Universitat d'Alacant
Universidad de Alicante

Propuesta y análisis de criptosistemas de clave pública basados en matrices triangulares superiores por bloques.

José Francisco Vicent Francés

Tesis Doctorales

www.eltallerdigital.com

UNIVERSIDAD de ALICANTE

Universidad de Alicante

Departamento de Ciencia de la Computación e Inteligencia Artificial

Propuesta y Análisis de Criptosistemas de Clave Pública Basados en Matrices Triangulares Superiores por Bloques

TESIS DOCTORAL

Presentada por:
José Francisco Vicent Francés

Dirigida por:
Dr. Antonio Zamora Gómez
Dr. Leandro Tortosa Grau

Dedicatoria

A Mari Angeles y Verónica.

A Ramón y María.

A Antonio y Leandro.

Agradecimientos

Muchas veces me he planteado la opción de comenzar y acabar esta sección únicamente con la palabra “Gracias.”... y nada más. Sencillamente por evitar el resumir en una sola página la mención de tantas personas a las que debo mucho.

Sería absurdo tratar de medir o concretar mi agradecimiento a Mari Angeles y a mi hija Verónica por su ayuda, apoyo incondicional, cariño y comprensión a lo largo de los años. En cada ejemplo, en cada línea y en cada palabra podría indicar la influencia de su apoyo, de sus sugerencias y de su compañía. Sólo espero ser capaz de devolverles algún día lo mucho que me han dado.

Al doctor Antonio Zamora, fuente de motivación y al que considero mi maestro y ejemplo de dedicación. Al fin y al cabo, la mera existencia de esta memoria se debe a su persistencia (y también paciencia). Durante estos años me ha demostrado su calidad como tutor y, sobre todo, su valía como persona.

Al doctor Leandro Tortosa por su asesoramiento científico, esfuerzo, dedicación y excelente dirección.

A mis padres, Ramón y María, siempre preocupados por mi felicidad, alentadores y reconfortantes en los momentos bajos y a los que es imposible pagar su cariño como se merecen. Es a ellos a quienes debo cuanto soy y agradezco de corazón su esfuerzo, apoyo y dedicación a lo largo de toda mi formación académica y profesional.

A mi hermanos, Ramón, Octavio y Marian, cómplices, consejeros y amigos, tanto en la infancia como en la madurez.

A los compañeros de departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Alicante por su amable e incondicional ayuda en la preparación de esta tesis.

A todos mis amigos, por confiar en mí pese a la distancia y los años.

Índice

Índice de figuras

Índice de tablas

Prólogo

Hasta hace pocos años la criptografía sólo resultaba interesante para agencias de seguridad, gobiernos, grandes empresas y delincuentes. Sin embargo, en poco tiempo, y debido al rápido crecimiento de las comunicaciones electrónicas, esta ciencia se ha convertido en un tema sugerente que despierta el interés del público en general. Destaca especialmente el cambio que ha sufrido la investigación en criptografía en el último tercio del pasado siglo, ya que ha pasado del tema clásico del cifrado y su seguridad a los más actuales campos de las firmas digitales y los protocolos criptográficos. Dicha variación es una consecuencia inmediata del impacto de las nuevas tecnologías en la sociedad, que cada vez demanda más servicios telemáticos seguros. Así, ante las situaciones de peligro nacidas a raíz de los nuevos servicios, se hacen necesarias soluciones diferentes.

La aparición de la criptografía asimétrica o de clave pública ha permitido que se desarrollen una serie de nuevas tecnologías como firma digital, autentificación de usuarios y el cifrado de datos sin intercambio previo de secretos (clave privada), que es muy importante en intranets, extranets y en general en comercio electrónico desde canales inseguros como Internet.

La tesis consta de cuatro capítulos. El primero: "Fundamentos y preliminares", es una introducción a los principios elementales de la criptografía. En él, se aborda en primer lugar una introducción histórica, después unos conceptos básicos tanto de seguridad informática como de criptografía, para acabar con una serie de herramientas matemáticas como son: grandes números, teoría de la información, operaciones bit, complejidad computacional, teoría de la divisibilidad, teoría de grupos y aritmética modular.

El segundo capítulo: "Criptosistemas de clave pública", presenta en profundidad y con formalismo matemático, la criptografía de clave pública, aplicada a esquemas basados en factorización (RSA) y a esquemas basados en el logaritmo discreto (Diffie-

Hellman, ElGamal, Massey-Omura).

El tercer capítulo: "Criptosistemas asimétricos basados en matrices triangulares superiores por bloques", muestra los distintos criptosistemas de clave pública propuestos, basados en matrices triangulares superiores por bloques con elementos en $\mathbb{Z}_p$ con p primo. Comienza con un estudio del criptosistema de Diffie-Hellman aplicado a dichas matrices, después se presenta un esquema aditivo, llamado así por basarse en una propiedad aditiva de las matrices triangulares, para seguidamente desarrollar un criptosistema modificado de Diffie-Hellman. El último esquema propuesto es el multiplicativo, que solventa todas las debilidades de los anteriores y presenta unos tiempos de ejecución equiparables a RSA con unos niveles de seguridad similares teniendo la ventaja de no usar número primos, con el consiguiente ahorro en análisis de primalidad. La sección siguiente estudia y desarrolla una versión optimizada del esquema multiplicativo y hace una comparativa con esquemas existentes. Para finalizar este apartado, y debido a la utilización en todos los esquemas de grandes potencias de matrices, se presenta un algoritmo de exponenciación rápida, aplicado a las matrices triangulares superiores por bloques.

El último capítulo expone las conclusiones derivadas del estudio y pretende abrir nuevas líneas de investigación futuras.

Se cierra la tesis con una serie de apéndices donde se describen ejemplos numéricos de los sucesivos criptosistemas descritos; tablas numéricas del tamaño del mínimo común múltiplo y del máximo común divisor; así como tablas de tiempos para la elección de la versión óptima del esquema multiplicativo y tiempos para la comparativa con tres criptosistemas estándar.

Capítulo 1

Fundamentos y preliminares

1.1 Introducción histórica

Desde los tiempos más remotos, las personas han utilizado diversos métodos con el fin de lograr que un mensaje no llegara a manos de personas no autorizadas a leerlo (véanse [10, 11, 28, 34, 64, 114]).

Algunos de los testimonios más antiguos sobre la ocultación de la escritura que se conocen se remontan a Herodoto, quien hizo una crónica de los conflictos entre Grecia y Persia en el siglo V a.C. Fue este método el que salvó a Grecia de ser ocupada por Jerjes, Rey de Reyes persa. Herodoto cuenta en su crónica que Demarato, un griego exiliado en la ciudad Persa de Susa, tuvo conocimiento de los preparativos de Jerjes para atacar Grecia y decidió alertar a los espartanos mediante un mensaje oculto en tablillas de madera. El método de ocultación consistió en retirar la cera de las tablillas, escribir el mensaje y luego volver a cubrir con cera.

Herodoto narró también otro incidente en el que la ocultación fue suficiente para conseguir el paso seguro de un mensaje. La crónica cuenta la historia de Histaiaeo, quien pretendía alentar a Aristágoras de Mileto para que se rebelara contra el rey de Persia. Para transmitir sus instrucciones de forma segura, Histaiaeo afeitó la cabeza de su mensajero, escribió el mensaje en su cuero cabelludo y luego esperó a que volviera a crecer el pelo. Al llegar a su destino, el mensajero se afeitó la cabeza y se la mostró al receptor a quien iba destinado el mensaje.

En los dos mil años que han transcurrido desde Herodoto, diversas formas de ocultación de mensajes han sido utilizadas por todo el mundo. Por ejemplo, en la China antigua se escribían mensajes sobre seda fina, que luego era aplastada hasta formar una pelotita diminuta que se recubría de cera. Entonces, el mensajero se tragaba la bola de cera. En el siglo XV, el científico italiano Giovanni Porta describió una antigua técnica

para esconder mensajes dentro de un huevo cocido haciendo una tinta con una mezcla de una onza de alumbre y una pinta de vinagre, y luego escribiendo en la cáscara. La solución penetra a través de la cáscara porosa y deja un mensaje en la superficie de la albúmina del huevo duro, que sólo se puede leer si se pela el huevo.

La ocultación de la información ofrece sin duda un nivel de seguridad, pero adolece de una debilidad fundamental. Si se descubre el mensaje, el contenido de la comunicación secreta se revela en el acto, es decir, interceptar el mensaje compromete inmediatamente toda la seguridad. Por eso, paralelamente al desarrollo de la ocultación de la información, se produjo la evolución de la criptografía, término derivado de los vocablos griegos "kryptos" que significa oculto y "graphein" que significa escritura. El objetivo de la criptografía no es ocultar la existencia de un mensaje, sino ocultar su significado, un proceso que se conoce como cifrado. La ventaja de la criptografía es que si el enemigo intercepta un mensaje cifrado, éste es ininteligible.

La criptografía clásica utiliza fundamentalmente técnicas de transposición y sustitución. En la transposición, las letras del mensaje simplemente se colocan de otra manera, generando así un anagrama. Para que esta transposición sea efectiva la combinación de letras necesita seguir un sistema sencillo, establecido previamente por el emisor y el receptor. Como ejemplo se tiene la transposición de "riel", en la que el mensaje se escribe alternando las letras en dos líneas separadas:

tu secreto es tu prisionero; si lo sueltas, tu eres su prisionero

t s c e o s u r s o e o i i u l a t e e s p i i n r
u e r t e t p i i n r s l s e t s u r s u r s o e o.

Otra forma de transposición es la producida en el primer aparato criptográfico militar de la Historia, el escitalo espartano (siglo V a.C.). Se trata de una vara de madera sobre la que se enrosca una tira de cuero o pergamino, tal como se muestra en la figura 1.1. El emisor escribe el mensaje a lo largo de la longitud del escitalo y luego desenrosca la tira, que ahora parece llevar una lista de letras sin sentido. Para recuperar el mensaje, el receptor simplemente enrosca la tira de cuero en torno a un escitalo del mismo diámetro que el usado por el emisor.

La alternativa a la transposición es la sustitución. Una de las descripciones más antiguas de cifrado por sustitución aparece en el Kamasutra, donde se recomienda a las mujeres que estudien el arte de la escritura secreta, proponiéndoles emparejar al azar las letras del alfabeto y posteriormente sustituir cada letra del mensaje original por su pareja. Esta forma de escritura secreta se conoce como cifrado por sustitución

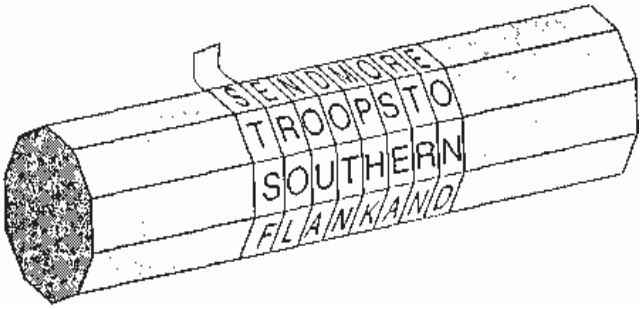

Figura 1.1: *Escitalo espartano*

dado que cada letra del texto llano se sustituye por una letra diferente.

La diferencia entre transposición y sustitución radica en la identidad y posición de cada letra. Mientras que en la transposición se cambia la posición manteniendo la identidad, en la sustitución es la identidad la que varía.

El primer uso documentado de un cifrado de sustitución con propósitos militares aparece en La Guerra de las Galias, de Julio Cesar (año 100-49 a.C.). César utilizó la escritura secreta tan frecuentemente que Valerio Probo escribió un tratado entero acerca de sus cifrados, que desgraciadamente no ha sobrevivido. Sin embargo, gracias a la obra de Suetonio, "Vidas de los Césares LVI", escrita en el siglo II de nuestra era, tenemos una descripción detallada de uno de los tipos de cifrado de sustitución utilizado por César. El emperador sencillamente sustituía cada letra del mensaje con la letra que está tres lugares más adelante en el alfabeto. Este método es llamado "cifra de cambio del César" o, simplemente, "cifra del César". Aunque Suetonio sólo menciona un cambio del César de tres lugares, es evidente que al utilizar cualquier cambio de entre 1 y 25 lugares es posible generar 25 cifrados diferentes.

El algoritmo de sustitución tiene la ventaja de su sencillez a la hora de ponerlo en práctica, a su vez ofrece un alto nivel de seguridad. Esta simplicidad y fortaleza hicieron que el cifrado de sustitución dominara el arte de la escritura secreta a lo largo del primer milenio de nuestra era. Los creadores de códigos habían desarrollado un sistema para garantizar la comunicación segura, de manera que no era necesario ningún nuevo avance. Muchos estudiosos antiguos consideraban que la cifra de sustitución era indescifrable gracias al gigantesco número de claves posibles y durante siglos esto pareció ser verdad. Sin embargo, los descifradores encontraron finalmente un atajo en

el proceso de examinar exhaustivamente todas las claves. En vez de tardar años en descifrar una cifra, el atajo podía revelar el mensaje en cuestión de minutos.

El gran paso adelante sucedió en Oriente y requirió de una brillante combinación de lingüística, estadística y devoción religiosa. Los eruditos árabes fueron capaces de encontrar un método para descifrar la cifra de sustitución monoalfabética, descubriendo que algunas letras son más corrientes que otras. Esta observación aparentemente inocua conduciría al primer gran avance hacia el criptoanálisis. Al Kindi en su tratado titulado "Sobre el descifrado de mensajes cifrados", expone el revolucionario sistema:

"Una manera de resolver un mensaje cifrado, si se sabe en qué lengua está escrito, es encontrar un texto llano diferente escrito en la misma lengua y que sea lo suficientemente largo para llenar alrededor de una hoja, y luego contar cuántas veces aparece cada letra. A la letra que aparece con más frecuencia se le llama "primera", a la siguiente en frecuencia "segunda", a la siguiente "tercera", y así sucesivamente, hasta que se haya cubierto todas las letras que aparecen en la muestra de texto llano. Luego se observa el texto cifrado que se quiere resolver y se clasifica sus símbolos de la misma manera. Se encuentra el símbolo que aparece con más frecuencia y se sustituye por la letra "primera" de la muestra de texto llano. El siguiente símbolo más corriente se sustituye por la forma de la letra "segunda", el siguiente en frecuencia se cambia por la forma de la letra "tercera", y así sucesivamente, hasta que se haya cubierto todos los símbolos del mensaje cifrado que se quiere resolver."

Entre los años 800 y 1200, Europa estaba estancada en la Edad Media y los únicos estudiosos de la escritura secreta eran los monjes. Se sentían intrigados con los secretos que escondía el Antiguo Testamento, donde se encuentran fragmentos de texto cifrados con Atbash. Esta es una forma tradicional de cifra de sustitución hebrea, consistente en tomar cada letra, anotar el número de lugares en que está con respecto al principio del alfabeto y sustituirla por la letra que se halla a un mismo número de lugares con respecto al final del alfabeto.

Los monjes europeos comenzaron a redescubrir viejas cifras de sustitución, inventaron otras nuevas y ayudaron a reintroducir la criptografía en la civilización occidental. Así, en el siglo XIV el uso de la criptografía se había extendido considerablemente.

El resurgimiento de las artes, las ciencias y la erudición durante el Renacimiento reforzó el desarrollo de la criptografía. Italia, por encontrarse en el corazón del Renacimiento, proyectó el ambiente ideal para la criptografía. La Italia renacentista, estaba conformada por ciudades estado independientes y las comunicaciones entre embajadores se hacían con textos cifrados, lo que fomentó que hubiese individuos que tratasen de

destruir esta seguridad. El primer gran criptoanalista europeo fue Giovanni Soro, nombrado secretario de cifras en Venecia en 1506. La reputación de Soro se extendió por toda Italia y fueron muchos los mensajes que llegaron a sus manos para ser descifrados.

Vista la debilidad del cifrado de sustitución monoalfabética, se intentó mejorar con la introducción de nulos, es decir, símbolos o letras que no eran sustitutos de letras auténticas, sino meros huecos que no representaban nada y cuyo único objetivo era confundir el análisis de frecuencia. Otro avance igualmente sencillo fue que los criptógrafos a veces deletreaban mal algunas palabras deliberadamente antes de cifrar el mensaje.

Durante siglos la cifra de sustitución monoalfabética simple había sido suficiente para asegurar la confidencialidad de los mensajes. El subsiguiente desarrollo del análisis de frecuencia, primero en el mundo árabe y luego en Europa, destruyó su seguridad.

En ninguna parte queda ilustrado más dramáticamente el impacto del análisis de frecuencias como en el caso de María Estuardo, reina de Escocia, condenada a morir por conspirar para asesinar a la reina Isabel II de Inglaterra. Todos los traidores habían sido detenidos y ejecutados, no obstante quedaba por comprobar la implicación de María Estuardo. La única prueba tangible en su contra eran una serie de cartas cifradas entre ella y los conspiradores (véase la figura 1.2). Thomas Phelippes, experto criptoanalista, descifró las cartas mediante un análisis de frecuencias y proporcionó la prueba evidente para condenarla.

