para el mensaje w, son los del par (R,s). Para verificar la firma, el proceso es el siguiente:

1. Bernardo utiliza el inverso de s para calcular $v_1 = s^{-1}H(w) \bmod p$ y, usando la función f, $v_2 = s^{-1}f(R) \bmod p$.
2. Después utiliza ambos enteros para obtener el punto

$$R' = v_1 P + v_2 K_A.$$

3. Bernardo da por válida la firma si $R' = R$.

La verificación es correcta ya que

$$R' = v_1 P + v_2 K_A = s^{-1}H(w)P + s^{-1}f(R)k_A P = s^{-1}\left(H(w) + f(R)k_A\right)P = rP = R.$$

Como se puede apreciar, la diferencia básica entre el ElGamal y el ECDSA consiste en la verificación de la firma. Si en ElGamal se debían realizar tres productos de un entero por un punto (múltiplos), en el ECDSA sólo son necesarios dos.

Capítulo 3

Sistema criptográfico lineal de curva elíptica basado en matrices

En este capítulo proponemos un nuevo problema matemático con las propiedades necesarias para definir criptosistemas de clave pública, es decir, proponemos una función de una vía cuya inversión es un problema computacionalmente intratable en un tiempo aceptable.

El problema se fundamenta en el ECDLP definido sobre matrices polinómicas. La diferencia básica es que si en el ECDLP se parte de un punto y se obtiene otro a partir de un escalar, en este sistema se parte de una n-tupla (columna) de puntos y se obtiene otra a partir de una matriz cuadrada de escalares. Para que los protocolos criptográficos como el de intercambio de claves funcionen, es preciso definir las matrices de escalares de forma que conmuten, por ello se hace de forma polinómica.

Con esta propuesta conseguimos incrementar la seguridad de un sistema aumentando el número de instancias necesarias para resolverlo. Para establecer el sistema necesitamos una curva elíptica, pero es posible incrementar la complejidad necesaria para romper el sistema de forma más rápida y eficiente que transmitiendo puntos de la curva de forma independiente.

Realizamos, además, un criptoanálisis del sistema detectando sus debilidades y verificando que, aún así, es más duro de resolver que el ECDLP.

3.1 Motivación: formas de definir problemas más seguros

Veamos las opciones más eficientes y viables si se desea aumentar la seguridad de un sistema criptográfico basado en curvas elípticas.

La primera y más obvia consiste en aumentar el tamaño de la curva elíptica que define el problema. Para ello se debe incrementar el tamaño del cuerpo subyacente. Sin embargo, esta tarea, como ya se ha comentado, requiere de un esfuerzo computacional muy elevado para calcular el número de puntos racionales del grupo de la curva, más aún cuando estamos aumentando el tamaño de la misma. La alternativa supone utilizar curvas con el orden del grupo ya calculado, como las curvas estándar propuestas por el NIST. El inconveniente es que éstas son muy pocas y además se conocen de antemano, y aunque hasta la fecha no se ha probado que esto suponga una ventaja para el criptoanalista, es posible que se descubran, si no debilidades, sí en cambio algunas formas de acelerar el ataque precisamente por el conocimiento anticipado que se tiene de las mismas.

Un método alternativo consiste en transmitir varios puntos de la misma curva de forma independiente. Sin embargo, como ya se vio, no es necesario el mismo esfuerzo para calcular una instancia del problema que para calcular varias. Esto es así porque se aprovecha la información obtenida de los puntos ya calculados para reducir la complejidad del siguiente. De esta forma, el incremento en el tamaño del problema sería del orden de la raíz cuadrada del número de puntos transmitidos [105].

Un ejemplo: supongamos un sistema con claves de 2^{200}; para romperlo serían necesarias alrededor de 2^{100} iteraciones usando un ataque de raíz cuadrada. Para aumentar el tamaño del problema a tan sólo 2^{110}, sería necesario transmitir aproximadamente medio millón de puntos.

Otra opción es recurrir a los sistemas habituales basados en el IFP y el DLP, donde definir un problema es una tarea casi inmediata. El inconveniente de estos sistemas es que utilizan claves ya de por sí muy grandes, y para pensar en aumentarlas se debe considerar la cantidad de recursos, sobre todo de tiempo, que se van a necesitar para procesar las claves. Debemos recordar, además, que existen algoritmos subexponenciales que resuelven estos problemas y que el incremento del tamaño de las claves es más rápido que el incremento en seguridad.

Otros esfuerzos que se han realizado para incrementar la seguridad, consisten en definir sistemas basados en el DLP con matrices. Desafortunadamente, como se vio en la sección 1.4, no han conducido a sistemas útiles en la práctica.

Como hemos visto, todas estas opciones plantean serios problemas para ser efectivas en la práctica, o bien no obtienen mejoras significativas. Por ello, el objetivo del problema matemático que presentamos en este capítulo es lograr un sistema criptográfico más seguro que el ECDLP, evitando las dificultades comentadas.

3.2 Descripción del sistema

Consideremos una curva elíptica $E/\mathbb{F}_q$, con $p=\#E(\mathbb{F}_q)$ un número primo. Sea n un entero positivo, consideramos la aplicación

$$\begin{aligned} \mathcal{M}at_n(\mathbb{F}_p)\times E(\mathbb{F}_q)^n &\to E(\mathbb{F}_q)^n \\ (A,\Pi) &\mapsto A\Pi \end{aligned}$$

donde $\mathcal{M}at_n(\mathbb{F}_p)$ es el conjunto de todas las matrices cuadradas de tamaño $n\times n$ con elementos en $\mathbb{F}_p$ y $E(\mathbb{F}_q)^n$ es el conjunto de n-tuplas (columnas) de puntos de $E(\mathbb{F}_q)$.

Si $A=[a_{i,j}]$ para $0\le i,j\le n-1$ y $\Pi=\begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_{n-1} \end{bmatrix}$, entonces $A\Pi=\begin{bmatrix} Q_0 \\ Q_1 \\ \vdots \\ Q_{n-1} \end{bmatrix}$, donde

$$Q_i=\sum_{j=0}^{n-1} a_{i,j}P_j\,.$$

Suponiendo que Π y $\Phi=A\Pi$ son conocidos, el problema consiste en encontrar la matriz A.

Como vemos, la exposición es similar al ECDLP, pero usando n-tuplas de puntos en lugar de puntos, y matrices cuadradas en lugar de escalares. El objetivo es definir un problema mayor, es decir, con claves más grandes, pero con el mismo tamaño de la curva elíptica con la que se realizan las operaciones.

El conjunto $\mathcal{M}at_n(\mathbb{F}_p)$ no excluye a las matrices no invertibles, ya que para nuestros propósitos no necesitamos invertir ninguna matriz como parte del problema a resolver. En cambio, sí es importante comprobar que los elementos utilizados como claves privadas en los protocolos criptográficos conmutan respecto a la operación que se les aplica. En este caso, como la multiplicación de matrices no es, en general, conmutativa, debemos restringirnos al conjunto

$$\mathbb{F}_p[M] = \{ f(M) \mid f(t) \in \mathbb{F}_p[t] \} \subseteq \mathcal{M}at_n(\mathbb{F}_p)$$

donde $\mathbb{F}_p[t]$ es el conjunto de polinomios en la variable t y coeficientes en el cuerpo $\mathbb{F}_p$, y $M \in \mathcal{M}at_n(\mathbb{F}_p)$ es una matriz cualquiera. De esta forma, si $A = f(M)$ y $B = g(M)$, como $f(t) \cdot g(t) = g(t) \cdot f(t)$, tenemos que $AB = BA$.

Ahora, si

$$A = a_{d-1}M^{d-1} + a_{d-2}M^{d-2} + \cdots + a_1 M + a_0 I$$

con $a_j \in \mathbb{F}_p$ para $0 \le j \le d-1$, considerando que M, Π y $\Phi = A\Pi$ son conocidos, el problema consiste en encontrar los escalares a_j para $0 \le j \le d-1$.

3.3 Órdenes y valores posibles

Las matrices creadas a partir de los escalares a_i, utilizadas como secreto a resolver, poseen n^2 elementos de $\mathbb{F}_p$, por tanto se podrían formar p^{n^2} matrices. Sin embargo, puesto que las matrices que nos interesan están en el conjunto $\mathbb{F}_p[M]$, por el teorema de Cayley-Hamilton sabemos que el grado máximo de éstos es $n-1$. Por tanto, si como máximo podemos tomar n escalares pertenecientes a $\mathbb{F}_p$, entonces tendremos p^n matrices posibles. Este valor debe coincidir con el número de posibles variaciones que se pueden formar con los elementos a_i, por tanto $d = n$, y para calcular la clave privada, éste es el valor máximo que se puede alcanzar. Por ello:

$$A = a_{n-1}M^{n-1} + a_{n-2}M^{n-2} + \cdots + a_1 M + a_0 I .$$

Como hemos visto, el valor p^n que tomamos para el problema a resolver excede ampliamente el valor máximo que podríamos tomar para un ECDLP definido sobre la misma curva elíptica, que sería de p.

Para la n-tupla Π, usada como generador y parte conocida del problema, obviamente podemos tomar cualquier variación de puntos de $E(\mathbb{F}_q)$. Como el número de puntos de $E(\mathbb{F}_q)$ es p, tenemos p posibles puntos para cada entrada en la n-tupla, por tanto tenemos p^n posibles n-tuplas. De la misma forma, las posibles n-tuplas Φ obtenidas de la operación $A\Pi$ serán p^n porque tienen el mismo tamaño que Π.

Debemos comprobar ahora que podemos obtener todas o casi todas las p^n n-tuplas posibles Φ a partir de cualquier n-tupla de puntos Π y las p^n matrices posibles A; para ello damos el siguiente teorema.

Teorema 3.1: *Consideremos las matrices M, Π y Φ definidas de la forma vista. Para cada n-tupla de puntos Φ existen n escalares a_i tales que*

$$(a_{n-1}M^{n-1} + a_{n-2}M^{n-2} + \cdots + a_1M + a_0I)\Pi = \Phi \tag{3.1}$$

con probabilidad

$$\frac{\prod_{i=0}^{n-1}(p^n - p^i)}{p^{n^2}} \approx 1.$$

Demostración: Los elementos de la matriz Π son puntos de $E(\mathbb{F}_q)$. Debemos tener en cuenta que p es un número primo y por tanto todo punto de la curva elíptica $E/\mathbb{F}_q$ se comporta como generador del grupo de puntos racionales $E(\mathbb{F}_q)$. Por otra parte, como el número p de puntos racionales de $E(\mathbb{F}_q)$ es primo, $E(\mathbb{F}_q) \cong \mathbb{Z}_p$, esto es, existe un isomorfismo entre el grupo de puntos de la curva y el grupo aditivo de los enteros módulo p. Este isomorfismo es inviable de calcular para los tamaños de p usuales, pero sabemos que existe y podemos aprovecharlo para extraer algunas propiedades.

Podemos ver por tanto (3.1) como un sistema de ecuaciones lineales con coeficientes en $\mathbb{Z}_p$, e incógnitas a_i para $0 \le i \le n-1$. Consideremos que $\tilde{\Pi}$ y $\tilde{\Phi}$ representan las imágenes mediante el isomorfismo citado de Π y Φ en $\mathbb{Z}_p$ respectivamente; el sistema (3.1) tendrá una única solución si la matriz de coeficientes del sistema

$$(a_{n-1}M^{n-1} + a_{n-2}M^{n-2} + \cdots + a_1M + a_0I)\tilde{\Pi} = \tilde{\Phi}$$

es invertible. Esta matriz de coeficientes es una matriz de tamaño $n \times n$ con elementos en $\mathbb{Z}_p$, por tanto debemos determinar cuándo estas matrices son invertibles.

Por una parte, el orden del grupo lineal general $GL(n, \mathbb{Z}_p)$ es $\prod_{i=0}^{n-1}(p^n - p^i)$; por otra parte, tenemos que el número total de matrices posibles de tamaño $n \times n$ y coeficientes en $\mathbb{Z}_p$ es p^{n^2}; en consecuencia, la probabilidad de que una matriz de tamaño $n \times n$ y coeficientes en $\mathbb{Z}_p$ sea invertible es

$$\frac{\prod_{i=0}^{n-1}(p^n - p^i)}{p^{n^2}}.$$

No es difícil comprobar que esta probabilidad es aproximadamente igual a 1. ■

3.4 Criptoanálisis del sistema

Procedemos ahora a realizar un criptoanálisis del sistema, es decir, a intentar resolverlo o a reducir su dureza encontrando un método alternativo a la búsqueda exhaustiva.

Hemos hallado una forma de reducir el sistema a varias instancias de un ECDLP. Aún así, continúa siendo un problema más duro de resolver que el ECDLP. Veamos el caso $n=2$.

Consideremos la clave privada $A=a_1M+a_0I$ formada mediante la matriz pública $M\in\mathcal{M}at_2(\mathbb{F}_p)$ y los escalares secretos $a_0,a_1\in\mathbb{F}_p$, con p primo. Se toma una 2-tupla de puntos

$$\Pi=\begin{bmatrix}P_0\\P_1\end{bmatrix}$$

públicamente y se procede a calcular el producto $A\Pi=\Phi$, donde la 2-tupla

$$\Phi=\begin{bmatrix}Q_0\\Q_1\end{bmatrix}$$

es la clave pública, teniendo que $P_i,Q_i\in E(\mathbb{F}_q)$. Procedemos a determinar A.

Dado que p es primo, todo punto de la curva elíptica es generador de la misma, por tanto podemos encontrar un escalar α tal que $\alpha P_0=P_1$. Esto supone resolver una instancia del ECDLP. Tenemos que

$$A=a_1M+a_0I=\begin{bmatrix}a_1m_{11}+a_0 & a_1m_{12}\\ a_1m_{21} & a_1m_{22}+a_0\end{bmatrix}$$

y por tanto

$$\begin{aligned}A\Pi&=\begin{bmatrix}a_1m_{11}+a_0 & a_1m_{12}\\ a_1m_{21} & a_1m_{22}+a_0\end{bmatrix}\begin{bmatrix}P_0\\ \alpha P_0\end{bmatrix}\\&=\begin{bmatrix}(a_1m_{11}+a_0+a_1m_{12}\alpha)P_0\\ (a_1m_{21}+a_1m_{22}\alpha+a_0\alpha)P_0\end{bmatrix}\\&=\begin{bmatrix}Q_0\\Q_1\end{bmatrix}.\end{aligned}\qquad(3.2)$$

Ahora se determinan los escalares β y γ tales que $\beta P_0=Q_0$ y $\gamma P_0=Q_1$, es decir, dos instancias más del ECDLP. Si sustituimos estos cambios en (3.2), obtendremos el sistema de ecuaciones lineales:

$$\begin{cases} a_1 m_{11} + a_0 + a_1 m_{12}\alpha = \beta \\ a_1 m_{21} + a_1 m_{22}\alpha + a_0\alpha = \gamma \end{cases}$$

del que podemos obtener a_0 y a_1 ya que el resto de elementos son conocidos.

La extensión a un sistema de tamaño $n > 2$ es casi inmediata. Primero tendríamos que resolver $n-1$ instancias para poner todos los puntos P_i en función de uno solo; segundo, plantear el sistema de ecuaciones; tercero, resolver n instancias para poner todos los puntos Q_i en función del único punto anterior; y por último, habría que resolver un sistema de n ecuaciones lineales con n incógnitas.

En conclusión, para romper un sistema en el que intervienen 2 n-tuplas y una matriz de tamaño $n \times n$ en el que, por tanto, se transmiten n puntos, tendremos que resolver $2n-1$ instancias de la curva elíptica cuando, en esta última, transmitiendo los puntos de forma independiente, haría falta resolver n instancias.

Veamos un ejemplo usando la información almacenada por el algoritmo Pollard-ρ para romper el sistema. Supongamos que L es el número de instancias a resolver y T el tamaño del problema. El coste computacional requerido R_L, como puede verse en [52], es aproximadamente $R_L = \sqrt{2LT}$. Consideremos un ECDLP con tamaños de clave de 200 bits, esto es, $T = 2^{200}$, para encontrar las claves privadas correspondientes a 7 instancias del mismo problema, sería necesario calcular:

$$R_7 = \sqrt{2 \cdot 7 \cdot 2^{200}} \approx 2^{101,9}$$

iteraciones. Si aplicamos el sistema descrito, sería necesario calcular

$$R_{13} = \sqrt{2 \cdot 13 \cdot 2^{200}} \approx 2^{102,35}$$

iteraciones. Esto representa un incremento de $2^{100,44}$ en el número de iteraciones requerido para encontrar las claves secretas.

3.5 Sistemas criptográficos

Es conveniente comprobar que el sistema se puede aplicar a la criptografía de clave pública, ya que poseer buenas propiedades como problema matemático no significa que funcione correctamente al usarlo como problema base de un protocolo criptográfico. Por

ello, procedemos a rediseñar los principales esquemas criptográficos basados en curvas elípticas, de intercambio de clave, de cifrado y de firma digital, con el sistema presentado.

3.5.1 Protocolo de intercambio de claves de Diffie-Hellman

Alicia y Bernardo desean compartir una clave para su posterior utilización. Consideremos la matriz M, necesaria para generar las claves privadas, y la n-tupla Π, necesaria para las claves públicas, definidas según se ha visto. Ambas matrices son conocidas públicamente.

1. Alicia toma n escalares $a_i \in \mathbb{F}_p$ y calcula su clave secreta

$$A = a_{n-1}M^{n-1} + a_{n-2}M^{n-2} + \cdots + a_1 M + a_0 I .$$

   Seguidamente calcula el producto $A\Pi = \Phi_A$ y transmite a Bernardo su clave pública, la n-tupla Φ_A, formada también por puntos de la curva elíptica $E/\mathbb{F}_q$.

2. Bernardo, asimismo, toma n escalares $b_i \in \mathbb{F}_p$ y forma su clave secreta

$$B = b_{n-1}M^{n-1} + b_{n-2}M^{n-2} + \cdots + b_1 M + b_0 I .$$

   Tras ello, calcula el producto $B\Pi = \Phi_B$ y envía a Alicia la n-tupla Φ_B, formada consecuentemente por n puntos de la curva.

3. Alicia, entonces, calcula $\Gamma_1 = A\Phi_B$.

4. Bernardo calcula de forma similar $\Gamma_2 = B\Phi_A$.

Es necesario que las matrices A y B conmuten respecto a la multiplicación para que las dos n-tuplas resultantes, Γ_1 y Γ_2, sean idénticas. Esto podemos asegurarlo por la forma polinómica de construir las matrices A y B. Por tanto tenemos que

$$\Gamma_1 = A\Phi_B = AB\Pi = BA\Pi = B\Phi_A = \Gamma_2$$

es el secreto compartido.

3.5.2 Esquema de cifrado de ElGamal

Consideremos la n-tupla Π formada por puntos de la curva elíptica $E/\mathbb{F}_q$, y la matriz $M \in \mathcal{M}at_n(\mathbb{F}_p)$ usada como base para generar las claves privadas. Todos estos parámetros, necesarios para establecer el sistema, se hacen públicos.

Alicia quiere enviar un mensaje de forma privada a Bernardo. Lo primero que debe hacer es codificar el mensaje en la forma de los elementos que usamos, es decir como n-tupla de puntos. Para ello se puede, por ejemplo, dividir el mensaje en n partes y utilizar la técnica de Koblitz vista en la sección 2.11. En cualquier caso, denotamos mediante Ψ el mensaje a cifrar codificado en forma de n-tupla de puntos.

1. Bernardo toma n escalares $b_i \in \mathbb{F}_p$ y forma su clave secreta

$$B = b_{n-1}M^{n-1} + b_{n-2}M^{n-2} + \cdots + b_1 M + b_0 I .$$

Después, calcula el producto $B\Pi = \Phi_B$ y envía a Alicia su clave pública, que será la n-tupla Φ_B.

2. Alicia toma n escalares $a_i \in \mathbb{F}_p$ para calcular su clave secreta

$$A = a_{n-1}M^{n-1} + a_{n-2}M^{n-2} + \cdots + a_1 M + a_0 I ,$$

y calcula el producto $A\Pi = \Phi_A$, tomando el resultado como su clave pública. Después, obtiene la clave de cifrado $\Gamma_1 = A\Phi_B$, que mantiene en secreto.

3. Alicia cifra entonces el mensaje haciendo $\Omega = \Gamma_1 + \Psi$. Por último, envía a Bernardo el par de n-tuplas de puntos (Φ_A, Ω).

4. Bernardo usa la clave pública de Alicia para obtener la clave de descifrado $\Gamma_2 = B\Phi_A$, y utiliza el opuesto de Γ_2, que consiste en la n-tupla formada por los opuestos de cada punto de la n-tupla Γ_2, para recuperar el mensaje: $\Psi' = \Omega - \Gamma_2$.

Se comprueba que

$$\Gamma_1 = A\Phi_B = AB\Pi = BA\Pi = B\Phi_A = \Gamma_2$$

y por tanto

$$\Psi' = \Omega - \Gamma_2 = \Omega - \Gamma_1 = \Omega - (\Omega - \Psi) = \Psi .$$

3.5.3 Esquema de cifrado Massey-Omura

Consideremos la curva elíptica $E/\mathbb{F}_q$, con $\#E(\mathbb{F}_q) = p$ primo, y la matriz $M \in \mathcal{M}at_n(\mathbb{F}_p)$, usada para generar las claves privadas. Esta información es pública.

Alicia quiere enviar a Bernardo un mensaje de forma confidencial codificado como n-tupla Ψ de puntos de la curva elíptica.

1. Alicia toma de forma aleatoria n escalares $a_i \in \mathbb{F}_p$ para calcular su clave privada A de la forma antes vista, pero además debe comprobar que A es invertible –para

valores prácticos en criptografía, esta probabilidad es casi 1 (véase el teorema 3.1)–. Tras ello calcula el producto $A\Psi$ y le envía la n-tupla resultante a Bernardo.

2. Éste, por su parte, toma de forma aleatoria n escalares $b_i \in \mathbb{F}_p$ y calcula su clave secreta B de la forma anterior, comprobando que es invertible. Después calcula $BA\Psi$ y le devuelve el resultado a Alicia.
3. Alicia debe calcular el inverso de la matriz obtenida previamente, A^{-1}, y multiplicarlo por la n-tupla recibida de Bernardo: $A^{-1}BA\Psi$. Por último, le envía el resultado a Bernardo.
4. Éste, finalmente, calcula el inverso de su clave, B^{-1}, y calcula la expresión $B^{-1}A^{-1}BA\Psi$.

Como las matrices A y B conmutan, Bernardo recupera el mensaje ya que

$$B^{-1}A^{-1}BA\Psi = B^{-1}A^{-1}AB\Psi = \Psi .$$

3.5.4 Firma digital de ElGamal

Para establecer el esquema de firma de ElGamal con el nuevo sistema como problema matemático subyacente, se precisan los parámetros $E(\mathbb{F}_q)$, p, Π, M, n, de la forma anterior, todos públicos.

Alicia desea enviar la firma de un mensaje W, que en este caso debe estar codificado como una matriz de tamaño $n \times n$ cuyos elementos w_{ij} son enteros que cumplen $0 \le w_{ij} < p$, es decir, son elementos de $\mathbb{F}_p$, de forma que el tamaño total del mensaje es menor que p^{n^2}. En el caso de que fuera mayor se debería usar una función *hash*.

En este caso necesitamos definir una función

$$f : E(\mathbb{F}_q)^n \to \mathcal{M}at_n(\mathbb{Z}_p)$$

que a una n-tupla de puntos de la curva elíptica le haga corresponder una matriz cuadrada de tamaño $n \times n$ con elementos en $\mathbb{Z}_p$. Es suficiente que los enteros de la matriz resultante sean grandes y que pocas n-tuplas se correspondan con la misma matriz de enteros. Para crearla de forma sencilla, definimos una función subyacente que transforme un punto en un escalar:

$$g : E(\mathbb{F}_q) \to \mathbb{Z}_p$$

Esta función g se define de la siguiente forma: si $T = (x, y)$ es un punto de $E(\mathbb{F}_q)$, entonces

$$g(T) = x' \bmod p,$$

donde x' es la representación de x en $\mathbb{Z}_q$. La obtención de esta representación es trivial. Consideremos la matriz M y la n-tupla $\Pi = [P_i]$, con $0 \le i \le n-1$; la función f se define entonces como

$$f(\Pi) = g(P_{n-1})M^{n-1} + g(P_{n-2})M^{n-2} + \cdots + g(P_1)M + g(P_0)I.$$

Pasos a seguir para la firma:

1. Alicia escoge de forma aleatoria n escalares y crea su clave privada A y su clave pública Φ_A de la forma antes vista.
2. Asimismo, toma n escalares $r_i \in \mathbb{F}_p$ y crea una segunda clave R de la misma forma que A, pero en este caso debe asegurarse de que R sea invertible. Después calcula el producto $\Phi_R = R\Pi$.
3. Por último, obtiene una matriz S de tamaño $n \times n$ como

$$S = (W - Af(\Phi_R))R^{-1} \bmod p.$$

La firma de Alicia para el mensaje W viene dada por el par (Φ_R, S).

Pasos a seguir para la verificación:

1. Bernardo, por una parte, calcula la expresión $\Delta_1 = f(\Phi_R)\Phi_A + S\Phi_R$.
2. Por otra, calcula $\Delta_2 = W\Pi$.
3. Finalmente, verifica la firma si $\Delta_1 = \Delta_2$.

Para comprobar que este esquema de firma funciona, basta ver que:

$$\Delta_1 = f(\Phi_R)\Phi_A + S\Phi_R = f(\Phi_R)A\Pi + SR\Pi = f(\Phi_R)A\Pi + (W - Af(\Phi_R))\Pi = W\Pi = \Delta_2,$$

ya que $f(\Phi_R)$ y A conmutan por la forma en que están definidas.

3.5.5 Firma digital de Schnorr

El esquema de firma digital de Schnorr se establece con los elementos $E(\mathbb{F}_q)$, p, Π, M, n, vistos en los anteriores sistemas. Esta información es conocida públicamente.

Alicia desea enviar la firma de un mensaje W a Bernardo. El mensaje en este caso podría venir codificado como entero sin necesidad de transformar en una matriz porque se le va a aplicar una función *hash* directamente. Sin embargo, la salida de la función *hash* debe ser una matriz de enteros, por lo que se consigue mayor homogeneidad si W se

encuentra en la forma de matriz de tamaño $n \times n$, con elementos enteros no necesariamente menores que p.

Por otra parte, se necesita una función que transforme una n-tupla de puntos en una matriz de enteros. Sirva la función descrita en el esquema de firma de ElGamal:

$$f : E(\mathbb{F}_q)^n \to \mathcal{M}at_n(\mathbb{Z}_p).$$

La función *hash* que se precisa, debe ser de la forma:

$$\chi : \mathcal{M}at_n(\mathbb{Z}_p) \times \mathcal{M}at_n(\mathbb{Z}_p) \to \mathcal{M}at_n(\mathbb{Z}_p).$$

Partiendo de las funciones *hash* usuales, no es difícil definir una función *hash* χ, bien considerando los elementos de las matrices de forma independiente, bien realizando alguna operación sobre ellos.

Para realizar la firma:

1. Alicia escoge de forma aleatoria n escalares y crea su clave privada A de la forma usual. Para obtener su clave pública, usa el opuesto de A para obtener:
$$\Phi_{-A} = -A\Pi$$
2. Después toma otros n escalares, crea una segunda clave privada R de la misma forma que A, y calcula $\Phi_R = R\Pi$.
3. Tras ello aplica la función f y la función *hash* χ para obtener
$$H = \chi\left(W, f(\Phi_R)\right).$$
4. El último paso consiste en obtener una matriz S de tamaño $n \times n$ como
$$S = (R + HA) \bmod p.$$

La firma de Alicia sobre el mensaje W consiste en el par de matrices (H, S).

Bernardo, para comprobar la firma, debe seguir los pasos siguientes:

1. Calcular la expresión $\overline{\Phi}_R = S\Pi + H\Phi_{-A}$.
2. Aplicar la función f sobre $\overline{\Phi}_R$ y la función *hash* sobre el resultado de la función y el mensaje, obteniendo:
$$H' = \chi\left(W, f(\overline{\Phi}_R)\right).$$
3. Bernardo verifica la firma si $H' = H$.

La validez del esquema se argumenta viendo que

$$\overline{\Phi}_R = S\Pi + H\Phi_{-A} = (R + HA)\Pi + H(-A\Pi) = (R + HA - HA)\Pi = R\Pi = \Phi_R$$

y por tanto

$$H' = \chi\left(W, f(\bar{\Phi}_R)\right) = \chi\left(W, f(\Phi_R)\right) = H .$$

La ventaja de este esquema de firma consiste en el ahorro que se obtiene al no tener que invertir ninguna matriz, operación muy costosa computacionalmente.

3.5.6 ECDSA

Para describir el esquema de firma ECDSA adaptado al sistema propuesto, se definen de forma similar a la del esquema de ElGamal los parámetros $E(\mathbb{F}_q)$, p, Π, M, n, y se hacen públicos.

En el esquema ECDSA, los estándares imponen el uso de la función *hash* SHA-1 antes de aplicar la firma. En cualquier caso, el mensaje o el resultado de la función *hash* debe codificarse como una matriz W de tamaño $n \times n$, siendo sus elementos w_{ij} enteros, con $0 \le w_{ij} < p$. Como puede verse, el tamaño total disponible para el mensaje es de p^{n^2}.

Se define una función f de la misma forma que se hizo en el esquema de firma de ElGamal, de manera que a una n-tupla de puntos de la curva elíptica le hace corresponder una matriz cuadrada de tamaño $n \times n$ con elementos en $\mathbb{Z}_p$.

Para realizar la firma del mensaje:

1. Alicia elige de forma aleatoria n escalares y crea su clave privada A y su clave pública Φ_A de la forma usual.
2. Después escoge n escalares r_i, con $0 \le r_i < p$, y crea una segunda clave R, de la misma forma que A, para calcular $\Phi_R = R\Pi$. La matriz R debe ser invertible.
3. Por último, debe obtener la matriz S de tamaño $n \times n$ con elementos en $\mathbb{Z}_p$ dada por

$$S = \left(W + f(\Phi_R)A\right)R^{-1} \bmod p . \tag{3.3}$$

asegurándose de que también sea invertible.

La firma de Alicia para el mensaje W consiste en la n-tupla de puntos Φ_R y la matriz S.

Para comprobar la firma:

1. Bernardo debe obtener S^{-1} y calcular las expresiones $V_1 = S^{-1}W \bmod p$ y $V_2 = S^{-1} f(\Phi_R) \bmod p$.
2. Tras ello, hace $\bar{\Phi}_R = V_1\Pi + V_2\Phi_A$.

3. Bernardo verifica la firma si $\bar{\Phi}_R = \Phi_R$.

Se comprueba que el esquema de firma presentado funciona ya que de la expresión (3.3) tenemos que

$$R = S^{-1}(W + f(\Phi_R)A) \bmod p,$$

y, por tanto,

$$\bar{\Phi}_R = V_1\Pi + V_2\Phi_A = S^{-1}W\Pi + S^{-1}f(\Phi_R)A\Pi = S^{-1}(W + f(\Phi_R)A)\Pi = R\Pi = \Phi_R.$$

3.6 Conclusiones

Basado en curvas elípticas, este sistema incrementa la dureza del ECDLP, uno de los problemas matemáticos punteros en criptografía de clave pública por su complejidad inherente y seguridad. Para ello, consigue aumentar el número de instancias necesarias para resolverlo, de forma que el coste necesario para romperlo por fuerza bruta –mediante algoritmos de raíz cuadrada– es superior al coste requerido para romper las instancias correspondientes del ECDLP.

Tras analizar las características criptográficas del sistema, hemos hallado que es posible atacar la curva elíptica subyacente reduciendo el problema mediante ecuaciones lineales. Aún así, como el propio análisis muestra, hemos logrado una mejora sobre el ECDLP.

Las características del problema matemático definido permiten que sea aplicable a la criptografía de clave pública. Para ello es necesario que las matrices usadas cumplan algunas propiedades y, en algunos casos, definir funciones adicionales. Como muestra de la utilidad del sistema, se ha aplicado a esquemas tanto de intercambio de claves, como de cifrado y de firma digital.

En los esquemas de firma de Schnorr y ECDSA, de forma particular, no necesitamos que las matrices estén construidas de forma polinómica, ya que no se requiere que conmuten. Por ello, en estos casos podemos tomar cualquier matriz perteneciente a $\mathcal{M}at_n(\mathbb{F}_p)$ ampliando en consecuencia el espacio de claves posible.

Capítulo 4

Sistema criptográfico no lineal de curva elíptica y matrices formales

Proponemos ahora un nuevo sistema basado en curvas elípticas y matrices formales como problema matemático para su uso en criptografía de clave pública. El sistema se fundamenta en una función unidireccional con trampa cuya inversión tiene una complejidad computacional muy elevada, por ello se considera intratable.

La finalidad de esta propuesta es similar a la de la presentada en el capítulo anterior: lograr un sistema criptográfico más seguro que el ECDLP, de forma que se sorteen los inconvenientes y dificultades que plantea la definición de sistemas criptográficos de clave pública más seguros. La dureza o complejidad del sistema es la implícita en el ECDLP, es decir, la más alta posible, ya que sólo puede atacarse mediante algoritmos de raíz cuadrada, que siempre son aplicables a un grupo. Esto es así porque el sistema utiliza curvas elípticas en última instancia. En cualquier caso, aunque no aumenta la complejidad inherente a la resolución de la curva elíptica en la que se basa, sí requiere un esfuerzo computacional mayor en cada iteración. El cómputo que se precisa en el proceso de cifrado no debe incrementarse de forma apreciable usando algoritmos de exponenciación rápida, pero el tiempo necesario para el descifrado sí debe aumentar en cierta medida.

El sistema es una variante del Problema del Logaritmo Discreto (DLP) con los elementos de determinado grupo formado por matrices formales compuestas por puntos de una curva elíptica y elementos de cierto cuerpo finito relacionado con la curva. El término

no lineal hace referencia al coeficiente que se usa como problema a resolver ya que se obtiene como una combinación no lineal de dos escalares escogidos de forma aleatoria.

Además, exponemos el protocolo de intercambio de claves de Diffie-Hellman con este sistema funcionando como problema matemático subyacente y un criptoanálisis de este protocolo revelando las deficiencias del problema.

4.1 Descripción del sistema

Dada una curva elíptica $E/\mathbb{F}_q$ con $p=\#E(\mathbb{F}_q)$ un número primo. Consideramos el conjunto

$$\xi=\left\{\begin{pmatrix} a & P \\ & b \end{pmatrix} \middle| \; a,b\in\mathbb{F}_p, \text{ y } P\in E(\mathbb{F}_q)\right\},$$

y se define una operación binaria como la multiplicación formal de matrices de la forma:

$$\begin{pmatrix} a & P \\ & b \end{pmatrix}\begin{pmatrix} c & Q \\ & d \end{pmatrix}=\begin{pmatrix} ac & aQ+Pd \\ & bd \end{pmatrix} \tag{4.1}$$

El siguiente resultado nos da alguna de sus propiedades.

Teorema 4.1: *La multiplicación formal de matrices definida por* (4.1) *es asociativa y posee elemento neutro.*

Demostración: Es fácil ver que

$$(M_1M_2)M_3=M_1(M_2M_3),$$

para todo $M_1,M_2,M_3\in\xi$, y

$$MI=M=IM,$$

para todo $M\in\xi$, donde

$$I=\begin{pmatrix} 1 & O \\ & 1 \end{pmatrix}\in\xi. \qquad \blacksquare$$

Notar que, por ejemplo,

$$\begin{pmatrix} 0 & P \\ & b \end{pmatrix}\begin{pmatrix} c & Q \\ & d \end{pmatrix}\neq\begin{pmatrix} 1 & O \\ & 1 \end{pmatrix}$$

para todo $b,c,d \in \mathbb{F}_p$ y $P,Q \in E(\mathbb{F}_q)$, por tanto existen elementos no invertibles en ξ.

Para conseguir que todos los elementos tengan inverso, debemos considerar sólo aquellos que contienen elementos a y b invertibles. Por ello se define el subconjunto $\zeta \subset \xi$ como

$$\zeta = \left\{ \begin{pmatrix} a & P \\ & b \end{pmatrix} \in \xi \mid a,b \neq 0 \right\}.$$

Es fácil ver que si

$$M = \begin{pmatrix} a & P \\ & b \end{pmatrix} \in \zeta,$$

entonces

$$M^{-1} = \begin{pmatrix} a^{-1} & -b^{-1}a^{-1}P \\ & b^{-1} \end{pmatrix} \in \zeta.$$

Como consecuencia de estas propiedades, el conjunto ζ con la operación definida es un grupo, que no es abeliano como se puede comprobar fácilmente.

Hemos creado un grupo usando la multiplicación formal de matrices en este conjunto sin necesidad de ningún tipo de artificio. Esto es posible porque el elemento que no aparece en la matriz impide que se multipliquen puntos de la curva elíptica, operación no definida.

Una vez determinado el grupo, se describe el sistema.

Dado el subgrupo de orden $p-1$ generado por una matriz conocida

$$M = \begin{pmatrix} a & P \\ & b \end{pmatrix} \in \zeta;$$

el sistema consiste en obtener el escalar $k \geq 0$ a partir únicamente de $P^{(k)}$, término que se obtiene al desarrollar la potencia *k*-ésima de *M*:

$$M^k = \begin{pmatrix} a & P \\ & b \end{pmatrix}^k = \begin{pmatrix} a^k & P^{(k)} \\ & b^k \end{pmatrix},$$

donde

$$P^{(k)} = \left(\sum_{i=0}^{k-1} a^{k-1-i} b^i \right) P = c^{(k)} P \tag{4.2}$$

para $k \geq 1$, y donde $P^{(0)} = O$, esto es, $c^{(0)} = 0$. Haremos referencia de forma general al elemento $c^{(k)}$ como el coeficiente.

Posteriormente veremos cómo escoger los elementos a y b de forma que M genere un subgrupo de orden máximo, es decir, de $p-1$ elementos.

Los elementos a^k y b^k se excluyen del problema ya que, de lo contrario, obtener k se reduciría directamente al DLP. Sin embargo, existe un problema intrínseco a esta exclusión, ya que si bien M puede ser un generador de un subgrupo de orden $p-1$ de ζ, y por tanto único para $k \in [0, p-2]$; en cambio $P^{(k)}$ no es único para estos valores de k, a pesar de que existen exactamente $p-1$ puntos diferentes en la curva, y por ser de tamaño primo todos actúan como generadores. Posteriormente veremos como paliar este problema.

Para analizar el sistema adecuadamente, debemos realizar un estudio del elemento $P^{(k)}$ y, consecuentemente, del coeficiente $c^{(k)}$.

4.2 Reducción de los elementos

Teorema 4.2: *Sean* $a,b \in \mathbb{F}_p^*$ *y* $P \in E(\mathbb{F}_q)$. *Si* $0 \le t \le k$ *entonces*

$$P^{(k)} = a^t P^{(k-t)} + P^{(t)} b^{k-t}.$$

Demostración: Consideremos el elemento $M = \begin{pmatrix} a & P \\ & b \end{pmatrix} \in \zeta$. Claramente tenemos que

$$M^t M^{k-t} = M^k,$$

es decir,

$$\begin{pmatrix} a^t & P^{(t)} \\ & b^t \end{pmatrix} \begin{pmatrix} a^{k-t} & P^{(k-t)} \\ & b^{k-t} \end{pmatrix} = \begin{pmatrix} a^k & P^{(k)} \\ & b^k \end{pmatrix},$$

y, por tanto,

$$a^t P^{(k-t)} + P^{(t)} b^{k-t} = P^{(k)}. \qquad \blacksquare$$

Para reducir el sumatorio que define el coeficiente $c^{(k)}$, veamos el siguiente teorema.

Teorema 4.3: *Sean* $a,b \in \mathbb{F}_p^*$, *con* $a \neq b$, *si* $k \ge 0$, *entonces*

$$c^{(k)} = \sum_{i=0}^{k-1} a^{k-1-i} b^i = (a^k - b^k)(a-b)^{-1}. \tag{4.3}$$

Demostración: Multiplicando la expresión (4.3) por $(a-b)$ tenemos que

$$(a-b)\sum_{i=0}^{k-1} a^{k-1-i}b^i = (a-b)(a^{k-1}+a^{k-2}b+\cdots+ab^{k-2}+b^{k-1})$$
$$= a^k + a^{k-1}b+\cdots+a^2b^{k-2}+ab^{k-1}-a^{k-1}b-a^{k-2}b^2-\cdots-ab^{k-1}-b^k$$
$$= a^k - b^k$$

lo que prueba el teorema. ■

4.3 Orden máximo de los elementos

Es necesario determinar los escalares a y b que podemos usar para que el orden de un elemento de ζ alcance el máximo valor posible.

Teorema 4.4: *Consideremos un elemento*

$$M = \begin{pmatrix} a & P \\ & b \end{pmatrix} \in \zeta \text{, con } a \neq b,$$

si $\text{mcm}(o(a), o(b)) = p-1$ *entonces*

$$o(M) = p-1.$$

Demostración: Debemos comprobar que $p-1$ es el menor entero positivo que cumple que $M^{p-1} = I$. Como $p-1 = \text{o}(a)\cdot j$ para algún j, entonces $a^{p-1} = a^{o(a)\cdot j} = 1^j = 1$, y ocurre lo mismo para b. Además, como $p-1$ es el mínimo común múltiplo de los órdenes de ambos elementos, no puede existir un número $h < p-1$ tal que $a^h = b^h = 1$. Por tanto, sólo debemos probar que $P^{(p-1)} = c^{(p-1)}P = O$ para todo $P \neq O$. Como $o(P) = p \neq c^{(p-1)}$, entonces debemos comprobar que $c^{(p-1)} = 0$.

Por (4.3), y aplicando el "pequeño" teorema de Fermat, $g^{p-1} \equiv 1 \mod p$, tenemos que:

$$\begin{aligned} c^{(p-1)} &= (b^{p-1} - a^{p-1})(b-a)^{-1} \\ &= (1-1)(b-a)^{-1} \\ &= 0 \end{aligned} \tag{4.4}$$

lo que prueba el teorema. ■

4.4 Análisis del coeficiente

Por la definición del sistema sabemos que $c^{(0)}=0$ y que $c^{(1)}=1$. Por otra parte, como hemos visto en (4.4), $c^{(p-1)}=0$, con $a\neq b$. Si no fuera así, es decir, si $a=b$, entonces $c^{(p-1)}\neq 0$ ya que

$$\begin{aligned} c^{(p-1)} &= a^{p-2}+a^{p-3}a+\cdots+a^{p-2} \\ &= (p-1)a^{p-2} \\ &= (p-1)a^{-1}, \end{aligned}$$

y esta expresión es igual a cero sólo si $p=1$. Además se podría realizar de forma inmediata la reducción

$$c^{(k)}=ka^{k-1}.$$

Sin embargo, el resto de valores posibles de $c^{(k)}$ para un a y b determinados no son únicos en el rango $[0,p-1]$; de hecho, sólo puede tomar $p-1$ valores posibles ya que $c^{(p-1)}$ vuelve a ser 0 debido a que superamos el orden del elemento del que forma parte.

Podemos ver por (4.3) que $c^{(k)}$ tomará los valores determinados por la relación:

$$(b^k-a^k)(b-a)^{-1}=c^{(k)},$$

es decir,

$$b(b^{k-1}-c^{(k)})=a(a^{k-1}-c^{(k)}).$$

De la misma forma, $a^k=b^k$ cuando se dé $c^{(k)}=0$, expresión que se cumple con mayor frecuencia que el resto, por lo que el valor cero de $c^{(k)}$ no debe utilizarse, además de que no podríamos usar los protocolos criptográficos. Para comprobar la frecuencia con la que se repite el valor $c^{(k)}=0$, veamos el siguiente teorema.

Teorema 4.5: *Sea* $k\in\left]0,p-1\right[$ *el menor valor, si existe, tal que* $c^{(k)}=0$, *con* $a\neq b$ *y* $a,b\neq 0$, *entonces* $k\mid p-1$ *y* $c^{(k\cdot l)}=0$ *para* $l\leq\dfrac{p-1}{k}$; *es decir, el coeficiente cero se repite* $\dfrac{p-1}{k}$ *veces.*

Demostración: si $c^{(k)}=0$, es porque $a^k=b^k$, y entonces cualquier múltiplo de k anula el coeficiente:

$$\begin{aligned} c^{(k\cdot l)} &= (b^{k\cdot l} - a^{k\cdot l})(b-a)^{-1} \\ &= \left((b^k)^l - (a^k)^l\right)(b-a)^{-1} \\ &= \left((b^k)^l - (b^k)^l\right)(b-a)^{-1} \\ &= 0\,. \end{aligned}$$

Para demostrar que $k \mid p-1$ hay que ver que no existen más valores, además de los múltiplos de k, que anulen el coeficiente. Supongamos que existe $d \in]k, p-1[$ tal que $k \nmid d$, es decir, d no es un múltiplo de k y sin embargo se cumple que $c^{(d)} = 0$. Entonces, si escogemos el mayor valor de l tal que $k \cdot l < d$, ocurre que:

$$\begin{aligned} a^d &= b^d \\ a^{k\cdot l} a^{d-k\cdot l} &= b^{k\cdot l} b^{d-k\cdot l} \\ a^{k\cdot l} a^{d-k\cdot l} &= a^{k\cdot l} b^{d-k\cdot l} \\ a^{d-k\cdot l} &= b^{d-k\cdot l} \end{aligned}$$

y por tanto $c^{(d-k\cdot l)} = 0$, siendo $d - k\cdot l < k$, lo cual es una contradicción ya que k era el menor valor que anulaba el coeficiente. ■

4.5 Distribución empírica del coeficiente

La distribución del coeficiente está basada en gran medida en la del logaritmo discreto, de ahí la dificultad para su caracterización, que parece inviable, lo que nos conduce a un estudio experimental de su comportamiento.

Para realizar las pruebas, hemos recorrido todos los valores de $c^{(k)}$ para $k \in [0, p-2]$. Como hemos visto en la sección anterior, el valor $c^{(k)} = 0$ ocurre con más frecuencia que los otros, por ello ha sido excluido de los cálculos estadísticos.

Según estas pruebas, la media aritmética de la cantidad de elementos que no aparecen ninguna vez se aproxima al 36,8%. Este valor es el mismo que el porcentaje de elementos que aparecen sólo una vez. La cantidad de elementos distintos que aparecen dos veces se acerca mucho al 18,4%, es decir, la mitad. La cantidad de elementos diferentes que aparecen tres veces se aproxima mucho al 6,1%, esto es, la tercera parte de los elementos que aparecen dos veces, y así sucesivamente.

Definimos una variable aleatoria X, que mide el número de repeticiones asociado a un valor calculado $c^{(k)}$ (de ahí que la probabilidad de los valores que no aparecen es cero). Tenemos que

$$P(X=1)=0,368$$
$$P(X=2)=0,368$$
$$P(X=3)=0,183$$
$$P(X=4)=0,061$$
$$\ldots$$

y de ahí definimos la función de cuantía

$$f(x)=P(X=x)=\frac{0,368}{x!}x.$$

Para que esté correctamente definida es preciso que

$$\sum_i f(x_i)=\sum_i \frac{0,368}{x_i!}x_i=1,$$

por ello afinamos el ajuste de la probabilidad inicial obteniendo:

$$P(X=1)=0,3678794411714423.$$

Esta conjetura nos conduce a obtener la cantidad máxima de repeticiones que esperamos que se produzcan según el tamaño de p.

Para que se dé un número determinado de repeticiones, es necesario que exista al menos un valor para el que se pueda dar esa cantidad de repeticiones, es decir,

$$f(x)p=\frac{0,368}{x!}xp\geq 1.$$

Aplicando logaritmos y despejando nos queda

$$\log_{10}\left(\frac{0,368}{x!}xp\right)\geq \log_{10}1,$$

es decir,

$$\log_{10}\frac{0,368}{(x-1)!}\geq -\log_{10}p.$$

Por ejemplo, para que exista un valor que se pueda repetir 10 veces, tenemos que

$$\log_{10}(0,368/(10-1)!)\approx -6,$$

por tanto p debería tener aproximadamente 6 dígitos en decimal, $\log_{10}p\approx 6$.

Las pruebas indican que con valores muy pequeños de p, la variabilidad es alta, pero con valores elevados, necesarios para implementaciones prácticas, los cálculos realizados se cumplen con bastante exactitud.

Es obvio que los valores del coeficiente que nos interesan son aquellos que no se repiten (pero aparecen una vez), ya que de esa forma una búsqueda exhaustiva necesitaría recorrer todos los valores en el intervalo $[0, p-2]$. Si el coeficiente apareciera dos veces, la probabilidad nos llevaría a recorrer la mitad de los valores, y así sucesivamente. Considerando que la complejidad del problema es de orden exponencial, esto no representa una gran merma. Por otra parte, aquellos valores del coeficiente que no aparecen, no suponen óbice alguno para mantener la dureza del problema, ya que al no conocerlos no podemos excluirlos de la búsqueda.

En una implementación práctica donde, por ejemplo, $p \approx 2^{200}$, es decir, el tamaño de p es de 200 bits, el coeficiente podría tomar aproximadamente el mismo valor 49 veces como máximo; esto reduciría el problema a uno equivalente de $2^{200}/49 \approx 2^{194}$ en el que no se producirían repeticiones. Sin embargo, un ataque por fuerza bruta no es tal, sino que se utilizan algoritmos de raíz cuadrada que siempre son aplicables a cualquier grupo. Por tanto, si el tamaño del problema inicial es de $\sqrt{p} = 2^{100}$, entonces el tamaño del problema equivalente será de $2^{100}/\sqrt{49} \approx 2^{97}$, como vemos, una merma de tan sólo 3 bits.

Esto ocurriría considerando el caso más desfavorable, en el que escogiéramos un valor del coeficiente que tuviera el número máximo de repeticiones, algo que es muy improbable que ocurra. Si calculamos la esperanza matemática del número de repeticiones, obtendremos:

$$E(X) = \sum_i x_i f(x_i) = \sum_i \frac{0{,}368}{x_i!} x_i^2 = 2\,.$$

Si ahora dividimos la dureza del problema planteado entre el número de repeticiones esperado, considerando algoritmos de raíz cuadrada, obtendremos un problema equivalente a $2^{100}/\sqrt{2} \approx 2^{99{,}6}$, valor que se ajusta mucho más a la realidad y que, como vemos, produce una merma en el problema casi inapreciable.

Para obtener unos datos estadísticos fiables, son necesarias miles de pruebas. De ellas se detalla una muestra representativa en la tabla 4.1. Para diferentes valores de p, a y b, las columnas numeradas indican la cantidad de valores que el coeficiente no toma (columna 0), la cantidad de valores que el coeficiente toma una vez (columna 1), la cantidad de valores que el coeficiente toma dos veces (columna 2), y así sucesivamente hasta el máximo número posible de repeticiones (9 en este caso). La columna $c^{(k)} = 0$ indica las veces que $c^{(k)}$ es nulo para esos valores de a, b y p. Por ejemplo: para $p = 26.099$, $a = 18.024$ y $b = 3.550$, la columna 0 indica que existen 9.648 elementos de $\mathbb{F}_p$ que no

aparecen, esto es, $c^{(k)}$ nunca toma esos valores. La columna 1 señala que hay 9.508 elementos de $\mathbb{F}_p$ que $c^{(k)}$ toma una sola vez. La columna 2 indica que existen 4.843 elementos de $\mathbb{F}_p$ que $c^{(k)}$ toma dos veces, y así sucesivamente.

Tabla 4.1: Veces que los elementos de $\mathbb{F}_p$ aparecen en $c^{(k)}$ para distintos valores de p, a y b

p	a	b	0	1	2	3	4	5	6	7	8	9	$c^{(k)}=0$
7	1	3	1	5									1
37	13	18	6	30									6
53	21	14	20	20	8	4							4
137	62	30	50	53	19	12	2						1
179	146	38	67	67	22	22							1
223	23	155	78	94	28	20	0	2					2
223	162	122	81	84	42	9	6						3
223	96	83	81	85	37	15	3	1					1
347	175	158	130	120	72	16	7	1					1
409	37	192	142	161	76	24	4	0	1				1
449	218	136	128	256	64								64
457	37	204	171	161	87	29	7	1					1
457	401	105	173	156	90	30	6	1					1
461	228	188	156	194	78	22	8	2					2
467	319	367	168	178	87	23	6	4					1
479	421	395	177	174	89	29	7	2					1
487	237	429	174	201	63	36	12						3
503	337	20	172	200	94	32	4						2
509	210	385	212	152	96	32	16						4
523	14	359	188	192	114	20	2	4	2				2
541	55	294	200	210	70	60							10
547	274	340	222	180	93	33	12	6					3
563	411	42	208	209	93	43	8	1					1
569	152	99	204	206	126	26	4	0	0	2			2
577	58	305	210	219	98	38	10	0	1				1
647	342	468	228	251	119	40	5	2	1				1
673	271	135	260	220	138	44	8	2					2
683	363	617	198	341	110	22	11						11
683	170	350	250	249	130	42	9	2					1
787	720	518	288	284	150	56	8						2
941	524	74	364	326	162	68	18	0	2				2
1.009	846	319	424	308	176	68	20	12					4
1.031	56	264	337	425	208	52	8						1
1.129	1.029	62	429	402	186	96	12	3					3
1.213	785	233	434	478	198	77	20	4	1				1
1.277	1.098	245	444	476	292	44	20						4
1.361	1.150	935	495	490	290	60	20	5					5
1.361	1.314	475	520	448	304	56	32						8
1.409	1.351	646	524	504	276	76	20	8					4
1.447	629	1.272	519	556	264	79	19	7	1	1			1
1.483	460	935	572	455	364	78	13						13
1.567	1.565	1.047	612	612	144	144	54						18
1.657	940	351	600	632	296	88	36	4					4
1.913	1.697	1.163	715	683	349	135	25	5					1
1.973	58	1.546	728	722	364	117	35	6					1

p	a	b	0	1	2	3	4	5	6	7	8	9	$c^{(k)}=0$
2.063	1.539	476	788	688	424	130	26	6					2
2.099	1.603	1.397	748	830	342	144	24	6	4				2
2.333	1.455	266	855	862	438	125	42	10					1
2.333	1.494	353	874	820	448	150	38	0	2				2
2.417	1.513	1.256	882	903	444	135	42	9	1				1
2.521	920	981	896	960	480	144	40						8
2.539	1.459	877	927	957	440	165	42	6	0	1			1
2.767	1.311	233	1.023	1.011	498	186	42	6					3
2.789	1.878	782	1.058	974	522	188	28	16	2				2
2.819	1.854	1.868	1.043	1.026	521	170	52	5	1				1
2.879	2.842	393	1.044	1.094	505	186	34	12	2	1			1
2.879	291	927	1.057	1.066	521	180	42	11	1				1
2.887	2.528	2.403	1.074	1.017	576	165	51	3					3
2.887	1.575	2.084	1.075	1.038	529	196	39	9					1
2.917	928	772	1.073	1.072	536	180	48	4	2	1			1
2.927	2.366	1.160	1.072	1.086	529	187	42	9	0	1			1
2.953	117	1.829	1.097	1.067	534	210	34	10					1
3.067	1.656	2.820	1.119	1.135	578	175	48	10	0	1			1
3.067	594	391	1.125	1.135	563	181	49	13					1
3.089	1.610	1.970	1.136	1.134	573	187	48	7	2	1			1
3.163	2.359	1.783	1.155	1.164	589	207	39	6	2				1
3.167	1.268	1.159	1.157	1.183	578	178	60	8	2				1
3.329	2.925	697	1.213	1.256	589	201	56	12	1				1
3.373	3.147	2.726	1.244	1.233	618	222	43	9	2	1			1
3.389	1.975	1.713	1.247	1.246	620	214	47	13	1				1
3.433	877	1.479	1.261	1.260	633	217	54	5	1	1			1
3.433	2.952	332	1.277	1.233	638	223	54	5	2				1
3.491	1.626	1.908	1.295	1.265	637	235	47	9	2				1
3.761	2.154	2.436	1.383	1.383	698	217	66	13					1
3.767	1.989	3.427	1.392	1.388	670	244	60	10	0	2			2
3.793	2.597	1.738	1.397	1.387	705	237	51	12	2	1			1
3.793	3.688	636	1.421	1.313	787	194	68	4	5				1
3.881	1.911	668	1.428	1.427	707	253	50	13	0	2			1
3.889	462	3.049	1.403	1.465	728	216	64	10	2				1
3.989	940	1.711	1.453	1.503	712	237	69	11	3				1
4.049	2.840	3.260	1.397	1.661	682	231	66	11					11
4.127	1.144	949	1.518	1.512	776	239	64	14	3				1
4.211	2.374	14	1.560	1.525	787	254	75	6	3				1
4.241	68	834	1.546	1.594	761	251	73	12	3				1
4.253	4.160	3.883	1.580	1.532	796	259	76	8	1				1
4.327	2.110	1.584	1.580	1.628	756	283	61	16	2				1
4.327	3.020	4.323	1.584	1.606	796	254	67	17	2				1
4.357	1.814	1.539	1.611	1.521	909	261	45	9					9
4.397	1.485	1.818	1.602	1.678	768	236	88	24					2
4.507	1.884	787	1.614	1.731	789	303	60	9					3
4.649	746	1.318	1.698	1.727	867	264	72	16	2	2			1
4.691	2.483	3.899	1.727	1.725	852	306	60	18	2				1
4.751	1.448	193	1.732	1.778	874	261	88	14	3				1
4.831	3.247	575	1.876	1.694	826	294	112	28					14
4.861	4.413	1.405	1.790	1.788	878	326	58	16	4				2
4.987	987	558	1.842	1.824	948	246	108	18					6
5.051	4.835	3.777	1.836	1.914	889	310	82	16	2	1			1
5.197	2.962	2.087	1.910	1.914	955	321	77	14	5				1
5.227	2.594	686	1.884	2.004	924	324	72	12	0	0	6		6

p	a	b	0	1	2	3	4	5	6	7	8	9	$c^{(k)}=0$
5.273	2.802	3.936	1.944	1.926	982	324	73	21	2				1
5.297	2.572	1.564	1.986	1.655	1.655								331
5.303	5.103	2.276	1.892	2.002	1.034	297	66	0	11				11
5.381	1.640	3.370	1.965	2.017	962	330	85	19	1	1			1
5.519	2.154	1.137	2.035	2.020	1.016	348	78	17	4				1
5.581	4.033	3.933	2.043	2.073	1.012	352	77	21	1	1			1
5.591	1.107	249	2.084	2.012	1.040	338	100	14	2				2
5.623	4.249	1.502	2.058	2.090	1.031	328	93	19	3				1
5.647	1.587	2.398	2.102	2.020	1.076	344	86	12	6				2
5.659	3.043	2.551	2.064	2.106	1.050	312	120	6					6
5.923	5.228	1.292	2.157	2.228	1.065	353	98	15	5	1			1
6.133	3.353	5.840	2.259	2.237	1.155	372	86	16	6	0	1		1
6.257	5.134	5.758	2.530	1.748	1.564	322	92						46
6.317	2.625	3.791	2.340	2.238	1.248	396	78	16					2
6.689	4.438	1.711	2.484	2.436	1.228	396	120	20	4				4
6.701	4.563	1.680	2.600	2.100	1.600	300	100						100
6.779	6.233	5.123	2.485	2.513	1.244	393	122	19	1	0	1		1
6.857	2.590	4.394	2.554	2.446	1.312	424	96	18	4	2			2
6.863	798	2.115	2.528	2.520	1.263	415	112	22	2				1
6.899	1.930	2.737	2.524	2.564	1.258	427	100	17	6	1	1		1
6.899	4.011	3.797	2.557	2.491	1.295	431	102	19	2	0	1		1
6.907	5.124	3.008	2.547	2.527	1.277	425	104	23	3				1
6.907	2.704	849	2.559	2.544	1.227	435	111	24	6				3
6.991	22	1.376	2.572	2.578	1.269	438	110	19	4				1
7.027	6.025	4.057	2.514	2.646	1.374	366	108	12	6				6
7.103	747	400	2.604	2.636	1.288	441	103	27	2	1			1
7.103	2.880	1.056	2.606	2.633	1.286	443	110	17	7				1
7.151	805	1.037	2.690	2.514	1.372	450	92	24	4	4			2
7.187	1.518	1.536	2.624	2.677	1.330	411	112	27	3	2			1
7.331	340	5.443	2.710	2.677	1.335	477	108	19	4				1
7.349	2.227	4.967	2.724	2.674	1.380	410	128	24	6	2			2
7.433	4.052	3.051	2.736	2.706	1.400	462	106	18	4				2
7.547	2.807	5.273	2.781	2.779	1.357	492	113	21	2	1			1
7.573	1.479	5.996	2.775	2.809	1.383	459	118	23	3	2			1
7.639	2.390	6.352	2.797	2.860	1.338	499	121	19	3	1			1
7.639	4.176	717	2.822	2.772	1.430	482	106	22	4				2
7.789	7.339	6.025	2.867	2.866	1.418	500	104	29	4				1
7.793	6.178	6.493	2.856	2.890	1.434	452	128	28	4				2
7.823	7.637	1.228	2.902	2.852	1.446	440	160	16	6				2
7.867	3.657	7.770	2.859	2.971	1.407	479	115	27	8				1
7.901	3.449	7.386	2.885	2.922	1.483	465	116	23	5	1			1
8.009	898	5.135	2.916	3.006	1.454	476	122	30	2	2			2
8.147	1.682	3.119	2.976	3.048	1.471	487	130	30	4				1
8.147	6.822	678	3.024	2.958	1.474	566	86	32	6				2
8.161	251	4.243	2.945	3.045	1.545	500	115	5	0	5			5
8.167	3.954	93	3.028	2.952	1.525	513	117	30	1				1
8.179	2.935	801	3.015	2.992	1.512	512	118	22	6	1			1
8.219	109	6.400	2.926	3.122	1.589	434	126	21					7
8.231	5.422	925	3.044	3.034	1.468	528	116	30	10				2
8.273	1.285	151	3.076	3.004	1.552	456	140	32	12				4
8.297	1.426	3.217	3.042	3.014	1.620	470	124	22	4				2
8.329	2.936	4.656	3.024	3.288	1.272	528	192	24					24
8.377	507	480	3.074	3.112	1.504	519	138	28	1				1
8.513	5.105	6.571	3.088	3.116	1.684	488	124	8	4				4

p	a	b	0	1	2	3	4	5	6	7	8	9	$c^{(k)}=0$
8.563	1.301	2.508	3.141	3.153	1.590	507	150	21					3
8.563	8.231	577	3.180	3.099	1.567	560	132	24					1
8.627	310	3.660	3.156	3.212	1.571	520	131	29	7				1
8.663	4.143	7.920	3.160	3.242	1.582	498	146	28	6				2
8.699	7.986	7.993	3.165	3.289	1.541	528	138	32	5				1
8.779	4.276	7.085	3.210	3.271	1.584	549	133	28	2	1			1
8.999	1.720	8.190	3.297	3.348	1.614	582	117	34	5	1			1
9.041	394	4.741	3.342	3.271	1.726	529	134	35	3				1
9.133	5.014	4.176	3.336	3.354	1.755	525	126	30	6				3
9.239	7.572	4.012	3.388	3.436	1.666	558	162	22	6				2
9.239	4.372	5.402	3.396	3.425	1.645	602	137	30	3				1
9.257	2.023	85	3.360	3.544	1.632	480	208	24	8				8
9.277	2.348	1.281	3.411	3.393	1.740	573	120	33	6				3
9.283	1.669	3.869	3.400	3.448	1.672	598	132	26	6				2
9.293	4.852	1.633	3.380	3.508	1.692	520	132	52	8				4
9.311	4.611	5.130	3.326	3.582	1.668	588	106	38	2				2
9.337	1.364	649	3.398	3.512	1.698	533	153	36	6				1
9.371	4.969	341	3.414	3.514	1.690	562	166	20	4				2
9.371	5.178	4.594	3.480	3.396	1.724	600	134	28	8				2
9.587	1.971	5.294	3.521	3.540	1.755	588	147	28	6	1			1
9.587	5.633	9.506	3.530	3.466	1.824	608	144	14					2
9.587	7.208	6.481	3.530	3.508	1.807	542	159	39	1				1
9.661	1.022	5.961	3.565	3.550	1.732	642	139	29	3				1
9.739	7.046	8.016	3.568	3.624	1.764	577	176	24	5				1
9.829	4.124	3.477	3.602	3.642	1.802	584	167	25	6				1
9.923	7.604	4.823	3.675	3.581	1.882	606	139	33	5	1			1
9.941	9.401	4.342	3.430	4.018	1.708	672	84	28					14
9.973	3.463	1.762	3.690	3.726	1.701	657	144	36	18				9
10.613	3.991	6.440	3.936	3.952	1.796	704	184	28	8	4			4
15.149	9.036	13.315	5.563	5.585	2.801	908	231	49	10	1			1
15.161	1.662	3.896	5.415	5.770	2.855	860	215	35	10				5
15.269	671	229	5.639	5.561	2.843	952	207	61	4	1			1
17.231	10.267	2.123	6.377	6.233	3.257	1.047	251	53	12				1
18.713	9.916	7.328	6.842	6.930	3.450	1.148	282	52	8				2
21.313	8.566	21.003	7.819	7.894	3.891	1.287	347	59	14	1			1
22.567	5.066	13.068	8.321	8.257	4.173	1.394	340	68	11	2			1
23.993	6.185	1.820	8.824	8.805	4.474	1.430	372	65	19	3			1
24.593	10.194	68	8.928	9.200	4.560	1.392	480	32					16
25.703	17.509	11.267	9.455	9.460	4.717	1.582	399	71	17	0	1		1
26.099	18.204	3.550	9.648	9.508	4.834	1.622	390	82	14				2
27.581	327	6.286	10.147	10.165	5.021	1.733	406	99	9				1
30.757	6.159	1.070	11.329	11.287	5.641	1.946	444	85	22	1	1		1
35.923	7.021	30.786	13.332	12.996	6.678	2.256	516	132	12				6
37.691	9.988	19.367	13.889	13.798	7.010	2.254	607	114	15	3			1
38.377	1.286	34.641	13.949	14.235	7.215	2.314	572	78	13				13
39.869	30.116	23.486	14.708	14.500	7.486	2.458	596	90	26	2	2		2
40.519	34.114	31.241	14.944	14.808	7.519	2.491	612	117	24	3			1
40.759	24.454	36.574	15.080	14.914	7.424	2.572	614	110	40	0	4		2
41.491	7.474	21.873	15.245	15.302	7.597	2.559	645	121	17	3	1		1
42.323	19.855	21.020	15.564	15.604	7.714	2.656	629	128	25	1	1		1
44.617	17.284	40.422	16.536	16.155	8.391	2.646	726	138	24				3
48.073	28.829	259	17.704	17.646	8.855	2.942	773	122	25	4	0	1	1
48.299	42.067	26.408	17.548	17.753	9.471	2.706	697	82	41				41
48.869	12.361	33.411	17.964	18.003	8.979	2.998	749	142	27	4	2		1

p	a	b	0	1	2	3	4	5	6	7	8	9	$c^{(k)}=0$
49.211	15.642	5.040	18.134	17.998	9.172	2.977	750	140	37	2			1
49.391	40.591	38.521	18.185	18.124	9.130	3.018	747	156	27	3			1
49.727	39.054	42.591	18.158	18.460	9.206	2.970	752	150	28	0	2		2
58.189	18.496	48.411	21.383	21.461	10.696	3.496	954	162	32	4			1
64.693	27.274	60.073	23.828	23.733	11.929	3.987	974	209	26	6			1
65.323	48.536	46.914	24.041	23.987	12.060	4.020	971	195	41	7			1
69.109	28.990	6.975	25.457	25.321	12.800	4.251	1.014	218	42	5			1
94.049	73.755	22.721	34.670	34.538	17.240	5.814	1.420	298	60	8			2
102.983	55.335	1.723	37.678	38.222	18.838	6.328	1.546	312	52	6			2
131.249	101.132	38.334	48.230	48.390	24.112	8.020	2.004	410	68	13	1		1
145.349	26.173	68.048	53.584	53.276	26.824	8.924	2.176	480	52	24	8		4
162.989	96.698	48.091	60.396	59.528	29.668	10.216	2.532	548	96	4			4
163.433	159.535	5.766	60.089	60.192	30.061	9.953	2.545	488	91	12	1		1
183.437	1.446	278	67.497	67.413	33.840	11.228	2.757	592	93	16			1
227.453	100.432	217.114	83.402	84.092	41.784	13.928	3.404	690	124	28			2
252.691	79.508	18.325	93.360	92.640	46.350	14.940	4.590	735	45	30			15
275.129	44.529	93.897	101.535	100.275	51.380	16.863	4.102	763	196	14			7
313.739	251.677	119.252	115.350	115.580	57.624	19.240	4.766	972	180	24	2		2
347.251	59.478	23.916	127.125	129.255	63.135	20.910	5.385	1.215	210	15			15
382.919	252.791	342.929	140.867	140.838	70.501	23.419	5.941	1.102	211	33	5	1	1
397.433	334.120	287.208	146.134	146.337	73.053	24.371	6.085	1.212	211	24	4	1	1
452.269	325.585	380.077	166.365	166.296	83.431	27.587	6.928	1.381	236	37	7		1
514.739	448.899	120.496	189.374	189.254	94.882	31.447	7.899	1.571	260	41	10		1
545.647	284.344	370.804	200.544	201.189	99.795	33.921	8.253	1.629	264	48	3		3
558.563	200.697	87.423	205.481	205.478	102.724	34.344	8.467	1.733	281	49	4	1	1
560.411	268.678	223.845	206.391	205.544	103.594	34.270	8.563	1.726	277	39	4	2	1
641.681	156.712	435.027	236.656	235.592	117.688	39.120	10.288	1.896	368	72			8
764.587	377.150	135.721	281.152	281.551	140.519	46.785	11.813	2.271	438	51	5	1	1
842.383	184.490	508.180	309.962	309.552	155.238	51.934	12.500	2.694	438	60	4		2
843.209	806.199	14.579	310.323	310.021	154.977	51.929	12.903	2.570	419	59	7		1
886.493	761.262	883.767	326.216	325.796	163.672	53.820	13.630	2.856	430	66	6		2
907.363	670.163	54.478	333.635	334.290	166.406	55.810	13.918	2.745	486	60	12		1
924.877	629.500	878.310	340.576	339.196	170.848	57.008	13.896	2.808	484	52	8		4
967.819	479.809	368.329	356.307	354.852	179.364	59.265	14.469	2.910	564	66	21		3

4.6 Búsqueda de coeficientes sin repeticiones

Como hemos visto, más de la tercera parte de los valores del coeficiente sólo aparecen una vez y, a pesar de que la dureza del sistema apenas se reduce por las repeticiones, es conveniente evitarlas. Para ello vamos a transformar el problema de obtener las repeticiones de un coeficiente dado, tarea complicada por su aparente comportamiento

altamente aleatorio, en el problema de obtener las raíces de un polinomio en un cuerpo finito.