Incumbía a los criptógrafos inventar un nuevo cifrado más sólido, algo que pudiese desconcertar a los criptoanalistas. Dicho cifrado surgió a finales del siglo XVI y su creador fue el polifacético erudito florentino Leon Battista Alberti. Una conversación sobre criptografía incitó a Alberti a escribir un ensayo sobre este tema, esbozando lo que él consideraba una nueva forma de cifra. Alberti propuso utilizar dos o más alfabetos cifrados, alternando entre ellos durante el cifrado, confundiendo de esta manera a los potenciales criptoanalistas.

"Alf. llano A B C D E F G H I J K L M N O P Q R S T U V X Y Z"

"Alf. cifrado1 F Z B V K I X A Y M E P L S D H J O R G N Q C U T W"

"Alf. cifrado2 G O X B F W T H Q I L A P Z J D E S V Y C R K U H N"

Por ejemplo, se puede cifrar utilizando dos alfabetos cifrados, con esto la palabra "HOLA" se cifraría como "AJPG".

A pesar de que Alberti había dado con el avance más significativo en más de mil años, no logró convertirlo en un sistema plenamente formado. Fueron varios los inte-

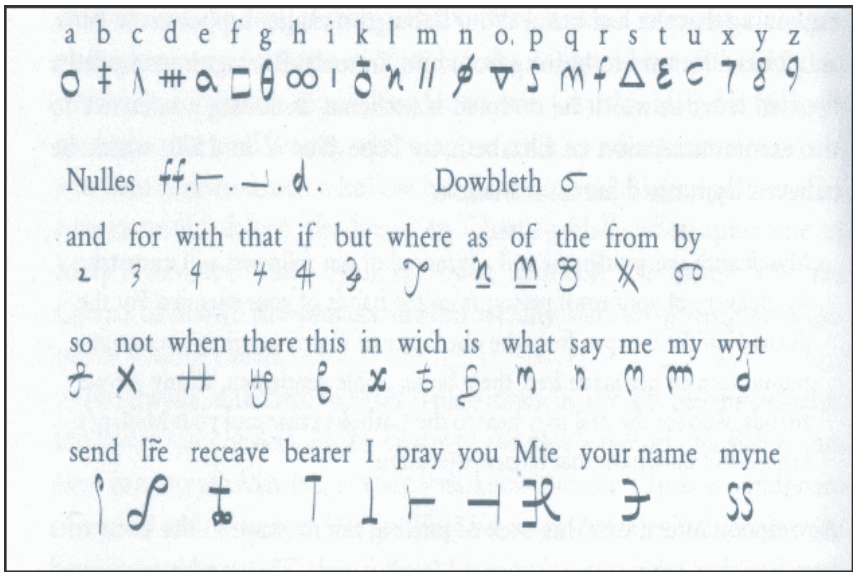

Figura 1.2: *Cifra de María Estuardo*

lectuales que estudiaron este sistema pero fue finalmente Blaise de Vigenère, un diplomático francés nacido en 1523, quién dio con un nuevo cifrado, coherente y poderoso.

Al nuevo método se le conoce como "cifra Vigenère" y su fuerza radica en que no utiliza uno, sino 25 alfabetos de cifrado distintos para ocultar un mensaje. El primer paso del cifrado era trazar lo que se denomina un cuadro Vigenère, tal y como se muestra en la figura 1.3. Se trata de un alfabeto llano seguido de 25 alfabetos de cifrado, donde cada uno de ellos comienza en la siguiente letra que el anterior. De esta forma, la línea 1 representa un alfabeto cifrado con un cambio del César de una posición, la segunda línea con un cambio César de dos posiciones, y así sucesivamente. Se puede cifrar cada letra del texto llano según uno de los 25 alfabetos cifrados.

La gran ventaja de la cifra Vigenère es que resulta inexpugnable para el análisis de frecuencia. El hecho de que una letra que aparece varias veces en el texto cifrado pueda representar en cada ocasión una letra diferente del texto llano genera una ambigüedad tremenda para el criptoanalista. La cifra Vigenère pertenece a una clase conocida como polialfabética puesto que emplea varios alfabetos en cada mensaje.

En el siglo XVIII el criptoanálisis empieza a industrializarse con equipos de criptoanalistas gubernamentales, los cuales trabajaban para descifrar cifras monoalfabéticas complejas. Estos equipos de trabajo consiguieron transformar en inseguras todas

Tabela de Vigenere

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y

Figura 1.3: *Cuadro de Vigenère*

las formas de cifra monoalfabética, con ello los criptógrafos se vieron forzados a adoptar la cifra Vigenère, más compleja pero más segura.

La figura más fascinante del criptoanálisis del siglo XIX es Charles Babbage, quien descifró la cifra de Vigenère. El criptoanálisis de Babbage consistió en buscar secuencias de letras que aparecían más de una vez en el texto cifrado. Lo más probable es que la misma secuencia de letras del texto llano haya sido cifrada usando la misma parte de la clave. El criptoanálisis de la cifra Vigenère realizado por Babbage fue conseguido a mediados del siglo XIX (1854), pero no publicó nunca su descubrimiento. No obstante, la técnica de Babbage, fue descubierta al mismo tiempo y de forma independientemente por Friedrich Wilhelm Kasiski, un oficial retirado del ejército prusiano, que en 1863 publicó "Die Geheimschriften uncí die Dechiffrir-kunst" (La escritura secreta y el arte del descifrado). Desde este momento esta técnica ha sido conocida como la prueba Kasiski y la contribución de Babbage ha sido en gran medida ignorada.

Gracias a los avances realizados por Charles Babbage y Friedrich Kasiski, la cifra Vigenère ya no era segura, por lo que los criptógrafos trataron de diseñar nuevos métodos de cifrado.

En 1883 Auguste Kerckhoffs escribió su tratado "La cryptographie militaire", el cual proporcionó a Francia una guía excepcional de los principios del criptoanálisis. Los militares franceses habían puesto en práctica las ideas de Kerckhoffs a escala industrial y gracias a ello formaron expertos con habilidades especialmente desarrolladas a fin de abordar una cifra particular. Por ello durante la primera guerra mundial los franceses fueron los criptoanalistas más eficaces.

La primera guerra mundial vio una serie de victorias de los criptoanalistas, que desde que resolvieran la cifra Vigenère en el siglo XIX, habían mantenido la ventaja sobre los criptógrafos. A finales de la guerra unos científicos estadounidenses realizaron un avance extraordinario: descubrieron que la cifra Vigenère podía utilizarse como base para una forma nueva y más formidable de cifrado.

La debilidad fundamental de la cifra Vigenère es su naturaleza cíclica. Si la clave tiene cinco letras, entonces cada quinta letra del texto llano está cifrada según el mismo alfabeto cifra. Si el criptoanalista logra identificar la longitud de la clave, el texto cifrado puede ser tratado como una serie de cifras monoalfabéticas y cada una de ellas se puede descifrar con el análisis de frecuencia. Sin embargo si la clave se hace tan larga como el propio texto, la técnica criptoanalítica desarrollada por Babbage y Kasiski no es válida.

Al finalizar la primera guerra mundial, el comandante Joseph Mauborgne, jefe de la investigación criptográfica del ejército de Estados Unidos, introdujo el concepto de la clave aleatoria. Se trataba de una clave que no poseía palabras reconocibles, sino letras mezcladas al azar y las utilizó como parte de la cifra Vigenère para proporcionar un nivel de seguridad sin precedentes. La primera fase del sistema de Mauborgne era compilar un gran cuaderno consistente en cientos de hojas de papel, conteniendo cada una de ellas una clave única formada por líneas de letras reunidas al azar. Habría dos copias del cuaderno, una para el emisor y la otra para el receptor. Para cifrar un mensaje, el emisor aplicaría la cifra Vigenère utilizando la primera hoja de papel del cuaderno como clave. El receptor puede descifrar fácilmente el texto cifrado usando la clave idéntica e invirtiendo la cifra Vigenère. Una vez que el mensaje ha sido enviado, recibido y descifrado con éxito, tanto el emisor como el receptor destruyen la hoja que ha servido como clave. Cuando se cifra el siguiente mensaje se usa la siguiente clave aleatoria del cuaderno y así sucesivamente. Como cada clave se utiliza una única vez, el sistema se conoce como "cifra de cuaderno de uso único".

Figura 1.4: *Disco de Alberti*

Se puede probar matemáticamente que es imposible que un criptoanalista descifre un mensaje codificado con una cifra de cuaderno de uso único.

Por fin, los criptógrafos habían encontrado un sistema indescifrable; sin embargo, la perfección de la cifra de cuaderno de uso único es teórica. En la práctica tiene fallos, ya que adolece de una dificultad fundamental y es el problema práctico de crear grandes cantidades de claves aleatorias. En un solo día, un ejército puede intercambiar cientos de mensajes con miles de caracteres cada uno, de modo que los operadores de radio requerirían un abastecimiento diario de claves equivalente a millones de letras colocadas al azar.

Para fortalecer sus cifras los criptógrafos se vieron obligados a abandonar el lápiz y el papel y a sacar partido de la tecnología más avanzada.

La primera máquina criptográfica conocida es el disco de cifras, inventado en el siglo XV por el arquitecto italiano León Alberti, uno de los padres de la cifra polialfabética. Tomó dos discos de cobre, uno ligeramente mayor que el otro, e inscribió el alfabeto al borde de ambos, colocando el disco pequeño sobre el grande y fijándolos con una aguja que servia de eje. Construyó algo similar al disco de cifras que se muestra en la figura 1.4. Los dos discos pueden hacerse girar independientemente de modo que los dos alfabetos pueden tener diferentes posiciones relativas y, por tanto, se puede utilizar para cifrar un mensaje con un simple cambio del César [106].

Por ejemplo, para remitir un mensaje con un cambio del César de cinco posiciones simplemente hay que girar los discos de modo que la A externa esté junto a la F interna

Figura 1.5: *Máquina Enigma*

y luego usar el disco de cifras en esta nueva posición.

En 1918 el inventor alemán Arthur Scherbius desarrolló una máquina criptográfica denominada Enigma que era esencialmente una versión eléctrica del disco de cifras de Alberti. La forma básica del invento de Scherbius consiste en tres elementos conectados por cables: un teclado para escribir cada letra de texto llano, una unidad modificadora que cifra cada letra de texto llano en la correspondiente letra de texto cifrado y por último un tablero expositor consistente en varias lámparas que indica la letra de texto cifrado. Para cifrar una letra el operador pulsa la letra apropiada en el teclado, lo que envía una pulsación eléctrica a través de la unidad modificadora central y llega al tablero, donde se ilumina la correspondiente letra de texto cifrado. El modificador define esencialmente un alfabeto cifrado y la máquina puede ser utilizada tanto para llevar a cabo una cifra de sustitución monoalfabética simple o polialfabética. El invento de Scherbius proporcionó al ejército alemán el sistema criptográfico más seguro del mundo. Por ello al estallar la segunda guerra mundial sus comunicaciones estaban protegidas con un nivel de cifrado sin precedentes. Los estadounidenses y los franceses intentaron abordar el descifrado de la máquina Enigma pero sus tentativas resultaron infructuosas.

Un alemán, Hans-Thilo Schmidt, hermano menor de Rudolph, jefe del personal de cuerpo de señales del ejercito, vendió información secreta sobre Enigma a las potencias extranjeras y gracias a esta traición, los aliados pudieron crear una réplica exacta de esta máquina. Sin embargo, esto no resultó suficiente para permitirles descifrar mensajes cifrados por Enigma, ya que todavía tenían que descubrir cuál de los billones de claves posibles fue utilizada para cifrarlo.

Los franceses e ingleses estaban convencidos que era imposible encontrar la clave requerida para descifrar un mensaje Enigma concreto, por lo que entregaron a sus aliados, los polacos, las fotografías de los documentos de Schmidt y dejaron la imposible tarea de descifrar Enigma en manos de un equipo de científicos capitaneados por Rejewski.

La estrategia de Rejewski para atacar Enigma se centró en el hecho de que la repetición es el enemigo de la seguridad, esta repetición conduce a patrones y es el arma favorita de los criptoanalistas. La repetición más obvia era la clave de mensaje, que se cifraba dos veces al principio de cada mensaje. ¿Cuál de las 10^{26} claves del día posible se relacionaba con un patrón concreto de las cadenas? El número de posibilidades era demasiado grande, sin embargo, el número total de posiciones de la máquina es el número de disposiciones de los modificadores multiplicado por el número de orientaciones de los modificadores ($6 \cdot 17576 = 105456$). De modo que, en vez de tener que preocuparse sobre cuál de las 10^{26} claves del día se asociaba con un juego de cadenas en particular, Rejewski podía ocuparse de un problema más sencillo: conocer cuál de las 105456 posiciones de los modificadores se asociaba con el número de conexiones en un juego de cadenas.

Su equipo realizó la tarea de probar cada una de las 105456 posiciones de los modificadores, catalogando la longitud de las cadenas generadas por cada una de ellas. Esta tarea requirió un año de labor, pero una vez recopilados todos los datos, Rejewski finalmente comenzó a descifrar Enigma, ya que podía encontrar la clave del día antes que acabara el propio día. Una vez que tenía esta clave, poseía la misma información que el receptor a quien iba dirigido el mensaje y, por tanto, podía descifrarlos fácilmente.

Los estudios realizados por los polacos pasaron a los franceses y británicos que siguieron con su análisis en el centro de Bletchley Park, donde se fue creando un centro de inteligencia aliada que descifraba Enigma. Una vez que habían dominado las técnicas polacas, los criptoanalistas del Bletchley comenzaron a inventar sus propios atajos para descubrir las clave de Enigma.

Al ir evolucionando la máquina Enigma se buscaban nuevos métodos de criptoaná-

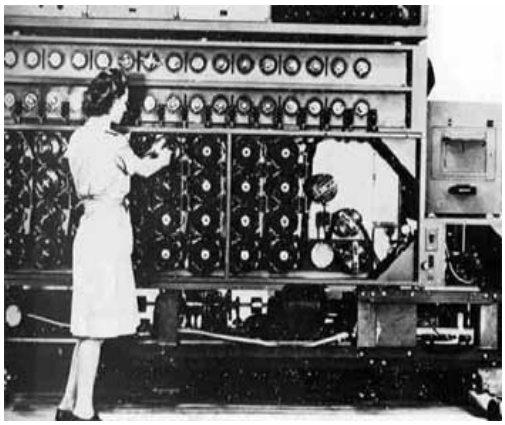

Figura 1.6: *Bomba de Turing*

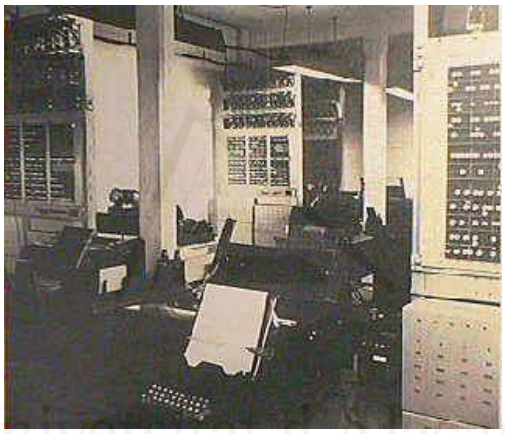

Figura 1.7: *Colossus*

lisis, pero fue Alan Turing quien identificó el punto mas débil de Enigma. Llegó a la conclusión de que había puntales en los criptogramas que son fragmentos de texto llano. Estos pueden asociarse a fragmentos de texto cifrado lo que resultaba una forma extraordinaria de criptoanálisis.

Durante la segunda guerra mundial, los criptoanalistas británicos consiguieron imponerse a los criptógrafos alemanes gracias a la tecnología de descifrado desarrollada en Bletchley Park. Además de la máquina utilizada para derrotar a la cifra de Enigma (llamada bombas de Turing, figura 1.6), los británicos desarrollaron otro instrumento de descifrado: Colossus (figura 1.7), utilizado para combatir la cifra Lorenz alemana.

La cifra Lorenz se empleaba para cifrar las comunicaciones más importantes de la inteligencia alemana y su funcionamiento era parecido al de la máquina Enigma pero

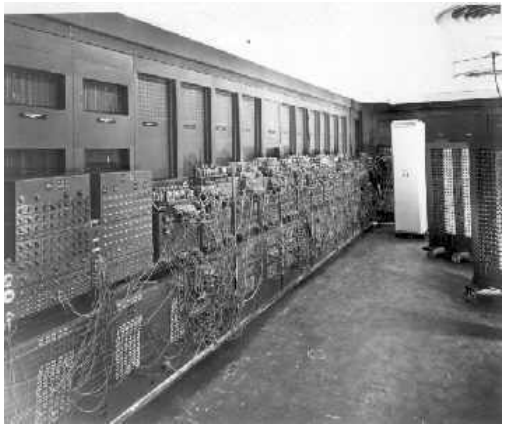

Figura 1.8: *Eniac*

con una complejidad muy superior. Esta cifra era demasiado potente para descifrarla utilizando los métodos manuales que se manejaban en ese momento. Era necesario una mecanización del proceso y con ello un aumento de la velocidad de las operaciones de descifrado. Fue Max Newman, un matemático de Bletchley, quien dio con una nueva manera de mecanizar el criptoanálisis de la cifra Lorenz. Inspirado por el concepto de la máquina universal de Alan Turing, creó una máquina capaz de adaptarse a diferentes problemas, el primer ordenador programable, Colossus.