Una vez escogido un valor k y obtenido el coeficiente $c^{(k)}$, hemos de determinar si existe algún valor $t \neq k$ tal que $c^{(t)} = c^{(k)}$. Si esto es así, tendremos que

$$(b^t - a^t)(b-a)^{-1} = (b^k - a^k)(b-a)^{-1}$$

y por tanto

$$b^t - a^t - (b^k - a^k) = 0. \tag{4.5}$$

Para plantear el sistema escogemos un generador g del grupo multiplicativo $\mathbb{F}_p^*$, que sabemos que existe por ser p primo; tras ello tomamos dos valores i y j que actúen como exponentes de g, de forma que $g^i = a$ y $g^j = b$; y por último tomamos un valor aleatorio k. Ahora podemos ver la ecuación (4.5) como

$$(g^j)^t - (g^i)^t = (g^j)^k - (g^i)^k,$$

es decir

$$(g^t)^j - (g^t)^i = (g^j)^k - (g^i)^k,$$

y, tras calcular el término independiente, podemos plantearla de la forma

$$z^j - z^i - \gamma = 0.$$

Ahora el problema consiste en determinar si el trinomio $f(z) = z^j - z^i - \gamma$ tiene una raíz o más. Sabemos que tiene una, obviamente $z = g^k$, porque la hemos usado para plantear la ecuación anterior, pero determinar si tiene más raíces puede ser muy duro de resolver si consideramos que el cuerpo finito sobre el que está definido el polinomio es muy grande.

Una forma sencilla sería tomar $j = 3$ e $i \in [1,2]$, de forma que $a = g$ o $a = g^2$, y $b = g^3$, con lo que $f(z)$ sería un polinomio de tercer grado, y, como sabemos que tiene una raíz en $z = g^k$, podemos obtener

$$h(z) = f(z)(z - g^k)^{-1}$$

donde $h(z)$ es de segundo grado. Sólo faltaría comprobar que el discriminante de este último no es residuo cuadrático, es decir $\sqrt{D(h(z))} \notin \mathbb{F}_p$, para asegurar que es irreducible y por tanto la única raíz de $f(z)$ es la que poseemos.

Mediante un simple mecanismo de prueba y error, no debemos tardar mucho en encontrar un valor de k adecuado ya que la cantidad de coeficientes que sólo aparecen una vez supera la tercera parte del total. Sin embargo, dar explícitamente el valor del

generador con i y j, y por tanto la relación entre a y b, facilitaría en cierta medida la búsqueda de k a partir de $c^{(k)}$.

Otra forma más elaborada consiste en tomar $i=1$, es decir $g=a$, de manera que el trinomio quede de la forma $f(z)=z^j-z-\gamma$. El uso del elemento a como generador no implica una forma más sencilla de encontrar la relación entre a y b, ya que el elemento b puede tomar cualquier valor en $\mathbb{F}_p$, excluyendo el cero y a, por tanto el coste de encontrarlo sería similar a la búsqueda de i y j a partir de un generador cualquiera. Sin embargo, determinar si este trinomio tiene una única raíz requiere de algoritmos de búsqueda de raíces de teoría de polinomios en cuerpos finitos que tienen una alta complejidad algorítmica, y por tanto son ineficientes en la práctica. Véase [55] para más información sobre la búsqueda de raíces de polinomios.

4.7 Sistemas criptográficos

Para estudiar la aplicabilidad del sistema a la criptografía de clave pública, vamos a utilizar el conocido protocolo de intercambio de claves de Diffie-Hellman. El protocolo funciona usando este problema matemático, pero no aumenta la dureza de un sistema basado en el Problema del Logaritmo Discreto de Curva Elíptica (ECDLP), como se verá en el criptoanálisis.

4.7.1 Protocolo de intercambio de claves de Diffie-Hellman

Consideremos el subgrupo de orden $p-1$ generado por el elemento

$$M=\begin{pmatrix} a & P \\ & b \end{pmatrix} \in \zeta$$

como se vio en la definición del sistema. Este elemento es público.

Para compartir una clave secreta:

1. Alicia toma un elemento $k \in \mathbb{F}_p$ como su clave privada, calcula

$$M^k=\begin{pmatrix} a^k & P^{(k)} \\ & b^k \end{pmatrix},$$

extrae el elemento $P^{(k)}$ y se lo envía a Bernardo.

2. A continuación, Bernardo toma un elemento $m \in \mathbb{F}_p$ como su clave privada, calcula

$$M^m = \begin{pmatrix} a^m & P^{(m)} \\ & b^m \end{pmatrix},$$

extrae el elemento $P^{(m)}$ y se lo transmite a Alicia.

3. Ella entonces calcula

$$\begin{pmatrix} a & P^{(m)} \\ & b \end{pmatrix}^k = \begin{pmatrix} a^k & \left(P^{(m)}\right)^{(k)} \\ & b^k \end{pmatrix}$$

y considera el elemento $\left(P^{(m)}\right)^{(k)}$.

4. Por último, Bernardo calcula

$$\begin{pmatrix} a & P^{(k)} \\ & b \end{pmatrix}^m = \begin{pmatrix} a^m & \left(P^{(k)}\right)^{(m)} \\ & b^m \end{pmatrix}$$

y considera el elemento $\left(P^{(k)}\right)^{(m)}$.

El protocolo funciona correctamente como consecuencia del siguiente teorema.

Teorema 4.6: *Con la notación anterior, se cumple que*

$$\left(P^{(m)}\right)^{(k)} = \left(P^{(k)}\right)^{(m)}. \tag{4.6}$$

Demostración: Por (4.2) tenemos que

$$\left(P^{(m)}\right)^{(k)} = \left(\sum_{i=0}^{k-1} a^{k-1-i} b^i\right) P^{(m)} = c^{(k)} P^{(m)} = c^{(k)} c^{(m)} P$$

y

$$\left(P^{(k)}\right)^{(m)} = \left(\sum_{i=0}^{m-1} a^{m-1-i} b^i\right) P^{(k)} = c^{(m)} P^{(k)} = c^{(m)} c^{(k)} P.$$

Como $c^{(k)}, c^{(m)} \in \mathbb{F}_p$, entonces $c^{(k)} c^{(m)} = c^{(m)} c^{(k)}$ y en consecuencia se verifica (4.6). ■

El secreto compartido que toman Alicia y Bernardo es por tanto $\left(P^{(m)}\right)^{(k)} = \left(P^{(k)}\right)^{(m)}$.

4.7.2 Criptoanálisis del protocolo de intercambio de claves de Diffie-Hellman

El problema planteado en este capítulo como función unidireccional con trampa consiste en encontrar el escalar k a partir de $P^{(k)}$. Este problema, aunque similar, no es exactamente igual que el problema de Diffie-Hellman, que consiste en encontrar el secreto compartido $\left(P^{(k)}\right)^{(m)}$ a partir de $P^{(k)}$ y $P^{(m)}$.

Como hemos visto, el protocolo de intercambio de claves funciona gracias a la conmutatividad que se da en la aplicación de la operación $\left(P^{(m)}\right)^{(k)}$ que resulta $\left(P^{(k)}\right)^{(m)}$. Sin embargo, a pesar de que el protocolo funciona, no es necesario encontrar k (o equivalentemente m) para reducir el sistema al ECDLP.

Es posible realizar un ataque por fuerza bruta sobre $P^{(k)}$ hasta encontrar un elemento α tal que $\alpha P = P^{(k)}$. Como vimos en la descripción del sistema, obtener la clave pública a partir de la clave privada y del punto base es equivalente a realizar un múltiplo de este punto. Por tanto α realiza la misma acción que la aplicación de la clave privada independientemente del punto al que se le aplica. Esto quiere decir que podemos obtener el múltiplo $\alpha P^{(m)}$ del punto $P^{(m)}$ obteniendo el secreto compartido $\alpha P^{(m)} = \left(P^{(m)}\right)^{(k)}$.

La seguridad del protocolo de intercambio de claves de Diffie-Hellman es similar tanto si aplicamos el sistema expuesto como si utilizamos el problema del logaritmo discreto con curvas elípticas. En consecuencia, este problema queda sólo como alternativa.

A los esquemas de cifrado se les puede aplicar el mismo criptoanálisis ya que basan su seguridad en la utilización de un clave secreta compartida, con lo que se podrían plantear también como alternativa sin obtener una mejora sobre los esquemas de cifrado bajo el ECDLP. Los esquemas de firma digital, aunque algo diferentes, tampoco aportan ventajas ya que un ataque por fuerza bruta sobre ellos produciría los mismos resultados que un ataque sobre un esquema de firma con el ECDLP.

4.8 Conclusiones

La búsqueda de sistemas capaces de incrementar la seguridad de los problemas existentes, evitando los inconvenientes inherentes a estos aumentos, es lo que nos ha movido a diseñar un nuevo sistema que relaciona el problema del logaritmo discreto de

ciertos elementos con las curvas elípticas. Para construir estos elementos, hemos usado curvas elípticas porque actualmente representan el mejor problema matemático que se conoce aplicable a la criptografía de clave pública. El objetivo era crear un problema con la misma dureza que el ECDLP, pero sin los complejos cálculos requeridos para establecer estos sistemas, sobre todo con tamaños de clave elevados.

El nuevo sistema está basado en dos problemas; esto es una ventaja porque resolver uno de ellos no implica resolver el problema, sino sólo una parte. Hemos analizado, además, los problemas que pueden reducir la dureza del sistema, como las repeticiones del coeficiente, mostrando que la disminución que producen en la seguridad es inapreciable, pero dando posibles vías para evitarlas.

Se ha diseñado una función unidireccional con trampa que funciona como problema subyacente de los protocolos criptográficos, como se ha visto con el de Diffie-Hellman; sin embargo, aunque el sistema es seguro porque no se conoce ningún algoritmo subexponencial capaz de atacar la curva en la que se basa, en la práctica sólo se puede plantear como sistema alternativo ya que no obtiene ventajas significativas sobre el ECDLP. La única mejora que se podría considerar es que un ataque por fuerza bruta –o por algoritmo de raíz cuadrada– requeriría de mayor esfuerzo de computación en cada iteración, mientras que en el cálculo de las claves apenas sería apreciable usando algoritmos de exponenciación rápida.

El estudio teórico del problema expuesto, por otra parte, es necesario como base para abordar el sistema del siguiente capítulo que, como veremos, evita la generación de un coeficiente que permita simular el comportamiento de las claves, por lo que sí resulta competente para su utilización en criptografía de clave pública.

Capítulo 5

Sistema criptográfico no lineal de curva elíptica y matrices formales triangulares por bloques

El sistema que presentamos ahora está basado en el introducido en el capítulo anterior, pero en lugar de usar matrices formales compuestas por elementos de determinado tipo, utiliza matrices formales compuestas por tres submatrices o bloques; dos de ellas compuestas a su vez por elementos de $\mathbb{F}_p$ y la restante submatriz compuesta por puntos de una curva elíptica. El término no lineal hace referencia a la combinación no lineal de matrices que aparece cuando se elevan los elementos a las sucesivas potencias. La complejidad inherente a estos elementos y la complejidad de las operaciones en la curva elíptica, junto con la imposibilidad de separarlos, es lo que da garantías de su dureza. Con este sistema se consigue incrementar el espacio de claves sin necesidad de aumentar el tamaño de la curva elíptica que se utiliza y, consecuentemente, sin los requerimientos computacionales inherentes al establecimiento de una curva elíptica de gran tamaño de forma aleatoria.

La propuesta es un sistema capaz de definir fácilmente problemas con una seguridad mayor que la que normalmente se determina, mientras que se mantiene la complejidad más alta posible. Esto es factible porque está basado en la dureza del ECDLP. El sistema utiliza una curva elíptica base, no necesariamente grande y por tanto fácilmente definible, o una de las curvas estándar propuestas por el NIST. Con esta curva elíptica es posible

establecer sistemas con claves del tamaño que se precise aumentando la seguridad de forma correspondiente.

Por último, se aplica este problema a algunos esquemas criptográficos usuales de clave pública, comprobando aquéllos en los que el sistema funciona y aquéllos en los que no, y argumentando los motivos.

5.1 Descripción del sistema

Sea $E/\mathbb{F}_q$ una curva elíptica con $p=\#E(\mathbb{F}_q)$ un número primo. Se considera el conjunto

$$\xi=\left\{\begin{pmatrix} A & \Pi \\ & B \end{pmatrix} \;\middle|\; A\in \mathcal{M}at_r(\mathbb{F}_p), B\in \mathcal{M}at_s(\mathbb{F}_p), \Pi\in \mathcal{M}at_{r\times s}\left(E(\mathbb{F}_q)\right)\right\}$$

donde $\mathcal{M}at_n(\mathbb{F}_p)$ denota el conjunto de todas las matrices cuadradas de tamaño $n\times n$ con elementos en $\mathbb{F}_p$, y $\mathcal{M}at_{r\times s}\left(E(\mathbb{F}_q)\right)$ denota el conjunto de todas las matrices de tamaño $r\times s$ formadas por puntos del grupo de puntos racionales $E(\mathbb{F}_q)$ de la curva elíptica.

Se define además una operación binaria sobre este conjunto como la multiplicación formal de matrices, esto es,

$$\begin{pmatrix} A & \Pi \\ & B \end{pmatrix}\begin{pmatrix} C & \Phi \\ & D \end{pmatrix}=\begin{pmatrix} AC & A\Phi+\Pi D \\ & BD \end{pmatrix}, \tag{5.1}$$

donde $A\Phi=[a_{ij}][P_{ij}]=[Q_{ij}]$ y $Q_{ij}=\sum_{k=1}^{r} a_{ik}P_{kj}$. De forma similar para ΠD.

Si observamos la estructura de los elementos, veremos que son matrices formales triangulares superiores por bloques. El bloque inexistente permite que se pueda realizar la multiplicación de matrices de la forma usual, ya que evita la multiplicación de dos bloques de puntos y, por tanto, la multiplicación de puntos de la curva elíptica, operación no definida.

El siguiente teorema proporciona las propiedades elementales de ξ.

Teorema 5.1: *La multiplicación formal de matrices definida en* (5.1) *es asociativa y posee elemento neutro.*

Demostración: Es fácil ver que

$$(M_1M_2)M_3 = M_1(M_2M_3)$$

para todo $M_1, M_2, M_3 \in \xi$, y

$$M\Gamma = M = \Gamma M$$

para todo $M \in \xi$, donde

$$\Gamma = \begin{pmatrix} I_r & O_{r\times s} \\ & I_s \end{pmatrix} \in \xi . \qquad \blacksquare$$

Notar que, por ejemplo, si A es una matriz no invertible, entonces el producto AC no puede ser I_r para todo $C \in \mathcal{M}at_r(\mathbb{F}_p)$, es decir,

$$\begin{pmatrix} A & \Pi \\ & B \end{pmatrix}\begin{pmatrix} C & \Phi \\ & D \end{pmatrix} \neq \begin{pmatrix} I_r & O_{r\times s} \\ & I_s \end{pmatrix}$$

para todo $C \in \mathcal{M}at_r(\mathbb{F}_p)$, $B, D \in \mathcal{M}at_s(\mathbb{F}_p)$ y $\Pi, \Phi \in \mathcal{M}at_{r\times s}\left(E(\mathbb{F}_q)\right)$, por tanto existen elementos no invertibles en ξ.

Para asegurar que sólo aparecen elementos invertibles, consideramos el conjunto

$$\zeta = \left\{ \begin{pmatrix} A & \Pi \\ & B \end{pmatrix} \in \xi \mid \mathrm{A} \in GL_r(\mathbb{F}_p), B \in GL_s(\mathbb{F}_p) \right\},$$

donde $GL_n(\mathbb{F}_p)$ denota el conjunto de todas las matrices invertibles de tamaño $n\times n$ con elementos en $\mathbb{F}_p$.

Es fácil ver que si $M = \begin{pmatrix} A & \Pi \\ & B \end{pmatrix} \in \zeta$, entonces

$$M^{-1} = \begin{pmatrix} A^{-1} & -A^{-1}\Pi B^{-1} \\ & B^{-1} \end{pmatrix} \in \zeta .$$

Queda demostrado, por tanto, el siguiente teorema.

Teorema 5.2: *El conjunto* ζ *junto con la multiplicación definida en* (5.1) *es un grupo.*

Además, se puede verificar fácilmente que no es un grupo abeliano.

Veamos ahora la descripción del sistema.

Dado el subgrupo generado por un elemento conocido públicamente

$$M = \begin{pmatrix} A & \Pi \\ & B \end{pmatrix} \in \zeta ,$$

el problema consiste en encontrar el escalar $k \geq 1$ a partir únicamente de $\Pi^{(k)}$, que es el término obtenido de calcular la potencia *k*-ésima de *M*:

$$M^k = \begin{pmatrix} A & \Pi \\ & B \end{pmatrix}^k = \begin{pmatrix} A^k & \Pi^{(k)} \\ & B^k \end{pmatrix},$$

donde

$$\Pi^{(k)} = \sum_{i=0}^{k-1} A^{k-1-i} \Pi B^i \text{ para } k \geq 1, \tag{5.2}$$

y $\Pi^{(0)} = O_{r\times s}$.

En este punto debemos observar que, a diferencia de lo que ocurría en el problema considerado en el capítulo 4, no es posible extraer la matriz Π como factor común ya que el producto de estos elementos no es conmutativo.

Posteriormente veremos cómo escoger las matrices A y B de forma que M genere un subgrupo con el mayor número de elementos posible.

Los elementos A^k y B^k se excluyen del problema para no reducirlo a un sistema semejante al del logaritmo discreto con matrices, problema ya tratado en [64]. Por otra parte, esta exclusión genera un inconveniente, ya que si bien M funciona como elemento generador del subgrupo y por tanto único para valores de k comprendidos en su orden, en cambio $\Pi^{(k)}$ no es único para esos valores de k y se repite de forma aparentemente aleatoria. Posteriormente analizaremos estas repeticiones y veremos que esto no supone una desventaja frente al ECDLP.

El siguiente teorema proporciona una herramienta útil para realizar cálculos en este grupo.

Teorema 5.3: *Consideremos el elemento* $\Pi^{(k)}$ *definido en* (5.2), *si* $0 \leq t \leq k$ *entonces*

$$\Pi^{(k)} = A^t \Pi^{(k-t)} + \Pi^{(t)} B^{k-t}. \tag{5.3}$$

Demostración: Dado $M \in \zeta$, tenemos que

$$M^t M^{k-t} = M^k,$$

esto es,

$$\begin{pmatrix} A^t & \Pi^{(t)} \\ & B^t \end{pmatrix} \begin{pmatrix} A^{k-t} & \Pi^{(k-t)} \\ & B^{k-t} \end{pmatrix} = \begin{pmatrix} A^k & \Pi^{(k)} \\ & B^k \end{pmatrix},$$

y por tanto

$$A^t \Pi^{(k-t)} + \Pi^{(t)} B^{k-t} = \Pi^{(k)}. \qquad \blacksquare$$

5.2 Orden de los elementos

Para que la variabilidad de la matriz $\Pi^{(k)}$ sea máxima, el orden del elemento M debe ser máximo. Por una parte, sabemos que p es primo, por tanto $E(\mathbb{F}_q) \cong \mathbb{Z}_p$ el grupo aditivo de $\mathbb{F}_p$. Este isomorfismo es razonablemente imposible de calcular para los valores usuales de p, pero sabemos que existe y podemos aprovechar sus propiedades. Vamos a considerar, por tanto, que el elemento M está formado por tres matrices de escalares:

$$M = \begin{pmatrix} A & \tilde{\Pi} \\ & B \end{pmatrix},$$

con los elementos de $\tilde{\Pi}$ en $\mathbb{Z}_p$. Viéndolo de esta forma, sabemos que (véase [46]):

$$\mathrm{o}(M) = \mathrm{mcm}\big(\mathrm{o}(A), \mathrm{o}(B)\big).$$

Por otra parte, para obtener una matriz de orden máximo, debemos usar la matriz asociada a un polinomio primitivo, esto es, sea

$$f(x) = x^n + d_{n-1}x^{n-1} + \cdots + d_1 x + d_0$$

un polinomio primitivo con $d_i \in \mathbb{Z}_p$. La matriz asociada a $f(x)$ es la matriz de tamaño $n \times n$ dada por:

$$D = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -d_0 & -d_1 & -d_2 & \cdots & -d_{n-2} & -d_{n-1} \end{pmatrix},$$

y su orden es $p^n - 1$. Por tanto

$$D^{p^n-1} \equiv I_{n\times n} \pmod{p}.$$

Es posible evitar que D tenga esta representación con casi todos sus elementos nulos mediante una sencilla operación: dada una matriz invertible H, la matriz HDH^{-1} es semejante a D, esto es, mantiene sus propiedades fundamentales, y entre ellas su orden (véase [55]).

En conclusión, si tomamos las matrices A y B del sistema de la forma vista, podemos asegurar que

$$o(M) = \mathrm{mcm}(p^r - 1, p^s - 1).$$

Ahora debemos hacer máximo el mínimo común múltiplo anterior. Sabemos por teoría de cuerpos ciclotómicos y raíces de la unidad que el polinomio $x^n - 1$ es divisible por $x^d - 1$ si $d \mid n$ (véase [55]); por tanto si tomamos r y s de manera que sean primos entre sí, minimizaremos el número de divisores comunes y el $\text{mcm}(p^r - 1, p^s - 1)$ será máximo.

Es usual en criptografía tomar valores grandes de p, del orden de cientos de dígitos en binario. En estos casos, es posible incrementar enormemente el tamaño del problema tomando valores de r y s muy pequeños. Por ejemplo, para tamaños de p del orden de 100, 160 y 200 bits, y para diferentes valores de r y s, podemos ver la tabla 5.1 que muestra el tamaño de M.

Tabla 5.1: Tamaños de M

r	s	$p \approx 2^{100}$	$p \approx 2^{160}$	$p \approx 2^{200}$
2	3	2^{400}	2^{640}	2^{800}
3	4	2^{600}	2^{960}	$2^{1.200}$
3	5	2^{700}	$2^{1.120}$	$2^{1.400}$
4	5	2^{800}	$2^{1.280}$	$2^{1.600}$
5	6	$2^{1.000}$	$2^{1.600}$	$2^{2.000}$

Más aún, también es posible alcanzar grandes tamaños de problema con valores muy pequeños de p si r y s toman valores relativamente grandes. Por ejemplo, podemos ver algunos casos en la tabla 5.2, que contiene los tamaños resultantes de M para diferentes valores de p, r y s.

Tabla 5.2: Tamaños de M

r	s	$p = 5$	$p = 13$	$p = 29$
31	32	2^{145}	2^{230}	2^{302}
47	48	2^{219}	2^{348}	2^{457}
60	61	2^{279}	2^{445}	2^{584}
130	131	2^{605}	2^{963}	$2^{1.264}$
216	217	$2^{1.004}$	$2^{1.599}$	$2^{2.099}$

La elección de los parámetros p, r y s debe hacerse siempre sin comprometer la seguridad de la curva elíptica. Debemos tener en cuenta que el grupo de puntos de una curva elíptica, como vimos en el capítulo 2, es isomorfo a $\mathbb{Z}_p$. Esto no supone un problema ya que establecer el isomorfismo es inviable para los tamaños de p usuales, sin embargo para valores pequeños de p se podría realizar de forma eficiente y el sistema

acabaría como un DLP sobre matrices, que se puede reducir a varios DLP sobre pequeñas extensiones de $\mathbb{F}_q$ (véase [64]).

Dado el orden de M, debemos considerar, además, el tamaño de los elementos que aparecen en el problema. Por una parte, el tamaño del entero k usado como elemento a encontrar es igual al orden de M. Por otra, el elemento $\Pi^{(k)}$ que determina el problema consiste en una matriz de tamaño $r \times s$ cuyos elementos son puntos de $E(\mathbb{F}_q)$. El tamaño de un punto de la curva elíptica consiste en el tamaño de una de sus coordenadas (usualmente x), ya que la otra (usualmente y) se suele obtener resolviendo la ecuación que define la curva. La elección de las dos componentes y resultantes de la ecuación suele dirimirse mediante un bit añadido a los bits de x. Por tanto, el tamaño de un punto tendrá un bit más que q. Esto determina finalmente que $\Pi^{(k)}$ estará formado por $\left(\lceil \log_2 q \rceil + 1\right) rs$ bits.

Junto a estas consideraciones, para elegir correctamente el tamaño de p, debemos tener en cuenta el ataque Pohlig-Hellman (visto en la sección 2.10), el cual establece básicamente que es posible reducir el problema del logaritmo discreto definido sobre un grupo finito, a varios problemas del logaritmo discreto de menor tamaño determinados por los factores que componen el cardinal del grupo original. Por ello, los cardinales de los grupos usados en estos problemas suelen ser primos o contener un factor primo grande. En el sistema planteado, el orden de M no es primo ya que viene determinado por un múltiplo. El factor primo más grande que se puede obtener será $p^r - 1$ ó $p^s - 1$ puesto que es posible hacer que estos elementos sean primos (como los primos de Mersenne para $p = 2$, por ejemplo). Debido a que la complejidad de resolución del problema vendrá determinada por el factor primo más grande que posea, se debe optimizar la elección de p, r y s de forma que uno de los factores sea lo más grande posible y el otro, que ya no debe ser necesariamente primo, lo menor posible. De esta forma se minimizará el número de puntos y por tanto de bits usados para $\Pi^{(k)}$, manteniendo el máximo nivel de complejidad posible.

5.3 Análisis y distribución empírica

El estudio experimental del sistema criptográfico no lineal de curva elíptica visto en el capítulo 4 era necesario por la aparente aleatoriedad del sistema, producida por la exponenciación discreta. En el problema tratado en este capítulo, la exponenciación se

entremezcla con la matriz de puntos de la curva, de forma que aumenta su errático comportamiento. Además, ni siquiera se dispone de una expresión equivalente para la submatriz $\Pi^{(k)}$ como ocurría en el sistema anterior. Con mayor motivo, por tanto, el estudio empírico se muestra como la alternativa a escoger, ya que su caracterización se muestra aún más inviable. Los resultados de este estudio, como veremos, son similares a los obtenidos en el sistema del capítulo 4.

Al hacer la descripción del sistema comentamos que la matriz $\Pi^{(k)}$ no era única a lo largo de todo el intervalo de elementos distintos $M \in \zeta$, esto es, para todo $0 < m,k < o(M)$, con $m \neq k$, es cierto que $M^m \neq M^k$, sin embargo, en algunos casos puede ocurrir que $\Pi^{(m)} = \Pi^{(k)}$. Se obtienen por tanto repeticiones en la matriz que usamos como problema a resolver. Esto produce una merma en la dureza del sistema aunque, como veremos, muy pequeña.

Para obtener el mayor número posible de matrices distintas $\Pi^{(k)}$, debemos maximizar el orden del elemento conocido *M*. Según hemos visto, el orden de un elemento depende únicamente de las matrices *A* y *B*. Por tanto, para contabilizar el número de matrices $\Pi^{(k)}$ que podemos obtener, es indiferente la matriz inicial que tomemos, con la excepción de $\Pi = O_{r\times s}$, ya que en este caso todas las matrices $\Pi^{(k)}$ serían iguales a $O_{r\times s}$.

No ocurre lo mismo cuando consideramos los polinomios primitivos elegidos para crear las matrices *A* y *B* que componen *M*. Dependiendo de estos polinomios, en algunos casos podemos obtener un número diferente de repeticiones de la matriz $\Pi^{(k)}$. Sin embargo, de acuerdo con las pruebas realizadas, no se observan cambios importantes en la cantidad de repeticiones producidas según el polinomio escogido.

La media aritmética de la cantidad de matrices $\Pi^{(k)} \in \mathcal{M}at_{r\times s}\left(E(\mathbb{F}_q)\right)$, con $k < o(M)$, que no aparecen ninguna vez se acerca al 36,8%. Este valor es aproximadamente el mismo que el porcentaje de matrices que aparecen sólo una vez. La cantidad de matrices diferentes que aparecen dos veces se aproxima al 18,4%, esto es, la mitad. La cantidad de matrices diferentes que aparecen tres veces se acerca al 6,1%, es decir, la tercera parte de las que aparecen dos veces, y así sucesivamente. Estos cálculos coinciden aproximadamente con los obtenidos en el sistema criptográfico no lineal del capítulo anterior.

Obtenemos de esta forma la función de cuantía:

$$f(x) = P(X = x) = \frac{0{,}368}{x!}x,$$

donde *x* es el número de repeticiones que la variable aleatoria *X* puede tomar.