Colossus junto con los documentos de su diseño fueron destruidos después de la guerra y a todo aquel que trabajó en este o en cualquier otro proyecto de Bletchley Park se les prohibió dar información. Los planos del primer ordenador de la historia se perdieron para siempre. Su lugar fue ocupado por el proyecto ENIAC (figura 1.8), desarrollado en la Universidad de Pensilvana de los Estados Unidos . A partir de entonces los criptoanalistas comenzaron a utilizar y comprender la potencia que los ordenadores proporcionaban para descifrar cualquier tipo de cifra.

En principio, el cifrado por ordenador estaba restringido a los gobiernos y al ejército, sin embargo, una serie de avances científicos y tecnológicos permitieron que este cifrado por ordenador fuera asequible para el público en general. En 1947 ATT inventó el transistor, una alternativa barata a la válvula electrónica y en 1953 IBM lanzó su primer ordenador. Más tarde, en 1959, con la invención del circuito integrado, se avanzó hacia una nueva era en la informática.

Durante los setenta, los ordenadores bajaron de precio y se comercializaron en mayor número. Numerosas empresas comenzaron a utilizarlo para cifrar sus mensajes privados, lo que creó un nuevo problema: la estandarización. Una compañía podía usar

un sistema de cifrado en particular para garantizar la seguridad de sus comunicaciones internas, pero no podía enviar un mensaje oculto a una organización externa a no ser que el receptor usara el mismo sistema de cifrado. Finalmente, el 15 de mayo de 1973, la Oficina Nacional de Estándares norteamericana planeó resolver el problema y solicitó propuestas de cifrado estándar que permitieran a las empresas comunicarse entre sí. El 23 de noviembre de 1976 se adoptó un estándar de cifrado con una longitud de clave de 56 bits, dándole el nombre de DES [88]. Superado el problema de la estandarización, se planteó una nueva dificultad: un método seguro de distribución de claves.

Es necesario que tanto el emisor como el receptor conozcan la clave para poder cifrar y leer el mensaje original, pero esto plantea una posible grieta de seguridad en el sistema de cifrado: ¿cómo transportar de forma segura la clave del emisor al receptor? Es decir, antes que dos personas puedan intercambiar un mensaje secreto deben compartir un secreto (la clave de cifrado y descifrado).

Fueron Whitfield Diffie y Martin Hellman [36, 35] quienes iniciaron el camino hacia la resolución de este problema. Tras varios meses de fracasos investigando funciones matemáticas bidireccionales o de doble vía, sus investigaciones se centraron en funciones unidireccionales que, como su nombre sugiere, son funciones fáciles de calcular en un sentido pero muy difícil en sentido contrario.

Después de dos años concentrándose en la aritmética modular y las funciones de una sola vía, el trabajo de Hellman comenzó a dar sus frutos. En la primavera de 1976 dió con una estrategia para resolver el problema de la distribución de claves. Demostró que el emisor y el receptor podían acordar una clave sin necesidad de reunirse, deshaciendo así un axioma que había durado siglos. La idea de Hellman se basaba en una función unidireccional de la forma Y^x $(mod\ P)$. Había conseguido invalidar uno de los axiomas menos cuestionados de la criptografía y ahora solo se necesitaba que otro perfeccionara el sistema de distribución de claves para hacerlo más eficaz.

Paralelamente al trabajo de Hellman, Diffie había estado probando otro tipo de enfoque para la resolución del problema de distribución de claves. Descubrió un nuevo tipo de cifra: la clave asimétrica. Hasta ese momento todas la técnicas de cifrado utilizadas habían sido simétricas, es decir, tanto el emisor y el receptor utilizaban la misma clave para cifrar y descifrar el mensaje. Sin embargo, en un sistema de clave asimétrica, la clave de cifrado y la de descifrado no son idénticas.

Fueron tres investigadores del laboratorio de informática del MIT (Massachussets Institute of Technology), Ronald Rivest, Adi Shamir y Leonard Adleman, quienes consiguieron encontrar una función que convirtiera la teoría de la clave pública en un

instrumento práctico y eficaz. Este sistema se denomina RSA en honor a sus tres descubridores Rivest, Shamir y Adleman [103].

En 1984 T. ElGamal creó otro criptosistema cuya seguridad se basa en una función unidireccional denominada logaritmo discreto, que todavía no ha podido resolverse de forma eficiente (véase [37]).

Las matemáticas proporcionan las herramientas adecuadas para la creación de algoritmos de ocultación de información. Así, algunas de las áreas de las matemáticas que se han usado (y se usan) para crear criptosistemas (véase [18]) son:

- En 1970 R. J. McEliece desarrolló un criptosistema de clave pública basado en códigos detectores - correctores de errores [78, 79].
- En los años 80 V. Varadharajan propuso distintas estructuras de anillos que pueden ser aplicados en la generalización del sistema RSA [92].
- En 1984 Lidl y Müller proponen polinomios de permutación [73].
- En 1985, de forma independiente V. Miller [84] y N. Koblitz [67, 68] usan la teoría de curvas elípticas para crear criptosistemas. Estas curvas fueron propuestas por Lenstra para factorizar números enteros.
- En 1988 J. Buchmann y H. Williams proponen usar campos cuadráticos reales e imaginarios [25, 107].
- En 1995 R. Scheidler y H. Williams usan campos ciclotómicos [108].
- Por último, existe otro tipo de protocolos que usa la teoría de incertidumbre y se le conoce como criptografía cuántica (véanse [14, 19, 17]).

1.2 Conceptos básicos

1.2.1 Seguridad en la información

Hay muchas formas de clasificar los servicios y mecanismos que proporcionan seguridad a los sistemas informáticos. Una de las clasificaciones que más se utilizan es la que se recoge en el estándar internacional "ISO 7498-2, Arquitectura de Seguridad", cuyas definiciones y términos se han extendido ampliamente.

Desde el punto de vista de esta norma, los servicios de seguridad son:

(a) Autenticación. Es la identificación ante un sistema, subsistema, red o aplicación, mediante algún mecanismo o combinación de mecanismos. Una vez que una entidad se ha autenticado puede que necesite volver a autenticarse para otros fines. Algunos mecanismos de autenticación son:
   (1) Nombre de usuario y contraseña.
   (2) Tickets de acceso. Consisten en identificadores temporales que son solicitados por el cliente a un servidor.
   (3) Certificados digitales. Incluyen información estandarizada como la clave pública del propietario, un nombre y fechas de caducidad. Asociado al certificado digital de cada entidad se encuentra la clave privada, complemento de la clave publica, que forma parte del certificado en sí.
   (4) Tarjetas inteligentes. Son elementos informáticos que pueden, entre otras cosas, contener claves privadas, certificados digitales u otra información sobre la entidad. Si la tarjeta inteligente contiene información suficientemente valiosa, puede protegerse mediante un número de identificación personal, que tiene la misma función ante la tarjeta que las contraseñas ante los identificadores de usuario.
   (5) Tokens (Fichas). Se utilizan en mecanismos robustos de autenticación y están protegidas por PIN o por otro mecanismo, como la generación de números mediante los token de securID.
   (6) Dispositivos biométricos. Realizan análisis estadísticos de patrones generados analizando alguna característica física de una persona (retina, huella digital, voz, etc.) para establecer una identificación personal.

(b) Control de acceso. Protege contra el uso no autorizado de recursos. Generalmente, hay un orden implícito, según el cual una entidad primero se identifica y autentifica y a continuación se le proporciona el acceso o se le deniega, basándose en

mecanismos de control de acceso asociados a las credenciales de la entidad. Cada entidad tiene sus permisos de acceso a cada recurso especificado.

Ejemplos de dichos mecanismos son:

(1) Listas de control de acceso. Consisten en listas de entidades que tienen autorización para acceder al un determinado recurso, junto a sus permisos de acceso.

(2) Etiquetas de seguridad. Son atributos asociados con una entidad que permiten la clasificación de la entidad en términos de nivel de seguridad.

(3) Roles. Es un atributo de privilegio que representa la posición o función de un usuario que busca una autenticación.

(4) Barreras físicas. El acceso físico a dispositivos del sistema y a la red debe limitarse mediante habitaciones cerradas con mecanismos de control de accesos.

(5) Firewalls. Son mecanismos de control con funciones de filtrado y proxy para evitar el acceso a determinados recursos o direcciones de la red.

(c) Confidencialidad. Consiste en proteger los datos trasmitidos o almacenados de difusiones no autorizadas. Este aspecto de la seguridad se ha vuelto cada vez más importante puesto que cada vez mayor cantidad de información se trasmite sobre redes inseguras y cada vez mayor volumen de información sensible llega a los equipos menos protegidos de una red. Se debe asumir que la red no es fiable y que la información trasmitida puede ser interceptada. En la actualidad el cifrado de la información es la principal herramienta utilizada para conseguir la confidencialidad.

(d) Integridad de Datos. Consiste en detectar cuando los datos almacenados o trasmitidos han sido modificados, borrados o reproducidos.

Algunos mecanismos para garantizar la integridad de los datos son:

(1) Checksums. Un número obtenido tomando un mensaje, dividiéndolo en bloques de longitud fija y sumándolos. La suma se envía con el mensaje. El receptor divide el mensaje en bloques del mismo tamaño, los suma y compara ambas sumas.

(2) Resúmenes de mensajes. Un resumen de mensaje es un número calculado utilizando el mensaje como dato de entrada en una función de cifrado.

(e) No rechazo. Consiste en asociar la identidad de un individuo con su participación en un proceso. Los mecanismos de no rechazo proporcionan pruebas de un intercambio digital significativo de algún tipo.

(f) Gestión. Consiste en la administración y gestión de los mecanismos asociados con

cada una de las categorías de seguridad siguientes:

(1) La directiva general de seguridad.
(2) Los mecanismos de seguridad específicos.
(3) Los eventos de seguridad.
(4) La auditoria de seguridad.
(5) El restablecimiento de la seguridad.

1.2.2 Criptografía

La criptografía hace años que dejó de ser un arte para convertirse en una técnica, o más bien un conglomerado de técnicas, que tratan sobre la protección de la información (véase figura 1.9). Entre las disciplinas que engloba cabe destacar la Teoría de la Información, la Teoría de Números y la Complejidad Algorítmica.

Existen dos trabajos fundamentales sobre los que se apoya prácticamente toda la teoría criptográfica actual. Uno de ellos, desarrollado por Claude Shannon en sus artículos "A Mathematical Theory of Communication" [111] y "Communication Theory of Secrecy Systems" [112], sienta las bases de la teoría de la información y de la criptografía moderna. A partir de entonces, y unido al desarrollo de la computación moderna, la criptografía alcanza nuevos horizontes. El segundo trabajo, desarrollado en el artículo "New directions in Cryptography" [35] por Whitfield Diffie y Martin Hellman en 1976, introduce el concepto de criptografía asimétrica, abriendo enormemente el abanico de aplicación de esta disciplina.

La palabra criptografía sólo hace referencia al uso de códigos, por lo que no engloba las técnicas que se usan para romper dichos códigos, conocidas como criptoanálisis. Las dos disciplinas están íntimamente ligadas y al conjunto de ambas y se le conoce con el nombre de criptología.

1.2.3 Criptosistemas

Se define un criptosistema como una quíntupla (M, C, K, E, D), donde:

- M representa el conjunto de todos los mensajes sin cifrar (lo que se denomina texto claro o plaintext) que pueden ser enviados. Son componentes de un mensaje inteligible (bits, bytes, pixels, signos, caracteres, etc.) que provienen de un alfabeto previamente establecido. Este lenguaje tiene unas reglas sintácticas y semánticas y en algunos casos (sistemas de cifrado clásicos) la longitud del alfabeto

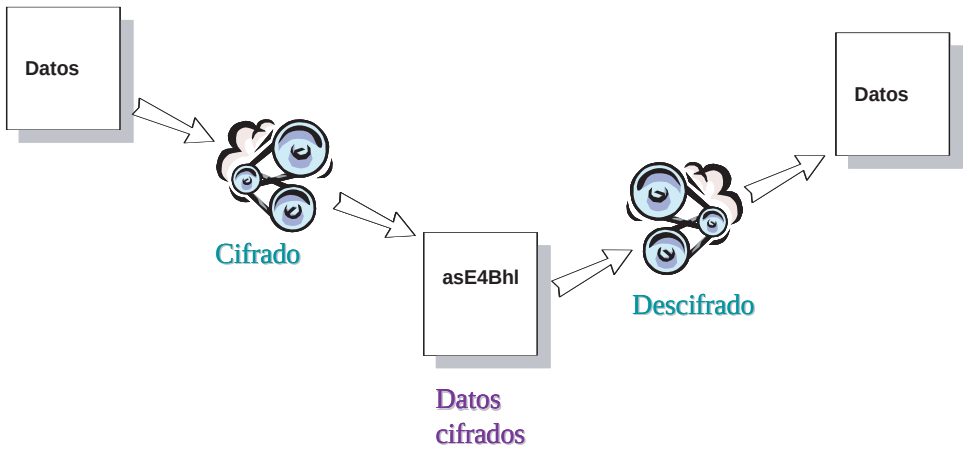

Figura 1.9: *Criptografía*

indicará el módulo en el cual se trabaja. En los modernos, no guarda relación. Además, hay mensajes con sentido y mensajes sin sentido,

$$M = \{m_1, m_2, m_3, \ldots, m_n\}.$$

- C representa el conjunto de todos los posibles mensajes cifrados, o criptogramas. Normalmente el alfabeto es el mismo que el utilizado para crear el mensaje en claro. Además, se supone que el espacio de los textos cifrados C y el espacio de los mensajes M (con y sin sentido) tienen igual magnitud y, en este caso, a diferencia del espacio de mensajes M, son válidos todo tipo de criptogramas,

$$C = \{c_1, c_2, c_3, \ldots, c_n\}.$$

- K representa el conjunto de claves que se pueden emplear en el sistema criptográfico. Se supone que es un conjunto altamente aleatorio de caracteres, palabras, bits, bytes, etc., en función del sistema de cifra. Al menos una de las claves en un criptosistema se guarda en secreto,

$$K = \{k_1, k_2, k_3, \ldots, k_n\}.$$

- E es el conjunto de transformaciones de cifrado o familia de funciones que se aplica a cada elemento de M para obtener un elemento de C. Existe una transformación diferente E_k para cada valor posible de la clave k. Es decir, E_k es una aplicación (con una clave k que está en el espacio de claves K) de M en C,

$$E_k : M \rightarrow C \quad \text{con} \quad k \in K.$$

En general el algoritmo de cifrado es de dominio público.

- D es el conjunto de transformaciones de descifrado, análogo a E. En este caso, D_k es una aplicación con una clave k (en el espacio de claves K) de C en M. Se utiliza el concepto de inverso. En consecuencia D_k es la operación inversa de E_k o bien, se utiliza la misma transformación E_k para descifrar pero con una clave k' que es la inversa de k dentro de un cuerpo,

$$D_k : C \rightarrow M \quad \text{con} \quad k \in K.$$

Todo criptosistema debe cumplir la sigueinte condición:

$$D_k(E_k(m)) = m,$$

es decir, dado un mensaje en claro m, si se cifra empleando la clave k y luego se descifra empleando la misma clave, se obtiene el mensaje original m.

Existen dos tipos fundamentales de criptosistemas:

(a) Criptosistemas simétricos o de clave privada (véase figura 1.10). Son aquellos que emplean la misma clave k tanto para cifrar como para descifrar. Presentan el inconveniente de que para ser empleados en comunicaciones, la clave k debe ser conocida tanto por el emisor como por el receptor, lo cual plantea el problema de transmitir la clave de forma segura.

(b) Criptosistemas asimétricos o de clave pública (véase figura 1.11). Son aquellos que emplean una doble clave (k_p, k_P). La clave k_p se conoce como clave privada y se emplea para la transformación de descifrado D, mientras que k_P se conoce como clave pública y sirve para el cifrado E.

En la práctica se emplea una combinación de estos dos tipos de criptosistemas, puesto que los segundos presentan el inconveniente de ser computacionalmente más costosos que los primeros, se cifra los mensajes (largos) mediante algoritmos simétricos, que suelen ser muy eficientes; luego se hace uso de la criptografía asimétrica para cifrar las claves simétricas (cortas).

1.2.4 Criptoanálisis

El criptoanálisis se define como la ciencia del descifrado de los criptogramas por análisis y deducción, sin tener conocimiento previo de la clave. No se considera criptoanálisis el descubrimiento de un algoritmo secreto de cifrado; se supone, por el contrario, que los algoritmos siempre son conocidos.

El criptoanálisis se realiza estudiando grandes cantidades de pares (mensaje, criptograma) generados con la misma clave. El mecanismo que se emplea para obtener estos

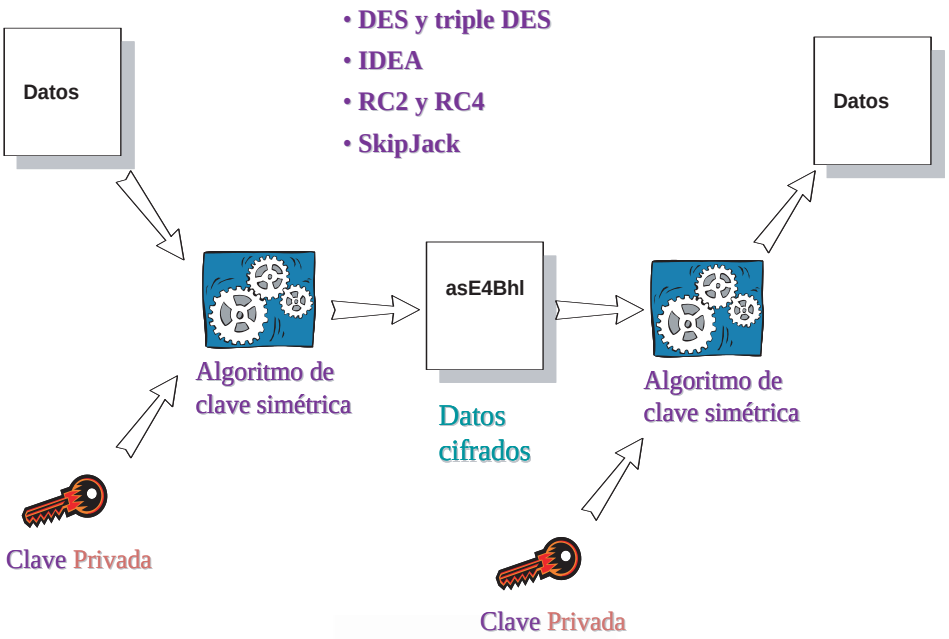

Figura 1.10: *Criptografía simétrica*

pares es indiferente y puede ser resultado de escuchar un canal de comunicaciones o puede ser que el objeto de ataque responda con un criptograma cuando se le envíe un mensaje, o que se tenga acceso al dispositivo de cifrado y éste permita efectuar operaciones pero no permita leer su clave (por ejemplo, las tarjetas de los teléfonos móviles GSM). Cuando el sistema es débil, pueden ser suficientes unos cientos de mensajes para obtener información que permita deducir la clave empleada.

Como el algoritmo de cifrado es conocido, se puede intentar criptoanalizar un sistema aplicando para cada una de las claves el algoritmo de descifrado hasta encontrar una salida que tenga sentido como posible texto claro. Este tipo de técnicas que buscan exhaustivamente en todo el espacio de claves K, se denominan "fuerza bruta" y no se consideran como auténticas técnicas de criptoanálisis, reservándose este término para mecanismos que explotan posibles debilidades intrínsecas en el algoritmo de cifrado. Se denomina ataque a cualquier técnica que permite recuperar un mensaje cifrado empleando menos esfuerzo computacional que el que se utiliza por fuerza bruta.

En cualquier criptosistema digno de interés el espacio de claves es lo suficientemente grande como para que los métodos basados en fuerza bruta sean inviables. No obstante, hay que tener en cuenta que la capacidad de cálculo de las computadoras crece a gran velocidad, por lo que algoritmos que hace unos años eran resistentes a ataques por

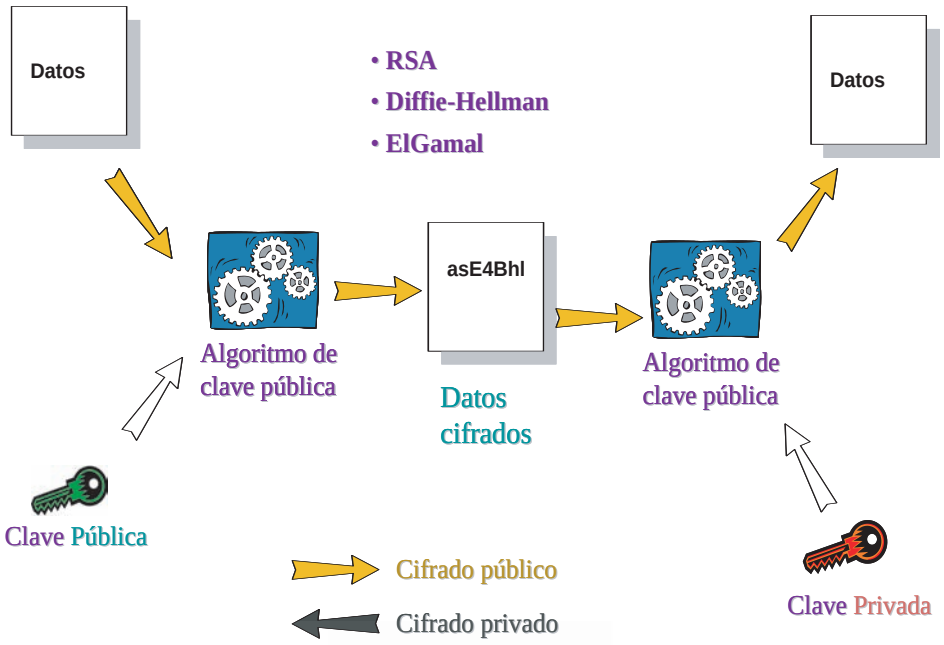

Figura 1.11: *Criptografía asimétrica*

fuerza bruta hoy pueden resultar inseguros. A pesar de eso, existen longitudes de clave para las que resulta imposible, empleando la computación actual, tener éxito al aplicar un método de este tipo.

Dos métodos de criptoanálisis simétrico que proporcionan resultados interesantes son el análisis diferencial y el análisis lineal. El primero de ellos, parte de pares de mensajes con diferencias mínimas (usualmente de un bit), estudia las variaciones que existen entre los mensajes cifrados correspondientes y trata de identificar patrones comunes. El segundo emplea operaciones XOR entre algunos bits del texto claro y algunos bits del texto cifrado, obteniendo finalmente un único bit. Si se realiza esto con muchos pares (texto claro, texto cifrado) se puede obtener una probabilidad p en ese bit que se calcula y si p está suficientemente sesgada (no se aproxima a $\frac{1}{2}$) se tiene posibilidad de recuperar la clave.

Un tipo de análisis para algoritmos asimétricos consiste en deducir la clave privada a partir de la pública. Por lo general, se suelen emplear técnicas analíticas que intentan resolver los problemas de elevado coste computacional (factorización, logaritmos discretos, etc) en los que se apoyan estos criptosistemas. Mientras estos problemas permanezcan sin resolver de forma eficiente se puede seguir confiando en estos algoritmos.

La gran variedad de sistemas criptográficos existentes produce necesariamente una

gran variedad de técnicas de criptoanálisis, cada una de ellas asociada a un algoritmo o familia de ellos.

1.2.5 Esteganografía

Del griego "steganos" (encubierto) y "graphos" (escritura) nace el término esteganografía que consiste en ocultar en el interior de una información, aparentemente inocua, otro tipo de información (cifrada o no).

Aunque la criptografía y la esteganografía pueden parecer en un principio términos equivalentes, o al menos similares, son distintos. La criptografía tiene su fuerza en la imposibilidad de comprender el mensaje, mientras que la esteganografía la tiene en el desconocimiento de que el mensaje existe. Son técnicas distintas e independientes, si bien pueden complementarse entre ellas (y de hecho lo suelen hacer).

Con el auge de la informática, el mecanismo esteganográfico más extendido está basado en las imágenes digitales y su excelente capacidad para ocultar información. Aunque existen varias formas de conseguirlo, la más básica consiste en sustituir el bit menos significativo de cada byte por los bits del mensaje que se quiere ocultar; dado que casi todos los estándares gráficos tienen una graduación de colores mayor de lo que el ojo humano puede apreciar, la imagen no cambiará su apariencia de forma notable. Otros elementos donde se puede ocultar información son las señales de audio y vídeo y aunque históricamente no han estado tan extendidas como la esteganografía en imágenes digitales, en los últimos tiempos el interés por los mecanismos de ocultación de información en formatos de audio (principalmente MP3) y vídeo ha ido en aumento.

Sea cual sea el tipo de información que se quiera ocultar y sea cual sea el medio en el que se quiere hacer, hay ciertas reglas básicas que se deben cumplir:

- La información (texto ASCII, hexadecimal, código morse...) que se quiere esteganogarfiar, debe ser primero convertida a binario, debido a la comodidad de trabajar con este código.
- No hay que permitir que un supuesto atacante obtenga el fichero original (anterior a la modificación), pues permitiría, mediante comparación, establecer pautas de cambios en la información que podría llevar a desentrañar el mensaje oculto.
- Las cabeceras de los ficheros no deben ser modificadas.
- No transmitir el algoritmo esteganográfico por un medio inseguro.

Sin llegar aún a la combinación de esteganografía y criptografía, es posible el uso de determinadas técnicas que permiten aumentar la eficacia de una información oculta

mediante esteganografía, por ejemplo:

- Uso de múltiples claves. Esta técnica es heredada directamente de la criptografía, pero con distinta forma de aplicación. Consiste en usar distintas codificaciones para cada porción arbitraria del mensaje a ocultar. Así, una frase de cinco palabras puede tener una clave de cifrado para cada una de las palabras. Naturalmente la clave ha de ser conocida por el destinatario.
- Esteganografía en capas. Mediante esteganografía en capas se establece una relación lineal entre los elementos ocultos. Así, el cifrado de la segunda palabra o letra de un mensaje depende de la primera, con lo que se establece un orden estricto de descifrado que impide obtener completamente el mensaje sin la primera parte. Únicamente se debe comunicar la clave y la pauta a seguir para encadenar los fragmentos.
- Adición de ruido. Además de modificar los bits necesarios para introducir el mensaje, se pueden modificar unos cuantos bits aleatorios del mensaje de forma que, aun teniendo el fichero original, un posible atacante deba conocer el sistema de cifrado usado.
- Uso de distintas magnitudes. Aunque lo habitual es variar en 1 bit el byte del mensaje original, nada impide hacerlo en más bits.

Actualmente, la esteganografía está íntimamente ligada a la criptografía, puesto que mediante la combinación de estas dos técnicas se establecen dos capas en la seguridad de la información:

- La capa más interna es la criptográfica y se encarga de la seguridad de los datos.
- La capa esteganográfica, que protege la integridad de la capa criptográfica.

1.2.6 Funciones hash

Matemáticamente se pueden definir las funciones resumen (funciones hash) como proyecciones de un conjunto, generalmente con un número elevado de elementos (incluso infinitos), sobre un conjunto de tamaño fijo y mucho más pequeño que el anterior. Para que una función resumen sea útil en aplicaciones criptográficas, debe cumplir los siguientes requisitos:

(a) La entrada puede ser de un tamaño indeterminado.
(b) La salida es de un tamaño fijo, varios órdenes de magnitud más pequeño que el anterior.
(c) Es de un único sentido.

(d) No presenta colisiones.

Partiendo de un mensaje determinado de cualquier tamaño, este mensaje se convierte mediante la función hash en un mensaje con una dimensión fija (generalmente de 160 bits). Para ello, el mensaje originario se divide en varias partes, cada una de las cuales tendrá ese tamaño (160 bits), y una vez dividido se combinan elementos tomados de cada una de las partes resultantes de la división para formar el mensaje resumen o hash, que tiene un tamaño fijo y constante (160 bits). Este resumen es el que se cifra utilizando la clave privada del emisor del mensaje.

1.2.7 Firma digital

La firma digital (véase figura 1.12) es una secuencia de datos anexos a un mensaje, resultado de aplicar un conjunto de algoritmos matemáticos, que permiten ofrecer ciertas garantías de seguridad sobre el mensaje objeto de firma. Estas garantías son dos: autenticación (quién es el autor) e integridad (no ha existido ninguna manipulación posterior de los datos).

Para poder equiparar firma digital y firma manuscrita, la firma digital debe cumplir una serie de normas:

(a) Debe ser barata y fácil de producir.
(b) Debe ser fácil de reconocer tanto por el propietario como por otros.
(c) Debe ser imposible de rechazar por el propietario.

Para ello existen 2 métodos de firma digital:

- **Firma digital con árbitro.** En la que dos usuarios con desconfianza mutua confían en un tercero. Se utilizan criptosistemas de clave única (una única clave para cifrar y descifrar). El emisor y el receptor tienen sus propias claves por lo que es el árbitro el encargado de recibir el mensaje del emisor y descifrarlo con la clave del emisor. De esta forma el emisor y el receptor no necesitan compartir claves.
- **Firma digital ordinaria.** En la que el firmante envía directamente la firma digital al destinatario y este debe poder comprobar la validez de la misma sin necesidad de árbitro. A este método pertenecen los sistemas de firmas actuales que se basan en criptosistemas de clave pública.

El modo de funcionamiento de la firma digital basado en clave pública es el siguiente:

(a) Cada participante tiene un par de claves, una se usa para cifrar y la otra para

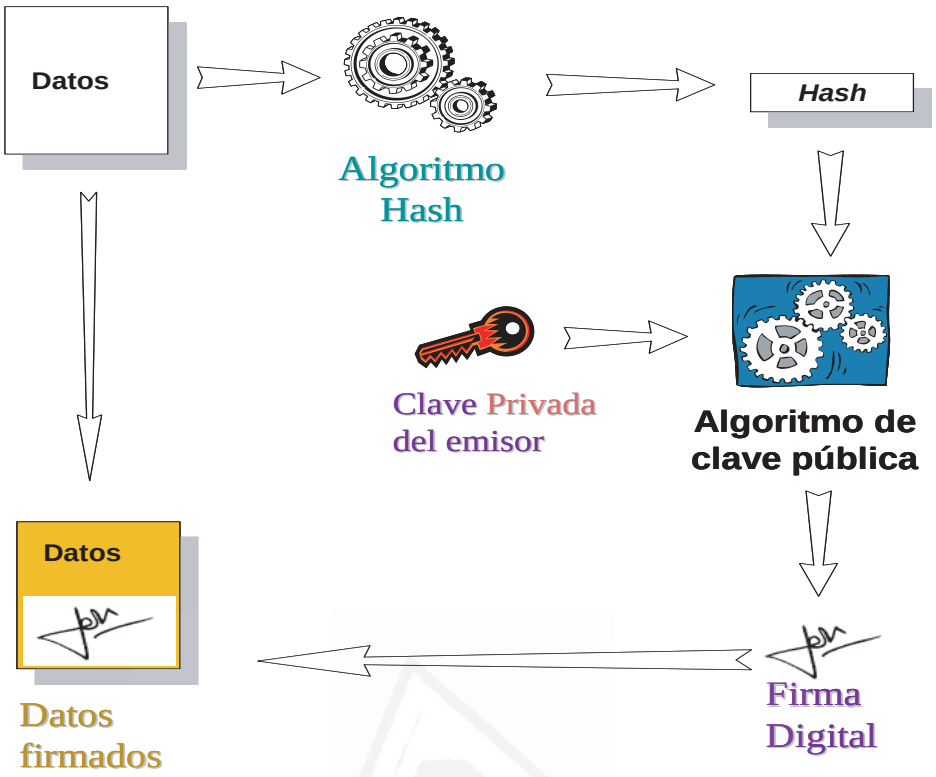

Figura 1.12: *Firma digital*

descifrar.

(b) Cada participante mantiene en secreto una de las claves (clave privada) y pone a disposición del público la otra (clave pública).

(c) El emisor calcula un resumen del mensaje a firmar (conjunto de datos de pequeño tamaño que tiene la propiedad de cambiar si se modifica el mensaje).

(d) El emisor cifra el resumen del mensaje con su clave privada (ésta es la firma digital que se añade al mensaje original).

El receptor, al recibir el mensaje, calcula de nuevo su resumen. Además, descifra la firma utilizando la clave pública del emisor obteniendo el resumen que el emisor calculó. Si ambos resúmenes coinciden entonces la firma es válida por lo que cumplen los criterios de autenticidad, integridad y no repudio, ya que el emisor no puede negar haber enviado el mensaje que lleva su firma (véase figura 1.13).

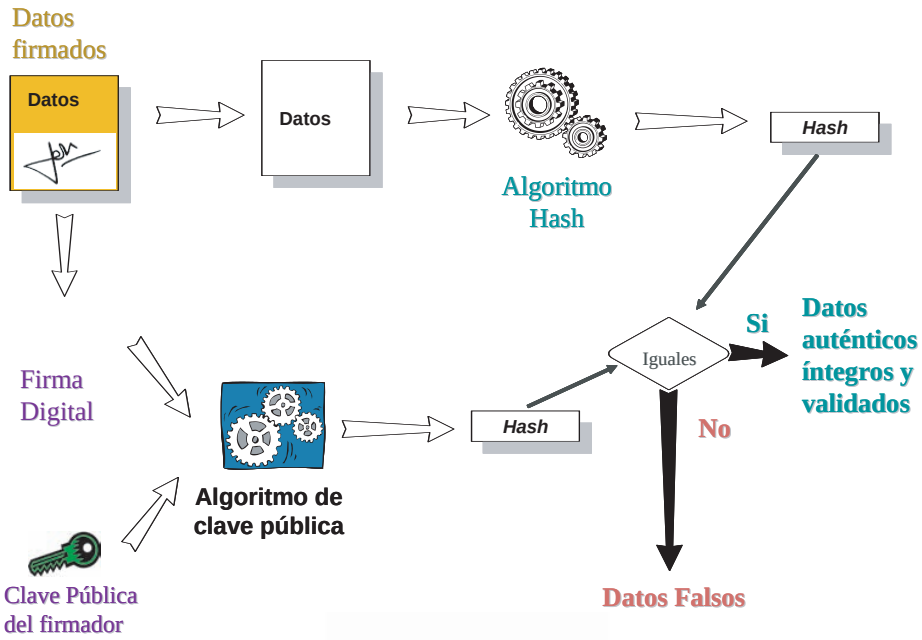

Figura 1.13: *Comprobación de la firma digital*

1.3 Teoría de la información

Desde el punto de vista de la ciencia de las comunicaciones, la información se define como la transmisión de una idea por medio de un mensaje convencional a través de un soporte espacio-temporal. En la transmisión de la información intervienen, básicamente, tres elementos: un emisor o fuente de información que selecciona un mensaje entre un conjunto posible, un canal por el que es transmitido el mensaje y el receptor del mismo.