Esta conjetura nos lleva a obtener la máxima cantidad de repeticiones que pueden darse de acuerdo con el tamaño de $o(M)$. Para que ocurra un cierto número de

repeticiones es necesario que exista, al menos, una matriz que pueda tomar ese número de repeticiones, esto es,

$$f(x)\cdot \mathrm{o}(M) = \frac{0,368}{x!} x \cdot \mathrm{o}(M) \geq 1,$$

por tanto

$$0,368 \cdot \mathrm{o}(M) \geq (x-1)!.$$

Por ejemplo, si $\mathrm{o}(M)$ tiene 50 bits, es decir, $\lceil \log_2 \mathrm{o}(M) \rceil = 50$, entonces $(x-1)! \leq 4,1433 \times 10^{14}$, por tanto $x \leq 18$ y éste será el número máximo esperado de veces que una matriz $\Pi^{(k)}$ puede aparecer.

Aunque los porcentajes pueden contener un error elevado (y por tanto las probabilidades calculadas), el número de veces que una matriz puede repetirse no variará significativamente.

Las matrices $\Pi^{(k)}$ que nos interesan son las que aparecen una vez porque una búsqueda exhaustiva obligaría a recorrer todas las posibles matrices $\Pi^{(k)}$ para $k < \mathrm{o}(M)$. Si la matriz $\Pi^{(k)}$ aparece dos veces, la probabilidad nos llevaría a recorrer la mitad de las matrices posibles, y así sucesivamente.

Para valores prácticos de p, por ejemplo, $p \approx 2^{160}$, esto es, 160 bits, $r = 2$ y $s = 3$, entonces $\mathrm{o}(M) \approx 2^{640}$ y k podría tomar un valor aproximado de 640 bits. En este caso $\Pi^{(k)}$ podría tomar el mismo valor 117 veces como máximo, y el problema equivalente sería de tamaño $2^{640}/117 \approx 2^{633}$ pero sin repeticiones. En la práctica no se utiliza la búsqueda exhaustiva puramente, sino que se procede con un ataque de raíz cuadrada, porque éste siempre es aplicable a un grupo. En consecuencia, si inicialmente el problema tenía un tamaño de $\sqrt{\mathrm{o}(M)} \approx 2^{320}$, entonces el problema equivalente será de $2^{320}/\sqrt{117} \approx 2^{316}$, es decir, una reducción en la dureza de sólo 4 bits.

De la misma forma que en el sistema anterior, debemos calcular la esperanza de la variable aleatoria X para conocer la reducción esperada en el tamaño del problema. Debemos considerar que los cálculos realizados son para el peor caso posible donde tengamos una matriz $\Pi^{(k)}$ en la que se dé el número máximo de repeticiones, y esto es razonablemente imposible.

El valor esperado del número de veces que una matriz puede aparecer es:

$$E(X) = \sum_i x_i f(x_i) = \sum_i \frac{0,368}{x_i!} x_i^2 = 2.$$

Ahora procedemos a realizar el cálculo anterior, pero actualizando el número máximo de repeticiones al número de repeticiones esperado. De esta forma, obtendremos un tamaño de problema con una reducción casi imperceptible: $2^{320}/\sqrt{2} \approx 2^{319,6}$.

Se proporciona una muestra de los cálculos realizados desde la tabla 5.3 hasta la 5.9. Cada tabla representa el número de veces que se repiten las matrices $\Pi^{(k)}$ para diferentes valores de r y s. Además, cada tabla está subdividida según el valor de p usado. Los polinomios $f(x)$ y $g(x)$ están representados por sus coeficientes, esto es, la secuencia $1\ a_{n-1}\ a_{n-2}\ \cdots\ a_1\ a_0$ representa el polinomio $x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$. El máximo número de matrices $\Pi^{(k)}$ que pueden aparecer es $o(M)$. Para este número de posibles matrices (y por tanto para r, s y p), y para diferentes polinomios primitivos $f(x)$ y $g(x)$, la tablas muestran la cantidad de matrices que no aparecen ninguna vez, la cantidad que aparece una vez, la cantidad que aparece dos veces, y así sucesivamente hasta alcanzar el número máximo de veces que una matriz puede aparecer (en este caso, sólo 9). Por ejemplo, para $p = 29$, con $f(x) = x^2 + x + 3$ y $g(x) = x^3 + x + 18$, la tabla 5.3 para $r = 2$ y $s = 3$ muestra que 270.563 matrices no aparecen ninguna vez, 265.482 matrices aparecen una vez, 136.619 matrices aparecen dos veces, y así de forma sucesiva.

Tabla 5.3: Veces que se repiten las matrices $\Pi^{(k)}$ para $r = 2$, $s = 3$

f(x)	*g(x)*	0	1	2	3	4	5	6	7	8	9
p = 5, *o(M)* = 744											
1 1 2	1 1 0 2	279	249	165	39	12					
1 4 2	1 3 2 3	277	267	123	77						
p = 7, *o(M)* = 2.736											
1 1 3	1 1 1 2	1.017	970	526	184	33	6				
1 6 3	1 6 4 4	1.005	1.057	343	331						
p = 11, *o(M)* = 15.960											
1 1 7	1 1 0 5	5.913	5.721	3.087	951	228	60				
1 4 2	1 1 0 5	5.764	6.070	2.805	1.035	255	31				
1 5 2	1 1 0 3	5.961	5.634	3.072	1.026	243	12	12			
1 8 8	1 1 0 3	5.715	6.723	1.329	2.193						
p = 13, *o(M)* = 30.744											
1 1 2	1 1 0 7	11.515	10.819	5.893	1.989	474	48	6			
1 3 7	1 1 0 7	11.397	11.189	5.479	2.197	404	78				
1 4 6	1 1 0 2	11.427	11.015	5.749	2.131	272	150				
1 4 11	1 1 0 2	10.915	13.273	2.197	4.359						
p = 17, *o(M)* = 88.416											
1 1 3	1 0 1 14	31.001	39.459	4.911	13.045						
1 5 7	1 0 1 14	32.898	31.542	16.980	5.319	1.479	147	51			
1 2 6	1 1 0 7	32.951	31.368	17.166	5.284	1.407	210	24	6		
1 2 12	1 1 0 7	32.821	31.924	16.141	6.365	710	455				
p = 19, *o(M)* = 137.160											
1 1 2	1 1 0 16	51.029	49.105	25.876	8.705	2.106	288	33	18		
1 1 3	1 1 0 16	47.873	61.921	6.859	20.507						
1 6 14	1 1 0 16	50.981	49.264	25.723	8.690	2.196	261	24	21		
1 1 2	1 1 0 6	50.771	49.619	25.731	8.633	1.934	394	72	6		
1 10 3	1 1 0 6	50.885	49.691	24.759	10.067	1.040	718				
1 7 14	1 1 0 6	50.930	49.307	25.929	8.375	2.261	322	36			

f(x)	*g(x)*	0	1	2	3	4	5	6	7	8	9
$p = 23$, $o(M) = 291.984$											
1 1 7	1 1 0 16	101.191	134.115	12.165	44.513						
1 2 5	1 1 0 16	108.199	105.312	55.296	17.735	4.449	888	96	9		
1 5 21	1 1 0 16	108.498	104.840	55.114	18.129	4.579	749	63	6	6	
1 4 7	1 1 0 6	108.536	104.722	55.087	18.448	4.391	647	126	27		
1 16 21	1 1 0 6	108.378	105.008	55.168	18.141	4.348	836	90	15		
1 3 11	1 1 0 6	107.933	106.222	54.354	17.806	4.787	804	78			
$p = 29$, $o(M) = 731.640$											
1 1 3	1 0 1 18	270.563	265.482	136.619	45.542	11.151	2.051	189	43		
1 7 2	1 0 1 18	270.981	264.867	136.617	46.020	10.599	2.277	255	24		
1 22 15	1 0 1 18	271.152	263.976	138.606	43.341	13.071	819	675			
1 5 2	1 1 0 3	269.695	267.120	137.235	43.064	12.131	2.037	357	1		
1 2 3	1 1 0 3	269.441	271.273	126.897	52.471	8.630	2.928				
1 5 10	1 1 0 3	270.940	265.103	136.125	46.371	10.717	2.046	302	24	12	
$p = 31$, $o(M) = 953.280$											
1 1 12	1 0 1 28	352.524	344.285	180.655	59.055	13.840	2.516	315	60	0	30
1 2 3	1 0 1 28	327.349	447.361	29.791	148.779						
1 30 24	1 0 1 28	352.733	346.127	177.115	59.831	14.432	2.598	396	48		
1 23 3	1 1 0 9	352.604	346.130	178.186	58.082	15.206	2.724	348			
$p = 37$, $o(M) = 1.924.776$											
1 1 5	1 1 0 24	711.119	701.506	357.007	117.938	31.557	4.830	687	114	18	
1 1 5	1 1 0 17	711.546	700.092	357.902	119.428	29.521	5.303	897	81	6	
$p = 41$, $o(M) = 2.894.640$											
1 1 12	1 0 1 35	1.070.583	1.049.892	545.670	169.059	53.976	2.433	3.027			
$p = 43$, $o(M) = 3.498.264$											
1 1 3	1 0 1 40	1.191.945	1.670.593	79.507	556.219						
$p = 47$, $o(M) = 4.983.456$											
1 1 13	1 1 0 42	1.841.058	1.815.442	924.856	310.126	74.858	14.630	2.102	348	30	6

Tabla 5.4: Veces que se repiten las matrices $\Pi^{(k)}$ para $r = 2$, $s = 5$

f(x)	*g(x)*	0	1	2	3	4	5	6
$p = 5$, $o(M) = 18.744$								
1 1 2	1 0 0 1 0 2	7.082	6.212	4.063	1.162	205	20	
1 2 3	1 0 0 0 4 2	7.064	6.296	3.929	1.240	205	10	
$p = 7$, $o(M) = 134.448$								
1 1 3	1 1 0 0 0 4	50.795	45.720	27.276	8.737	1.635	285	
1 2 5	1 1 0 0 0 4	50.555	46.143	27.150	8.630	1.755	195	20
$p = 11$, $o(M) = 1.932.600$								
1 1 7	1 0 1 1 0 9	722.777	679.879	374.183	123.534	28.097	3.415	715

Tabla 5.5: Veces que se repiten las matrices $\Pi^{(k)}$ para $r = 2$, $s = 7$

f(x)	*g(x)*	0	1	2	3	4	5
$p = 5$, $o(M) = 468.744$							
1 1 2	1 1 0 0 0 0 0 2	17.6850	156.939	98.625	31.059	4.977	294
1 2 3	1 1 0 0 0 0 0 2	17.7016	156.427	99.123	30.977	4.865	336

Tabla 5.6: Veces que se repiten las matrices $\Pi^{(k)}$ para $r = 3$, $s = 4$

f(x)	*g(x)*	0	1	2	3	4	5	6	7
$p = 5$, $o(M) = 19.344$									
1 1 0 2	1 1 0 1 3	7.521	6.086	4.227	1.236	274			
1 0 3 2	1 0 1 2 2	7.461	6.278	3.987	1.404	190	24		
$p = 7$, $o(M) = 136.800$									
1 1 1 2	1 1 1 0 3	52.109	46.049	27.477	9.145	1.750	258	12	
1 0 3 2	1 0 1 3 5	52.385	44.954	29.400	7.198	2.863			
$p = 11$, $o(M) = 1.947.120$									
1 1 0 5	1 0 0 1 2	732.663	677.966	378.316	125.540	27.805	4.334	460	36

Tabla 5.7: Veces que se repiten las matrices $\Pi^{(k)}$ para $r = 3$, $s = 5$

f(x)	*g(x)*	0	1	2	3	4	5	6
$p = 5$, $o(M) = 96.844$								
1 1 0 2	1 0 0 1 0 2	37.437	31.206	20.155	6.916	1.070	60	
1 0 3 2	1 0 0 0 4 3	37.819	30.088	20.976	7.040	921		
$p = 7$, $o(M) = 957.942$								
1 1 1 2	1 1 0 0 0 4	365.279	322.200	191.707	64.666	12.210	1.790	90

Tabla 5.8: Veces que se repiten las matrices $\Pi^{(k)}$ para $r = 3$, $s = 7$

f(x)	*g(x)*	0	1	2	3	4	5
$p = 5$, $o(M) = 2.421.844$							
1 1 0 2	1 1 0 0 0 0 0 2	938.851	773.800	510.000	170.240	27.441	1.512

Tabla 5.9: Veces que se repiten las matrices $\Pi^{(k)}$ para $r = 4$, $s = 5$

$f(x)$	$g(x)$	0	1	2	3	4	5
$p = 5$, $o(M) = 487.344$							
1 1 0 1 3	1 0 0 1 0 2	189.818	154.350	102.918	34.234	5.664	360

5.4 Sistemas criptográficos

Como comentamos en los sistemas anteriores, no todos los problemas matemáticos son útiles en criptografía de clave pública. Veremos ahora el sistema propuesto aplicado al protocolo de intercambio de claves de Diffie-Hellman y al esquema de cifrado de ElGamal, y analizaremos la falta de aplicabilidad al resto de esquemas vistos en anteriores capítulos.

5.4.1 Protocolo de intercambio de claves de Diffie-Hellman

Consideremos el subgrupo generado por el elemento

$$M = \begin{pmatrix} A & \Pi \\ & B \end{pmatrix} \in \zeta ,$$

donde ζ representa el grupo definido en el presente sistema. Denotamos con $E/\mathbb{F}_q$ la curva sobre la que está definido el grupo ζ, con $p = \#E(\mathbb{F}_q)$ un número primo. Toda esta información es pública.

Alicia y Bernardo desean compartir un secreto. Para ello:

1. Alicia toma una clave privada k, con $1 \le k < \mathrm{o}(M)$ de forma aleatoria, calcula

$$M^k = \begin{pmatrix} A^k & \Pi^{(k)} \\ & B^k \end{pmatrix},$$

de la forma vista en la definición del sistema, y transmite $\Pi^{(k)}$ a Bernardo.

2. De forma similar, Bernardo toma aleatoriamente una clave secreta m, con $1 \le m < \mathrm{o}(M)$, calcula

$$M^m = \begin{pmatrix} A^m & \Pi^{(m)} \\ & B^m \end{pmatrix},$$

y transmite $\Pi^{(m)}$ a Alicia.

3. Entonces, Alicia calcula

$$\begin{pmatrix} A & \Pi^{(m)} \\ & B \end{pmatrix}^k = \begin{pmatrix} A^k & \left(\Pi^{(m)}\right)^{(k)} \\ & B^k \end{pmatrix}$$

y considera el elemento $\left(\Pi^{(m)}\right)^{(k)}$.

4. Bernardo, igualmente, calcula

$$\begin{pmatrix} A & \Pi^{(k)} \\ & B \end{pmatrix}^m = \begin{pmatrix} A^m & \left(\Pi^{(k)}\right)^{(m)} \\ & B^m \end{pmatrix}$$

y considera el elemento $\left(\Pi^{(k)}\right)^{(m)}$.

El siguiente teorema verifica el funcionamiento del protocolo.

Teorema 5.4: *Con la notación anterior, se cumple que*

$$\left(\Pi^{(m)}\right)^{(k)} = \left(\Pi^{(k)}\right)^{(m)}.$$

Demostración: Cuando hicimos la descripción del sistema, vimos que

$$\Pi^{(k)} = \sum_{i=0}^{k-1} A^{k-1-i}\Pi B^i,$$

por tanto podemos hacer

$$\begin{aligned}
\left(\Pi^{(m)}\right)^{(k)} &= \sum_{i=0}^{k-1} A^{k-1-i}\Pi^{(m)}B^i \\
&= A^{k-1}\Pi^{(m)} + A^{k-2}\Pi^{(m)}B + \cdots + A\Pi^{(m)}B^{k-2} + \Pi^{(m)}B^{k-1} \\
&= A^{k-1}\left(\sum_{i=0}^{m-1} A^{m-1-i}\Pi B^i\right) + A^{k-2}\left(\sum_{i=0}^{m-1} A^{m-1-i}\Pi B^i\right)B + \cdots \\
&\qquad \cdots + A\left(\sum_{i=0}^{m-1} A^{m-1-i}\Pi B^i\right)B^{k-2} + \left(\sum_{i=0}^{m-1} A^{m-1-i}\Pi B^i\right)B^{k-1} \\
&= \sum_{i=0}^{m-1} A^{k+m-2-i}\Pi B^i + \sum_{i=0}^{m-1} A^{k+m-3-i}\Pi B^{i+1} + \cdots + \sum_{i=0}^{m-1} A^{m-i}\Pi B^{i+k-2} + \sum_{i=0}^{m-1} A^{m-1-i}\Pi B^{i+k-1}
\end{aligned}$$

$$
\begin{aligned}
&= A^{k+m-2}\Pi + A^{k+m-3}\Pi B + A^{k+m-4}\Pi B^2 + \cdots + A^{k}\Pi B^{m-2} + A^{k-1}\Pi B^{m-1} + \\
&\quad + A^{k+m-3}\Pi B + A^{k+m-4}\Pi B^2 + \cdots + A^{k}\Pi B^{m-2} + A^{k-1}\Pi B^{m-1} + A^{k-2}\Pi B^{m} + \\
&\quad + \cdots + \\
&\quad + A^{m}\Pi B^{k-2} + A^{m-1}\Pi B^{k-1} + \cdots + A\Pi B^{k+m-3} + \\
&\quad + A^{m-1}\Pi B^{k-1} + A^{m-2}\Pi B^{k} + \cdots + \Pi B^{k+m-2} \\
&= A^{k+m-2}\Pi + 2A^{k+m-3}\Pi B + 3A^{k+m-4}\Pi B^2 + \cdots + 3A^2\Pi B^{k+m-4} + 2A\Pi B^{k+m-3} + \Pi B^{k+m-2}.
\end{aligned}
$$

La distribución de los coeficientes que aparecen como resultado de esta expresión puede obtenerse de la siguiente forma:

Si $k < m$ tenemos la serie de coeficientes:

$$\underbrace{1,2,3,4,\ldots,k-2,k-1,}_{k-1}\ \underbrace{k,k,\ldots,k,}_{m-k+1}\ \underbrace{k-1,k-2,\ldots,4,3,2,1}_{k-1}$$

Si $k > m$ tenemos la serie de coeficientes:

$$\underbrace{1,2,3,4,\ldots,m-2,m-1,}_{m-1}\ \underbrace{m,m,\ldots,m,}_{k-m+1}\ \underbrace{m-1,m-2,\ldots,4,3,2,1}_{m-1}$$

Si $k = m$ la serie de coeficientes resultante es:

$$\underbrace{1,2,3,4,\ldots,k-2,k-1,}_{k-1}\ \underbrace{k,}_{1}\ \underbrace{k-1,k-2,\ldots,4,3,2,1}_{k-1}$$

Como las series son iguales intercambiando k por m, podemos suponer sin pérdida de generalidad que $k \le m$. Ahora podemos simplificar la expresión anterior obteniendo

$$\left(\Pi^{(m)}\right)^{(k)} = \sum_{i=0}^{k-2}(i+1)A^{k+m-2-i}\Pi B^{i} + kA^{k-1}\Pi^{(k+1)}B^{k-1} + \sum_{i=0}^{k-2}(i+1)A^{i}\Pi B^{k+m-2-i},$$

y por tanto

$$\left(\Pi^{(m)}\right)^{(k)} = \sum_{i=0}^{k-2}(i+1)\left(A^{k+m-2-i}\Pi B^{i} + A^{i}\Pi B^{k+m-2-i}\right) + kA^{k-1}\Pi^{(k+1)}B^{k-1}.$$

Al desarrollar $\left(\Pi^{(k)}\right)^{(m)}$ llegaremos a la misma expresión tanto si $k \le m$ como si $k > m$ ya que ambos términos conmutan respecto a la suma, de ahí que

$$\left(\Pi^{(m)}\right)^{(k)} = \left(\Pi^{(k)}\right)^{(m)}$$

para cualquier k y m, quedando demostrado el teorema. ■

El secreto compartido por Alicia y Bernardo es por tanto $\left(\Pi^{(m)}\right)^{(k)} = \left(\Pi^{(k)}\right)^{(m)}$.

Si comparamos el secreto compartido de este sistema con el del sistema del capítulo 4, comprobaremos que el entrelazamiento que se produce entre las matrices debido a la no conmutatividad de la multiplicación de las mismas, impide extraer como factor común la

matriz de puntos Π, de forma que la única opción que parece viable para atacar el sistema es la de encontrar alguna de las claves privadas mediante la búsqueda exhaustiva.

5.4.2 Esquema de cifrado de ElGamal

En el esquema de cifrado de ElGamal hemos de partir de los mismos elementos públicos que los del protocolo de intercambio de claves de Diffie-Hellman.

Alicia desea enviar un mensaje de forma privada a Bernardo. El mensaje debe codificarse como una matriz Ψ de tamaño $r \times s$ cuyos elementos pertenecen al grupo de puntos $E(\mathbb{F}_q)$ de la curva elíptica (recordemos que r y s son los tamaños de la matrices cuadradas A y B, respectivamente). La técnica más apropiada para codificar el mensaje, normalmente alfanumérico, en una matriz de tales características, parece ser la de Koblitz vista en la sección 2.11, sólo que en este caso habría que dividir el tamaño q del cuerpo donde está definida la curva en $r \cdot s$ partes para poder multiplexar los puntos de forma equitativa.

El esquema consiste en:

1. En primer lugar, Bernardo toma aleatoriamente un entero m, con $1 \le m < \mathrm{o}(M)$, calcula

$$M^m = \begin{pmatrix} A^m & \Pi^{(m)} \\ & B^m \end{pmatrix},$$

y envía a Alicia su clave pública, que es el elemento $\Pi^{(m)}$.

2. En segundo lugar, y de forma similar, Alicia toma un entero k de forma aleatoria, con $1 \le k < \mathrm{o}(M)$, calcula

$$M^k = \begin{pmatrix} A^k & \Pi^{(k)} \\ & B^k \end{pmatrix},$$

y extrae su clave pública $\Pi^{(k)}$.

3. Para obtener la clave de cifrado, Alicia calcula

$$\begin{pmatrix} A & \Pi^{(m)} \\ & B \end{pmatrix}^k = \begin{pmatrix} A^k & \left(\Pi^{(m)}\right)^{(k)} \\ & B^k \end{pmatrix}$$

y extrae el elemento $\left(\Pi^{(m)}\right)^{(k)}$ con el que cifra el mensaje Ψ haciendo

$$\Omega = \left(\Pi^{(m)}\right)^{(k)} + \Psi .$$

Tras ello, envía a Bernardo el mensaje cifrado Ω junto con su clave pública $\Pi^{(k)}$. Observar que ambos son matrices de puntos de tamaño $r \times s$.

4. Por último, Bernardo obtiene la clave de descifrado mediante el cálculo de

$$\begin{pmatrix} A & \Pi^{(k)} \\ & B \end{pmatrix}^m = \begin{pmatrix} A^m & \left(\Pi^{(k)}\right)^{(m)} \\ & B^m \end{pmatrix}$$

y la extracción del elemento $\left(\Pi^{(k)}\right)^{(m)}$, y descifra el mensaje haciendo

$$\Psi' = \Omega - \left(\Pi^{(k)}\right)^{(m)}.$$

Se comprueba que

$$\left(\Pi^{(m)}\right)^{(k)} = \left(\Pi^{(k)}\right)^{(m)}$$

de la misma forma que en el protocolo de Diffie-Hellman y, por tanto,

$$\Psi' = \Omega - \left(\Pi^{(k)}\right)^{(m)} = \Omega - \left(\Pi^{(m)}\right)^{(k)} = \Omega - (\Omega - \Psi) = \Psi.$$

5.4.3 Disertación sobre los esquemas de firma digital

Después de analizar los esquemas de firma digital sobre este sistema como problema matemático subyacente, hemos llegado a la conclusión de que su aplicación a los mismos no parece viable. Veamos por qué.

El esquema de firma digital de ElGamal sobre curvas elípticas está basado en una expresión del tipo

$$w \equiv ut + kf(\Pi^{(t)}) \pmod{\mathrm{o}(M)}, \tag{5.4}$$

siendo todos los términos enteros, y donde $\mathrm{o}(M)$ es el cardinal del grupo usado en el problema; w es el mensaje a firmar; t y k son sendas claves privadas de Alicia; $\Pi^{(t)}$ es la clave pública correspondiente a t; $f(\Pi^{(t)})$ es una función que asigna un entero a una clave pública; y u es el entero que verifica la ecuación.

Es posible crear una expresión de este tipo utilizando el sistema propuesto. Codificar el mensaje a transmitir w como entero menor que $\mathrm{o}(M)$ es una tarea sencilla. A partir de t y k deben obtenerse las claves públicas $\Pi^{(t)}$ y $\Pi^{(k)}$, respectivamente, de la misma forma que en los esquemas vistos en los apartados anteriores. La función f se puede definir de varias formas válidas. La más sencilla consiste en concatenar las coordenadas x de los puntos que componen la matriz $\Pi^{(t)}$, es decir, obtener un entero de la forma

$$x_{11} \| x_{12} \| \cdots \| x_{1u} \| x_{21} \| \cdots\cdots \| x_{tu}$$

que como máximo alcanzará el valor q^{tu}. Como dicho resultado debe hacerse módulo $o(M)$, entonces tendremos que existirán puntos con la misma imagen, pero la probabilidad de utilizar dos puntos cuya imagen coincida es de $1/o(M)$, lo cual es imposible en la práctica. Una vez obtenidos todos los términos que aparecen en la expresión (5.4), se obtiene el valor de u.

La firma de Alicia para el mensaje w estará compuesta por u y $\Pi^{(t)}$.

Para verificar la firma, Bernardo calcula las expresiones $\Delta_1 = f(\Pi^{(t)})\Pi^{(k)} + u\Pi^{(t)}$ y $\Delta_2 = \Pi^{(w)}$, y verifica la firma si $\Delta_1 = \Delta_2$.

El problema reside en que de la expresión

$$\Delta_1 = f(\Pi^{(t)})\Pi^{(k)} + u\Pi^{(t)} = f(\Pi^{(t)})\sum_{i=0}^{k-1} A^{k-1-i}\Pi B^i + t^{-1}\left(w - kf(\Pi^{(t)})\right)\sum_{i=0}^{t-1} A^{t-1-i}\Pi B^i ,$$

por la imposibilidad de extraer términos del sumatorio (k y t), parece inviable llegar a la expresión

$$\Delta_2 = \Pi^{(w)} = \sum_{i=0}^{w-1} A^{w-1-i}\Pi B^i .$$

Las expresiones necesarias en los otros esquemas vistos de firma digital son similares y el razonamiento se puede realizar de forma análoga, llegando a las mismas conclusiones. Por tanto, se descarta el uso de este sistema para el planteamiento de esquemas de firma digital.

El esquema de cifrado Massey-Omura es algo diferente. Recordemos que se utilizaba el propio mensaje como elemento base al que se le aplicaban las claves privadas mediante una comunicación continua entre Alicia y Bernardo. En este caso no era necesario extraer los términos del sumatorio, sólo era necesario anularlos para recuperar el mensaje cifrado como matriz. Esta operación, aunque más sencilla, no parece viable con el sistema propuesto. Aún siendo posible encontrar el inverso de los elementos que intervienen, la complejidad creada por el entrelazamiento de las matrices no permite aplicar la multiplicación de las inversas para eliminar las matrices A y B recuperando el mensaje.

5.5 Conclusiones

Si el sistema matricial no lineal de curva elíptica –el sistema visto en el capítulo 4– está basado fundamentalmente en la exponenciación discreta con un punto de una curva elíptica, el presente sistema va más allá y aumenta la seguridad al entremezclar dicha exponenciación con bloques de matrices, una de ellas formada con puntos de una curva elíptica, de forma que el coste de separar ambos problemas tenga una dureza similar al de romperlo.

Los sistemas conocidos con mayor dureza, es decir, los que requieren de algoritmos de ataque con mayor complejidad, son aquellos que sólo pueden ser atacados por fuerza bruta, por tanto los únicos ataques viables son los de raíz cuadrada. El ECDLP posee esta dureza, pero el DLP y el IFP no.

El sistema que hemos propuesto está basado en última instancia en el ECDLP para aprovechar la dureza intrínseca de éste. Utiliza un grupo cuyos elementos están diseñados de forma que intervengan dos problemas, la exponenciación de matrices sobre cuerpos finitos y las operaciones con curvas elípticas. De esta forma alcanza una complejidad similar a la del ECDLP, aunque mermada por el orden máximo alcanzable por un elemento del grupo y por el ataque Pohlig-Hellman. La ventaja estriba en que permite aumentar el tamaño de las claves tanto como se quiera mediante simples operaciones sobre los puntos de una curva elíptica, evitando los grandes requerimientos de cálculo que se precisarían para establecer el ECDLP con el mismo tamaño de claves.

Hemos considerado, por otra parte, el impacto que podrían tener las repeticiones del elemento que se utiliza como base del problema, verificando que no afecta de forma apreciable a la seguridad del mismo.

Por último, hemos aplicado este problema matemático a los esquemas criptográficos usuales de clave pública, comprobando su funcionalidad con el protocolo de intercambio de claves de Diffie-Hellman y con el esquema de cifrado de ElGamal. Sin embargo, no hemos podido aplicarlo a los esquemas de firma digital en general, y hemos argumentando los motivos.

Capítulo 6

Conclusiones y líneas futuras de investigación

En los últimos años, el auge imparable de Internet y de las comunicaciones en general ha hecho necesario el uso de la criptografía de forma cotidiana. La misma herramienta que ha permitido este desmesurado incremento de las comunicaciones, la informática en general, es la misma capaz de atacar sin cuartel la seguridad de éstas. Cada cierto tiempo aparecen nuevos sistemas que permiten romper con mayor celeridad los problemas matemáticos existentes. A la vez, el continuo avance en la velocidad de cálculo de los procesadores, junto con las técnicas de proceso conjunto colaborativo –en redes o en Internet– hacen que los sistemas que hoy son seguros no lo sean tanto en pocos años. Según el valor o vida útil de la información, será preciso un término temporal durante el cual se asegure la misma. No resulta nada fácil determinar hasta qué momento estará segura la información, por lo que es conveniente mantener siempre un margen de fiabilidad. Por otra parte, resulta que no es posible idear sistemas que aumenten la complejidad de los existentes ya que hay ataques –los de raíz cuadrada– que siempre son aplicables. Por tanto, la solución no parece estar en diseñar nuevos sistemas que requieran algoritmos de ataque más complejos, sino en aumentar el tamaño de los problemas para incrementar así la seguridad. De los sistemas criptográficos de clave pública conocidos, los de curva elíptica son los que mayor complejidad muestran, sólo atacables por algoritmos de raíz cuadrada, y por tanto de complejidad exponencial. A pesar de que en la mayoría de casos en el resto de sistemas –DLP o IFP, por ejemplo– es fácil aumentar los

tamaños de clave, usando curvas elípticas resulta una tarea difícil que requiere de grandes recursos computacionales.

Los sistemas que hemos presentado en esta memoria se han diseñado con el objetivo de lograr, en último término, esquemas criptográficos más seguros que los existentes. Para ello, hemos tratado de hacer más costosos de realizar los ataques de raíz cuadrada, bien aumentando el número de instancias necesarias para resolver el problema, bien aumentando el espacio de claves posible.

Los tres nuevos sistemas que hemos propuesto, vistos como funciones unidireccionales con trampa, funcionan perfectamente y cumplen todos los requisitos teóricos para su uso en criptografía de clave pública. La aplicabilidad desarrollada en los esquemas criptográficos usuales dan mejores resultados en unos sistemas que en otros. En concreto, con el primer problema –capítulo 3– se obtienen sistemas algo más seguros que el habitual de curvas elípticas. Además, se consigue aplicar a todos los esquemas criptográficos usuales de clave pública basados en el logaritmo discreto. Con el segundo sistema –capítulo 4– no se obtienen ventajas significativas como problema matemático, aunque sí funciona como problema subyacente del protocolo de intercambio de claves de Diffie-Hellman, quedando sólo como alternativa. El tercer sistema –capítulo 5– se muestra más intratable, aumentando la seguridad mediante el incremento del espacio de claves disponible. La aplicabilidad de este último a los esquemas criptográficos es desigual. Se consigue usar como problema base en el intercambio de claves y en el cifrado, no así en los esquemas de firma digital.

En el diseño de estos nuevos sistemas o funciones unidireccionales con trampa se han abierto muchas vías de investigación, unas relacionadas con estos sistemas y otras planteando otros nuevos. Respecto a los sistemas propuestos, resultaría interesante profundizar en el estudio de las repeticiones que se producen en el segundo y tercer sistema, problema estrechamente relacionado con el del logaritmo discreto, y probablemente reducible al mismo. Por otra parte, en el tercer sistema se debe establecer un equilibrio entre el tamaño de las matrices y el de la curva elíptica. Ello depende de la facilidad de encontrar el isomorfismo entre el grupo de puntos de una curva elíptica $E(\mathbb{F}_q)$, con $\#E(\mathbb{F}_q) = p$, y $\mathbb{Z}_p$. Aquí sería posible investigar usando la teoría de grupos adaptada a las especiales características de las curvas elípticas.

En cuanto a los protocolos y esquemas criptográficos, se podrían buscar modificaciones del tercer sistema que permitieran de alguna forma su aplicabilidad a los esquemas de firma digital. Asimismo, sería interesante adaptar otros protocolos, aunque sean menos conocidos, a los sistemas planteados, para así buscar nuevas características de las funciones unidireccionales con trampa que mejoren la adaptabilidad de las mismas a la criptografía de clave pública.