Se pueden producir diferencias entre el mensaje emitido y el recibido, debido a perturbaciones denominadas ruido. Con objeto de preservar al mensaje de los efectos causados por el ruido se añade un codificador, cuyo objetivo es doble: por un lado transformar el mensaje previamente a su transmisión adaptándolo al canal y por otro lado mejorar la eficiencia del mismo. En el extremo opuesto del canal se agrega un decodificador, que realiza la operación inversa, trata de reconstruir el mensaje original.

La teoría de la información se encarga de la medida de la información transmitida a través de un canal durante el proceso de comunicación. Una aportación muy importante a esta idea es la realizada por Shannon al afirmar que la cantidad de información está asociada a la libertad de elección de cada mensaje que puede pasar por el canal y por tanto a la probabilidad de que ocurra un determinado suceso, es decir, a su incerti-

dumbre. Cuanto más improbable sea un mensaje, más importante será la información que dé. La incertidumbre mencionada anteriormente se define como la falta de certeza o seguridad ante una situación. Disminuir la incertidumbre aumenta la información.

En 1948 Shannon dió una medida de la incertidumbre, $I(a_k)$, sobre la ocurrencia de un suceso a_k perteneciente a un conjunto de sucesos mutuamente excluyentes

$$S = \{a_1, a_2, ..., a_n\}.$$

Siendo $p_k = P(a_k)$ con $k = 1, 2, ..., n$ las probabilidades de que ocurran cada uno de los sucesos de S y sabiendo que

$$\sum_{k=1}^{n} p_k = 1,$$

existen dos propiedades que debería cumplir la función de incertidumbre I:

- Cuanto menor sea la probabilidad de ocurrencia de un suceso, mayor debería ser su incertidumbre, es decir
$$I(a_k) = f\left(\frac{1}{p_k}\right).$$
- Considerando los n sucesos de S equiprobables y suponiendo m realizaciones del experimento "elegir un suceso de S", parece lógico que la incertidumbre de las m realizaciones sea m veces mayor que la incertidumbre de un solo suceso, por lo que se exige a esa función de incertidumbre que cumpla
$$I(a_{i_1}, a_{i_2}, \ldots, a_{i_n}) = mI(a_{i_k}),$$
es decir
$$f\left(\frac{1}{\frac{1}{n^m}}\right) = mf\left(\frac{1}{\frac{1}{n}}\right).$$

Como se ve, la función logaritmo se ajusta a estas dos condiciones.

Sea E un suceso que puede presentarse con probabilidad $P(E)$, cuando E tiene lugar, se dice que se ha recibido

$$I(E) = \log_a \frac{1}{P(E)} = -\log_a P(E)$$

unidades de información.

La función $I(E)$ mide el grado de incertidumbre sobre la realización u ocurrencia de un suceso E, además, la elección de la base del logaritmo equivale a elegir

una determinada unidad de información. Si se elige base 2, la unidad de información correspondiente se denomina bit (binary unit)

$$I(E) = \log_2 \frac{1}{P(E)} \text{ bits.}$$

Si se tiene dos sucesos equiprobables, $P(E_1) = P(E_2) = 1/2$, entonces la incertidumbre de ambos sucesos es $I(E_1) = I(E_2) = 1$ bit. Es decir, un bit es la cantidad de información obtenida al especificar una de dos alternativas equiprobables.

Empleando logaritmos naturales, la unidad de información recibe el nombre de nat (natural unit)

$$I(E) = \ln \frac{1}{P(E)} \text{ nats.}$$

Y si se emplea base 10, la unidad de información recibe el nombre de dit (decimal unit) o Hartley

$$I(E) = \log_{10} \frac{1}{P(E)} \text{ dits.}$$

Dado que $\log_a x = \frac{\log_b x}{\log_b a}$, se tiene que 1 dit = 3,32 bits y 1 nat = 1,44 bits.

En 1949, Shannon, con la publicación de su artículo "Communication Theory of Secrecy Systems" [112] basado en su obra sobre la Teoría de la Información, sienta las bases para el tratamiento matemático de la criptología obteniendo conclusiones válidas para cualquier criptosistema. Entre estas bases, se tiene una serie de definiciones importantes:

- Seguridad teórica. Un criptosistema alcanza seguridad teorica si no se puede obtener ninguna información del texto en claro incluso cuando el criptoanalista dispone de tiempo y recursos ilimitados. En realidad, en todos los criptosistemas el conocimiento de texto cifrado supone alguna información sobre el texto en claro.
- Sistema rompible. Un sistema criptográfico es rompible si a través del análisis de texto cifrado se puede determinar de una forma única el texto en claro. Todos los criptosistemas aceptados en la actualidad son rompibles en el sentido de que no alcanzan la seguridad teórica, ahora bien son computacionalmente seguros.
- Seguridad práctica. Un criptosistema cumple los requisitos de la seguridad práctica si no se puede romper en un tiempo razonable con los recursos computacionales disponibles.

El problema de la transmisión de un mensaje a través de un canal con ruido es análogo al problema del secreto en un sistema criptográfico. En el primero, el emisor envía un mensaje m y se recibe m', el receptor debe recuperar m a partir del mensaje distorsionado para lo cual se añaden al mensaje original unos símbolos redundantes que permiten la detección y corrección de errores. En el segundo, el emisor del mensaje m lo distorsiona (lo cifra) y lo convierte en un mensaje cifrado m', el receptor debe conseguir m a partir de m' utilizando la clave si se trata del legítimo destinatario o sin ella si se trata de un criptoanalista.

1.4 Complejidad computacional

El elemento básico de la teoría de la complejidad computacional es el algoritmo y los recursos computacionales a los que se recurre para llevarlos a cabo son dos: tiempos de proceso y necesidad de almacenamiento en memoria.

Intuitivamente, un algoritmo es un procedimiento computacional que toma una entrada y produce una salida; es decir, es un programa escrito para una máquina determinada en un cierto lenguaje de programación y que alimentado con una entrada, produce una salida, utilizando como recursos computacionales el tiempo de ejecución y la cantidad de memoria.

Un algoritmo determinístico es aquel que sigue la misma secuencia de operaciones cada vez que se ejecuta con la misma entrada, mientras que un algoritmo probabilístico realiza ciertas decisiones aleatorias en el curso de su ejecución, por lo que no sigue necesariamente la misma secuencia de operaciones aunque se le proporcione sucesivamente la misma entrada.

Los recursos computacionales hacen referencia a la cantidad de memoria requerida y por eso, es conveniente precisar el tamaño de los números a utilizar. Cualquier entero n puede ser expresado en base $b > 1$ de la siguiente manera:

$$n = d_{k-1}b^{k-1} + d_{k-2}b^{k-2} + \ldots + d_1 b + d_0 = \sum_{i=0}^{k-1} d_i b^i, \tag{1.1}$$

con $d_{k-1} \neq 0$, $0 \leq d_i < b$, $1 \leq i \leq k-1$.

De la expresión (1.1) se deduce que

$$b^{k-1} \leq n \leq b^k$$

y tomando logaritmos

$$k - 1 \leq \log_b n \leq k.$$

Por lo tanto, el número de dígitos de n en la base b es la parte entera por defecto más 1

$$k = \lfloor \log_b n \rfloor + 1.$$

Si la expresión de n en base b dada en (1.1) se escribe como

$$n = (d_{k-1}, d_{k-2}, \ldots, d_1, d_0),$$

entonces n se llama k-dígito en base b. Si $b = 2$, n se dice un k-bit, cada d_i es un bit, d_{k-1} es el bit más significativo y d_0 el menos significativo o bit de paridad puesto que el número n es par si $d_0 = 0$ e impar si $d_0 = 1$.

Así, el tamaño de un entero es el número total de bits necesarios para representarlo en binario

$$\lfloor \log_2 n \rfloor + 1.$$

Existe una clasificación de los problemas matemáticos en función de la clase de algoritmos que se conoce para resolverlos y del tiempo necesario para determinar una solución del mismo. Esta clasificación se conoce como "Clases de complejidad" [81]. Se dice que un problema es de decisión, si su solución es la respuesta SI o NO. Un problema tiene una complejidad de clase P si pertenece al conjunto de todos los problemas de decisión que son resolubles en tiempo polinómico.

Recientemente M. Agraval, N. Kayal y N. Saxena [2] han demostrado que el problema de decidir si un número es primo o no, es un problema de clase P. Igualmente, un problema tiene una complejidad del tipo NP si es un problema para el que se puede obtener una respuesta SI en un tiempo polinómico, utilizando alguna información adicional. Y por último, un problema de decisión tiene una complejidad de tipo $co - NP$, si se puede obtener una respuesta NO en tiempo polinómico, utilizando alguna información adicional.

Es claro que se verifican las siguientes inclusiones

$$P \subseteq NP$$

y

$$P \subseteq co - NP.$$

Sin embargo las siguientes preguntas están aún sin resolver, aunque se cree que la respuesta es no a todas ellas:

(a) $P = NP$.
(b) $NP = co - NP$.
(c) $P = NP \cap co - NP$.

Dados dos problemas de decisión P_1, P_2, se dice que P_1 se reduce en un tiempo polinómico a P_2, representado como $P_1 <_p P_2$, si existe un algoritmo que resuelve P_1 y usa como subrutina un algoritmo para resolver P_2, de modo que el algoritmo que resuelve P_1 es de tiempo polinómico si también lo es para P_2. De forma resumida se puede decir que $P_1 <_p P_2$ implica que P_2 es al menos tan difícil como P_1.

Un problema de decisión P_1 se dice que es NP-completo si $P_1 \in NP$ y $P_2 <_p P_1$ para cualquier P_2 de tipo NP.

Los problemas NP-completos son los problemas más difíciles del tipo NP, en el sentido de que son al menos tan difíciles como cualquier otro problema.

1.5 Operaciones bit

La teoría de la complejidad computacional no debe depender de la velocidad de computación de la máquina empleada en cada momento, por eso se define una unidad llamada operación bit, que no dependa del procesador que se utilice.

Sean a y b enteros tales que $b \leq a$ y a de longitud k, para sumar a y b se expresan ambos en binario y si b no fuera un k-bit, se llenaría con ceros por la izquierda hasta dicho tamaño. A continuación se procede a sumar cada una de las k columnas observando el bit superior, el bit inferior y el acarreo.

Una operación bit consiste en considerar los dos bits a sumar y el acarreo y ejecutar una vez los siguientes pasos:

(a) Si ambos bits son 0 y no hay acarreo; el resultado es 0.
(b) Si ambos bits son 0 y el acarreo es 1; o si un bit es 0, el otro 1 y no hay acarreo, el resultado es 1.
(c) Si ambos bits son 1 y no hay acarreo; o si un bit es 0, el otro 1 y el acarreo es 1; el resultado es 0 y el acarreo 1, que se coloca en la siguiente columna.
(d) Si ambos bits son 1 y el acarreo es 1; el resultado es 1 y el acarreo 1, que se coloca en la siguiente columna.

La suma de dos enteros a y b, uno de k-bits y el otro de h-bits, requiere máx(k, h) operaciones bit. La resta necesita el mismo número de operaciones bits que la suma.

La multiplicación de dos enteros a y b, uno de k-bits y el otro de h-bits, con $k \geq h$,

requiere $(k+h)h$ operaciones bit. La división requiere el mismo número de operaciones bit que el producto.

Para expresar el número de operaciones bit de un algoritmo se utiliza la notación $O(f)$. Sean $f(x)$ y $g(x)$ dos funciones que toman valores positivos para cualquier entero n positivo, entonces se escribe $f = O(g)$ si existe una constante positiva c de modo que $f(n) \leq cg(n)$.

Ejemplo.

$$2n^2 + 3n - 5 = O(n^2),$$

pues

$$2n^2 + 3n - 5 \leq 3n^2.$$

Según esta notación, la suma de dos números enteros, n y m, de k dígitos requiere

$$O(k) = O(\log n)$$

operaciones bit, mientras que el producto de dos números n y m con k y h dígitos respectivamente, requiere

$$O(kh) = O(\log n \log m)$$

operaciones bit. El mejor orden de complejidad conocido para multiplicar dos enteros de k dígitos es

$$O(k \log k \log\log k) < O(k^2).$$

Un algoritmo tiene orden de complejidad polinómico si su ejecución depende polinomicamente del tamaño de la entrada; es decir, si n es la entrada y d es un entero, entonces el número de operaciones bit del algoritmo es $O(\log_d n)$.

Igualmente un algoritmo tiene orden de complejidad probabilístico si es un algoritmo aleatorio cuyo orden de complejidad esperado es polinómico.

Si un algoritmo tiene orden de complejidad exponencial el número de operaciones bit depende exponencialmente de la entrada; es decir, si n es la entrada y d es un entero, entonces el número de operaciones bit del algoritmo es

$$O(n^d).$$

Un algoritmo tiene orden de complejidad subexponencial si su orden es

$$O(L[x, c, \alpha]),$$

donde

$$L[x, c, \alpha] = e^{(c+\epsilon)(\ln x)^{\alpha}(\ln \ln x)^{1-\alpha}} = \exp((c+\epsilon)(\ln x)^{\alpha}(\ln \ln x)^{1-\alpha}),$$

siendo x el tamaño de la entrada, c una constante, $\epsilon = o(1)$ y $0 < \alpha < 1$.

Si en esta expresión anterior se hace $\alpha = 0$, el orden de complejidad será polinómico en $\ln x$,

$$O(L[x, c, \alpha]) = O(\exp((c+\epsilon)(\ln \ln x))).$$

Y si $\alpha = 1$, el orden será exponencial en $\ln x$,

$$O(L[x, c, 1]) = O(\exp((c+\epsilon)(\ln x))).$$

Un algoritmo de orden de complejidad subexponencial es asintóticamente mucho más rápido que un algoritmo cuyo orden de complejidad es exponencial en el tamaño de la entrada.

La diferencia entre un orden polinómico y uno exponencial se puede ver en el siguiente ejemplo. Dados dos algoritmos, uno exponencial, AE, y otro polinómico, AP, cuyos ordenes de complejidad son $O(n^3)$ y $O(\log^3 n)$ respectivamente. Si se dobla la entrada para el algoritmo exponencial, su orden de complejidad será $O(8n^3)$; es decir, se incrementa 8 veces al doblar la variable, mientras que para conseguir el mismo aumento con el algoritmo polinómico se necesita doblar el número de bits de la entrada $O(21 \log n)^3$.

El algoritmo de Euclides para calcular el mcd de dos números enteros a y b, es una división reiterada como sigue:

$$a = b\ q_0 + r_0,\ r_0 < b,$$
$$b = r_0\ q_1 + r_1,\ r_1 < r_0,$$
$$r_0 = r_1\ q_2 + r_2,\ r_2 < r_1,$$
$$\ldots$$

En este algoritmo, el tamaño del resto se reduce por lo menos a la mitad cada dos pasos. Por tanto hay que hacer como máximo $2[\log a]$ divisiones; es decir, el número de divisiones requiere $O(\log a)$ operaciones bit.

Cada una de las divisiones involucra dos números menores que a, luego, como máximo, el número de operaciones bit en cada división es $O(\log^2 a)$. Por consiguiente el orden de complejidad es polinómico

$$O(\log a)\ O(\log^2 a) = O(\log^3 a).$$

1.6 Teoría de grupos

Los esquemas criptográficos presentados en este trabajo se construyen a partir de una de las estructuras algebraicas más estudiadas: la estructura de grupo. A continuación se expresan algunas nociones elementales de teoría de grupos.

Definición 1.1: Se denomina grupo a un conjunto G no vacío dotado de una operación binaria $*$ que cumple:

(a) La operación $*$ es asociativa.
(b) Existe un elemento $e \in G$ tal que $e * g = g * e = g, \forall g \in G$. El elemento e se dice elemento neutro o identidad del grupo.
(c) Para cada $a \in G$, existe $b \in G$ tal que $a * b = b * a = e$. El elemento b se dice inverso de a y se denota por a^{-1}.

Si además se tiene $a * b = b * a$ para todo $a, b \in G$, el grupo G se dice que es abeliano.

Habitualmente, se denota la operación binaria $*$ por $\cdot$ o con una mera yuxtaposición de elementos, refiriéndose a ella como producto asociado al grupo, además e_G es el elemento neutro de un grupo G.

Definición 1.2: Dado un grupo G se denomina orden de G, denotado por $|G|$, al cardinal del conjunto subyacente G.

Definición 1.3: Sean G y H grupos. Una aplicación $f : G \longrightarrow H$ se dice que es un homomorfismo si $f(ab) = f(a)f(b), \forall a, b \in G$.

Definición 1.4: Un homomorfismo inyectivo (respectivamente suprayectivo) se denomina monomorfismo (respectivamente epimorfismo).

Definición 1.5: Un homomorfismo biyectivo se denomina isomorfismo y, si se establece de un grupo en sí mismo, se denomina automorfismo.