A partir de las características observadas en los sistemas propuestos, se pueden idear otros nuevos que podrían mejorar los expuestos. Sería posible diseñar sistemas de curva elíptica basados en matrices que combinaran no sólo escalares con puntos de una curva elíptica, sino también escalares con varias curvas elípticas isomorfas entre sí. Para ello, la n-tupla base del primer sistema, por ejemplo, podría estar formada por puntos de curvas elípticas diferentes pero isomorfas, relacionadas por un residuo cuadrático. Otros sistemas basados en matrices podrían combinar escalares con puntos, pero también puntos entre sí, no sólo utilizando la suma de puntos, sino también definiendo alguna operación adicional de la forma $\vartheta : E(\mathbb{F}_q) \times E(\mathbb{F}_q) \rightarrow E(\mathbb{F}_q)$.

Apéndice A

Consideraciones de implementación y pruebas

El presente apéndice se destina a la implementación práctica de los sistemas diseñados. No siempre los estudios teóricos conllevan obtener un modelo adaptable; es frecuente que aparezcan problemas en la implementación práctica que, a veces, pueden llegar a hacer la adaptación inviable. El diseño práctico de los sistemas propuestos en la presente memoria demuestra con casos genéricos que estos sistemas tienen aplicación práctica.

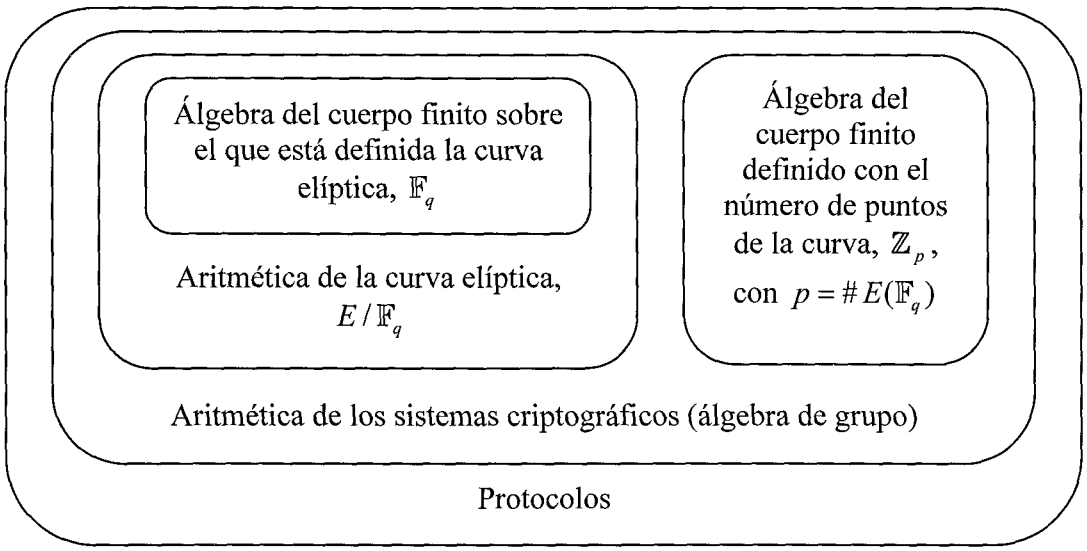
A.1 Consideraciones de implementación

Para la programación hemos utilizado el lenguaje C con el entorno de desarrollo Dev-C++ (*software* libre distribuido bajo la *GNU General Public License*) que utiliza el compilador GCC (*GNU Compiler Collection*).

Para poner en marcha los sistemas hemos implementado diversos módulos con las funciones necesarias para desarrollar los protocolos. Éstos precisan de la aritmética de los grupos definidos en los problemas matemáticos que, a su vez, necesitan la aritmética de los puntos de la curva elíptica así como el álgebra del cuerpo finito determinado por el

número de puntos de la curva. Este cuerpo finito se define como $\mathbb{Z}_p$ ya que $p = \#E(\mathbb{F}_q)$ es un número primo. Por último, las operaciones con los puntos de la curva elíptica se desarrollan utilizando el cuerpo finito $\mathbb{F}_q$ sobre el que está definida la misma. Podemos ver estas relaciones gráficamente en la siguiente figura:

Figura A.1: Relación entre los módulos



Antes de hacer la implementación de los módulos, es preciso especificar los tipos y estructuras de datos, y ello depende de la curva elíptica que se desee usar y, por tanto, del cuerpo sobre el que esté definida. Por razones de eficiencia hemos escogido una de las curvas especificadas por el NIST [69]. Este organismo determina diez cuerpos finitos sobre el que definir curvas elípticas, cinco cuerpos sin extensión $\mathbb{F}_q$ de característica q, un número primo impar, y cinco cuerpos extensión $\mathbb{F}_{2^m}$ de característica 2. En la siguiente tabla mostramos estos diez cuerpos finitos.

Tabla A.1: Cuerpos finitos para la definición de curvas elípticas.

$\mathbb{F}_q$	$\mathbb{F}_{2^m}$
$q = 2^{192} - 2^{64} - 1$	$\mathbb{F}_{2^{163}}$
$q = 2^{224} - 2^{96} + 1$	$\mathbb{F}_{2^{233}}$
$q = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$	$\mathbb{F}_{2^{283}}$
$q = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$	$\mathbb{F}_{2^{409}}$
$q = 2^{521} - 1$	$\mathbb{F}_{2^{571}}$

Para cada $\mathbb{F}_q$ este organismo recomienda una curva elíptica aleatoria, mientras que para cada $\mathbb{F}_{2^m}$ recomienda una curva elíptica aleatoria y una curva de Koblitz. Los primos q de los cuerpos $\mathbb{F}_q$ han sido seleccionados por ser primos de Mersenne o similares con un tamaño en bits múltiplo de 32 (excepto $q = 2^{521} - 1$); de esta forma se consigue una reducción modular muy eficiente.

Las curvas elípticas sobre cuerpos finitos de característica un número primo impar se determinan mediante la ecuación $y^2 = x^3 + ax + b$. La siguiente tabla muestra los parámetros determinados por el NIST para la definición de las mismas (el prefijo `0x` indica que el número está en base hexadecimal, y la barra (\ ) al final de línea indica que el número continúa en la línea siguiente).

Tabla A.2: Curvas elípticas recomendadas por el NIST sobre cuerpos finitos de característica prima impar.

$\mathbb{F}_q, \quad y^2 = x^3 + ax + b, \quad \#E(\mathbb{F}_q) = p$
$q = 2^{192} - 2^{64} - 1$ a = -3 b = 0x 64210519 E59C80E7 0FA7E9AB 72243049 FEB8DEEC C146B9B1 p = 0x FFFFFFFF FFFFFFFF FFFFFFFF 99DEF836 146BC9B1 B4D22831
$q = 2^{224} - 2^{96} + 1$ a = -3 b = 0x B4050A85 0C04B3AB F5413256 5044B0B7 D7BFD8BA 270B3943 2355FFB4 p = 0x FFFFFFFF FFFFFFFF FFFFFFFF FFFF16A2 E0B8F03E 13DD2945 5C5C2A3D
$q = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$ a = -3 b = 0x 5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E \ 27D2604B p = 0x FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2 \ FC632551
$q = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$ a = -3 b = 0x B3312FA7 E23EE7E4 988E056B E3F82D19 181D9C6E FE814112 0314088F \ 5013875A C656398D 8A2ED19D 2A85C8ED D3EC2AEF p = 0x FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF C7634D81 \ F4372DDF 581A0DB2 48B0A77A ECEC196A CCC52973
$q = 2^{521} - 1$ a = -3 b = 0x 00000051 953EB961 8E1C9A1F 929A21A0 B68540EE A2DA725B 99B315F3 \ B8B48991 8EF109E1 56193951 EC7E937B 1652C0BD 3BB1BF07 3573DF88 \ 3D2C34F1 EF451FD4 6B503F00 p = 0x 000001FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF \ FFFFFFFF FFFFFFFA 51868783 BF2F966B 7FCC0148 F709A5D0 3BB5C9B8 \ 899C47AE BB6FB71E 91386409

Notar que el NIST define estas curvas con $a=-3$ para conseguir algoritmos de duplicación de puntos más eficientes (usando coordenadas Jacobianas [15]). La elección del elemento a con este valor no supone perder generalidad, ya que aproximadamente la mitad de las curvas poseen una curva isomorfa en la que $a=-3$. En estas curvas, además, el número de puntos del grupo de puntos $E(\mathbb{F}_q)$ es un número primo, $\#E(\mathbb{F}_q)=p$.

De las curvas de la tabla A.2, hemos escogido la curva sobre $\mathbb{F}_q$, con $q=2^{192}-2^{64}-1$, por razones de eficiencia; además no es una curva excesivamente grande, lo que permite, teniendo en cuenta el uso de matrices, mostrar los datos de forma más o menos clara.

Las palabras usadas son de 32 bits, por tanto para almacenar los elementos de 192 bits del cuerpo son necesarias 6 palabras. Esta exactitud redunda en la eficiencia de almacenamiento.

La mayoría de los métodos utilizados pueden encontrarse en [12], pero además indicamos algunas alternativas eficientes de implementación.

El primer módulo a implementar, obviamente, es el del álgebra de $\mathbb{F}_q$. Las funciones necesarias son la suma, el opuesto, la multiplicación, el inverso, la potencia y la reducción modular. La suma y el opuesto modular se diseñan fácilmente y son muy rápidas. La multiplicación clásica es bastante más costosa y requiere realizar una reducción posterior. Una buena alternativa puede ser el método de Karatsuba [45]. Aunque no imprescindible, es conveniente además, definir un algoritmo que realice el cuadrado de un elemento. En nuestro caso, éste se encuentra implícito en la función que realiza la potencia. Para esta última el método utilizado es el binario de corte general (repetición de cuadrados y multiplicaciones).

Para realizar una reducción genérica sobre cualquier cuerpo se puede utilizar el método de Barrett o el de Montgomery [9]. Aquí hemos utilizado la reducción de Solinas, específica para este cuerpo, pero mucho más rápida [102].

Por último, para obtener el inverso de un elemento del cuerpo, hemos usado una variante del algoritmo extendido de Euclides.

El siguiente módulo necesario es el de la aritmética de la curva elíptica, es decir, la suma y duplicación de puntos, el opuesto y el múltiplo –o multiplicación por un escalar– de un punto. La representación de los puntos se realiza mediante coordenadas afines, con dos elementos del cuerpo para las coordenadas x e y, aunque para almacenar un punto sólo es necesario un elemento para la coordenada x y un bit que indique cuál de las dos coordenadas y posibles es la pertinente. Otra representación, usada también con frecuencia, es la de coordenadas proyectivas, en la que se transforman las inversiones de los elementos del cuerpo –operación más costosa– en multiplicaciones y cuadrados. En

esta representación es necesario transformar la ecuación de la curva a su forma proyectiva. Podemos ver una implementación eficiente en [23].

La suma y duplicación de puntos, y el opuesto de un punto, se logran aplicando directamente la formulación mostrada en el capítulo 2. Hemos implementado el múltiplo de un punto mediante el clásico método binario para un punto genérico (repetición de sumas y duplicaciones). Cuando el punto base se conoce de antemano es posible acelerar estos cálculos usando algo de memoria para almacenar los múltiplos del punto por potencias de 2: $2P, 2^2P, \ldots$ [11]. En nuestro caso, dado que usamos varios puntos, esta técnica no es muy útil.

Por último, es conveniente definir una función que obtenga puntos aleatorios, de forma que se tome aleatoriamente un elemento del cuerpo para la coordenada x, y se obtenga su correspondiente coordenada y, si existe; de lo contrario se repite el proceso. Esta tarea no debe ser muy costosa ya que, aunque el método es probabilístico, la probabilidad de obtener una x perteneciente a un punto de forma aleatoria está próxima a $1/2$ (véase la subsección 2.6.1). Además, puede ser de ayuda a la hora de revisar errores utilizar una función que nos diga si un punto pertenece a la curva. Para ello se puede emplear el símbolo de Legendre [46].

Hasta aquí alcanza una implementación usual con curvas elípticas. Los restantes módulos son necesarios únicamente para desarrollar los sistemas planteados en esta memoria.

Una vez implementadas las operaciones de la curva, es necesario desarrollar la aritmética del cuerpo finito $\mathbb{Z}_p$ determinado por el número de puntos de la curva elíptica, $p = \#E(\mathbb{F}_q)$. En una implementación usual con curvas elípticas no es necesario, ya que sólo se realiza una operación de múltiplo de un punto, y el escalar usado ya viene limitado por q.

El número primo p se aproxima mucho a 2^{192}, pero sin llegar a alcanzarlo, lo que nos permite almacenar los elementos en 6 palabras exactamente. De esta forma se mantiene la eficiencia en el almacenamiento y además posibilita la reutilización de algunas funciones auxiliares definidas en el desarrollo de la aritmética de $\mathbb{F}_q$. Por lo demás, hemos definido las funciones de suma, opuesto, multiplicación, inverso, potencia y reducción modular de forma similar a como se ha hecho con $\mathbb{F}_q$, con los cambios adecuados. La mayor desventaja se encuentra en la implementación de la reducción, ya que en este caso es necesario utilizar un algoritmo genérico en lugar de la rápida reducción que se conseguía con el cuerpo determinado por el NIST.

Para implementar la aritmética del sistema 1 –el sistema desarrollado en el capítulo 3– hemos creado las funciones necesarias para realizar las operaciones entre matrices

cuadradas de tamaño $n \times n$ con elementos en $\mathbb{Z}_p$, es decir, la suma, el opuesto, la multiplicación por un escalar, el producto y el inverso matricial. Para este último hemos usado el algoritmo de Gauss. Además, hemos creado una función para definir las matrices de forma polinómica. Todas las operaciones utilizan el álgebra de $\mathbb{Z}_p$ definida en el módulo anterior.

Por otra parte, hemos creado las funciones necesarias para operar con *n*-tuplas de puntos, léase suma y opuesto de estas *n*-tuplas, pero también el producto de una *n*-tupla de puntos por una matriz de escalares. Otra función implementada es la que transforma una *n*-tupla de puntos en una matriz de forma polinómica (véase subsección 3.5.4), necesaria en los esquemas de firma.

Para obtener la función *hash* que toma dos matrices y devuelve una (véase subsección 3.5.5), hemos utilizado como base la estándar SHA-1 [67]. No es necesario implementar esta función ya que existen numerosos desarrollos en C (y en otros lenguajes) que pueden encontrarse en Internet. Recordemos que esta función *hash* devuelve una salida fija de 160 bits para una entrada de cualquier tamaño. La función que hemos diseñado toma los elementos de cada matriz situados en la misma posición y realiza las siguientes operaciones:

1. Toma los 96 bits más significativos del primer elemento y los 96 bits menos significativos del segundo (ambos elementos poseen 192 bits), los concatena en ese mismo orden obteniendo 192 bits que hace pasar a través de la SHA-1 resultando 160 bits. Como éstos forman 5 palabras, se toma la primera, tercera y quinta, de forma que conseguimos 96 bits.
2. Después realizamos la misma operación con los 96 bits menos significativos del primer elemento y los 96 bits más significativos del segundo, obteniendo otros 160 bits a la salida de la SHA-1, de los que tomamos la primera, tercera y quinta palabra consiguiendo otros 96 bits.
3. El resultado de la función son 192 bits formados por los 96 primeros bits obtenidos en la parte más significativa y los 96 últimos en la parte menos significativa.

Esta función, claramente, no es la mejor que se podría diseñar partiendo de la SHA-1, sin embargo permite trabajar con palabras completas de 32 bits, y esto proporciona una indudable facilidad y eficiencia en la implementación.

La aritmética del sistema 2 –sistema desarrollado en el capítulo 4– la hemos implementado diseñando dos funciones, una que realiza el producto de dos elementos del grupo definido en este sistema, y otra que realiza la potencia de un elemento elevado a un escalar. Ambas funciones, claro está, basadas en las operaciones del cuerpo finito $\mathbb{Z}_p$

determinado por el orden de la curva elíptica. Para realizar la potencia se ha utilizado el método binario de cuadrados y productos sucesivos.

Por último, para implementar la aritmética del sistema 3 –el desarrollado en el capítulo 5– hemos diseñado básicamente las mismas operaciones que en el sistema anterior (el producto y la potencia), pero en este caso se requieren varias funciones subyacentes como el producto de matrices cuadradas, el producto de una matriz de escalares por una de puntos (y viceversa), y la suma y diferencia de matrices de puntos. Para realizar estas funciones hemos utilizado algoritmos similares a los de los anteriores sistemas, pero siempre manteniendo el diseño incremental que proporciona el encapsulamiento de los niveles inferiores, es decir, el álgebra del cuerpo finito $\mathbb{Z}_p$ definido por el número de puntos de la curva elíptica $p = \#E(\mathbb{F}_q)$, y las operaciones con los puntos de la misma que a su vez se sostienen sobre el álgebra del cuerpo finito $\mathbb{F}_q$ sobre el que está definida.

En las secciones restantes los números se encuentran expresados en hexadecimal; una barra (\ ) al final de línea indica que el número continúa en la línea siguiente; los números entre paréntesis al principio de línea indican posiciones de los elementos dentro de una matriz; y las letras *x* e *y* delante de un número hacen referencia a las coordenadas *x* e *y*, respectivamente, de un punto. La curva elíptica que se utiliza en todos los esquemas, como hemos dicho, es la que aparece en la primera fila de la tabla A.2.

A.2 Pruebas del sistema 1

Hemos tomado $n = 3$ como tamaño de las matrices usadas para no hacer demasiado farragosa la lectura de las pruebas. En todos los esquemas se precisa una matriz *M* de tamaño 3×3, con elementos en $\mathbb{Z}_p$, donde $p = \#E(\mathbb{F}_q)$. Con esta matriz se construyen las claves públicas como matrices polinómicas. También se necesita una 3-tupla Π formada por puntos de la curva elíptica. Ambas matrices, obtenidas de forma aleatoria, se utilizan en todos los esquemas de este sistema, por lo que se han escogido las mismas:

M

```
(0,0):   B7C01E3A 52BCE140 EF49B131 653DAFFC F6A25700 88D77B68
(0,1):   2EDAFD04 B924A0AC 333F353C 348C1ADE F0704304 0F8236ED
(0,2):   FDB08568 ABD62B70 0A86BD19 47A3D480 67A08828 8396FE7C
(1,0):   6856A8A9 9CAD60F2 EB5D5F94 6F97F07C E6742A40 13C738A2
(1,1):   29CCBF15 FC12D164 88884258 4F9A747E 28C62732 5A8C1D18
(1,2):   7DFA6AC3 5309AF08 19CB75E0 98885240 647762D4 2238ECA6
```

```
(2,0):   7BC662A0 8FC5F59C DE57DF00 D0F58868 AC7F0348 834AA119
(2,1):   24F74CE0 159016F0 5BB28500 B7551540 8D2DFF80 1A311B6C
(2,2):   26AFB4A6 3663BD34 6CABB1A0 3E92BEB0 DE725020 F17E9F0C
```

Π

```
(0)      x:  BF5AB51A 4AF1D380 E079EAD0 39E7E720 2B7C7558 8C3607D0
         y:  E67EC0D4 C9671315 67B66E84 A4B6A649 4CF6300C A47ADD22
(1)      x:  48CEE1B0 CD4C612E B59058CC DBF71F98 62180CEA C9E6CB18
         y:  ED020E98 0DCF94F9 1736A0E1 7F87306F E2AC25A9 62BEBDDC
(2)      x:  DCF3DE33 B8AEED92 ED38660C 957DBC0E 018E0EBA FF467440
         y:  5660E9EA 4100D064 1401E50A 4EB0E574 E036A9FA 462D06C1
```

A.2.1 El problema matemático

Para plantear el problema, se escogen de forma aleatoria 3 escalares a_i de $\mathbb{Z}_p$:

```
a0:      6F21A724 A02406B8 D34BBF78 49F60D3F 1599972B EA970B90
a1:      1CBB1ECC 65A945AC 675420F3 608B0D28 05A1E500 C239C228
a2:      65860646 4A9B8E7C A989FF4E 4A22510B D889E720 6DF63A88
```

Con ellos y con la matriz M se calcula la matriz $A = a_2M^2 + a_1M + a_0I$:

A

```
(0,0):   9C087035 CE2272EB 4D3793DB 95C475A0 56320604 E0F24199
(0,1):   088FF908 85C7103E 13F1493A 9408B776 AC8D7AC7 5B711DF0
(0,2):   350DB58A E9B27043 7A675CB8 77E74979 34D30F06 43366A81
(1,0):   D3478E29 058E0EB6 26D70952 12F98FC2 5FB734BC 8CF4E200
(1,1):   5E6C7E78 19395EBE CE7C951C 2AA32B0F 4A289459 0324B255
(1,2):   E85425BD 0BFC0922 75947A81 F41959F4 90F38CE2 7C9CC078
(2,0):   EF50B979 AE186A8A AA915F4E 1D26D776 F990F8E8 03AD2C44
(2,1):   90308BCA 16A49F8F 2E107CFF 9CE76D71 39DC0D61 2B544FD0
(2,2):   5AAD208F BD68602E 42FBB52D D19BE393 16AC888C 224C065A
```

Tras ello se obtiene la 3-tupla de puntos $\Phi = A\Pi$:

Φ

```
(0)      x:  709C7BD0 A3EB2364 96C211E3 BA8AE64C AE126F2E 0FED73EA
         y:  D8C0937A 1DE7CD25 BED11DDF 53157EE9 30D670A4 097C11A6
(1)      x:  AB0CAAA9 EE75E7CA F71994B2 3039EA4C 8BEE1DCC 099BBE2F
         y:  DC5325EE 5C675FD1 8CAEBA40 AA7E0568 E9583E9C 7BC853EB
(2)      x:  F397E7F4 B02FB502 0C3EC0B3 CE3F2633 71C96515 6C9C3771
         y:  F498AB6D E83FFAB3 2E978885 F6511332 629CE138 4B1C6D46
```

El problema consiste en recuperar los escalares a_2, a_1 y a_0 a partir de M, Π y Φ.

A.2.2 Protocolo Diffie-Hellman

Alicia toma 3 escalares de $\mathbb{Z}_p$ de forma aleatoria:

a_0 : 6754F5F0 EF00093B 415E4E40 A2020318 BAB6A850 24067388

a_1 : 0ABBD8C6 B0929673 1C70F8C0 A1821190 71AD2070 17564B50

a_2 : EC76C0E0 508544B0 780304DE A3151840 D437DA60 57DD6F1A

Construye su clave privada A usando la matriz pública M:

A

```
(0,0):  AAD3849F 1BB6ED42 FE4A8A67 FA4FFD01 ACAD8772 F7201C30
(0,1):  E88FC8C1 C35E96D7 70C763A7 650BB186 F77083F4 90AD94B9
(0,2):  9868EAAB 2D59C6C8 BB0003AA 8FFCAE8C 96A3E973 80B3FCA9
(1,0):  D1B5ED98 B96BF9AD 3536831D FF08EDAC FBC31BD1 90108B43
(1,1):  6A76B2B8 FA9762D9 8311EEFC CB9C95DE 15998532 F26AA0FF
(1,2):  AF2F17BB 93749D41 A8B9B837 0747AFBA C213C41A 31335CFD
(2,0):  590D3A55 69063113 A3AD7D90 DE26789F 591B7468 2DE084C0
(2,1):  A6F746F0 4E9E257A CD1ABAF4 8EF63454 3C091ACC 8E475A12
(2,2):  ACB8DEAC C6AD26E7 DAC1B9F3 C2346A6E 7417EC36 DE214AA3
```

Por último, obtiene su clave pública:

Φ_A

```
(0):    x:  01FC19C7 D270F7EA 0D376FCD C25B6550 021D41B6 56359D22
        y:  A4693B42 EAE04A11 4FC1E1A1 6435A5C6 1000434B 209AB757
(1):    x:  4A81D828 DF15F863 BC06910D D3DCF7AE 96A8899C B61C402C
        y:  527AB0C3 E92577EF 0F556240 5E8A7B73 B6C2A0C7 25115913
(2):    x:  D1921FDA 35F77F2C 8E7295E5 979A7100 AC37267F 19662E72
        y:  09C97FDC 0F9AED58 119FB4DA C57A307A 733852E3 FEB3AFD9
```

Bernardo toma 3 escalares de forma aleatoria:

b_0 : 8336D012 ACFDFF16 B21F7320 D4008D0A 63053F7C D5749CEE

b_1 : 77579D1C 0EACBAAC D71A9A40 C8530AC0 FF0DA084 61E35800

b_2 : ADFA518C AF706860 1AF6D460 1614ADEE 6F10BF30 BAA3A2FC

Obtiene su matriz privada B:

B

```
(0,0):  027971BC 82F09586 EEAE4FFA EAB21EEF 04E8B49D 1BF91613
(0,1):  0EE3FE23 8C1E6F68 8E72ADA4 2A563F99 1F09E8EF 597A0C01
(0,2):  5B0EF16D D1539E9C 9DCDE407 A1ED5A7A 82FF0E5B 6B42B6DD
(1,0):  FB2AD76F 96055592 619F1538 C98F5627 23DA8DDB C935AB49
(1,1):  096BDB01 42B81585 0DBE7216 486D8BA8 D3169DE6 F731CCF0
(1,2):  89CF6460 7A0C20E8 4F969322 5175FD01 80F41089 2E9215C7
(2,0):  4D7CE491 F3B1D2A2 A86EEFD1 6F208B72 EEC51C2E 2C795938
(2,1):  2767782B 06DE6CE6 59EBC482 16F44AFD 312B3653 A3E5DB2C
(2,2):  AC7146DD 5E383BAA 013DBA8E 7DB2BB02 3D50704C 6FEFB41F
```

Por último, calcula su clave pública:

Φ_B

```
(0):    x:  029B4802 D2E357BD 230622E2 E13D5B09 46FFDEA8 94964332
        y:  0D0A72CD 0F68383C 2958415D 59BEA61F FB7AE150 8DAC5BDD
(1):    x:  272DEE03 B014EE09 5D2E4092 27DEC365 384AACB2 0DEF8A11
        y:  5FD2B5BB 92369F9B 99873828 FAAF3C17 9C63306D C9066B4F
(2):    x:  4B68288B DEA81945 6FEF46F9 9588AD21 F25D4840 03BF00A3
        y:  A779C4FC 71285079 43A21FBD 885F4B25 119448D8 07D793B8
```

Alicia obtiene el secreto compartido calculando $\Gamma_1 = A\Phi_B$:

Γ_1

```
(0):    x:  010DEFEA A5062863 C130DCAA 07FA8689 56652373 4820D30F
        y:  C553344E D9DDBB23 7A8C2555 C5B91510 883AEA0B AA9C8661
(1):    x:  5F744D48 216279E7 A9F102E5 A412A8F3 FE2DA56C 8A5D5FDE
        y:  15F63C63 16FA042E 63734D51 35F21290 C814544B 706F2FDD
(2):    x:  3CBDBD41 3855499A 242196E5 A3BF8AE6 CE815B16 B5C119EA
        y:  E57BAE83 DE114344 43B7A5A8 74B2DB5A B317A1ED 94B5F6C9
```

Bernardo obtiene el secreto compartido calculando $\Gamma_2 = B\Phi_A$:

Γ_2

```
(0):    x:  010DEFEA A5062863 C130DCAA 07FA8689 56652373 4820D30F
        y:  C553344E D9DDBB23 7A8C2555 C5B91510 883AEA0B AA9C8661
(1):    x:  5F744D48 216279E7 A9F102E5 A412A8F3 FE2DA56C 8A5D5FDE
        y:  15F63C63 16FA042E 63734D51 35F21290 C814544B 706F2FDD
(2):    x:  3CBDBD41 3855499A 242196E5 A3BF8AE6 CE815B16 B5C119EA
        y:  E57BAE83 DE114344 43B7A5A8 74B2DB5A B317A1ED 94B5F6C9
```

Como puede verse, $\Gamma_1 = \Gamma_2$.

A.2.3 Esquema de cifrado ElGamal

Bernardo escoge 3 escalares de forma aleatoria:

```
b0:     2B953F73 8252DAE8 D04C528E DBB98B80 FC010540 B54AC94C
b1:     5D9DDF00 3BB40820 12C2F9C9 91D0DDDF 9BD240A8 973E7940
b2:     5CD19380 FE715802 8889B865 B58C2CD8 10CB4E10 FE0FFCF3
```

Calcula su clave privada:

B

```
(0,0):  88D8AEC0 7EA5011F FBAAA8D9 9703AF97 6E131ECD B66E0EC6
(0,1):  DD8D1B32 5038DBBC 8AD10AE3 B010689A 9B2F5B10 3103980C
(0,2):  B351138D D11BFD9A FCB8B417 84D68507 C6F20611 EDF9C724
(1,0):  EF16A2B9 C943930B 52284AA0 0D919B99 4FB04D6B 653435C6
(1,1):  677A9947 624385FC D9ED27C8 00CE167D 0CE95576 E9AAF095
(1,2):  368675F9 C719EBC9 365CC92B 9AE5D310 D7FA99CF 7ACCFB54
(2,0):  901AE46B 8A4119E5 67A8B314 62B588A5 A2872B84 3914A740
(2,1):  BF1C077F 6A5E5F0A 65628FDB 195A3F43 08659A30 760DF7D4
```

```
(2,2):   C7702C8C AEE98086 68509450 A572CAF7 260932ED 50DCFBA6
```

Calcula su clave pública Φ_B y se la envía a Alicia:

Φ_B

```
(0):    x:  2F710DBC D75A6E25 CA13F96A CD9F54BD 0489A8FA 52CAF89B
        y:  6BFA7A43 28047B51 B9490392 BFA7C2F4 2DB640B6 D3C40FB0
(1):    x:  59B3B914 01BBB428 CA9086E5 B02320F6 30BFC54E 4500C613
        y:  36F70551 F4EF5F63 A816BA86 09A9AAA2 E08B6A6C 50C3806B
(2):    x:  785806BD 6C86CBB2 0F42A8CC 0026B2A0 A21B6D23 5FF7C090
        y:  A22623EC AAD0B974 6AE66A04 F6852BF9 2837A5F3 D0245CDC
```

Alicia toma de forma aleatoria 3 escalares:

```
a0:     82DE2080 BC94EFB8 F60CBF72 2BD2A050 5F5E4A57 EB56E5F6
a1:     7D7F3AF8 C4A6E2E0 24CA8B7C D0A90A00 4B829E90 42EDA31E
a2:     4C94044D 7DF51610 9A6CCA68 F1C5A478 6C115000 DB208F80
```

Calcula su clave privada, la matriz A:

A

```
(0,0):  D026A51C 463B23BB 795313EF 1E12089D AAE9B0F4 F8BFBF5E
(0,1):  1B2AFC6B 3EF88C14 1CD3A431 E4D5FD9F 4F34616D 8249F87E
(0,2):  0DAD74AF AA8AE043 FE29556E D341E8D0 4F33DC2C D981701C
(1,0):  42672924 C4FA097F 23DA7E6D 4F93B6A7 E4B7E830 C2FF006B
(1,1):  41298B59 E882D23C F912055D 1C26E16D E15A1889 3E178A93
(1,2):  EB72985A A102A49E A94D85AF 52FD3F94 EC71B626 4E2DC3BF
(2,0):  6B6A9D4D 552AEF58 378A24A3 90516785 AD5E27C1 C696CE11
(2,1):  78182098 4B7C1533 FFB7163F 39D21D46 1B5C29C1 4A9961F3
(2,2):  A5C5B1DB 85D8F538 624FCC15 1DE2D767 950CBBD3 D699CA9D
```

Obtiene su clave pública:

Φ_A

```
(0):    x:  B7A9CB59 283AC9AD FEFD8FCD 3A5BD5FE 23C5B736 30E00F6F
        y:  5018F04B D7ADEB2F 70F59ECE F0EAA122 A811BDAB CC8D236E
(1):    x:  7D5EAABA 0EF93304 E20F36E2 C75D2489 7FF391BF 72C4B50E
        y:  90195F7A 544F843F F4D85540 7B85906A FF7A087D 716F9B1D
(2):    x:  A1AE3CD6 398A6739 B37B97EE 1F4A479C B6DBDA85 ABA89257
        y:  B3C59BFF ABB5937F 0D5644A8 6B958314 BFC0B715 13C9CE4A
```

Alicia obtiene su clave secreta de cifrado $\Gamma_1 = A\Phi_B$:

Γ_1

```
(0):    x:  723DACDA C8881E1D 4F8CEF0E 9371C313 A9ED352C 83959606
        y:  78C49FDB 96B5D76A 54A21C20 F694B9FB F9C5D763 659FB485
(1):    x:  A92B446C 3705F5A8 17AEFAAE 0E342993 2529AAA1 DC3A0D0A
        y:  E87E1758 F567C7D1 DC580F29 08435DAE B8F51627 BA690BC8
(2):    x:  4B966FD7 3B1EA10F A45DBA2C C05A629F 8DA98A4B 57EF3C33
        y:  35D21850 C04016C2 A3CA01E1 D5F44A1B 04963989 5AC3C5C2
```

Alicia toma el mensaje codificado como matriz de 3 puntos (hemos tomado uno de forma aleatoria):

Ψ (mensaje a cifrar)

```
(0):    x:  73ADADB4 38A32559 FBE8CA40 003F2C84 51F2F3E2 F97B3FE2
        y:  1C086FCA 61E2B90B D808868E 8FB657EE 82CF482B 58AC1FBE
(1):    x:  AA0D5A3C E64852FA 976DE020 AB32D558 9E92BC6A 22F8B0E8
        y:  63E4693B 6B8CE984 B2B4A1F7 829B724A 18EFC730 188EC666
(2):    x:  D0FAA480 0656E1CE AD067640 38C15CF9 5F34AE58 7D5C94D0
        y:  A7EB0782 AA2DC524 8EF476DB 6BF34A61 2C3A8831 1494883C
```

Alicia cifra el mensaje mediante $\Omega = \Gamma_1 + \Psi$ y se lo envía a Bernardo:

Ω (mensaje cifrado)

```
(0):    x:  3FDF1A84 65C2D0CF EBA0A207 5F2F74D8 106C2F71 21B136F1
        y:  988643A2 C7602DF1 7E99E556 7B8475B9 389BC6C3 858974BD
(1):    x:  A1F87BCC C4459F58 15B74301 98DB40E4 2C0AC470 354F06AC
        y:  4B0440E2 046762F5 300AA1D5 F9911E7D 7941B124 10906FC7
(2):    x:  9541C818 522F4EC3 4D0D0266 201A23E6 6D16E393 A1388DFA
        y:  68F51B83 C9FF067C F0B68900 ABD60D75 910528A3 6EE609C1
```

Bernardo obtiene su clave secreta de descifrado $\Gamma_2 = B\Phi_A$ (que es la misma que la de cifrado):

Γ_2

```
(0):    x:  723DACDA C8881E1D 4F8CEF0E 9371C313 A9ED352C 83959606
        y:  78C49FDB 96B5D76A 54A21C20 F694B9FB F9C5D763 659FB485
(1):    x:  A92B446C 3705F5A8 17AEFAAE 0E342993 2529AAA1 DC3A0D0A
        y:  E87E1758 F567C7D1 DC580F29 08435DAE B8F51627 BA690BC8
(2):    x:  4B966FD7 3B1EA10F A45DBA2C C05A629F 8DA98A4B 57EF3C33
        y:  35D21850 C04016C2 A3CA01E1 D5F44A1B 04963989 5AC3C5C2
```

Bernardo descifra el mensaje mediante $\Psi' = \Omega - \Gamma_2$.