Teorema 1.1: *El conjunto de los automorfismos de un grupo G dado, tiene a su vez estructura de grupo con la operación composición de aplicaciones y se denota $Aut(G)$.*

Definición 1.6: Sea G un grupo y $x \in G$, la aplicación $f_x : G \longrightarrow G$ definida por $f_x(g) = xgx^{-1}$ es un automorfismo de G y se denomina automorfismo interno.

Definición 1.7: Dados dos elementos $h, g \in G$, se dice que son conjugados si existe un automorfismo interno de G que transforma h en g. Por ejemplo, existe $x \in G$ tal que $h = xgx^{-1}$.

Teorema 1.2: *La conjugación es una relación de equivalencia en G cuyas clases de equivalencia se denominan clases conjugadas.*

Definición 1.8: Sea G un grupo, un subconjunto no vacío $H \subset G$ se dice subgrupo de G si es estable respecto de la ley interna de G y tiene a su vez estructura de grupo con la restricción de la operación de G.

Teorema 1.3: *Un subconjunto no vacío H de G es un subgrupo si $\forall s, t \in H$ se tiene que $s^{-1} \in H$ y $st \in H$.*

Definición 1.9: Un subgrupo invariante por los automorfismos internos de G (i.e., tal que $xHx^{-1} = H, \forall x \in G$) se denomina normal o invariante.

Teorema 1.4: *Todo grupo G tiene al menos dos subgrupos normales, él mismo y el formado por el elemento neutro e_G. Además, estos dos subgrupos se dicen impropios y cualquier otro subgrupo se dice subgrupo propio de G.*

Se escribe $H \leq G$ para denotar que H es un subgrupo de un grupo G y si H es normal, se escribe $H \triangleleft G$. Para cualquier $X \subseteq G$ se denomina $\langle X \rangle_G$ (o $\langle X \rangle$, si no

hay ambigüedad) al menor subgrupo de G que contiene a X y se denomina subgrupo generado por X.

Definición 1.10: Al menor subgrupo normal de G que contiene a X se le denomina subgrupo normal generado por X y escribe $\langle X \rangle^G$.

Definición 1.11: Si $X = \{g\}$, para algún $g \in G$, el grupo $\langle X \rangle$ se denota por $\langle g \rangle$ y se dice que es un grupo cíclico. Se denomina orden de g al orden del grupo generado por g, $|\langle g \rangle|$.

Definición 1.12: Se denomina cogrupo a la izquierda (derecha) de G a todo conjunto de la forma gH (Hg) donde H es un subgrupo de G y $g \in G$.

Teorema 1.5: *El número de cogrupos a izquierda de H coincide con el número de cogrupos a derecha de H y se denomina índice de H en G (denotado $|G : H|$).*

Nótese que gH (respectivamente Hg) es la clase de equivalencia del elemento g respecto de la relación de equivalencia

$$g_1 \Re g_2 \Leftrightarrow g_1^{-1} g_2 \in H \quad (\textit{respectivamente } g_2 g_1^{-1} \in H).$$

Teorema 1.6: *El grupo G es una unión disjunta de cogrupos*

$$G = \bigcup_{i=1}^{|G:H|} g_i H, \quad g_i \in G, \quad (\textit{respectivamente } G = \bigcup_{i=1}^{|G:H|} H x_i, \quad x_i \in G).$$

Definición 1.13: El conjunto

$$\{g_i, i = 1, \ldots, |G : H|\} \ (\{x_i, i = 1, \ldots, |G : H|\})$$

se denomina conjunto completo de representantes a la izquierda (derecha) o transversal a la izquierda (derecha) de H en G.

La representación de G como unión disjunta de cogrupos permite demostrar el siguiente teorema.

Teorema 1.7: *(Teorema de Lagrange).*

Sea G un grupo y $H \leq G$. Entonces

$$|G| = |G : H||H|,$$

en particular, si G es finito $|H|$ divide a $|G|$.

Si $H \triangleleft G$, para cada $g \in G$ los cogrupos a izquierda y a derecha coinciden,

$$\forall g \in G, gH = Hg.$$

El conjunto de clases de equivalencia asociado a la relación de equivalencia $\Re$ anterior, se denota por G/H.

Teorema 1.8: *El conjunto G/H tiene estructura de grupo con la operación heredada de G y se denomina grupo cociente de G por H.*

$$g_1H * g_2H := g_1g_2H \quad \forall g_1, g_2 \in G/H.$$

Definición 1.14: Un grupo G se denomina simple si no posee ningún subgrupo normal propio.

Otra noción elemental que se utiliza a menudo es la de acción de un grupo sobre un conjunto, en la que juegan un papel fundamental los denominados grupos de permutaciones.

Definición 1.15: Sea X un conjunto cualquiera. El conjunto de biyecciones de X en sí mismo forma un grupo con la composición habitual de aplicaciones, denominado grupo de permutaciones sobre X. Dicho grupo se escribe S_X y depende sólo del cardinal de X. Así, si X es finito y está formado por n elementos puede denotarse alternativamente por S_n.

Definición 1.16: Sea G un grupo y X un conjunto. Una aplicación

$$\phi : G \times X \longmapsto X$$

denotada por $\phi(g, x) = gx$ se dirá acción de G sobre X si:

(a) $ex = x \ \ \forall\, x \in G$.
(b) $g(hx) = (gh)x \ \ \forall g, h \in G, x \in X$.

Definición 1.17: Se dice que G actúa sobre x, si existe un homomorfismo

$$\psi : G \longmapsto S_X \text{ con } \psi(g)(x) = gx = \phi(g, x), \ \ (g \in G, x \in X).$$

Definición 1.18: La acción es fiel si el homomorfismo asociado, ψ, es inyectivo, o, equivalentemente si la acción de G sobre X, ϕ, cumple

$$\phi(g, x) = x, \forall x \in X, \ \ g = e_G.$$

A través de la noción de acción se puede ver cada grupo como un grupo de permutaciones que actúa sobre sí mismo sin más que considerar la acción fiel de G sobre sí mismo definida por la multiplicación a izquierda ($\phi(g, x) := gh$).

Teorema 1.9: *Todo grupo G es isomorfo a un subgrupo del grupo de permutaciones $S_{|G|}$.*

Los grupos de permutaciones juegan un papel fundamental en criptografía, pues, en muchas ocasiones, las aplicaciones de cifrado no son más que permutaciones del conjunto de mensajes.

El uso de los grupos en criptografía está muy extendido y algunos de estos criptosistemas se pueden ver en [8, 40, 46, 52, 66, 76, 93, 94].

1.7 Cuerpos finitos

Definición 1.19: Sea A un conjunto y sean $+$ y $\cdot$ dos operaciones binarias. Se dice que la terna $(A, +, \cdot)$ es un anillo conmutativo o abeliano si se cumplen las siguientes propiedades:

(a) $(A, +)$ es un grupo abeliano.
(b) $(A, \cdot)$ tiene las propiedades asociativa, tiene elemento neutro y es distributiva respecto a $+$.

Ejemplo.

La terna $(\mathbb{Z}, +, \cdot)$ es un anillo conmutativo, cuyo elemento neutro para la suma es el 0 y para el producto el 1.

Definición 1.20: Un cuerpo es un conjunto F provisto de dos operaciones internas, $(F, +, \cdot)$, de modo que tanto $(F, +)$ como $(F, \cdot)$ son grupos conmutativos, y el producto es distributivo respecto a la suma.

Definición 1.21: Un cuerpo finito es un cuerpo con un número finito de elementos. Se suele escribir $\mathbb{F}_q$ para indicar un cuerpo finito con q elementos (orden q).

Ejemplo.

Si p es primo, cada entero no divisible por p tiene inverso módulo p; por tanto, $\mathbb{Z}_p$ es un cuerpo. El cuerpo $\mathbb{Z}_p$ desempeña un papel fundamental en la teoría de cuerpos finitos, como indica el siguiente teorema:

Teorema 1.10: *Se verifican las siguientes propiedades:*

(a) *El orden de un cuerpo finito $\mathbb{F}$ es igual a la potencia (m) de un número primo (p), p^m.*
(b) *Solo hay un cuerpo finito de p^m elementos.*

Definición 1.22: Se denomina característica del cuerpo, al número primo p mencionado en el teorema anterior (1.10).

Teorema 1.11: *Sea $\mathbb{F}$ un cuerpo de orden $q = p^m$, entonces cualquier subgrupo finito $\mathbb{F}^*$, es cíclico. En particular, $\mathbb{F}^*$ es cíclico de orden $q - 1$.*

1.8 Teoría de la divisibilidad

Dados $a, b \in \mathbb{Z}$, se dice que a divide a b si existe otro entero c tal que $b = ac$. Esta expresión se denotará por $a|b$. Un divisor d de un número a se dice propio si no es ni 1 ni el propio número. Algunas propiedades elementales de la divisibilidad son:

(a) $a|a$.
(b) Si $a|b$ y $b|c, \rightarrow a|c$.
(c) Si $a|b$ y $b|c, \rightarrow a|(bx + cy), \forall x, y \in \mathbb{Z}$.
(d) Si $a|b$ y $b|a, \rightarrow a = \pm b$.

La operación de dividir un número entero a entre otro número entero positivo b, proporciona como resultado un cociente c y un resto r, de modo que $a = bc + r$, siendo $0 \leq r < b$; además c y r son únicos.

Un número entero se dice primo si no tiene divisores propios, es decir, si sólo es divisible por sí mismo y por la unidad. De igual forma se define la "función recuento de números primos" así:

$$\pi : [1, \infty) \rightarrow \mathbb{R},\ \pi(x) = |\{ \text{ todos los números primos } p \text{ tales que } p \leq x\}|.$$

Esta función recuento de número primos, $\pi(x)$, para valores grandes de x, se aproxima a la expresión $\frac{x}{\ln x}$; de forma más precisa, se verifica que:

$$\lim_{x \rightarrow \infty} \frac{\ln x}{x} \pi(x) = 1. \tag{1.2}$$

Ejemplo.

Si $x = 10^{10}$, el número de primos menores que x es

$$\pi(10^{10}) = 455052511.$$

Es decir, alrededor del 21,97 % de los números menores que 10^{10} son primos. Mientras que el valor dado por la aproximación de la expresión (1.2) equivale al 23,02 %

$$\left\lfloor \frac{10^{10}}{\ln 10^{10}} \right\rfloor = 434294481.$$

Teorema 1.12: *(Teorema fundamental de la aritmética).*

Todo número compuesto $n \geq 2$ admite una factorización única como producto de potencias de primos de la forma

$$n = \prod_{i=1}^{k} p_i^{e_i} = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k},$$

donde p_i son enteros primos tales que $p_1 < \cdots < p_k$ y e_i enteros positivos.

Ejemplo.

El número $n = 36655171356678336$ tiene la siguiente descomposición en factores:

$$36655171356678336 = 2^6 \cdot 3^2 \cdot 7^2 \cdot 11^3 \cdot 31237^2.$$

Para cualquier par de enteros a, b, se verifica que

$$ab = mcd(a, b)mcm(a, b)$$

y estos números enteros se dice que son primos entre si o coprimos si se verifica que $mcd(a, b) = 1$.

Teorema 1.13: *(Teorema de la división de Euclides).*

Dados dos números enteros a, b tales que $a > b > 0$, se verifica:

$$mcd(a, b) = mcd(b, r),$$

donde r es el resto de dividir a entre b, es decir, $a = bc + r$, con $b > r$.

Algoritmo de Euclides.

Este algoritmo tiene como entrada dos enteros a, b con $a > b > 0$, y como salida el $mcd(a, b)$. Consiste en realizar divisiones sucesivas en las que se toma como dividendo y como divisor en cada división, el divisor y el resto, respectivamente, de la división anterior, hasta que el resto se haga cero, entonces, el último divisor es el máximo común divisor.

Si $r_0 = a$, $r_1 = b$, para $k \geq 1$, se escribe

$$r_{k-1} = r_k c_k + r_{k+1},\ r_{k+1} < r_k.$$

Utilizando el teorema de Euclides de forma iterada se tiene:

$$mcd(a,b) = mcd(r_1, r_2) = \cdots = mcd(r_{n-1}, r_n) = mcd(r_n, 0) = r_n.$$

El cálculo del $mcd(a,b)$ de dos enteros, $a > b$, mediante el Algoritmo de Euclides requiere $O((\lg a)^3)$ operaciones bit. Se trata por consiguiente de un algoritmo de orden de complejidad polinómico.

Optimizando la programación se puede rebajar el orden mencionado anteriormente (véanse [9, 31]) hasta convertirlo en $O(\lg a \ \lg b)$.

Del cómputo del máximo común divisor por el algoritmo de Euclides se deduce el siguiente teorema.

Teorema 1.14: *(Identidad de Bezout).*

Sean $a, b \in \mathbb{Z}$ y $d = mcd(a,b)$, existen dos números enteros u, v, tales que

$$d = ua + vb.$$

Este teorema establece la existencia de los valores u y v, pero no prueba que sean únicos. De hecho, cualquier otro par de valores u' y v' de la forma

$$u' = u \pm k\frac{b}{d}$$

o

$$v' = v \pm k\frac{a}{d},$$

verifican también la Identidad de Bezout, donde el signo positivo o negativo en cada una de las expresiones coincide con el de u y v.

Algoritmo de Euclides Extendido.

Este algoritmo tiene como entrada dos enteros a y b tales que $a > b > 0$ y como salida los números enteros u, v y d tales que

$$d = mcd(a,b) = ua + vb.$$

Para ello se utiliza el algoritmo de Euclides sin más que ir despejando desde la última división obtenida hasta llegar a la primera, de modo que se obtiene d como función de a y de b. Este algoritmo de Euclides extendido requiere de $O((\lg a)^3)$ operaciones bit y al igual que el algoritmo de Euclides, optimizando la programación se puede mejorar este orden de modo que sea $O(\lg a \ \lg b)$ operaciones bit.

1.9 Aritmética modular

Definición 1.23: Sea $n \in \mathbb{Z}$ tal que $n \geq 2$. Dados $a, b \in \mathbb{Z}$, se dice que a y b son congruentes módulo n, (representado por $a \equiv b\ (mod\ n)$), si su diferencia es un múltiplo de n, es decir, $n|(a-b)$, o dicho de otra forma, si a y b tienen el mismo resto al ser divididos por n. De esta definición se deduce que si r es el resto de dividir a entre n, entonces $a = r\ (mod\ n)$.

Esta relación de congruencia que se acaba de definir es una relación binaria en $\mathbb{Z}$, que es de equivalencia puesto que cumple las propiedades reflexiva, simétrica y transitiva.

Definición 1.24: La relación binaria de equivalencia de las congruencias módulo n divide a $\mathbb{Z}$ en clases de equivalencia de modo que si $a \in \mathbb{Z}$, la clase de equivalencia de a es

$$[a] = \{b \in \mathbb{Z} \ / \ b = a\ (mod\ n)\}.$$

Si $a > n$, se tiene $a = cn+r$ con $0 \leq r < n$, entonces $a = r\ (mod\ n)$. Por tanto, cada entero a es congruente con un único entero $r \in [0, n-1]$. Dado que a y r están en la misma clase de equivalencia, r puede ser considerado como el representante canónico de dicha clase de equivalencia y como cada clase módulo n admite un representante canónico $r \in [0, n-1]$, se puede efectuar la siguiente identificación:

$$\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$$

donde $\mathbb{Z}_n$ es el conjunto de las clases de equivalencia de los enteros módulo n.

Ejemplo.

Si $n = 7$, se tiene

$$[4] = \{\ldots, -10, -3, 4, 11, 18, \ldots\} = \{4 + 7k \text{ tal que } k \in \mathbb{Z}\},$$

y si $n = 15$

$$[4] = \{\ldots, -26, -11, 4, 19, 34, \ldots\} = \{4 + 15k \text{ tal que } k \in \mathbb{Z}\},$$

con lo que

$$\mathbb{Z}_7 = \{0, 1, 2, 3, 4, 5, 6\},$$

$$\mathbb{Z}_{15} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}.$$

En el conjunto $\mathbb{Z}_n$ se pueden realizar las operaciones de suma y producto ordinarias, siempre módulo n.

Definición 1.25: Si $a = b \ (mod\ n)$ y $a' = b' \ (mod\ n)$, entonces

$$a \pm a' = b \pm b' \ (mod\ n).$$

Definición 1.26: Si $a = b \ (mod\ n)$ y $a' = b' \ (mod\ n)$, entonces

$$aa' = bb' \ (mod\ n).$$

La suma y la resta de dos enteros a, b, módulo n, requieren $O(\lg n)$ operaciones bit y el producto de dos enteros módulo n requiere $O((\lg n)^2)$ operaciones bit.

Con las operaciones suma y producto ordinarias, $\mathbb{Z}_n$ es un anillo conmutativo.

Definición 1.27: Un elemento $a \in \mathbb{Z}_n$ es invertible módulo n si existe $c \in \mathbb{Z}_n$ tal que $ac \ (mod\ n) = 1$. Tal entero c se denota por a^{-1}. En este caso se puede efectuar la división de cualquier elemento $b \in \mathbb{Z}_n$ por a, de modo que $b/a \ (mod\ n) = ba^{-1} \ (mod\ n)$.

Definición 1.28: Un elemento no nulo $a \in \mathbb{Z}_n$ es un divisor de cero si existe otro elemento no nulo $b \in \mathbb{Z}_n$ tal que $ab \ (mod\ n) = 0$.