Ψ' (mensaje recuperado)

```
(0):    x:  73ADADB4 38A32559 FBE8CA40 003F2C84 51F2F3E2 F97B3FE2
        y:  1C086FCA 61E2B90B D808868E 8FB657EE 82CF482B 58AC1FBE
(1):    x:  AA0D5A3C E64852FA 976DE020 AB32D558 9E92BC6A 22F8B0E8
        y:  63E4693B 6B8CE984 B2B4A1F7 829B724A 18EFC730 188EC666
(2):    x:  D0FAA480 0656E1CE AD067640 38C15CF9 5F34AE58 7D5C94D0
        y:  A7EB0782 AA2DC524 8EF476DB 6BF34A61 2C3A8831 1494883C
```

Como puede verse, Bernardo recupera el mensaje original.

A.2.4 Esquema de cifrado Massey-Omura

Alicia toma el mensaje a cifrar como matriz de 3 puntos (hemos escogido uno de forma aleatoria):

Ψ (mensaje a cifrar)

```
(0):    x:  C8E8E598 4447E7FC FF893390 C716CE00 2FA53ECC 2CF6EA37
```

```
          y:  294C1A37 1E1D6A43 63576D10 178F1049 0FD24082 F2CCBF9F
(1):      x:  903C2380 0B715100 2712C700 999F8740 16E87725 C5E29B84
          y:  5BD1DCBE CAE506C7 790FEA66 96D02D6E B59A7603 4AC23574
(2):      x:  1B50B6C0 040E122C 37EFF8C2 A9BD9D00 A0BB24F0 CA46C7B8
          y:  3E0B5E69 ADF86B4E D0CC2C51 5CA429D7 C99783B2 D83BCC54
```

Alicia escoge 3 escalares de forma aleatoria:

a_0: A54C79F8 20AB0808 DAEF4FD8 ABA189D0 5D67C7A0 DD9DFC30

a_1: B556A268 CE03D270 4C8B6F0C 585CAEA0 E1D122C8 4880EAA0

a_2: 59DE3A33 7EC33AB4 F4331900 E0AD2180 652B453E 56C13F52

Calcula su clave privada:

A

```
(0,0):  B464CC33 F6C5EA37 EC08AEFF 1666CB8F BDD570FF CC189FD9
(0,1):  002C802D 772CFA6D 9D104C0F E2246437 D4F1FC33 E224808B
(0,2):  78E43966 6E8EABF7 86139788 6E8D90DB 9DD31952 56B6DE64
(1,0):  D9B39BB3 ECC7D785 6FBF660D E8590C54 6094A2A1 4A2E3E13
(1,1):  86CDC54B 9400B94A 23EE291E 787E35F6 ED5273F3 3EF9B431
(1,2):  C73B324F A88E7308 D11E3276 7670907D 70A599E4 A1771652
(2,0):  6624F0D1 B729EE78 9A42A61C 28A978A4 6B074314 32D6CD84
(2,1):  737ECA4C 044D42CD 722C2B18 9562C3F2 584AA8C2 C847B446
(2,2):  AEB41C12 25D57B5A 07CC063F 9CFD7B42 21912077 4E2D7C82
```

Cifra el mensaje haciendo $A\Psi$ y se lo envía a Bernardo:

$A\Psi$

```
(0):      x:  36A4B502 E2BE3EFA 59F459DA F5A41551 7810CBA3 43816FE4
          y:  3329CCAA FA0D648F 7E11AFCA 6ACAC820 9ED73E19 ED9EFC05
(1):      x:  F55AD16C B9B837C0 F8B39514 94DD4A5B 8FE8D1A3 B3DA2750
          y:  D41BEE4E F8102DB0 9B003FCB FCC73EE6 CDC647BA 7B94207E
(2):      x:  0F4034BB 27653DD7 D4A833A4 3B181239 C75A030B 7F7D94A2
          y:  B9811D2D 49CD46B9 C7007ECE F3B0460C 302A79C3 A8D70F94
```

Bernardo escoge de forma aleatoria 3 escalares:

b_0: 4C4861C0 E1A92F3C 16F22450 0CB6E890 D4DB1C2B D6F43B80

b_1: A665D2C8 F510F3DC 4A7BAAB0 C8078D03 C1D6FB8F E3017658

b_2: 027F793A A2E93666 BB1EB8C0 44713116 8A592DDF 8F0C9B68

Calcula su clave privada:

B

```
(0,0):  469E227B FA6B8B9D 9D99DA36 B2417C34 CD7B2D8F 7098AA45
(0,1):  4DD52AD3 45CF93F8 C495E6FF CCD1F79A A03DE6AB 975F5773
(0,2):  A896DD54 8CC696F1 44B715B1 B311D25F 1635060A 748CD116
(1,0):  94595CCC A204196D 0CE9671F 54363091 47D7CD05 BDD7407F
(1,1):  6E548A07 8261CB97 9F88CCB6 39141759 17177A20 A81A8949
(1,2):  617821E9 8D5D0BA0 AA57AE6C D256B93F 620E3802 165DD19D
(2,0):  74EBEC88 7B35025A 66318F2A A9132F40 B18BC725 BC6C4E59
(2,1):  BB2F2FD1 80E467DA BC2F76FB 9D5F90ED 621BDC5A 2C51A7BE
(2,2):  B6DA5F58 F65C8160 0EB6B057 2884D1EC 585DD85E 0BDA67B8
```

Recibe $A\Psi$, calcula $BA\Psi$ y se lo envía a Alicia:

$BA\Psi$

```
(0):    x:  7BB7FDFA DAA3DBCD 06D06E5D D1029885 F96D4A8E 0B43C90A
        y:  680C58DE 2EDA3B84 8256C961 AF9065C1 C9CF67A4 F10AEE1A
(1):    x:  809CA9F6 A7963AFD E7D40C48 74B84F83 C6C912A1 57DFB46F
        y:  2D6E4A92 F07EE797 53CBAF0C 204E7A67 EE97F87F 01E4167A
(2):    x:  1B83C055 F22D34AE E73585EE 6847A67A B50AB664 3613CC2F
        y:  FC34C182 9F5CC99F 447EC391 359974D9 A31AE7F3 E01E176A
```

Alicia recibe $BA\Psi$, calcula $A^{-1}BA\Psi$ y se lo envía a Bernardo:

$A^{-1}BA\Psi$

```
(0):    x:  489CE1A1 5C9F95B6 2A274557 B35E59F9 357A504E 9B0EA2EC
        y:  52E3CB81 E9F7D652 410389CE 8B09397A 98EBCD4C 88D5F5E1
(1):    x:  48E9D69C EE178640 E66E392A D7FB449D 007C8F7C C45E6180
        y:  64EDE66E 4BEF0B94 7CFE66DA 6F8FF665 58A66A6A 5F11878A
(2):    x:  AA262CFD F3E114C7 52AD5C39 80876D44 C6546537 5784E873
        y:  E4A6EE9C 8F150DA4 F78E9734 C8B44835 2BD469EE 56A572E6
```

Bernardo recibe $A^{-1}BA\Psi$ y recupera el mensaje haciendo $B^{-1}A^{-1}BA\Psi$:

$B^{-1}A^{-1}BA\Psi$

```
(0):    x:  C8E8E598 4447E7FC FF893390 C716CE00 2FA53ECC 2CF6EA37
        y:  294C1A37 1E1D6A43 63576D10 178F1049 0FD24082 F2CCBF9F
(1):    x:  903C2380 0B715100 2712C700 999F8740 16E87725 C5E29B84
        y:  5BD1DCBE CAE506C7 790FEA66 96D02D6E B59A7603 4AC23574
(2):    x:  1B50B6C0 040E122C 37EFF8C2 A9BD9D00 A0BB24F0 CA46C7B8
        y:  3E0B5E69 ADF86B4E D0CC2C51 5CA429D7 C99783B2 D83BCC54
```

Como puede verse, $B^{-1}A^{-1}BA\Psi = \Psi$.

A.2.5 Esquema de firma ElGamal

El mensaje a firmar está codificado como matriz de escalares de $\mathbb{Z}_p$ (hemos escogido uno de forma aleatoria):

W

```
(0,0):  D6A27C0D C8F3EDE2 17C0E6D0 2537ECA4 AA934F51 33919300
(0,1):  86D602C0 27750480 FA5C5F17 7ED43DCC 46FCF0D0 048047F8
(0,2):  984F5B10 664C4D0C 176F1990 031FDA92 ABD3F1A8 5ADCD6D6
(1,0):  3F5BCF54 99582800 657CE800 FD1E0800 4288D050 DED3581C
(1,1):  84F5042A BF2D35EE A17E5830 6D0873C8 890692C2 E5D4657A
(1,2):  A83C4665 0C8FCBA0 2B61C780 883194A0 739E9CF4 337AAE6A
(2,0):  4B505621 0558BAF0 EFF58E78 D38B4AC9 619452A0 82C540F8
(2,1):  3E3C139D AD68289C D0FE6C8A D94FA58C 39865310 3A1FFA58
(2,2):  A19D1E40 60631D26 CE949C9E E4398D76 4005D974 C3F30E08
```

Alicia toma 3 escalares aleatoriamente:

a_0 : 96D2FA4E C86F8854 7D811050 78BE5098 31ACE823 2228C640

a_1 : 1B7AA656 A48E6836 C805CCD4 3CF4A1D8 1C8CE108 533204FC

a_2 : 6B7E5518 F225C128 93B1C33E 5EE1886D 6B6A0404 25DEB350

Calcula su clave privada:

A

(0,0): C1AEE0A3 84D0254A 9A11ED32 73FABE26 7A7CDBB5 2999CE13
(0,1): 0AEB53BA F80E772D 2AC6490B DDC547D1 A9246C7F 78154188
(0,2): 5ED95E44 8D969631 6799428B 441BC0DD 856AF2AE 4753A978
(1,0): 786C1A15 2834DEC2 4EFEE4F2 EF058526 EDD19BEC 3E77D797
(1,1): 5BCCDCD7 ABA7BA18 E2DEFD3F 4A48E312 FFBC6211 395F8AA9
(1,2): AB7ACBC3 1DF906F2 C8C9D1EB 02B3084C 4BB0415C 0CF72697
(2,0): 44A59AA6 18C79CC3 9B1B1ECB 7F5A647F 29809645 BEC7E3FC
(2,1): 146F4BF3 35B0ED83 47388CA6 05827940 2FCC17DF C3A3236D
(2,2): F4D0D438 78A7FBF5 E9683C52 924A49CC AF556460 3102BB04

Calcula su clave pública:

Φ_A

(0): x: 130C5411 E189A819 42C1819B 0B7FFF75 9E4A7200 6FB40567
y: FEA8D9DB 782912AB 70E48841 7BB88A0A 7C3FD5B7 8EBFDD13
(1): x: 2CFDC24C 95B58D79 124078A7 51B1DFE1 8DF775E2 805E6B4C
y: D78CE691 F8199498 07ED1AF4 24A580F4 38C2DBCA 998C815E
(2): x: D347E258 E714BF5C 2F1F1D9B FA302D4B CD222F91 D55A808B
y: 5C9C3757 35875D4D 545B3AB4 02370010 015C8170 0D22EA4C

Alicia escoge otros 3 escalares:

r_0 : 63D74D08 0F464958 8CC5AFFC E9AEFAC6 17C4D900 47E72784

r_1 : 1A85A6F6 7EC67DA0 B48F5BFC 7738FC36 BEDC30AB 5993013C

r_2 : 44079BB9 D998907C D1E725CF 77E668C8 D6454A00 B678B330

Calcula una segunda clave privada:

R

(0,0): BFB4FD7F 5EE97568 8FA7417A 364B1A01 AE24D5DC 3F101D4D
(0,1): E4911C7F 559BCE6E 0D07EDF3 83419B10 C4B668AA B611564B
(0,2): 80190968 4FFCD7E1 57755936 FFF575E2 6C19059F 3D01A106
(1,0): ABE0074D 40FDB470 706D1055 027D6342 D74D32AF CD881FF1
(1,1): 2C285AF4 3BEB822A 5AE17A75 79CAE238 9B18EAE3 8A3EE006
(1,2): E5FC657F 709436DD EC7A189A 354CE5DB 2F60D688 C809569F
(2,0): 10D891AE 9EB80DA8 4776A720 86432936 0B370583 8B623BFA
(2,1): 9DF9B1EE 32972CE8 E219CF62 4AA1F68A 236FBB31 CD42E8A4
(2,2): AB9C6615 A0E9BD1C 98F4A500 E173BFB3 39B166A5 928C73CA

Obtiene la clave pública correspondiente a R:

Φ_R

(0): x: 6FA5853B F940390C 7D8BD942 CDD26C83 BC64A5F2 1D324532
y: 7C136BA8 5580EBD4 9C0A6C1E 9C917A3B FB29139A 97E22BD6
(1): x: 02E9899E D6238F9C 5A65DD91 B4E87FDE B57F4B06 8061CC71

```
          y:  E727D1F5 3F520E52 EF0A9BBE 33D27767 383854F8 015902C0
(2):      x:  40A2F555 7FA00F49 591E5A9F 54EE8D1A C3F5C5F7 8ED7E315
          y:  16E74F37 0F63ABCF 809F3634 B2E58551 517B8AE7 617D6F18
```

Alicia obtiene $S = (W - Af(\Phi_R))R^{-1} \bmod p$, donde f es la función definida en la subsección 3.5.4:

S

```
(0,0):    17E0DFFD 1664F98E A6A001C4 513669CC 16D7E40F 8A7B5AA0
(0,1):    7CF5C165 7F3F5505 7648F764 329CE7A3 CAF0E3BC CA485D67
(0,2):    537E1B62 5F041A02 106FEA48 294B80FC 34247673 CCDBC71F
(1,0):    18B6D8EC 4313273C C73A73CC 99E361C3 81D1D0AF 59ECEACB
(1,1):    07E0D192 63FA0B7F 01FA3CCC 8F02B083 5EB01FFC 28C16B8F
(1,2):    729B899A 213BB6DB 89AE699C D784A2E4 4907DE37 8141BCBE
(2,0):    3149860B 115144ED 4028B62B 35FEF8F6 BA1AD2EA 0C2FB406
(2,1):    0A5AD294 DF047029 83FE4E27 8FE58AA4 33ADC95E D43A4839
(2,2):    588CA1C7 2CF77682 44A6153B 82E7B363 ED380E54 D7699F43
```

Bernardo calcula $\Delta_1 = f(\Phi_R)\Phi_A + S\Phi_R$:

Δ_1

```
(0):      x:  89FF4AA7 3ABC92C1 7AE4026B F972B6C1 EC85BD2E 597D4B35
          y:  29217FA3 D9369B0B BB355671 F9E3902D 5680DD93 04D68234
(1):      x:  91539CDB F32245A8 0ADBF098 DB805A89 A57A5008 3855F3DD
          y:  ADC35BF6 7472A85E 0A678BE5 CBF986EB C40B18C8 97ADAA44
(2):      x:  876C00A3 6DD81DEC 905A9358 799CE2CD 7F897254 57883945
          y:  5D2A3A44 99C0644D 03BFA860 30B0ECEC 6E7E86F0 0C6DD146
```

Además calcula $\Delta_2 = W\Pi$:

Δ_2

```
(0):      x:  89FF4AA7 3ABC92C1 7AE4026B F972B6C1 EC85BD2E 597D4B35
          y:  29217FA3 D9369B0B BB355671 F9E3902D 5680DD93 04D68234
(1):      x:  91539CDB F32245A8 0ADBF098 DB805A89 A57A5008 3855F3DD
          y:  ADC35BF6 7472A85E 0A678BE5 CBF986EB C40B18C8 97ADAA44
(2):      x:  876C00A3 6DD81DEC 905A9358 799CE2CD 7F897254 57883945
          y:  5D2A3A44 99C0644D 03BFA860 30B0ECEC 6E7E86F0 0C6DD146
```

La firma se verifica, como puede verse, porque $\Delta_1 = \Delta_2$.

A.2.6 Esquema de firma Schnorr

El mensaje está codificado como matriz de escalares de $\mathbb{Z}_p$ (hemos escogido uno de forma aleatoria):

W

```
(0,0):    F843DFD4 2E7A079C 7EA0AD56 1B6F4E8E EB1D47D4 57F4C220
(0,1):    E755779C 9E61F7A0 825D01D8 6BD6E191 8404542F 9A90F550
(0,2):    048AB310 ACF02246 231ED636 28358980 E1E7235B D895A552
```

```
(1,0):   E2B9DE58 FD56AE58 575E8316 8369E648 9E63A500 133A1624
(1,1):   AC62995A 30FA3AD0 6AF11308 9A4D72E4 B563C024 FDC68114
(1,2):   3CA35688 8093F128 0EBA636C F3C3D810 03CFC0A0 62A03E10
(2,0):   628823B0 272D645A A08A234A 83A192D6 07FF30F8 B778FB24
(2,1):   B41AD4DC 69B145D5 EDD19C58 654F4E80 B4CA9600 E440A3B0
(2,2):   1000CA74 1B76D72C 003AEC70 48D720AC 2EEF6D3A 6E29FE0C
```

Alicia toma aleatoriamente 3 escalares:

a_0 : 13771780 1AAEC83E 178A3364 93405718 2998F658 D59330B4

a_1 : 7005A922 7AD7FAFA 104BAEC0 E3D33274 91767082 77B86978

a_2 : F69478D8 92354F82 428A1BB0 6EDDB7D7 D763CF08 C961F2F8

Calcula su clave privada:

A

```
(0,0):   1380CAA7 5391BA9A B6E4D275 55E0776D 0561D608 D0881333
(0,1):   26D27C70 60B82EE9 C07739D9 7F1C0E79 1900DD17 F599A19A
(0,2):   9667DB94 EFB34B94 0F9C3DC8 DBA6F99A CF631787 B41A7515
(1,0):   0F6F8F77 855A6536 2AB14D30 C203CF21 5416BD20 70711C5C
(1,1):   1463544D 1DC2358B 57BBD848 0B0CA69A F730CAAC 08296C83
(1,2):   5C71F6C6 A9C3B348 57BCC7B6 80465E49 E6CE134C 5A539741
(2,0):   16AF261C 656D2921 A2CE3F43 A45D0CF9 1F508D23 27BB35F8
(2,1):   F2BC1DA2 99AA79C5 F37260EE AC0F4615 BCB0B06A 625DE0BC
(2,2):   3B3659A5 2E4A126B 7C66129F B05AB999 87949992 A340C3F0
```

Calcula su clave pública de la forma $\Phi_{-A} = -A\Pi$:

Φ_{-A}

```
(0):   x:  151FB410 A6C6E83C C5D43224 29467689 4DB141D9 10A82A63
       y:  79E432F9 A1167B69 4546DC9E 9785778D 99860811 C36EDE2B
(1):   x:  16BC4F46 F5D2C9C6 100990A8 9FEC37EB 45315A9C 5C0D613E
       y:  C466B814 6C6FC519 4F3247D3 6072F952 05A7BB80 3C880AA1
(2):   x:  58F7AB39 8367DD77 736601D6 6AE5C875 12E49581 61F9A821
       y:  00BC829C E1DBEA60 1F01506B FC37D981 837957D9 73704B9B
```

Alicia toma otros 3 escalares:

r_0 : 7D260340 1DBFA850 FA71E3A5 95D75376 18E8EDCF 8728F799

r_1 : 54C0D554 9A8385AC 7AD62040 3E6330DC DA222297 AE51D270

r_2 : B47FA2DC 99763F3E AFAA3A95 6FB133C3 4678F5FC 1128640A

Calcula otra clave privada:

R

```
(0,0):   329402F2 6F815C7D B14D54AE FD1CD09C F83DC304 12577C49
(0,1):   B848664C 8613F910 F5DC6091 69CDA12F DAD41C02 4862F48A
(0,2):   7A9B0EA4 8DD44C41 8CD77807 A1CCDFF6 3092B907 D4F8521A
(1,0):   F504E478 FB89E3DB 55503E97 9EF9D965 EEDBA247 19486EC8
(1,1):   49AD9724 A5791C47 99B11737 6AB450E1 2E89E63C 7546D5C6
(1,2):   DAD638A9 CC47B601 547A8E62 EE96A892 8D361E2E DCBD51F4
(2,0):   1ED3EC35 2A21DAE0 6263AF9A B014D53B 71D79A9C EC7A7938
(2,1):   60E042D1 8FCE309C 57900086 D4D92F52 971B35CD 7DC00737
(2,2):   207A76F4 51371628 1A87DC89 F5E79D97 19EF4DEB 5C4B3E89
```

Obtiene la clave pública $\Phi_R = R\Pi$:

Φ_R

```
(0):    x:  2A649C10 449C8DD4 AFE0E0E1 4237EDC7 8ABD52D6 9C1790DA
        y:  67733609 96637621 0452B19F 11EFBA6C 1F000BB2 B7AA7AB6
(1):    x:  85FF3C45 8D27905C DD110AAD 9165CF52 7C1A7788 F9B585EB
        y:  6584E900 30ED4ED5 D3CA04A4 5CC3F147 FBBD5CC3 88F6E8EE
(2):    x:  9C4E053D 61145A15 3194AF46 135AC6A9 D37153EA 77BFBB11
        y:  F812C518 E0251808 E6B09068 12061CCC A88AC56F 99042794
```

Alicia aplica la función *hash* χ definida en la sección A.1 de la forma $H = \chi(W, f(\Phi_R))$, donde f es la función definida en la subsección 3.5.4:

H

```
(0,0):  78AC4C71 3DA7749A 54F5A486 89BC336C ACEBD28C 37B07306
(0,1):  AABA36F5 4B2486E2 9252C6A5 4FDCB794 FA717CC0 F4BCA275
(0,2):  9BA05834 E5F679B7 C596A045 CD064316 19F9D513 CD81E5BA
(1,0):  94672D4B FFBC3A64 C8DE795D 643426D2 34234250 75608518
(1,1):  FB3826AC F3D4190B 2060D577 04E54AC4 A5A23C61 345A87ED
(1,2):  88C3E299 9946CF7D 3C7E44F1 5E85F563 4C7CB9B1 70D7FE87
(2,0):  E1BEF0A0 9F5BD467 76C8731B 720CCEC3 F7FE52B0 4294E9DD
(2,1):  C5290BD6 EB660E7A C676FB6A 0877328C F1CE0FE2 90CE7B4E
(2,2):  A6BB9B7C 71F0C92E CB0E0D4A 0C9D547C DF17A71F 1BCFE622
```

Por último, Alicia calcula $S = (R + HA) \bmod p$:

S

```
(0,0):  684765DB CA9B9403 F2A63331 8D87BA9C 484633C9 CD65C80F
(0,1):  92A81E3D DF88E3A6 B6C7E252 C3299F67 F3A2F1AC A275D829
(0,2):  D4D4F261 2783B690 72DA99D2 ED14715D ABCE4CFF C6C04AF4
(1,0):  6CD2E555 1CE7CB73 AADC950A 2549B52E 5ABA9773 AC39D093
(1,1):  D823FE5A 656EDAF8 13F2062C 3BE27801 8BC75B03 AD52FBB2
(1,2):  06161FC4 51A90672 A6A1096A 45C8E24E D347B118 3BAF9F58
(2,0):  57436A6F C249BD62 F4170806 46B4F872 CE3C8301 AEBAC58D
(2,1):  48239FE6 7AB85E37 DAA4864D 02A61234 9712913B 956C274B
(2,2):  DBE9556E 55C5F929 F1809E96 03CF4B9D D0ACDFCC A786AE2D
```

Bernardo calcula $\bar{\Phi}_R = S\Pi + H\Phi_{-A}$:

$\bar{\Phi}_R$

```
(0):    x:  2A649C10 449C8DD4 AFE0E0E1 4237EDC7 8ABD52D6 9C1790DA
        y:  67733609 96637621 0452B19F 11EFBA6C 1F000BB2 B7AA7AB6
(1):    x:  85FF3C45 8D27905C DD110AAD 9165CF52 7C1A7788 F9B585EB
        y:  6584E900 30ED4ED5 D3CA04A4 5CC3F147 FBBD5CC3 88F6E8EE
(2):    x:  9C4E053D 61145A15 3194AF46 135AC6A9 D37153EA 77BFBB11
        y:  F812C518 E0251808 E6B09068 12061CCC A88AC56F 99042794
```

Calcula $H' = \chi(W, f(\bar{\Phi}_R))$:

H'

```
(0,0):  78AC4C71 3DA7749A 54F5A486 89BC336C ACEBD28C 37B07306
(0,1):  AABA36F5 4B2486E2 9252C6A5 4FDCB794 FA717CC0 F4BCA275
(0,2):  9BA05834 E5F679B7 C596A045 CD064316 19F9D513 CD81E5BA
```

```
(1,0):    94672D4B FFBC3A64 C8DE795D 643426D2 34234250 75608518
(1,1):    FB3826AC F3D4190B 2060D577 04E54AC4 A5A23C61 345A87ED
(1,2):    88C3E299 9946CF7D 3C7E44F1 5E85F563 4C7CB9B1 70D7FE87
(2,0):    E1BEF0A0 9F5BD467 76C8731B 720CCEC3 F7FE52B0 4294E9DD
(2,1):    C5290BD6 EB660E7A C676FB6A 0877328C F1CE0FE2 90CE7B4E
(2,2):    A6BB9B7C 71F0C92E CB0E0D4A 0C9D547C DF17A71F 1BCFE622
```

Bernardo verifica la firma, como puede verse, ya que $H' = H$.

A.2.7 Esquema de firma ECDSA

El mensaje viene codificado como matriz de tamaño 3×3 con elementos en $\mathbb{Z}_p$ (hemos escogido uno aleatoriamente):

W

```
(0,0):    A4D6F560 87A19B32 CB0EBBF0 B7CA7DE8 7C3AC82C 76D61BC0
(0,1):    03D91EBC 2BAFCAB9 505634E0 95E57848 EBE23010 7ADF7900
(0,2):    A48B6F4A DF4D8AAE 4D2F75AC 45A20A99 0D54AF44 4EDB2837
(1,0):    39FAD970 D2708868 92657A0E 228EED0C B1195D70 421054C0
(1,1):    308465E0 18380834 705D19C0 E7A5816E A35FDFB0 7239116B
(1,2):    28BC1708 808B2928 A9A5425C 73B262A8 80BE2F00 00C32BD9
(2,0):    3C468B12 9C12072F C4228566 6A1C0969 F942B9B0 DD110FC0
(2,1):    9C9B59F8 0D633C80 F029727E B1694BEC 839B222C 82317DB0
(2,2):    B1083DE0 E68D27D6 55039BA4 6BDC1600 3CF4C6E2 CAE44E88
```

Alicia toma 3 escalares de forma aleatoria:

```
a0:       253DF268 CDED7137 16BB8270 87A7AD7C 28484255 ACBB6574
a1:       2D780FA6 857C8C88 F1DD64F2 D86A63C4 CBCB78E0 66043813
a2:       F42D614A C55378FE 492D9608 2331F350 70DD0720 5EFF6FB2
```

Calcula su clave pública:

A

```
(0,0):    54BE369D BEF629E3 99F407E0 2863202E DDA270DE D413F431
(0,1):    206C6024 B5070D85 816F2C7C 53566E22 2807F336 95354E36
(0,2):    EE6FF402 43AC9AD7 114A62FF C46DDAA8 F85FAAB5 71D114E1
(1,0):    37D6B9CA B7A3D432 DB9EB6E2 27F13E4F 14E427DD C5669035
(1,1):    D8F303B8 7D838AC1 E18D5628 4C2E9311 3657EC79 C2FA85F3
(1,2):    4A54F680 7B99D375 224C47EF 53EEB1C7 572A1769 0CB1616F
(2,0):    6F7F3FDA B8EEBBDB A5BC2932 A1857397 026F5EE1 FCCBF0AD
(2,1):    DAE1B921 D78D84FF 7B296049 3CFA4CD1 C1BF03C6 8A9B50F1
(2,2):    9F54824A B1436C03 D515DEC7 7A0502EB E510B4F0 D55FB0AA
```

Obtiene su clave pública de la forma usual:

Φ_A

```
(0):      x:  5603F73F FDB61B7E 3C79916A 94030523 FDBF4C2B F63464C8
          y:  CAC17B38 6D9DCF21 006AD951 B1861206 69979AC8 6B9AD13B
(1):      x:  4636F4C3 2C2C7E19 78C0DE5E 8DC70A55 96A9FBC6 EBE60A16
          y:  29311DF1 79E16FB1 69489C23 48DFCDD8 C00C4ABE E2EB0FD8
```

```
(2):    x:  62C80657 A1DE3EFC 8EDDF852 C31EC635 2CE5DCF8 0FD8E0C0
        y:  D2EF6FBA 500C2EBD 025A04EA D3AEF1FF A28645CD EF13503C
```

Alicia toma otros 3 escalares:

r_0:
```
52AA2964 196C3050 B1F202C0 C479C8E0 3AB5316E 9EB015C8
```

r_1:
```
140D9678 A3CC8672 36A9D394 FC26E800 29369AD4 7389E01C
```

r_2:
```
ACD8BBE0 7718F370 E14D8A80 EB6D23B0 A1213DAC DA9A6876
```

Calcula la clave privada:

R
```
(0,0):  6248D436 AC80C145 9F9C2F12 DFA20A8C B55AF890 41E9A2D7
(0,1):  ACED0EB9 AAABD76E 0701F73C C7E6F99F 7B77FA7F D0D3156D
(0,2):  F9A4384E 88108DCD B7280D31 C8D6DB37 0F1B70D0 E90F3BCA
(1,0):  F6AD12F8 9D1EFFFF 6CEF0C6D 3F3460C8 EC230F90 57AB36BC
(1,1):  76988A1F BA6E2C0B BEF79D4B 2902B2CB 34C3122A 7BCB0D0F
(1,2):  EA681C03 75E77150 972D5045 A2D14CFD 40F6E649 6CEEE948
(2,0):  5E2AB8B2 C0494ACD 480A5D31 7D3D037D B4908986 AD87ECE6
(2,1):  CC6700BA 9CF49E10 74B1FC33 16714496 E050FDF9 59BAE434
(2,2):  9B34084A 93B22554 ECEF0619 F3FEF788 B871EB60 9AB0E7FB
```

Obtiene la clave pública correspondiente:

Φ_R
```
(0):    x:  9C272BCA 6D0C920E 631783E1 B62ED52E D3A62E00 DAB1D5AD
        y:  454E0B42 86A2C461 140A9A3F D63B5B69 58FDF21F 446BBE09
(1):    x:  867035A5 6971EDA6 7820F01C DF7FE3E2 D10C2B56 0064DE7F
        y:  BEACCBC7 DEC15818 65CC4C24 9A5ADE2E C9EDFDB0 3E385086
(2):    x:  6041D376 BEF55763 81054CF7 A2ECB88E A319F630 2DC18F69
        y:  030B83E3 69D2B459 6F45B207 3C876B59 BAA15C78 9173495D
```

Calcula la matriz $S = \left(W + f(\Phi_R)A\right)R^{-1} \bmod p$:

S
```
(0,0):  34383541 D9919F8A B11D88E6 4BCE1965 3463D429 54887C49
(0,1):  CAE96537 C6EA4069 02640ED9 E95BD669 211014FA 6AEF7F0A
(0,2):  3AA25EF8 9FB81CF3 FA054F0C 3019827F C7B9554C BC0BFD35
(1,0):  2D711510 8AD2D2D6 EB145533 84898A9D 587A60E5 041F6E2C
(1,1):  FBB257A8 4E1EFFDD 5023D47D 8CE7C25A 89CC1E2A 6DD488B7
(1,2):  8AB88BC7 87BA2A09 060FE6A4 CF983D0A 228696F9 51B46382
(2,0):  21B9E3D3 5A3271CB 394243B3 344602CD F3DD20F7 42AFD5F7
(2,1):  58FB5DA6 A3C2F57A FB7A79D4 48ED08AE 15DB4135 9AF83A43
(2,2):  96236444 A04FD79E 8A8BE54D FEF89363 044C6B29 92523CA3
```

Bernardo obtiene $\bar{\Phi}_R = \left(S^{-1}W\right)\Pi + \left(S^{-1}f(\Phi_R)\right)\Phi_A$:

$\bar{\Phi}_R$
```
(0):    x:  9C272BCA 6D0C920E 631783E1 B62ED52E D3A62E00 DAB1D5AD
        y:  454E0B42 86A2C461 140A9A3F D63B5B69 58FDF21F 446BBE09
(1):    x:  867035A5 6971EDA6 7820F01C DF7FE3E2 D10C2B56 0064DE7F
        y:  BEACCBC7 DEC15818 65CC4C24 9A5ADE2E C9EDFDB0 3E385086
(2):    x:  6041D376 BEF55763 81054CF7 A2ECB88E A319F630 2DC18F69
```

y : 030B83E3 69D2B459 6F45B207 3C876B59 BAA15C78 9173495D

Bernardo verifica la fima ya que $\overline{\Phi}_R = \Phi_R$.

A.3 Pruebas del sistema 2

Para definir este sistema es necesario utilizar un elemento $M = \begin{pmatrix} a & P \\ & b \end{pmatrix}$ generador de un subgrupo de ζ descrito en la sección 4.1, donde $a,b \in \mathbb{Z}_p$ y $P \in E(\mathbb{F}_q)$. El cálculo de $P^{(k)}$ se realiza mediante la ecuación definida por (4.2), sin embargo, en aras de la claridad, se muestran también los términos a^k y b^k. La curva usada es la misma que en el sistema 1, indicada en la tabla A-2. El elemento M se ha obtenido de forma aleatoria:

M

a : 1033C218 FDAB0926 B0A7CCFC 6195CC90 FB516213 479A32F0

b : 7E48DCC2 A491CFA0 2DA31EE4 C29AF734 DC08AE7E 7BAA6414

P : x : 43F495C3 275B50B4 E21B5A08 F0AE155F 6211B658 11CBED7C

y : A30D7251 70A3B4BF CC033749 4D87FEFB 6752642D E8EBA046

A.3.1 El problema matemático

Para plantear el problema, escogemos de forma aleatoria un escalar k de $\mathbb{Z}_p$:

k : 01E6EA60 623C73B0 8F731580 B1CE0178 63FB07F4 319CD400

Calculamos la potencia k-ésima de M y obtenemos:

M^k

a^k : E5A65F82 F35CC28F F18BFFC0 FA1B21CA F48CD93A B18A7D32

b^k : C4C2FAA3 E0139831 89EBFE9F 6472F914 4DA0089B 4CC41731

$P^{(k)}$: x : 0FFA3F38 A7F94271 ACC5952E 13DD0D26 84265BB0 F9A7C411

y : A95F34B9 EB2D46B0 7D87DBB1 430B222F 862A3654 430B0ABB

El problema consiste en recuperar el escalar k a partir de M y $P^{(k)}$.

A.3.2 Protocolo Diffie-Hellman

Alicia toma aleatoriamente un escalar k de $\mathbb{Z}_p$:

k: BDB30160 E1A30E68 58F04D10 C10AB594 36C9A5FC 4C9CC636

Eleva M a k y le envía el punto resultante $P^{(k)}$ a Bernardo:

M^k

a^k: F5058091 80C000BF 6086195E C376F25C 7DF37549 6AE84817

b^k: D498522A 22E36FA6 1A0C3499 689AB07B 57826E77 1175485C

$P^{(k)}$: x: EA6F0FDF 912321A8 30C25521 ACA0B802 AF942069 A709FA61
y: ABF6A851 E407A6C6 D04D62EE B19247BC BBBE71BB F8716D50

Bernardo toma de forma aleatoria un escalar m de $\mathbb{Z}_p$:

m: 1499A1A4 57BC3480 B1A9D02E D6790C64 1A5074A0 91F762A0

Calcula la potencia m-ésima de M y le envía el punto resultante a Alicia:

M^m

a^m: 8A924ECE 6CF179F1 BDAF4C78 B13CD483 F1841522 1B0731E4

b^m: EDC0AF9F 304F47C4 B75D8385 D3ABA9A2 E9B35CD8 0506F46A

$P^{(m)}$: x: D18DD686 54089EAB 3FA535C6 DBFF435E 18426DB4 BBA5954F
y: 6AC3BB5E AB707A26 5036F9FC E2A774FD B49D92DF BE7B686C

Alicia toma a, b y el punto $P^{(m)}$, y calcula la potencia k-ésima de $\begin{pmatrix} a & P^{(m)} \\ & b \end{pmatrix}$:

$$\begin{pmatrix} a^k & \left(P^{(m)}\right)^{(k)} \\ & b^k \end{pmatrix}$$

a^k: F5058091 80C000BF 6086195E C376F25C 7DF37549 6AE84817

b^k: D498522A 22E36FA6 1A0C3499 689AB07B 57826E77 1175485C

$\left(P^{(m)}\right)^{(k)}$: x: 61C54890 0743E288 E0324DB3 94ADBCCF ACC53C5F CD76F2C3
y: 674F709C 88C67569 BCCAFEC3 49B2E9DF DD8E3EB3 341430B7

Bernardo toma a, b y el punto $P^{(k)}$, y calcula la potencia m-ésima de $\begin{pmatrix} a & P^{(k)} \\ & b \end{pmatrix}$:

$$\begin{pmatrix} a^m & \left(P^{(k)}\right)^{(m)} \\ & b^m \end{pmatrix}$$

a^m: 8A924ECE 6CF179F1 BDAF4C78 B13CD483 F1841522 1B0731E4

b^m: EDC0AF9F 304F47C4 B75D8385 D3ABA9A2 E9B35CD8 0506F46A

$\left(P^{(k)}\right)^{(m)}$: x: 61C54890 0743E288 E0324DB3 94ADBCCF ACC53C5F CD76F2C3

y : 674F709C 88C67569 BCCAFEC3 49B2E9DF DD8E3EB3 341430B7

Alicia y Bernardo toman $\left(P^{(m)}\right)^{(k)}$ y $\left(P^{(k)}\right)^{(m)}$, respectivamente, como secreto compartido ya que ambos puntos son iguales.

A.4 Pruebas del sistema 3

En este sistema necesitamos un elemento $M = \begin{pmatrix} A & \Pi \\ & B \end{pmatrix}$ generador de un subgrupo de ζ descrito en la sección 5.1, donde $\mathrm{A} \in GL_2(\mathbb{Z}_p)$, $\mathrm{B} \in GL_3(\mathbb{Z}_p)$ y $\Pi \in \mathcal{M}at_{2\times 3}\left(E(\mathbb{F}_q)\right)$. Como vemos hemos escogido los tamaños de 2×2 y 3×3 para los bloques A y B respectivamente. La curva usada es la misma que en los otros dos sistemas, indicada en la tabla A.2. El elemento M se ha obtenido de forma aleatoria y se va a utilizar el mismo en todos los esquemas:

M

Bloque A:

```
(0,0):   E37DA779 E401E100 17AC912C 655E1850 9ADFCEA4 FA73A994
(0,1):   537693BE 615F7C58 E6C27A96 E6C7FF4E 5E4555E8 A42B714A
(1,0):   75F16240 65A85108 C56F6F12 95D214A0 054A504F E985BF50
(1,1):   7B8F7CD8 B9CCBE88 0DC78A20 05FE0FD8 A7917FCC B1F75AA8
```

Bloque B:

```
(0,0):   78EE2540 EB28C29C D9D500F4 CA19220A E537964C C0960098
(0,1):   62BD2A2C BC921938 7C658853 DAB02D22 7FBB2D30 8902F43E
(0,2):   C9571777 1B2C0448 AAA015C4 96064BC4 C690E680 BC5FC970
(1,0):   151E20B0 4012B460 BFE5483A 90DF5B90 D1FF3F36 5F6611CC
(1,1):   8B01D8C8 A5CF3170 B5A0A0FC AECA9A46 5960A2FE 7CCF2BAE
(1,2):   045952F0 F314A69C 5652EE1A C991AD00 0D1F52D4 32A82600
(2,0):   CC9EB7C1 0ABF6A00 3AE1C3C0 05762E06 C0D71260 1AAE195D
(2,1):   5198AC00 831CBEEC 409E6CDE 51C892A8 53F72066 D6299B62
(2,2):   C42BDBB4 2C7DA500 D69D5994 306964D8 2EB90558 31D083C0
```

Bloque Π:

```
(0,0):   x:  C5E8C9B0 D2AB2F8F 81DCFFAE E5486040 E3D8D6EA 10927F15
         y:  67C4267B 2BB44B09 067262F6 3B8FB5E3 0AF86852 29C0FF6C
(0,1):   x:  1ABDCF35 C1D717AB 0CE4A3D8 8BA135F1 BD12C542 AB8EB090
         y:  F49B0698 12FCA278 28C6677D E4131F66 E95B6E8C F957F63A
(0,2):   x:  4C7752B4 052EB554 229DF460 6349AECE 0BCDCD88 DD7CE5EE
         y:  644927D5 13CB3C5E 9B12064D A846C6FF 13351EEB 1A40380C
(1,0):   x:  E3800310 C4B57600 F62B54CB D53E9410 10A18954 2A44BBD2
         y:  CF5FFCDD D3F3BA62 8CB41DE5 045834A7 3F14FD60 73218BAA
```

```
(1,1):   x:  A84DDC48 9FCC69C8 64130230 E702BC5A 5A4D8065 DC4D28B4
         y:  2B7A9EF7 BCA45ED9 C4FA407C 773344E1 8BDA77A0 1DADDE3E
(1,2):   x:  497E6B65 D1D83208 78E75E78 82681244 EE6F9818 A30B5A00
         y:  ED22DAB5 6C88D264 75A06E0C 0D58B410 310B4448 E09BD8D0
```

A.4.1 El problema matemático

El problema se plantea tomando un escalar aleatorio $1 \le k \le o(M)$, donde $o(M) = \text{mcm}(p^2 - 1, p^3 - 1)$, por tanto k estará formado por 768 bits:

```
k:       5210B948 35CC0D64 2C049760 D3FC3E84 6C3A783F 62048D19 \
         6730915A 90E14A50 A9C8D343 03B0EA08 B08C24BA 01924A22 \
         1C8D5028 2828042B 3A004B4A 0564216A 4D174C05 C015232E \
         094DA510 A867A414 984892B0 5910EB04 B3669B80 808E2316
```

Calculamos la potencia k-ésima de M y obtenemos:

M^k

Bloque A^k:

```
(0,0):   CFAE2F43 8F99328E ED8E17D0 98DA2A95 0121C868 E7017544
(0,1):   F93F359C 48CE9CE1 0F50BA95 302A09F7 E2BE5751 8E9C301B
(1,0):   88B159E3 74C25208 383F4312 33357BCC E4CB14A8 05E398C0
(1,1):   D01EA0E2 1631C048 2C9D66FC BCF6091E 0210344C 5126147B
```

Bloque B^k:

```
(0,0):   F98DC796 A8AAD8E6 AF6E1BA2 94624EED D683DF25 67A86964
(0,1):   B06DBE6E ECCE8569 E8F2BFFE 6C7A52C8 710C6117 E1355F02
(0,2):   ACC20CA0 A4AEDE4F EB6D97A9 822E3F29 FAC3DA77 877E02B6
(1,0):   8F7D0898 92296431 021C8B8B BE479496 658690DC C5C7BA97
(1,1):   4F414274 109731CD C528C7F4 F42C1919 34AF305F 158C106A
(1,2):   03EB660D 9FA619D6 3D5E34AA 9C071F54 47002039 BD162B8B
(2,0):   E3057F9F 5E9AB959 0B77F385 24616CC4 1A5ADC57 A62B739F
(2,1):   D62053B1 D9334159 6524C5A7 9F1E9821 F2A7692F 57C13631
(2,2):   2E484615 4B1D659D 7E004F8C 53CB348C D8564229 71E54781
```

Bloque $\Pi^{(k)}$:

```
(0,0):   x:  23DBE4C3 2C1B7248 7A92C8D1 E17070BA 71D77E55 457169BD
         y:  90FEFB70 65BB6E13 5D48A6EC A7BCABC1 78CE4A44 305A3C83
(0,1):   x:  05F8E81E 798FE708 563400D3 5EA682A4 D4D5321C 6AE3E4BA
         y:  74C0C374 55958B46 555035B3 D6C698CF 5892EFD3 207500DC
(0,2):   x:  0E61C34B AA0AAD61 F9D77486 8C18FF2D D7E4D0D6 37C8BE7B
         y:  EAD1DEF7 E908E37E 2D4FFB15 41543E6C DEB4F793 3633F35A
(1,0):   x:  A8B08F31 8F824359 E1F8FF59 1CADFF83 49789DCC 17B9BC2D
         y:  B386BDCC C82970EA 8D7D071D 2E554BF5 9C8CF336 99954915
(1,1):   x:  722492F4 FC6560C7 38D70F65 78F00337 02555AF5 EEE375F8
         y:  CF8CECF4 18F04866 C05A5759 BD9E820B A8203616 B13B953F
(1,2):   x:  0EE286F8 27A964FA EDCC1D02 2584FD89 EC871F24 2938DBB4
         y:  2DBAA287 DF8674D8 FAAD117C 724A0D37 048BB153 4461B2A6
```

El problema consiste en recuperar el escalar k de 768 bits a partir de M y $\Pi^{(k)}$.

A.4.2 Protocolo Diffie-Hellman

Alicia toma aleatoriamente un escalar k de 768 bits:

```
k :     E430F990 C7D86CF2 F13211A8 A6DE3E40 EFE358D8 0E29FBC8 \
        1F3A8F40 1AEE4130 F1DD4E00 7B5B94ED 1E96D5B8 5F054380 \
        CA2A5E80 C3BC55A8 0D7D7476 A6C64260 66AF6040 FB2A807C \
        60430360 7FEC6400 0E70404D D2095500 20713508 597C94C0
```

Eleva M a k y le envía el bloque $\Pi^{(k)}$ resultante a Bernardo:

M^k

Bloque A^k:

```
(0,0):  81EE4064 73FCA136 B332B225 4D4BF5F9 FD1B92B8 77CCC476
(0,1):  03AA8D9E CF302885 7BCE03FD 3648CBBF 156D82DE BA253D42
(1,0):  775FD5F9 3BB915EB 6046AE02 70C35B72 6984270D EAD905DF
(1,1):  E8D72CD3 C4956294 E97DDE83 1C6CE94E A4408B45 165D35EB
```

Bloque B^k:

```
(0,0):  A4D8915F 0E91B52B 0274040E F917390B 6D5E2747 38A8092F
(0,1):  890248CC C4C3037A 6C0BADE9 E4F186D9 6843D412 E9B1AEF0
(0,2):  1810A63F F754E2AF 3CA8482A 88A0130E 578C215F EB8D8557
(1,0):  85065E39 A7A844C4 77E6F58C 88FB18A4 C72B5800 2587E822
(1,1):  546E431D 4C327DFE 60CFBF0C 0272A301 1D0DBD01 B5C51068
(1,2):  734F9597 187CA6E9 6018D673 2109FE54 8C08956C F6C6F5F2
(2,0):  08D5583A D57A5F02 7A460E4E 97E7E9B5 941D64CF 6D972AF3
(2,1):  276DA89B D0CB050B 6BF6E48E CA57113F 2A98EE9D 382ABF5D
(2,2):  FFC325BF C777940B 4B6C6E24 70D935AC 3FB0F346 9E5AD256
```

Bloque $\Pi^{(k)}$:

```
(0,0):  x :  77B0689F B1B5644C 1B1BBD37 A02C8EA4 9DFD6F73 D47FD6FC
        y :  9BA6599A 57DF4376 F1EF43E0 BC63F7A1 DD6967A5 FB5457A0
(0,1):  x :  596646BF E7E6DDC3 F609F0B1 38093AA9 ADD13A38 542FB0CD
        y :  B07A7961 CE0610FC F5E8E946 46DF96E7 A62A4AE4 E24ACCCD
(0,2):  x :  D4C18D3C 4331277B 4B7E6B5D 045EFC3F 1017D27E 437B71E6
        y :  F5E013B4 0AB323C9 2BEA5A62 2A16D067 99E7A59F 1607D856
(1,0):  x :  60E55A7F A4D13908 B7F80DC1 5A61BABA 3958DD06 339A5B46
        y :  DF33903F FAB3EE15 9AE0BC4A 6EB8D771 252858D0 CB993740
(1,1):  x :  BCDD0F01 16B7FB02 2D0976EF 1DD50E36 B818766A 4A3F96C8
        y :  2935CDE5 345D0518 7ADE6A8F 09D3B9BB B37FF019 4D861143
(1,2):  x :  2CB5125D ADC269AB 8A772336 3C513393 EC19FA54 4FB429EF
        y :  DAA137BD 8E3AE13D 180EF7DB E88C4237 CB2F777F 3745370F
```

Bernardo toma de forma aleatoria un escalar m de 768 bits:

```
m :     B01DFD80 F4CB0A45 29A6FD29 91DE53E2 3F11FE5A BB78ED30 \
        7337ABE3 05793896 CFA2C4A0 6D90EED4 673EF4FB 4F5E5030 \
        D07719E8 151408A0 FB6F64EC FC999888 0C350760 29BC91DC \
        3FF8AE36 3E6572D9 E308ED6A BF8A3828 C749F500 972FA2FB
```

Calcula la potencia m-ésima de M y le envía el bloque $\Pi^{(m)}$ resultante a Alicia:

M^m

Bloque A^m:

```
(0,0):   1ED1B8C3 E1893802 77FDE30B AD96AEC4 F1FD0D36 EA501612
(0,1):   570372FE D2E7ABC1 B62BF429 11898DDC C70EBC75 414CC56D
(1,0):   FC444ED3 F3FDA091 877671A7 D0382A2E DC4BA5A8 DC842B36
(1,1):   6436A38A C0D337D1 C303E6F5 D1C90EB1 44C308EC 319F5D31
```

Bloque B^m:

```
(0,0):   AD5E41BC 0CCE3BD9 82A887CE 84A945D6 2B0BCDA7 ECA921D4
(0,1):   3522B8A2 5B57879E 9DDEECAE 1D255942 230324A8 3E0EFA16
(0,2):   0BD0B0B7 6AC4A66F EBD035B3 558641A3 8BD103FA 0ED40588
(1,0):   0C3317E1 6E36AB9A 2D8E73C5 D965B86A 5CB5ACB6 2E3F261D
(1,1):   E72A08A1 EAD8D67F 4A857E8C 3EB1D5BA 5693F0B0 DDBF24EE
(1,2):   1A374C6E 6675B1E5 60ED0D63 E06A42C0 EA706006 E721CE88
(2,0):   261EF1FE BD60C92D 7FDA0155 B046DC63 97AC9938 2FFFAF5D
(2,1):   064C1B12 DE5D6840 4FC5735A 95F992C0 137BC006 D658EA0E
(2,2):   26A04588 14AD575B B92679E3 A4392DF7 B4D085BA 71D0E8EF
```

Bloque $\Pi^{(m)}$:

```
(0,0):   x:  2DD4DE66 3746DFFB F764A880 D4C6A6C6 C556A163 204DB33D
         y:  90D1D8AF ED541DA5 CE582650 B19BA35E F466E458 2E3877E2
(0,1):   x:  E874F050 BF235527 2B8821B3 AB52B022 6CA3D4CA 75FB7AC2
         y:  DA69894E BE2CC11D DD3D60D5 7CA409AA 1437E46D 3C29DFB5
(0,2):   x:  3A8F4C75 F517270F 76590DB3 9A5B1BDE FE666EEB F8AAD28E
         y:  B7398A24 EBF5337A 6609850E 28998FE4 E2EB01DB EFBC9748
(1,0):   x:  6E0DC915 E6F3D0DF 47A4B3C6 1001D741 97F97283 E2594B5B
         y:  B1082337 A2BF139F 7BA963FD 2D846B18 4F1E9507 D1E5FC8D
(1,1):   x:  B37B888A 81D9F249 4AF1654C 6813F731 854A7E4D 663BFFD6
         y:  A0513D83 87804BA2 F1407476 38A73698 3F9EC980 99B5E7B8
(1,2):   x:  CD745D11 59EF28ED D02B8FF5 5B1253BE 1B3A8F30 6FE70863
         y:  74B81604 694AE037 4EFCF5B6 5F1CACAC 92352208 15B7C694
```

Alicia toma A, B y el bloque $\Pi^{(m)}$, y calcula la potencia k-ésima de $\begin{pmatrix} A & \Pi^{(m)} \\ & B \end{pmatrix}$:

$$\begin{pmatrix} A^k & \left(\Pi^{(m)}\right)^{(k)} \\ & B^k \end{pmatrix}$$

Bloque A^k:

```
(0,0):   81EE4064 73FCA136 B332B225 4D4BF5F9 FD1B92B8 77CCC476
(0,1):   03AA8D9E CF302885 7BCE03FD 3648CBBF 156D82DE BA253D42
(1,0):   775FD5F9 3BB915EB 6046AE02 70C35B72 6984270D EAD905DF
(1,1):   E8D72CD3 C4956294 E97DDE83 1C6CE94E A4408B45 165D35EB
```

Bloque B^k:

```
(0,0):   A4D8915F 0E91B52B 0274040E F917390B 6D5E2747 38A8092F
(0,1):   890248CC C4C3037A 6C0BADE9 E4F186D9 6843D412 E9B1AEF0
(0,2):   1810A63F F754E2AF 3CA8482A 88A0130E 578C215F EB8D8557
(1,0):   85065E39 A7A844C4 77E6F58C 88FB18A4 C72B5800 2587E822
(1,1):   546E431D 4C327DFE 60CFBF0C 0272A301 1D0DBD01 B5C51068
(1,2):   734F9597 187CA6E9 6018D673 2109FE54 8C08956C F6C6F5F2
(2,0):   08D5583A D57A5F02 7A460E4E 97E7E9B5 941D64CF 6D972AF3
(2,1):   276DA89B D0CB050B 6BF6E48E CA57113F 2A98EE9D 382ABF5D
```

```
(2,2):   FFC325BF C777940B 4B6C6E24 70D935AC 3FB0F346 9E5AD256
```

Bloque $\left(\Pi^{(m)}\right)^{(k)}$:

```
(0,0):   x:  3767F8A5 400748D6 8AC974FE 1E606CB7 952C61FA 536C4A2D
         y:  1381156E 0525D7FF 4C8A6CC7 21D8C34E C95C9F1E 05D28DC7
(0,1):   x:  28328BD2 5FFEFC00 493AAFE2 19E7AD19 D856C7C3 F7E6BDFA
         y:  D1813C1B 4DC48828 30104D1B 000B4867 43178DCC E5D31488
(0,2):   x:  D4E2AF3F 82ADD8B2 6E3FFCE5 7C2CC17E 02EF0773 4539A03A
         y:  92F78D44 1B7DEAE7 21F00166 9AA10AA3 59DFA2EB 63295911
(1,0):   x:  7FCFC695 32BF9CC3 70BFBA45 326CD7EA C8100603 BCE797ED
         y:  96DEDE6C 1EE0E850 E4EB0CE7 531D387D A76C478C 2E7A4F8D
(1,1):   x:  9B97BD85 8C80F5E4 E76206EE FD94657E E949CEB7 B9124547
         y:  3F0F1A01 E3849C1C D3E8497A 3C6F8E5E C6F1FD54 E86501BF
(1,2):   x:  FBCF270A AF21E25A 3F0EA505 B9AF9DFC 65377B16 167755DB
         y:  D81AC3D6 1390BC88 F3AAD8B7 A9C4B836 791C9871 2B1A601F
```

Bernardo toma A, B y el bloque $\Pi^{(k)}$, y calcula la potencia m-ésima de $\begin{pmatrix} A & \Pi^{(k)} \\ & B \end{pmatrix}$:

$$\begin{pmatrix} A^m & \left(\Pi^{(k)}\right)^{(m)} \\ & B^m \end{pmatrix}$$

Bloque A^m:

```
(0,0):   1ED1B8C3 E1893802 77FDE30B AD96AEC4 F1FD0D36 EA501612
(0,1):   570372FE D2E7ABC1 B62BF429 11898DDC C70EBC75 414CC56D
(1,0):   FC444ED3 F3FDA091 877671A7 D0382A2E DC4BA5A8 DC842B36
(1,1):   6436A38A C0D337D1 C303E6F5 D1C90EB1 44C308EC 319F5D31
```

Bloque B^m:

```
(0,0):   AD5E41BC 0CCE3BD9 82A887CE 84A945D6 2B0BCDA7 ECA921D4
(0,1):   3522B8A2 5B57879E 9DDEECAE 1D255942 230324A8 3E0EFA16
(0,2):   0BD0B0B7 6AC4A66F EBD035B3 558641A3 8BD103FA 0ED40588
(1,0):   0C3317E1 6E36AB9A 2D8E73C5 D965B86A 5CB5ACB6 2E3F261D
(1,1):   E72A08A1 EAD8D67F 4A857E8C 3EB1D5BA 5693F0B0 DDBF24EE
(1,2):   1A374C6E 6675B1E5 60ED0D63 E06A42C0 EA706006 E721CE88
(2,0):   261EF1FE BD60C92D 7FDA0155 B046DC63 97AC9938 2FFFAF5D
(2,1):   064C1B12 DE5D6840 4FC5735A 95F992C0 137BC006 D658EA0E
(2,2):   26A04588 14AD575B B92679E3 A4392DF7 B4D085BA 71D0E8EF
```

Bloque $\left(\Pi^{(k)}\right)^{(m)}$:

```
(0,0):   x:  3767F8A5 400748D6 8AC974FE 1E606CB7 952C61FA 536C4A2D
         y:  1381156E 0525D7FF 4C8A6CC7 21D8C34E C95C9F1E 05D28DC7
(0,1):   x:  28328BD2 5FFEFC00 493AAFE2 19E7AD19 D856C7C3 F7E6BDFA
         y:  D1813C1B 4DC48828 30104D1B 000B4867 43178DCC E5D31488
(0,2):   x:  D4E2AF3F 82ADD8B2 6E3FFCE5 7C2CC17E 02EF0773 4539A03A
         y:  92F78D44 1B7DEAE7 21F00166 9AA10AA3 59DFA2EB 63295911
(1,0):   x:  7FCFC695 32BF9CC3 70BFBA45 326CD7EA C8100603 BCE797ED
         y:  96DEDE6C 1EE0E850 E4EB0CE7 531D387D A76C478C 2E7A4F8D
(1,1):   x:  9B97BD85 8C80F5E4 E76206EE FD94657E E949CEB7 B9124547
         y:  3F0F1A01 E3849C1C D3E8497A 3C6F8E5E C6F1FD54 E86501BF
(1,2):   x:  FBCF270A AF21E25A 3F0EA505 B9AF9DFC 65377B16 167755DB
```

```
y :  D81AC3D6 1390BC88 F3AAD8B7 A9C4B836 791C9871 2B1A601F
```

Alicia y Bernardo toman $\left(\Pi^{(m)}\right)^{(k)}$ y $\left(\Pi^{(k)}\right)^{(m)}$, respectivamente, como secreto compartido ya que ambos bloques, como puede verse, son iguales.

A.4.3 Esquema de cifrado ElGamal

Bernardo toma de forma aleatoria un escalar m de 768 bits:

```
m :     F4C4CB50 F7EFAF88 CCC92323 53D6C279 9A5E9936 5F8618B9 \
        28E4F470 E8ECA2E8 BF942F7C 5B7C6CE0 FBAF1396 B0F33108 \
        8ACE49FC 41C52D36 49CE8D44 E69E4184 62478788 EF56D8DC \
        FC3F13B4 45B8B590 DAAA7059 41EBDD22 67241A30 51FF96EA
```

Calcula la potencia m-ésima de M y le envía el bloque $\Pi^{(m)}$ resultante a Alicia:

M^m

Bloque A^m:

```
(0,0):  498AD05A 2A7A270C 9EC62AB3 F09761F5 CFD32E02 0DA94179
(0,1):  1733F91D 236D96AC 973782B1 8B3C6423 1336AF40 693DD015
(1,0):  52732282 0ECFB266 7C06C030 4ED9BC0B 923D14B2 F2F5F158
(1,1):  5EA70996 0504D6CE D557382F 7AC5125B F8D75EDC 5DB5FB5F
```

Bloque B^m:

```
(0,0):  6BFB5B63 F4CAF1AD 15BC255F EEBDAFDE 8B858C71 C30475BF
(0,1):  6D7D414D 482A5A5C A5A824D1 DAEA60B5 D69B4209 AD04135D
(0,2):  3AC0BAD2 9808A715 002D917B AC1BC380 4952F5DA D7C3D85F
(1,0):  3D66B58D 62879164 152BA925 2E990C59 BA86C35A D952614D
(1,1):  2B97C0AB DEBBA850 C29B1CDB 00D7D596 8BA81FCF 6932D9C6
(1,2):  75D885B1 8D3A93F9 C2EFEDE8 6C4B886D C47B9C0E 27C0ED51
(2,0):  1EF727BC 115CD88A F3A85B52 817FAEC7 E1EC8347 E1E961E0
(2,1):  E2EC0C95 321A4C72 F53F91BB C6029278 92835D94 EB1B12C6
(2,2):  8AA4CFBE A862F4EC 63531B55 E14D4321 5AD856AA FE39901F
```

Bloque $\Pi^{(m)}$:

```
(0,0):  x :  F753659C 8964E184 790D347A 73B0B101 2C276DA8 85E86636
        y :  18E034E3 ACE4AE8B 5862FD84 84658AC2 FA7A2A5A 1B11EDBD
(0,1):  x :  DE585623 516B5F13 DFEB838A FD75C465 80BA60E8 5C355DB7
        y :  57DE6670 24299B7C DD5CD67F 696B2D44 1AB58DA1 8C828148
(0,2):  x :  C4431DC9 6E46A15E 947E9E94 6AEF35FC 6D96AC6C C88937A2
        y :  FD2A7B1B EB26CF5B 08B8090D D76D086A B1E94693 A6948028
(1,0):  x :  F327FDA1 3909BE53 2F325F2D BB3B6FB8 049FB8F9 ADF5443A
        y :  C24437B3 9377151E F8613240 6114BBB1 A1DBA5DC 536E6BA4
(1,1):  x :  5AFB3CAB 8E0EFB1C 90CDB08B 9F09CF2E 559ED4F4 D2F024CC
        y :  D5F6746A AADC5274 CF674D18 C688EA6D 4742F8E5 B9EF41D7
(1,2):  x :  EC12A1AA 9007A347 CC5E1DF7 0114A7DC D70F7D6F A84C78E6
        y :  BCAAF225 6435A526 D7A40060 D73BDF33 9270F4EE 550C853B
```

Alicia toma aleatoriamente un escalar k de 768 bits:

```
k :     259F27EF 940C22C0 94605B5A 12A1BE10 A097D5BE 6C3DCA08 \
```

```
AC9E61A0 726E3263 EEAEF3B6 967F0788 584D65D8 DA9B19DA \
30439DF8 8B23F78A 1E36E625 ECF9D0B8 E2982DF6 A89C6EBC \
1CEF0C04 2BC41338 EA560AD9 72EE0764 7EC0B500 0128C560
```

Eleva M a k y le envía el bloque $\Pi^{(k)}$ resultante a Bernardo:

M^k

Bloque A^k:

```
(0,0):   256939F1 D807D559 5F437B9B BB42DCEC CD705783 6B890610
(0,1):   5B0AD172 75EE52B9 98AB9864 0EE4F931 3D45713C CCD6203E
(1,0):   B50C780A 38D6C2B4 6BA66429 1864B950 DEA1CDCD 40D7C87A
(1,1):   21DBA04F B21FE47A 531DADDE 9D3DE476 8919A370 9DACBDCB
```