Resulta inmediato comprobar que en $\mathbb{Z}_n$ todos los divisores de n son divisores de cero y que son invertibles todos los enteros positivos menores que n y primos con n; por consiguiente, si n es primo, todos los enteros positivos menores que n son invertibles (véase [54]).

Un algoritmo que permite calcular el inverso de un entero a módulo n es el siguiente:

Algoritmo para el cálculo del inverso módulo n.

Tiene como entrada dos enteros n, a tales que $n > a > 0$ y proporciona como salida el inverso de a módulo n (si existe). Para ello basta con utilizar la identidad de Bezout de la siguiente manera: sea $a \in \mathbb{Z}_n$, si a es invertible se tiene que $mcd(n, a) = 1$, por lo que existen enteros u, v tales que

$$un + va = 1.$$

Tomando módulo n resulta

$$un + va \ (mod\ n) = va \ (mod\ n) = 1,$$

es decir,

$$a^{-1} = v \ (mod\ n).$$

Este algoritmo necesita $O(\lg a)(\lg n)$ operaciones bit.

Un teorema fundamental para resolver ecuaciones que involucran congruencias es el Teorema chino del resto.

Teorema 1.15: *(Teorema chino del resto).*

Sean $n_1, n_2, \ldots n_k$, enteros primos entre sí dos a dos, es decir

$$mcd(n_i, n_j) = 1, \quad i \neq j.$$

El sistema de congruencias simultáneas

$$x = a_1 \ (mod\ n_1),$$

$$x = a_2 \ (mod\ n_2),$$

$$\ldots$$

$$x = a_k \ (mod\ n_k),$$

tiene una solución única $n = n_1 n_2 \cdots n_k$.

Este teorema garantiza la existencia de solución de un sistema de congruencias. Para calcular la solución se utiliza el siguiente algoritmo.

Algoritmo de Gauss.

La solución x al sistema de congruencias simultáneas del Teorema chino del resto viene dada por

$$\sum_{i=1}^{k} a_i N_i M_i \ = x \ (mod\ n).$$

Donde $N_i = n/n_i$ en $\mathbb{Z}$ y $M_i = N_i^{-1} \ (mod\ n_i)$. Además, N_i siempre es invertible módulo n_i pues $mcd(N_i, n_i) = 1$.

Este algoritmo de Gauss [9, 31] se puede ejecutar en $O((\lg n)^2)$ operaciones bit, donde n es el módulo de la congruencia.

Definición 1.29: Se denomina grupo de las unidades de $\mathbb{Z}_n$ al conjunto de sus elementos invertibles y se denota por $\mathbb{Z}_n^*$. Por definicion, sus elementos son:

$$\mathbb{Z}_n^* = \{a \in \mathbb{Z} \ / \ mcd(a, n) = 1\}.$$

En particular, si n es un primo, $\mathbb{Z}_n^* = \mathbb{Z}_n - \{0\}$.

Con la operación del producto ordinario, $\mathbb{Z}_n^*$ es un grupo conmutativo.

Definición 1.30: Sea $n \geq 1$ un entero y $\phi(n)$ el número de enteros en el intervalo $[1, n]$ que son primos con n. La función $n \longmapsto \phi(n)$ se denomina indicador ϕ de Euler (Euler Totient Function).

A continuación se enumeran las propiedades fundamentales del indicador de Euler.

(a) Si p es un número primo, entonces $\phi(p) = p - 1$.

(b) El indicador ϕ de Euler es multiplicativo, es decir, dados dos enteros, m y n, tales que $mcd(m, n) = 1$, entonces $\phi(mn) = \phi(m)\phi(n)$.

(c) Si $n = p^e$, con p primo, entonces $\phi(n) = \phi(p^e) = (p - 1)p^{e-1}$.

(d) Si $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$, con $p_1, p_2, \cdots, p_k$ factores primos, se deduce que

$$\phi(n) = n \prod_{i=1}^{k} (1 - \frac{1}{p_i}).$$

(e) Si n es un entero, entonces

$$\sum_{d|n, d>0} \phi(d) = n.$$

Ejemplo.

Dado $p = 23$ se tiene que $\phi(23) = 22, n = 729 = 2^3 \cdot 3^2 \cdot 11$, luego

$$\phi(729) = 729 \prod_{i=1}^{3} (1 - \frac{1}{p_i}) = 729(1 - \frac{1}{2})(1 - \frac{1}{3})(1 - \frac{1}{11}) = 240.$$

Se enuncian ahora dos teoremas fundamentales en la teoría de números como son los de Euler y Fermat.

Teorema 1.16: *(Teorema de Euler).*

Para todo elemento $a \in \mathbb{Z}_n^*$, se verifica que:

$$a^{\phi(n)} \ (mod\ n) = 1.$$

Si n es un número primo se conoce como congruencia de Fermat.

Teorema 1.17: *(Congruencia de Fermat).*

Si p es un número primo, para todo $a \in \mathbb{Z}_n$, se verifica

$$a^p \ (mod\ p) = a.$$

Y si a no es divisible por p

$$a^{p-1} \ (mod\ p) = 1.$$

De estos dos teoremas se deducen propiedades interesantes:

(a) Si n es un entero de modo que $r = s \ (mod\ \phi(n))$, entonces $\forall a \in \mathbb{Z}_n$ también se verifica $a^r \ (mod\ n) = a^s \ (mod\ n)$.
Ejemplo.
Para calcular 12^{428} en $\mathbb{Z}_{729}$, basta con tener en cuenta que
$\phi(729) = 240,\ 482 \ (mod\ 240) = 2 \longrightarrow 12^{482} \ (mod\ 729) = 12^2 \ (mod\ 729) = 144.$
El resto de dividir 12^{244} entre 23 se determina de forma análoga, como
$\phi(23) = 22$ y $244 \ (mod\ 22) = 2$ se tiene que $12^{244} \ (mod\ 23) = 12^2 \ (mod\ 23) = 144.$

(b) Si el orden de $a \in \mathbb{Z}_n^*$ es r y además $a^s \ (mod\ p) = 1$, entonces r divide a s. En particular, r divide a $\phi(n)$.

(c) El conjunto $\mathbb{Z}_n^*$ tiene un generador si y solo si $n = 2, 4, p^k, o\ 2p^k$, donde p es un primo y k un entero positivo.

(d) Si g es un generador de $\mathbb{Z}_n^*$, entonces $g^i \ (mod\ n) = a$, es también un generador de $\mathbb{Z}_n^*$ si y solo si $mcd(i, \phi(n)) = 1, \quad 0 \leq i \leq \phi(n)$. Por tanto, el número de generadores de $\mathbb{Z}_n^*$ es $\phi(\phi(n))$.

(e) Un elemento $g \in \mathbb{Z}_n$ es generador de $\mathbb{Z}_n^*$ si y sólo si para cada divisor d de $\phi(n)$ se tiene que $g^{\phi(n)/d} \ (mod\ n) \neq 1$.

Capítulo 2

Criptosistemas de clave pública

2.1 Esquemas basados en factorización.

Uno de los pilares que cimientan la criptografía de clave pública se sustenta en el problema de factorización de enteros que consiste en encontrar para un $N \in \mathbb{Z}$, el conjunto de números primos $p_1, p_2, \ldots, p_k$ tales que $N = p_1 p_2 \ldots p_k$ (véase [102]). El esquema más importante basado en este problema de factorización es RSA y es uno de los sistemas criptográficos más analizados e implementados en la breve historia de la criptografía de clave pública (véase [85]). Otras muchas construcciones criptográficas, con utilidades muy diversas, se fundamentan también en el problema de factorización (véanse [70, 75, 99, 122]).

2.1.1 Criptosistema RSA

En 1978 R.L. Rivest, A. Shamir y L. Adleman (véase [103]) idearon un sistema criptográfico que basa su seguridad en la dificultad de la factorización de un entero de gran tamaño y consta de tres partes:

(a) Generación de claves
(b) Cifrado del mensaje
(c) Descifrado del mensaje

2.1.1.1 Generación de claves

(a) Se eligen dos números primos p y q suficientemente grandes.

(b) Se calcula n como producto $n = pq$ y se considera como conjunto de mensajes a utilizar el grupo multiplicativo $\mathbb{Z}_n^*$ cuyo orden es $\phi(n)$, donde $\phi(n)$ es la función indicador de Euler definida en la sección 1.9,

$$\phi(n) = \phi(pq) = (p-1)(q-1).$$

(c) Se elige $e \in \mathbb{Z}$, tal que $1 < e < \phi(n)$ de modo que sea primo con el orden del grupo, es decir

$$mcd(e, \phi(n)) = 1.$$

(d) Mediante el algoritmo de Euclides extendido se calcula el inverso de e en $\mathbb{Z}_{\phi(n)}^*$, es decir, se calcula $d \in \mathbb{Z}$ tal que

$$ed = 1 \ (mod\ \phi(n)),$$

con $1 < d < \phi(n)$.

(e) La clave pública está constituida por la pareja (n, e) y la clave privada es el número d. Los números $p, q, \phi(n)$ también deben permanecer secretos.

Definición 2.1: Los valores e y d se denominan exponente de cifrado y exponente de descifrado, respectivamente.

Definición 2.2: El entero n se denomina módulo del criptosistema RSA.

En ocasiones, en lugar de utilizar el valor $\phi(n) = \phi(pq) = (p-1)(q-1)$, se utiliza el valor $\lambda = mcm(p-1, q-1)$, que es conocido como exponente universal de n (véase [81]). La razón de este cambio se debe a que se verifica

$$\phi(n) = (p-1)(q-1) = mcm(p-1, q-1)mcd(p-1, q-1).$$

En cualquier caso, si p y q se eligen de forma aleatoria, se espera que $mcd(p-1, q-1)$ sea pequeño y, por tanto, $\phi(n)$ y λ sean aproximadamente del mismo tamaño.

2.1.1.2 Cifrado de mensajes

Si un usuario U desea enviar un mensaje m a otro usuario V, debe realizar las operaciones siguientes:

(a) Obtiene la clave pública de V, (n_V, e_V).
(b) Representa el mensaje m como un elemento de $\mathbb{Z}_{n_V}^*$, es decir, como un entero de rango $\{0, 1, 2, \ldots, n_V - 1\}$.
(c) Envía a V el valor del criptograma $c = m^{e_V} (mod\ n_V)$.

En RSA los mensajes que se transmiten son elementos de $\mathbb{Z}_n^*$ y si se desea transmitir un mensaje más largo, se debe dividir en bloques de tal manera que cada bloque sea un elemento de $\mathbb{Z}_n^*$.

2.1.1.3 Descifrado de mensajes

Para recuperar el mensaje original m, el usuario V utiliza su clave privada d_V y calcula

$$c^{d_V}\ (mod\ n_V) = (m^{e_V})^{d_V}\ (mod\ n_V) = m^{e_V d_V}\ (mod\ n_V) = m.$$

Ejemplo.

Se utilizan primos pequeños para ejemplificar el método RSA.

Determinación de la clave

(a) El usuario V elige dos número primos de forma secreta, los multiplica para obtener n, y determina el indicador de Euler

$$p = 383, q = 521,$$

$$n = pq = 383 \cdot 521 = 199543,$$

$$\phi(n) = (p-1)(q-1) = 382 \cdot 520 = 198640.$$

(b) Elige el exponente de cifrado, $2 < e < 198640$, por ejemplo $e = 3$ y comprueba que

$$mcd(3, 198640) = 1.$$

(c) Utiliza el algoritmo de Euclides extendido para obtener el inverso

$$198640 = 66213 \cdot 3 + 1,$$

de donde

$$-66213 \cdot 3\ (mod\ 198640) = 132427 \cdot 3\ (mod\ 198640) = 1,$$

es decir, $d = 132427$.

(d) Finalmente, el usuario V da a conocer su clave pública,

$$(n, e) = (199543, 3),$$

manteniendo oculta su clave privada $d = 132427$, así como los restantes valores, $p = 383, q = 521$ y $\phi(n) = 198640$.

Cifrado del mensaje en claro

Si el usuario U desea enviar el mensaje "RSA" a V realiza los siguientes pasos:

(a) Obtiene la clave pública de V, $(199543, 3)$.

(b) Codifica el mensaje a enviar como un número menor que $n = 199543$. Para ello se considera la tabla

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Se emplea base 26 para representar cualquier palabra. De este modo

$$26^3 = 17576 < n = 199543 < 456976 = 26^4,$$

es decir, cada mensaje parcial puede contener como máximo 3 caracteres. En el ejemplo $R \to 17,\ S \to 18,\ A \to 0$, con lo que el mensaje es

$$RSA \to m = 17 \cdot 26^2 + 18 \cdot 26 + 0 = 911.$$

(c) A continuación el usuario U cifra m calculando

$$\begin{aligned} c &= m^e \ (mod\ n) \\ &= m^3 \ (mod\ n) \\ &= m^2 m \ (mod\ n) \\ &= 911^2 \cdot 911 \ (mod\ 199543) \\ &= 31749 \cdot 911 \ (mod\ 199543) \\ &= 189147. \end{aligned}$$

Transforma este número a base 26 para convertirlo en caracteres,

$$\begin{aligned} 189147 &= 26 \cdot 7274 + 23, \\ 7274 &= 26 \cdot 279 + 20, \\ 279 &= 26 \cdot 10 + 19, \end{aligned}$$

de donde

$$c = 189147 = 10 \cdot 26^3 + 19 \cdot 26^2 + 20 \cdot 26 + 23 \longmapsto KTUX.$$

Descifrado del mensaje

Una vez que el usuario V recibe el criptograma enviado por U: "KTUX", lo representa como un número en base 26, obteniendo $c = 189147$ y procede del siguiente modo:

(a) Utiliza su clave privada $d = 132427$ y calcula

$$m = c^d \ (mod\ n) = 189147^{132427} \ (mod\ 199543).$$

Como

$$\begin{aligned} d &= 132427 \\ &= 100000010101001011_{(2} \\ &= 2^{17} + 2^{10} + 2^8 + 2^6 + 2^3 + 2 + 1, \end{aligned}$$

el usuario V realiza las siguientes potencias (modulo n)

$$\begin{aligned} 189147^2 \ (mod\ 199543) &= 189147 \cdot 189147 \ (mod\ 199543) = 124053, \\ 189147^{2^2} \ (mod\ 199543) &= 124053^2 \ (mod\ 199543) = 191106, \\ 189147^{2^3} \ (mod\ 199543) &= 191106^2 \ (mod\ 199543) = 145661, \\ 189147^{2^4} \ (mod\ 199543) &= 145661^2 \ (mod\ 199543) = 118817, \\ &\cdots \ \cdots \\ 189147^{2^{17}} \ (mod\ 199543) &= 173001^2 \ (mod\ 199543) = 90974. \end{aligned}$$

A continuación multiplica las potencias que aparecen reduciendo módulo n después de cada producto, es decir

$$189147 \cdot 124053 \ (mod\ 199543) = 190964,$$

$$\begin{aligned} 190964 \cdot 145661 \ (mod\ 199543) &= 112090, \\ 112090 \cdot 133139 \ (mod\ 199543) &= 128626, \\ 128626 \cdot 188504 \ (mod\ 199543) &= 45574, \\ 45574 \cdot 18 \ (mod\ 199543) &= 22160, \\ 22160 \cdot 90974 \ (mod\ 199543) &= 911\ . \end{aligned}$$

(b) Finalmente recupera el mensaje original sin más que escribir el valor obtenido $m = 911$ en base 26, es decir:

$$m = 911 = 17 \cdot 26^2 + 18 \cdot 26 + 0 \longmapsto RSA.$$

2.1.2 Firma digital en RSA

Para firmar digitalmente un mensaje, el remitente debe llevar a cabo determinados cálculos con el mensaje que desea enviar y con su clave privada. De este modo, cada mensaje lleva su propia firma y se puede comprobar que el remitente es el único que lo ha podido firmar, dado que es el único que posee su clave privada.

La firma digital del esquema RSA es explícita (añadida como una marca inseparable al mensaje), pública (permite identificar al remitente ante cualquier persona) e irrevocable porque el receptor puede probar que el remitente escribió el mensaje.

El protocolo de firma digital consta de dos partes: la firma del mensaje y el proceso de verificación.

Sean (n_U, e_U) y d_U las claves pública y privada, respectivamente, de un usuario U, y sean (n_V, e_V) y d_V, las claves del usuario V.

Si el usuario U desea enviar, junto con el criptograma c del mensaje m, su firma digital para ese mensaje, procede a ejecutar el siguiente protocolo:

(a) Calcula mediante el exponente de descifrado, d_U, el valor de

$$r = m^{d_U} (mod\ n_U),$$

que es la rúbrica para el mensaje m.

(b) Luego cifra el valor anterior con la clave pública de V,

$$s = r^{e_V} (mod\ n_V).$$

El mensaje cifrado que el usuario U envía a V es la pareja (c, s) y es claro que solo U puede firmar el mensaje, dado que es el único que conoce su clave privada d_U.

Para que V pueda verificar que la firma corresponde a U solo tiene que ejecutar los siguientes pasos:

(a) Recupera la rúbrica del usuario U para el mensaje m calculando

$$s^{d_V} \ (mod\ n_V) = (r^{e_V})^{d_V} \ (mod\ n_V) = r^{e_V d_V} \ (mod\ n_V) = r.$$

(b) Posteriormente V comprueba si la rúbrica cifrada coincide con el mensaje

$$r^{e_U} \ (mod\ n_U) = m^{d_U e_U} \ (mod\ n_U) = m.$$

En el caso de que el resultado anterior no coincida con el valor de m obtenido en el proceso de descifrado, el mensaje original es rechazado.