Bloque B^k:

```
(0,0):   CCF8C3E2 5F129941 FAAA041F D088CEF6 168B8A87 C91D1B78
(0,1):   799BBC66 86C0A0F2 E6CEFDF2 94C2498B D032B127 E6825021
(0,2):   0AA4F750 756896A5 899D6CC3 5B361F87 38156C71 3C266422
(1,0):   26F39991 0D952C07 83A4B6B2 07008EBF F13A29FD 693992DC
(1,1):   414483CF 4CF2692F 217E9B31 86AF5EC8 14F93E3B 3C0B2D69
(1,2):   4FCC2291 CD3D8850 0AA224A8 9E704CBA B60BC700 5034B02C
(2,0):   DD9FB601 3E3F9FA4 77E1C7D3 C82B4447 0652A64D 96AFC8DA
(2,1):   2E322711 4EE47525 B643635F 89BB1AD1 5A1D1F5F 352B115A
(2,2):   16E712C8 5E76C6DB 2D32C0FB 833CB4B1 74838C9C 2BC6D670
```

Bloque $\Pi^{(k)}$:

```
(0,0):   x:  7D1DF55A 367512E0 7050A69C 04448274 25667456 FEFD40A0
         y:  DE69E76E 5B9B3B90 B7DDD3A8 5D74031D A2D486DF 724D923C
(0,1):   x:  655FEF1D C951DB30 235CA2B0 35E57A27 1F40C9BE 79DD195B
         y:  34EFFBD1 43CD6555 FC664D86 59B146C2 86C83B22 CF9DB9AE
(0,2):   x:  4AE967B2 589B24AC ABD59778 50A594E6 030938F1 A63A71BE
         y:  B778CB4E 1ADCA2BF D6B7396D 256D9B82 2FD51933 B79E8186
(1,0):   x:  0AA4D195 6A35850A 65A2E697 CA7ACF8B BB372535 35930B89
         y:  702AF174 91CD321B 3588576E 24894AE0 A719B4A0 D6E8CC31
(1,1):   x:  7AB37C0B 88577B99 D98A2679 69BAEA2C B8FA859A 3600D800
         y:  E52AFD06 008EBD01 034A1C52 966F2D9C 37E4D806 A5239F77
(1,2):   x:  62FF3860 8C749ED0 526DD7B1 32255D5F 8FC35359 1DDC2BDE
         y:  8CD5DC9C 8F04931E D06BA9E3 99B35842 0833DC79 BB0EA465
```

Alicia toma A, B y el bloque $\Pi^{(m)}$, y calcula la potencia k-ésima de $\begin{pmatrix} A & \Pi^{(m)} \\ & B \end{pmatrix}$:

$$\begin{pmatrix} A^k & \left(\Pi^{(m)}\right)^{(k)} \\ & B^k \end{pmatrix}$$

Bloque A^k:

```
(0,0):   256939F1 D807D559 5F437B9B BB42DCEC CD705783 6B890610
(0,1):   5B0AD172 75EE52B9 98AB9864 0EE4F931 3D45713C CCD6203E
(1,0):   B50C780A 38D6C2B4 6BA66429 1864B950 DEA1CDCD 40D7C87A
(1,1):   21DBA04F B21FE47A 531DADDE 9D3DE476 8919A370 9DACBDCB
```

Bloque B^k:

```
(0,0):   CCF8C3E2 5F129941 FAAA041F D088CEF6 168B8A87 C91D1B78
```

```
(0,1):   799BBC66 86C0A0F2 E6CEFDF2 94C2498B D032B127 E6825021
(0,2):   0AA4F750 756896A5 899D6CC3 5B361F87 38156C71 3C266422
(1,0):   26F39991 0D952C07 83A4B6B2 07008EBF F13A29FD 693992DC
(1,1):   414483CF 4CF2692F 217E9B31 86AF5EC8 14F93E3B 3C0B2D69
(1,2):   4FCC2291 CD3D8850 0AA224A8 9E704CBA B60BC700 5034B02C
(2,0):   DD9FB601 3E3F9FA4 77E1C7D3 C82B4447 0652A64D 96AFC8DA
(2,1):   2E322711 4EE47525 B643635F 89BB1AD1 5A1D1F5F 352B115A
(2,2):   16E712C8 5E76C6DB 2D32C0FB 833CB4B1 74838C9C 2BC6D670
```

Bloque $\left(\Pi^{(m)}\right)^{(k)}$:

```
(0,0):   x:  2BA6F98C 278230AD 70791EBC 017AD169 B6264049 76C7EF5B
         y:  4D6C72C5 8BF6F163 62070F10 D673BD95 83DBBBBD 4C6F0212
(0,1):   x:  C57EC8FD A92DD51A E640303E F4EB4DCF C4F78039 5E020B78
         y:  F26BA006 000731C6 873E1CAB 30AEB215 C4D5D93D A65165AE
(0,2):   x:  DAE12544 844FB899 8E067F58 0885FE96 09122584 4B53F599
         y:  82CF4DEB B083E005 D826693A DCB29E96 F2D17A43 0FFA4E15
(1,0):   x:  0A4A298E E095411E 6F22817D A01558F0 C612DD6B 7D37A4CD
         y:  AEFF06B9 2A6C9C79 01669A43 91175C83 1F6B59F7 E899ABC2
(1,1):   x:  FFD1AD8C 579321FA 575A5775 6FC1750A 8EC9A73B 9A8A7732
         y:  47AA9CA7 FEFB2CFF ADD249FA D9FFB1FC 427CC3D8 DDED7BB8
(1,2):   x:  4160399F B1CA80E6 8D764B35 50BE807C 9B83E40B 6B76E566
         y:  119A73A4 F72BC894 88437CDB AAC3E098 66BB3102 8BAAF148
```

Alicia toma el mensaje codificado como bloque de 2×3 puntos (hemos tomado uno de forma aleatoria):

Ψ (mensaje a cifrar)

```
(0,0):   x:  BBE0D5F4 99A288D2 62CD818D 684160F0 BA99D9D6 088B3020
         y:  B9172DA2 EA0D334C 8D66511D 7631F572 00CB698C 6A887A30
(0,1):   x:  30B60C6A 2C00DCC0 56761A72 9C921EC0 0998FF48 CFF84B4C
         y:  6BBADB98 93F1105F B1256555 4157DA60 7671E173 9347384A
(0,2):   x:  C53A8D47 CC154130 4F3414A0 486290B4 D6F1AC99 77A3139A
         y:  F86F269D 3A5B65E6 A56B5521 A859A475 F12B94CA 00B7B5DB
(1,0):   x:  5F303800 67270936 566F626C 364BD308 70815F6A 42A2D790
         y:  F47F802E 7C052E0B E575C2E6 EC5316B4 083910A0 B8846A3E
(1,1):   x:  C99C2599 18CFBF65 AE4403D0 F6194498 165EC540 9F4550F8
         y:  62F64EBC 60E689A1 C25D303B 7DC9B1A0 98D262A4 A25FB5BF
(1,2):   x:  0BF60F40 A1145B12 D4E346B0 02D0CA24 6E933591 80A1AA5F
         y:  EE12AA0E D622E4F0 E167F46C D197EDAF 2B82F892 A7189DA1
```

Alicia toma $\left(\Pi^{(m)}\right)^{(k)}$ como clave secreta de cifrado, y calcula $\Omega=\left(\Pi^{(m)}\right)^{(k)}+\Psi$ para cifrar el mensaje, enviándoselo a Bernardo:

Ω (mensaje cifrado)

```
(0,0):   x:  3C28123E 8DA815D7 9BEC0E6B C5184B57 3A31B6AE 91BE8D78
         y:  5A32C467 BAF27F67 49662A63 D6DF50AD D679EFA5 6B3D05FD
(0,1):   x:  97741116 DBD3D695 146215F7 4728125B BE63EACF FFFE0FF9
         y:  45A97081 A74E0310 48D1B0D1 29C1FBDE 661E7F84 C63CC067
(0,2):   x:  F8CA9819 C0D1DCE0 0AF03D70 A362F5EE AC7B535B 4868FE99
         y:  8AEBADF0 7C61C9F6 806BDEEF E0569CF5 725EC0CB 50B8CD84
(1,0):   x:  0EC2F566 366C38C6 181AF9E9 A5EF5E1A 54B01C52 6D080E89
         y:  5A3B47B5 23AF6F48 631758D3 66CF368F 6CD6007B B9685857
```

```
(1,1):   x:  81B6ED1D CFFC3786 9B201EC5 290CE981 C23EE17E 6F09571E
         y:  67397979 0B47EC05 D638F425 643E4C01 93770D41 47E1B670
(1,2):   x:  680631EF F5023420 66CFC3DB FCDC60A9 B528ADD3 A436016C
         y:  90A6A547 A957CBA7 D9AF53D4 BD97E912 C3E5F3F6 0A9A076D
```

Bernardo toma A, B y el bloque $\Pi^{(k)}$, y calcula la potencia m-ésima de $\begin{pmatrix} A & \Pi^{(k)} \\ & B \end{pmatrix}$:

$$\begin{pmatrix} A^m & \left(\Pi^{(k)}\right)^{(m)} \\ & B^m \end{pmatrix}$$

Bloque A^m:

```
(0,0):   498AD05A 2A7A270C 9EC62AB3 F09761F5 CFD32E02 0DA94179
(0,1):   1733F91D 236D96AC 973782B1 8B3C6423 1336AF40 693DD015
(1,0):   52732282 0ECFB266 7C06C030 4ED9BC0B 923D14B2 F2F5F158
(1,1):   5EA70996 0504D6CE D557382F 7AC5125B F8D75EDC 5DB5FB5F
```

Bloque B^m:

```
(0,0):   6BFB5B63 F4CAF1AD 15BC255F EEBDAFDE 8B858C71 C30475BF
(0,1):   6D7D414D 482A5A5C A5A824D1 DAEA60B5 D69B4209 AD04135D
(0,2):   3AC0BAD2 9808A715 002D917B AC1BC380 4952F5DA D7C3D85F
(1,0):   3D66B58D 62879164 152BA925 2E990C59 BA86C35A D952614D
(1,1):   2B97C0AB DEBBA850 C29B1CDB 00D7D596 8BA81FCF 6932D9C6
(1,2):   75D885B1 8D3A93F9 C2EFEDE8 6C4B886D C47B9C0E 27C0ED51
(2,0):   1EF727BC 115CD88A F3A85B52 817FAEC7 E1EC8347 E1E961E0
(2,1):   E2EC0C95 321A4C72 F53F91BB C6029278 92835D94 EB1B12C6
(2,2):   8AA4CFBE A862F4EC 63531B55 E14D4321 5AD856AA FE39901F
```

Bloque $\left(\Pi^{(k)}\right)^{(m)}$:

```
(0,0):   x:  2BA6F98C 278230AD 70791EBC 017AD169 B6264049 76C7EF5B
         y:  4D6C72C5 8BF6F163 62070F10 D673BD95 83DBBBBD 4C6F0212
(0,1):   x:  C57EC8FD A92DD51A E640303E F4EB4DCF C4F78039 5E020B78
         y:  F26BA006 000731C6 873E1CAB 30AEB215 C4D5D93D A65165AE
(0,2):   x:  DAE12544 844FB899 8E067F58 0885FE96 09122584 4B53F599
         y:  82CF4DEB B083E005 D826693A DCB29E96 F2D17A43 0FFA4E15
(1,0):   x:  0A4A298E E095411E 6F22817D A01558F0 C612DD6B 7D37A4CD
         y:  AEFF06B9 2A6C9C79 01669A43 91175C83 1F6B59F7 E899ABC2
(1,1):   x:  FFD1AD8C 579321FA 575A5775 6FC1750A 8EC9A73B 9A8A7732
         y:  47AA9CA7 FEFB2CFF ADD249FA D9FFB1FC 427CC3D8 DDED7BB8
(1,2):   x:  4160399F B1CA80E6 8D764B35 50BE807C 9B83E40B 6B76E566
         y:  119A73A4 F72BC894 88437CDB AAC3E098 66BB3102 8BAAF148
```

Bernardo toma $\left(\Pi^{(k)}\right)^{(m)}$ como clave secreta de descifrado y calcula $\Psi' = \Omega - \left(\Pi^{(k)}\right)^{(m)}$ para descifrar el mensaje:

Ψ' (mensaje recuperado)

```
(0,0):   x:  BBE0D5F4 99A288D2 62CD818D 684160F0 BA99D9D6 088B3020
         y:  B9172DA2 EA0D334C 8D66511D 7631F572 00CB698C 6A887A30
(0,1):   x:  30B60C6A 2C00DCC0 56761A72 9C921EC0 0998FF48 CFF84B4C
         y:  6BBADB98 93F1105F B1256555 4157DA60 7671E173 9347384A
(0,2):   x:  C53A8D47 CC154130 4F3414A0 486290B4 D6F1AC99 77A3139A
```

```
          y:  F86F269D 3A5B65E6 A56B5521 A859A475 F12B94CA 00B7B5DB
(1,0):    x:  5F303800 67270936 566F626C 364BD308 70815F6A 42A2D790
          y:  F47F802E 7C052E0B E575C2E6 EC5316B4 083910A0 B8846A3E
(1,1):    x:  C99C2599 18CFBF65 AE4403D0 F6194498 165EC540 9F4550F8
          y:  62F64EBC 60E689A1 C25D303B 7DC9B1A0 98D262A4 A25FB5BF
(1,2):    x:  0BF60F40 A1145B12 D4E346B0 02D0CA24 6E933591 80A1AA5F
          y:  EE12AA0E D622E4F0 E167F46C D197EDAF 2B82F892 A7189DA1
```

Como puede verse, Bernardo recupera el mensaje original.

Bibliografía

[1] ANSI X9.62, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), American National Standards Institute, 1999.

[2] ANSI X9.63, Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography, American National Standards Institute, 2001.

[3] R. BALASUBRAMANIAN, N. KOBLITZ, "The improbability that an elliptic curve has subexponential discrete log problem under the Menezes-Okamoto-Vanstone algorithm", *Journal of Cryptology*, 11: 141-145, 1998.

[4] M. BELLARE, R. CANETTI, H. KRAWCZYK, "A modular approach to the design and analysis of authentication and key exchange protocols", *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing*, 1998.

[5] I. BLAKE, G. SEROUSSI, N. SMART, *Elliptic Curves in Cryptography*, London Mathematical Society Lecture Note Series 265, Cambridge University Press, 1999.

[6] S. BLAKE-WILSON, A. MENEZES, "Entity authentication and authenticated key transport protocols employing asymmetric techniques", *Proceedings of the 5th International Workshop on Security Protocols*, Lecture Notes in Computer Science, 1361: 137-158, Springer-Verlag, 1997.

[7] D. BLEICHENBACHER, "Generating ElGamal signatures without knowing the secret key", *Advances in Cryptology – EUROCRYPT '96*, Lecture Notes in Computer Science, 1070: 10-18, Springer-Verlag, 1996.

[8] D. BONEH, R. DEMILLO, R. LIPTON, "On the importance of checking cryptographic protocols for faults", *Advances in Cryptology – EUROCRYPT '97*, Lecture Notes in Computer Science, 1233: 37-51, Springer-Verlag, 1997.

[9] A. BOSSELAERS, R. GOVAERTS, J. VANDEWALLE, "Comparison of three modular reduction functions", *Advances in Cryptology – CRYPTO '93*, Lecture Notes in Computer Science, 773: 175-186, Springer-Verlag, 1994.

[10] A. BOSSLAERS, H. DOBBERTIN, B. PRENEEL, "The RIPEMD-160 Cryptographic Hash Function", *Dr. Dobb's Journal*, enero 1997.

[11] E. BRICKELL, D. GORDON, K. MCCURLEY, D. WILSON, "Fast exponentiation with precomputation", *Advances in Cryptology – EUROCRYPT '92*, Lecture Notes in Computer Science, 658: 200-207, 1993.

[12] M. BROWN, D. HANKERSON, J. LÓPEZ, A. MENEZES, "Software implementation of the NIST elliptic curves over prime fields", *Centre of Applied Cryptographic Research, University of Waterloo*, Technical report CORR 2000-56, 2000.

[13] CERTICOM, "Remarks on the security of the elliptic curve cryptosystem", Certicom whitepaper, 1997.

[14] CERTICOM, "The Certicom ECC Challenge", disponible en el sitio web de Certicom: http://www.certicom.com/download/aid-111/cert_ecc_challenge.pdf

[15] D. CHUDNOVSKY, G. CHUDNOVSKY, "Sequences of numbers generated by addition in formal groups and new primality and factorization tests", *Advances in Applied Mathematics*, 7: 385-434, 1987.

[16] J.-J. CLIMENT, F. FERRÁNDEZ, "A new cryptosystem based on elliptic curves and polynomial matrices", *WSEAS Transactions on Business and Economics*, 1(3): 261-265, 2004.

[17] J.-J. CLIMENT, F. FERRÁNDEZ, "A new cryptosystem based on elliptic curves and polynomial matrices", *Electronic Proceedings of the WSEAS Conference*, C. Manikopoulos; B. Tafaghodina; L. Simoni; D. Politis; V. Kluev; A. Genco; J. Quadrado; C. D'Attelis; N. Mastorakis (editores), noviembre 2004.

[18] J.-J. CLIMENT, F. FERRÁNDEZ, "A nonlinear cryptosystem based on elliptic curves", *WSEAS Transactions on Business and Economics*, 1(3): 253-260, 2004.

[19] J.-J. CLIMENT, F. FERRÁNDEZ, "A nonlinear cryptosystem based on elliptic curves", *Electronic Proceedings of the WSEAS Conference*, C. Manikopoulos; B. Tafaghodina; L. Simoni; D. Politis; V. Kluev; A. Genco; J. Quadrado; C. D'Attelis; N. Mastorakis (editores), noviembre 2004.

[20] J.-J. CLIMENT, F. FERRÁNDEZ, L. TORTOSA, "New cryptographic protocols based on elliptic curves and polynomial matrices" (en proceso de evaluación).

[21] J.-J. CLIMENT, F. FERRÁNDEZ, J. VICENT, A. ZAMORA, "A nonlinear elliptic curve cryptosystem based on matrices" (en proceso de evaluación).

[22] H. COHEN, *A Course in Computational Algebraic Number Theory*, Springer-Verlag, Graduate Texts in Mathematics 138, 1993.

[23] H. COHEN, A. MIYAJI, T. ONO, "Efficient elliptic curve exponentiation using mixed coordinates", *Advances in Cryptology – ASIACRYPT '98*, Lecture Notes in Computer Science, 1514: 51-65, 1998.

[24] E. DE WIN, S. MISTER, B. PRENEEL, M. WIENER, "On the performance of signature schemes based on elliptic curves", *Algorithmic Number Theory*, Lecture Notes in Computer Science, 1423: 252-266, Springer-Verlag, 1998.

[25] H. DELFS, H. KNEBL, Introduction to Cryptography: Principles and Applications, Springer-Verlag, Berlin, 2002.

[26] W. DIFFIE, M. HELLMAN, "New directions in cryptography", *IEEE Transactions on Information Theory*, 22(6): 644-654, 1976.

[27] T. ELGAMAL, "A public key cryptosystem and a signatute scheme based on discrete logarithms", *IEEE Transactions on Information Theory*, 31: 469-472, 1985.

[28] N. ELKIES, "Elliptic and modular curves over finite fields and related computational issues", In *Computational Perspectives on Number Theory: Proceedings of a Conference in Honor of A.O.L. Atkin*, American Mathematical Society International Press, 7: 21-76, 1998.

[29] A. ESCOTT, J. SAGER, A. SELKIRK, D. TSAPAKIDIS, "Attacking elliptic curve cryptosystems using the parallel Pollard rho method", *CryptoBytes – The Technical Newsletter of RSA Laboratories*, 4(2): 15-19, 1999.

[30] G. FREY, H. RÜCK, "A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves", *Mathematics of Computation*, 62: 865-874, 1994.

[31] S. FUMY, P. LANDROCK, "Principles of key management", *IEEE Journal on Selected Areas in Communications* 11(5): 785-793, 1993.

[32] S. GALBRAITH, N. SMART, "A cryptographic application of Weil descent", *Codes and Cryptography,* Lecture Notes in Computer Science, 1746: 191-200, Springer-Verlag, 1999.

[33] R. GALLANT, R. LAMBERT, S. VANSTONE, "Improving the parallelized Pollard lambda search on binary anomalous curves", *Mathematics of Computation*, 69: 1699-1705, 2000.

[34] P. GAUDRY, F. HESS, N. SMART, "Constructive and destructive facets of Weil descent on elliptic curves", *Journal of Cryptology*, 15: 19-46, 2002.

[35] D. GORDON, "A survey of fast exponentiation methods", *Journal of Algorithms*, 27: 129-146, 1998.

[36] D. GORDON, "Discrete logarithms in GF(p) using the number field sieve", *SIAM Journal on Discrete Mathematics*, 6: 124-138, 1993.

[37] D. HANKERSON, J. LÓPEZ, A. MENEZES. "Software implementation of elliptic curve cryptography over binary fields", *Centre of Applied Cryptographic Research, University of Waterloo*, Technical report CORR 2000-42, 2000.

[38] IEEE Std 1363-2000, IEEE Standard specifications for public key cryptography, 2000.

[39] ISO/IEC 14888-3, *Information technology – Security techniques – Digital signatures with appendix*, Part 3: Certificate-based mechanisms, 1998.

[40] ISO/IEC 15946, *Information Technology – Security Techniques – Cryptography Techniques Based on Elliptic Curves*, Part 1: General (2002), Part 2: Digital Signatures (2002), Part 3: Key Establishment (2002), Part 4: Digital Signatures Giving Message Recovery (2004), 2004.

[41] T. IZU, J. KOGURE, M. NORO, K. YOKOYAMA, "Efficient implementation of Schoof's algorithm", *Advances in Cryptology – ASIACRYPT '98,* Lecture Notes in Computer Science, 1514: 66-79, Springer-Verlag, 1999.

[42] M. JACOBSON, N. KOBLITZ, J. SILVERMAN, A. STEIN, E. TESKE, "Analysis of the xedni calculus attack", *Designs, Codes and Cryptography*, 20: 41-64, 2000.

[43] D. JOHNSON, A. MENEZES, "The Elliptic Curve Digital Signature Algorithm (ECDSA)", *Centre of Applied Cryptographic Research, University of Waterloo*, Technical report CORR 99-34, 1999.

[44] A. JOUX, R. LERCIER, "Improvements to the general number field sieve for discrete logarithms in prime fields", *Mathematics of Computation*, 72(242): 953-967, 2003.

[45] D. KNUTH, The Art of Computer Programming – Seminumerical Algorithms, 3rd edition, Addison-Wesley, 1998.

[46] N. KOBLITZ, *A Course in Number Theory and Cryptography*, Springer-Verlag, New York, 1987.

[47] N. KOBLITZ, "CM-curves with good cryptographic properties", *Advances in Cryptology – CRYPTO '91*, Lecture Notes in Computer Science, 576: 279-287, Springer-Verlag, 1992.

[48] N. KOBLITZ, "Constructing elliptic curve cryptosystems in characteristic 2", In *Advances in Cryptology – CRYPTO '90*, Lecture Notes in Computer Science, 537: 156-167, Springer-Verlag, 1991.

[49] N. KOBLITZ, "Elliptic curve cryptosystems", *Mathematics of Computation*, 48: 203-209, 1987.

[50] N. KOBLITZ, A. MENEZES, S. VANSTONE, "The state of elliptic curve cryptography", *Designs, Codes and Cryptography*, 19: 173-193, 2000.

[51] P. KOCHER, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems", *Advances in Cryptology – CRYPTO '96*, Lecture Notes in Computer Science, 1109: 104-113, Springer-Verlag, 1996.

[52] R. KUHN, F. STRUIK, "Random walks revisited: extensions of Pollard's rho algorithm for computing multiple discrete logarithms", In *Selected Areas in Cryptography – SAC 2001*, Lecture Notes in Computer Science, 2259: 212-219, Springer, Berlin, 2001.

[53] G. LAY, H. ZIMMER, "Constructing elliptic curves with given group order over large finite fields", In *ANTS-1: Algorithmic Number Theory*, Lecture Notes in Computer Science, 877: 250-263, Springer-Verlag, 1994.

[54] R. LERCIER, F. MORAIN, "Counting the number of points on elliptic curves over finite fields: strategies and performances", *Advances in Cryptology – EUROCRYPT '95,* Lecture Notes in Computer Science, 921: 79-94, Springer-Verlag, 1995.

[55] R. LIDL, H. NIEDERREITER, *Introduction to Finite Fields and their Applications*, Cambridge University Press, 1994.

[56] J. MASSEY, J. OMURA, "Method and apparatus for maintaining the privacy of digital messages conveyed by public transmission", *U.S. Patent* # 4,567,600, 1986.

[57] R. MCELIECE, *Finite Fields for Computer Scientists and Engineers*, Kluwer Academic Publishers, Boston, 1987.

[58] J. MCKEE, "Subtleties in the distribution of the numbers of points on elliptic curves over a finite prime field", *Journal of the London Mathematical Society*, 59: 448-460, 1999.

[59] W. MEIER, O. STAFFELBACH, "Efficient multiplication on certain nonsupersingular elliptic curves", *Advances in Cryptology – CRYPTO '92,* Lecture Notes in Computer Science, 740: 333-344, Springer-Verlag, 1993.

[60] A. MENEZES, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, Boston, 1993.

[61] A. MENEZES, T. OKAMOTO, S. VANSTONE, "Reducing elliptic curve logarithms to logarithms in a finite field", *IEEE Transactions on Information Theory*, 39: 1639-1646, 1993.

[62] A. MENEZES, P. OORSCHOT, S. VANSTONE, *Handbook of Applied Cryptography*, CRC Press, New York, 1997.

[63] A. MENEZES, S. VANSTONE, "A note on cyclic groups, finite fields, and the discrete logarithm problem", *Applicable Algebra in Engineering, Communication and Computing*, 3: 67-74, 1992.

[64] A. MENEZES, Y. WU, "The Discrete Logarithm Problem in GL(n,q)", *Ars Combinatoria*, 47: 23-32, 1998.

[65] V. MILLER, "Uses of elliptic curves in cryptography", *Advances in Cryptology – CRYPTO 85*, Lecture Notes in Computer Science, 218: 417-426, Springer-Verlag, 1986.

[66] F. MORAIN, "Building cyclic elliptic curves modulo large primes", *Advances in Cryptology – EUROCRYPT* '91. Lecture Notes in Computer Science, 547: 328-336, Springer-Verlag, 1991.

[67] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, NIST FIPS PUB 180-1, "Secure Hash Standard", *Federal Information Processing Standards Publication* 180-1, U.S. Department of Commerce, Abril 1995.

[68] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, NIST FIPS PUB 186, "Digital Signature Standard", *Federal Information Processing Standards Publication* 186, U.S. Department of Commerce, mayo 1994.

[69] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, NIST FIPS PUB 186-2, "Digital Signature Standard (DSS)", *Federal Information Processing Standards Publication* 186-2, 2000.

[70] K. NYBERG, R. RUEPPEL, "A new signature scheme based on the DSA giving message recovery", *1st ACM Conference on Computer and Communications Security*, ACM Press, 58-61, 1993.

[71] A. ODLYZKO, "Discrete logarithms in finite fields and their cryptographic significance", *Advances in Cryptology – EUROCRYPT '84*, Lecture Notes in Computer Science, 209: 224-314, Springer-Verlag, 1985.

[72] A. ODLYZKO, "The future of integer factorization", *CryptoBytes – The technical newsletter of RSA Laboratories*, 1(2): 5-12, 1995.

[73] R. ODONI, V. VARADHARAJAN, R. SANDERS, "Public key distribution in matrix rings", *Electronics Letters*, 20: 386-387, 1984.

[74] R. OORSCHOT, M. WIENER, "Parallel collision search with cryptanalytic applications", *Journal of Cryptology*, 12: 1-28, 1999.

[75] S. POHLIG, M. HELLMAN, "An improved algorithm for computing logarithms over GF(p) and its cryptographic significance", *IEEE Transactions on Information Theory*, 24: 106-110, 1978.

[76] D. POINTCHEVAL, J. STERN, "Security proofs for signature schemes", *Advances in Cryptology – EUROCRYPT '96*, Lecture Notes in Computer Science, 1070: 387-398, Springer-Verlag, 1993.

[77] J. POLLARD, "Monte Carlo methods for index computation mod p", *Mathematics of Computation*, 32: 918-924, 1978.

[78] M. RABIN, "Digitalized signatures and public-key function as intractable as factorization", *MIT/LCS/TR-212, MIT Laboratory for Computer Science*, 1979.

[79] R. RIVEST, "The MD5 Message Digest Algorithm", Request For Comments 1321, *Internet Engineering Task Force*, abril 1992.

[80] R. RIVEST, A. SHAMIR, L. ADLEMAN, "A method for obtaining digital signatures and public key cryptosystems", *Communications of the ACM*, 21: 120-126, 1978.

[81] T. SATOH, K. ARAKI, "Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves", *Comentarii Mathematici Universitatis Sancti Pauli*, 47: 81-92, 1998.

[82] B. SCHNEIER, *Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C*, NY: John Wiley & Sons, New York, 1996.

[83] C. SCHNORR, "Efficient signature generation for smart cards", *Journal of Cryptology*, 4(3): 161-174, 1991.

[84] R. SCHOOF, "Counting points on elliptic curves over finite fields", *J. Théorie des Nombres de Bordeaux*, 7: 219-254, 1995.

[85] R. SCHOOF, "Elliptic curves over finite fields and the computation of square roots mod p", *Mathematics of Computation*, 44: 483-494, 1985.

[86] M. SCHROEDER, *Number Theory in Science and Communication*, Springer Series in Information Sciences, 7, Springer-Verlag, Berlin, 1986.

[87] R. SCHROEPPEL, H. ORMAN, S. O'MALLEY, O. SPATSCHECK, "Fast key exchange with elliptic curve systems", *Advances in Cryptology – CRYPTO '95*, Lecture Notes in Computer Science, 963: 43-56, Springer-Verlag, 1995.

[88] I. SEMAEV, "Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p", *Mathematics of Computation*, 67: 353-356, 1998.

[89] SETI, sitio web del proyecto: http://www.setiathome.ssl.berkeley.edu/

[90] A. SHAMIR, "Factoring large numbers with the TWINKLE device", In proceedings of *Cryptographic Hardware and Embedded Systems: First International Workshop, CHES'99*, Lecture Notes in Computer Science, 1717: 2-12, Springer-Verlag, 1999.

[91] J. SILVERMAN, *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics 106, Springer-Verlag, New York, 1986.

[92] J. SILVERMAN, "The xedni calculus and the elliptic curve discrete logarithm problem", *Designs, Codes and Cryptography*, 20: 5-40, 2000.

[93] J. SILVERMAN, J. SUZUKI, "Elliptic curve discrete logarithm problem and the index calculus", *Advances in Cryptology – ASIACRYPT '98*, Lecture Notes in Computer Science, 1514: 110-125, Springer-Verlag, 1999.

[94] R. SILVERMAN, "An analysis of Shamir's factoring device", *RSA Laboratories Bulletin*, 3, May 3, 1999.

[95] R. SILVERMAN, J. STAPLETON, Contribution to ANSI X9F1 working group, 1997.

[96] G. SIMMONS, *Contemporary Cryptology: The Science of Information Integrity*, NJ: IEEE Press, Piscataway, 1992.

[97] G. SIMMONS, "Cryptology", *Encyclopaedia Britannica*, 15ª ed., 1993.

[98] S. SINGH, *Los Códigos Secretos*, Debate, Madrid, 2000.

[99] N. SMART, "The discrete logarithm problem on elliptic curves of trace one", *Journal of Cryptology*, 12: 193-196, 1999.

[100] J. SOLINAS, "An improved algorithm for arithmetic on a family of elliptic curves", *Advances in Cryptology – CRYPTO '97*, Lecture Notes in Computer Science, 1294: 357-371, Springer-Verlag, 1997.

[101] J. SOLINAS, "Efficient arithmetic on Koblitz curves", *Designs, Codes and Cryptography*, 19: 195-249, 2000.

[102] J. SOLINAS, "Generalized Mersenne numbers", *Centre for Applied Cryptographic Research, University of Waterloo*, Technical report CORR 1999-39, 1999.

[103] W. STALLINGS, *Cryptography and Network Security*, Prentice Hall, Upper Saddle River, 2003.

[104] D. STINSON, *Cryptography: Theory and Practice*, CRC Press, Boca Raton, 1995.

[105] E. TESKE, "Square-Root algorithms for the discrete logarithm problem (a survey)", *Centre for Applied Cryptographic Research, University of Waterloo*, Technical report CORR 2001-07, 2001.

[106] L. WASHINGTON, *Elliptic Curves: Number Theory and Cryptography*, CRC Press, Boca Raton, 2003.

[107] M. WELSCHENBACH, *Cryptography in C and C++*, Apress, Berkeley, 2001.

[108] M. WIENER, R. ZUCCHERATO, "Faster attacks on elliptic curve cryptosystems", *Selected Areas in Cryptography*, Lecture Notes in Computer Science, 1556: 190-200, Springer-Verlag, 1999.

UNIVERSIDAD DE ALICANTE

Comisión de Doctorado

Reunido el Tribunal que suscribe en el día de la fecha acordó otorgar, por unanimidad a la Tesis Doctoral de Don/Dña. Francisco Antonio Ferrández Agulló la calificación de Sobresaliente "cum laude" (10)

Alicante 13 de mayo de 2005

El Presidente, El Secretario,

UNIVERSIDAD DE ALICANTE

Comisión de Doctorado

La presente Tesis de D. Francisco Antonio Ferrández Agulló ha sido registrada al Folio con el n.º 1077-56P del registro de entrada correspondiente.

Alicante 13 de Mayo de 2005

El Encargado del Registro,