El ataque contra el protocolo de la firma digital de un mensaje con el criptosistema RSA es el mismo que el utilizado para romper el propio criptosistema, dado que en ambos se realizan las mismas operaciones.

2.1.3 Criptoanálisis elemental RSA

2.1.3.1 Elección de los primos p y q

Los primos p y q deben elegirse de forma que factorizar n sea computacionalmente muy costoso, por lo que estos deben ser grandes, además se les debe exigir que tengan aproximadamente la misma longitud, dado que si uno de ellos es mucho más pequeño que el otro, es más fácil de obtener (véase [81]) y, determinado uno de ellos, el otro se calcula sin más que dividir el módulo RSA entre el primo ya calculado.

Actualmente, se recomienda que p y q tengan cada uno de ellos una longitud mínima de 512 bits, y por tanto, n tenga, 1024 bits (alrededor de 309 dígitos). Los valores de p y q no deben estar demasiado cercanos, ya que en tal caso si, por ejemplo $p < q$, entonces

$$\frac{q-p}{2},$$

es un número pequeño y se tiene que

$$\frac{p+q}{2} \approx \sqrt{n},$$

de modo que la diferencia

$$\left(\frac{p+q}{2}\right)^2 - n = \left(\frac{p-q}{2}\right)^2,$$

también es pequeña.

Como el segundo miembro es un cuadrado perfecto, con pocos tanteos se puede encontrar un entero $x > \sqrt{n}$, tal que $x^2 - n = y^2$.

Entonces se obtienen $p = x - y$ y $q = x + y$, con lo que n queda factorizado.

Además $mcd(p-1, q-1)$ debe ser pequeño, ya que en caso contrario

$$u = mcm(p-1, q-1) = \frac{\phi(n)}{mcd(p-1, q-1)}$$

es pequeño en comparación con $\phi(n)$. Ahora bien, cualquier inverso de e módulo u, d', tal que $ed'\ (mod\ u) = 1$ sirve como exponente de descifrado; esto es, se verifica que

$$m^{ed'}\ (mod\ n) = m.$$

Si u es pequeño, esta propiedad se aprovecha para romper el criptosistema llevando a cabo los siguientes pasos:

(a) Se elige un valor para u y se calcula el inverso d' de e módulo u.

(b) Se cifran varios mensajes sucesivos con el criptosistema y se intenta descifrarlos utilizando d'. Si en alguno de los ensayos se tiene éxito el sistema queda roto y en caso contrario se repite el paso (a).

Si u es realmente pequeño en comparación con $\phi(n)$ el éxito de los pasos anteriores se alcanza con relativa eficacia computacional.

Por otra parte, $p-1$ y $q-1$ deben contener factores primos grandes ya que, en caso contrario, se tiene que los factores de $\phi(n) = (p-1)(q-1)$ también son pequeños. En esta situación, si todos los factores son menores que una cota M, es posible averiguar todos los candidatos v a ser $\phi(n)$ y probar si c elevado a la potencia $(v+1)/e$ (supuesto que sea un número), proporciona un mensaje legible.

Definición 2.3: Un primo p se dice robusto si verifica las tres propiedades siguientes:

(a) $p-1$ tiene un factor primo grande r.

(b) $p+1$ tiene también un factor primo grande s.

(c) $r-1$ tiene también un factor primo grande t.

2.1.3.2 Elección del exponente de cifrado e

En general, una vez que se ha determinado un valor grande para el módulo RSA, se recomienda seleccionar un exponente de cifrado e pequeño para tratar de facilitar la

tarea de cifrado, es decir, que sea eficiente.

Por esta razón se recomienda que el exponente de cifrado sea 3 o 65537, y esta recomendación se basa en dos hechos:

(a) Tanto 3, como 65537 son números primos.

(b) La expresión binaria de estos números es muy sencilla, lo que facilita mucho el cálculo de $m^e \ (mod\ n)$. Para $e = 3 = 11_{(2}$, el cálculo de m^3 en $\mathbb{Z}_n$,

$$m^3 = m^2 m = (mm)m,$$

sólo necesita dos multiplicaciones modulares; mientras que para calcular m^{65537} en $\mathbb{Z}_n$, hacen falta 17 multiplicaciones modulares, pues

$$e = 65537 = 2^{16} + 1 = 10000000000000001_{(2}$$

y por tanto

$$m^{65537} = m^{2^{16}+1} = m^{2^{16}} m.$$

Ahora bien, para calcular $m^{2^{16}} (mod\ n)$ basta con calcular m^2, y luego ir elevando al cuadrado las potencias que se van obteniendo:

$$m^{2^2} = m^2 m^2, \quad m^{2^3} = m^{2^2} m^{2^2}, \quad m^{2^4} = m^{2^3} m^{2^3},$$

$$\dots$$

$$m^{2^{14}} = m^{2^{13}} m^{2^{13}}, \quad m^{2^{15}} = m^{2^{14}} m^{2^{14}}, \quad m^{2^{16}} = m^{2^{15}} m^{2^{15}}.$$

Por otra parte, varios usuarios pueden tener el mismo exponente de cifrado, e, pero en este caso deben tener distinto n, ya que si no fuera así, el conocimiento que un usuario tiene de su par (e, d), le permitiría conocer el exponente de descifrado de todos los demás, puesto que todos tendrían la misma clave.

No se deben utilizar valores pequeños de e y a la vez enviar un mismo mensaje a varios destinatarios. Si por ejemplo, $e = 3$, y alguien envía el mismo mensaje, m a tres destinatarios diferentes, cuyos módulos fueran n_1, n_2, n_3, se obtienen los siguientes criptogramas: $c_i = m^3 \ (mod\ n_i)$ para $i = 1, 2, 3$. Entonces, un atacante puede intentar resolver el siguiente sistema de congruencias

$$x \equiv c_1 \ (mod\ n_1),$$

$$x \equiv c_2 \ (mod\ n_2),$$

$$x \equiv c_3 \ (mod\ n_3).$$

Si consigue resolverlo (fácil utilizando el Teorema chino del resto si los módulos RSA son primos entre sí dos a dos), se tiene que $x = m^3 < n_1 n_2 n_3$ y extrayendo la raíz cúbica de x, se recupera el mensaje original.

La misma situación se presenta cuando los mensajes no son exactamente iguales, pero sí muy parecidos, y son conocidas las variaciones entre los mismos.

Para prevenir estos ataques, se debe generar secuencias de bits de longitudes adecuadas, de modo que sean añadidas al comienzo del mensaje antes de cifrarlo. Cada una de estas secuencias de bits se debe generar de forma independiente para cada uno de los mensajes a enviar, este proceso se conoce como "salar" el mensaje (véase [56,58]).

Se debe evitar cifrar mensajes cortos cuando se utilizan exponentes de cifrado pequeños. En este caso, si se verifica que $m < n^{1/e}$, entonces m se recupera a partir de la expresión $c = m^e \ (mod\ n)$ sin más que calcular la raíz e-ésima ordinaria de c en $\mathbb{N}$, que existe porque m es precisamente esa raíz. Este problema también se puede evitar "salando" los mensajes originales.

2.1.3.3 Elección del exponente de descifrado d

El exponente de descifrado d, debe ser de longitud aproximadamente igual a la de n. Si

$$\text{longitud en bits}(d) \ \leq \ \frac{1}{4} \ \text{longitud en bits}(n)$$

entonces existe un algoritmo eficiente para calcular d (véase [121]).

2.1.3.4 Propiedades del mensaje

En ocasiones algunas propiedades del mensaje facilitan su criptoanálisis cuando se cifra con RSA. Algunas de estas debilidades son:

Mensajes previsibles

Si el espacio de mensajes es pequeño o si el contenido del mensaje es previsible, un criptoanalista puede descifrar un mensaje simplemente cifrando todos los posibles mensajes hasta obtener alguno que sea igual al criptograma, con lo que se obtiene la clave. De nuevo "salar" el mensaje original puede ser un método para prevenir este tipo de ataques.

Mensajes inocultables

Definición 2.4: Un mensaje m es inocultable si se cifra a sí mismo, es decir, si verifica

$$m^e = m \ (mod \ n).$$

La existencia de tales mensajes es clara, por ejemplo, $m = 0, 1, n-1$. De hecho, el número de mensajes inocultables (véase [20]) es

$$(1 + mcd(e-1, p-1))(1 + mcd(e-1, q-1)).$$

Dado que $e-1, p-1$ y $q-1$ son todos pares, se deduce que al menos hay 9 mensajes inocultables. Si se eligen p, q y e aleatoriamente, la proporción de mensajes inocultables es muy pequeña y puede afirmarse que no afecta a la seguridad de RSA.

Multiplicidad del mensaje

Si m_1, m_2 son dos mensajes en claro, y c_1, c_2 sus respectivos criptogramas, se verifica la siguiente propiedad multiplicativa

$$(m_1 m_2)^e = m_1^e m_2^e = c_1 c_2 \ (mod \ n),$$

es decir, el criptograma correspondiente a un mensaje que es producto de dos mensajes, $m = m_1 m_2$, es el producto de los dos criptogramas, $c = c_1 c_2$.

Esta propiedad permite elaborar el siguiente ataque adaptable a c:

Si un adversario activo, C, desea descifrar el criptograma $c = m^e \ (mod \ n)$, de un mensaje enviado al usuario V, tiene que interceptar c, elegir un entero aleatorio $k \in \mathbb{Z}_n^*$ y calcular $\bar{c} = ck^e \ (mod \ n)$. A continuación C envía $\bar{c}$ a V, quien calcula $\bar{m} = \bar{c}^d \ (mod \ n)$ para el adversario. Ahora bien, como

$$\bar{m} = \bar{c}^d \ (mod \ n) = c^d (k^e)^d \ (mod \ n) = c^d k \ (mod \ n) = mk \ (mod \ n),$$

el atacante recupera el mensaje en claro original sin más que calcular

$$m = \bar{m} k^{-1} \ (mod \ n).$$

Este ataque puede evitarse imponiendo algunas propiedades estructurales a los mensajes en claro (véase [81]).

Ataques cíclicos

Sea $c = m^e \ (mod\ n)$ un texto cifrado. Dado que el proceso de cifrado no es más que una permutación del espacio de mensajes $\{0, 1, \ldots, n-1\}$, existe un entero positivo k tal que

$$c^{e^k} \ (mod\ n) = c,$$

por lo que se verifica

$$c^{e^{k-1}} \ (mod\ n) = m.$$

A partir de esta propiedad, un adversario puede llevar a cabo un ataque cíclico a RSA sin más que calcular los valores

$$c^e \ (mod\ n), c^{e^2} \ (mod\ n), c^{e^3} \ (mod\ n), \ldots$$

hasta obtener c, en cuyo caso el valor de m es el penúltimo valor computado.

De forma más general, se conoce como ataque cíclico generalizado, a un ataque encaminado a localizar el menor entero positivo k tal que

$$u = mcd(c^{e^k}) > 1.$$

Si se tiene $c^{e^k} \ (mod\ p) = c$ y $c^{e^k} \ (mod\ q) \neq c$ entonces $u = p$. De forma similar, si $c^{e^k} \ (mod\ p) \neq c$ y $c^{e^k} \ (mod\ q) = c$ entonces $u = q$. En cualquiera de los dos casos, se factoriza n y un atacante puede recuperar el exponente de descifrado d.

Por otra parte si $c^{e^k} \ (mod\ p) = c$ y $c^{e^k} \ (mod\ q) = c$ se verifica que $u = n$ y $c^{e^k} \ (mod\ n) = c$. En efecto, k es el menor entero positivo para el que se verifica la congruencia anterior y el ataque tiene éxito pues $c^{e^{k-1}} \ (mod\ n) = m$, que puede ser computado de forma eficiente. Ahora bien, el ataque cíclico generalizado se considera como un método para factorizar n más que un ataque al esquema RSA.

2.1.4 Primalidad en RSA

El primer paso para la generación de claves de RSA consiste en obtener de forma aleatoria dos números primos grandes, hecho que se conoce como problema de la primalidad.

La dificultad práctica del problema está precisamente en el tamaño de los números ya que determinar la factorización de un número o demostrar que no la tiene, es una tarea difícil. Por esta razón se han desarrollado métodos que permiten analizar la primalidad de un número sin necesidad de estudiar si tiene o no factorización.

Un método consiste en comprobar si el número candidato a ser primo cumple ciertas condiciones que verifican todos los números primos, de modo que si las cumple se le considera primo y si no verifica una de las propiedades se le considera compuesto. Sin embargo, la verificación de tales propiedades requiere, en general, mucho esfuerzo computacional si se quiere establecer con rigor la primalidad del número. Por ello, en la práctica se suele recurrir a métodos que exigen comprobaciones menos costosas, pero que, en contrapartida, no dan una respuesta correcta con absoluta seguridad; esto motiva las siguientes definiciones.

Definición 2.5: Se llama test de primalidad a un algoritmo determinista que decide si un número candidato a ser primo realmente lo es, basándose en el cumplimiento de ciertas propiedades por parte del número candidato.

Definición 2.6: Se llama test de pseudoprimalidad a un algoritmo que determina si un número candidato es compuesto, basándose en el cumplimiento de ciertas propiedades por parte del número candidato.

Es decir, un test de primalidad decide con total seguridad sobre la primalidad de un número. Sin embargo, un test de pseudoprimalidad sólo es capaz de determinar con toda seguridad que el candidato es compuesto, mientras que la decisión de que el candidato sea primo sólo es probabilística.

Definición 2.7: Un número n que pasa un test de pseudoprimalidad se denomina un primo probable.

2.1.4.1 Test de primalidad

Estos test de primalidad exigen una mayor cantidad de recursos computacionales que los de pseudoprimalidad, por lo que son poco utilizados. Su utilización se suele reservar para garantizar la primalidad de un número del que se tienen fundadas sospechas de que es primo. Los más conocidos son:

(a) Test de las divisiones sucesivas.
- Algoritmo de las divisiones sucesivas.
- Algoritmo de las divisiones incompletas.

(b) Test de Lucas-Lehmer [9].
- Algoritmo de Lucas-Lehmer.

(c) Test de Pocklington-Lehmer [24, 95].
- Algoritmo de Pocklington-Lehmer.

2.1.4.2 Test de pseudoprimalidad

Estos tests determinan si un número es compuesto basándose en el cumplimiento de ciertas propiedades, entre los más utilizados están:

(a) Test de Slovay-Strassen [31, 115].
- Algoritmo de Slovay-Strassen.

(b) Test de Miller-Rabin [83, 100].
- Algoritmo de Miller-Rabin.

(c) Test de Miller-Rabin modificado [81].
- Algoritmo de Miller-Rabin modificado.

2.2 El logaritmo discreto

Sea G un grupo abeliano finito, g un elemento de ese grupo y G^* el subgrupo de G generado por g. Dado $h \in G^*$, el Problema del Logaritmo Discreto (DLP) consiste en encontrar un entero n tal que $g^n = h$. Conocidos g y n es computacionalmente sencillo calcular h, sin embargo, dados g y h, se considera intratable (computacionalmente hablando) determinar n. Este hecho no es aplicable para ciertos grupos G como se verá más adelante.

Ejemplo.

Sea $p = 97$ y $\mathbb{F}_{97}^*$ un subgrupo cíclico de $\mathbb{Z}_{97}$ de orden $q = 96$ generado por $g = 5$. Como $5^{32} \ (mod\ 97) = 37$, se tiene que $\log_5 37 \ (mod\ 97) = 32$.

Sean g_1 y g_2 dos generadores de un grupo cíclico G de orden q y sea $h \in G$ y supongamos $x_1 = \log_{g_1} h, \quad x_2 = \log_{g_2} h, \quad y = \log_{g_1} g_2$, entonces $g_1^{x_1} = h = g_2^{x_2} = (g_1^y)^{x_2}$ y por tanto $x_1 = yx_2 \ (mod\ q)$. Es decir, la dificultad del DLP es independiente del generador escogido.

Así pues, cualquier algoritmo que calcule logaritmos en base g_1 puede utilizarse para calcular logaritmos en cualquier otra base g_2, siendo g_1 y g_2 generadores del grupo.

Definición 2.8: Definición formal del logaritmo discreto.

Sea $\mathbb{F}_q$ un cuerpo finito, donde $q = p^n$ (con p primo) y $\mathbb{F}_q^* = \mathbb{F}_q - \{0\}$. Dado $g \in \mathbb{F}_q$ un elemento primitivo o generador de $\mathbb{F}_q$ e y un elemento arbitrario de $\mathbb{F}_q^*$, el logaritmo discreto de y en base g se define como:

$$\log_g y = x \Leftrightarrow \ g^x = y \text{ en } \mathbb{F}_q \ \text{ y } \ 0 \leq x \leq q-2.$$

Gauss [41] se refiere al logaritmo discreto como el índice de un número y da algunas propiedades básicas (módulo $q-1$):

- $\log_g(y_1 y_2)\ (mod\ q-1) = \log_g y_1 + \log_g y_1\ (mod\ q-1)$.
- $\log_g(\frac{y_1}{y_2})\ (mod\ q-1) = \log_g y_1 - \log_g y_1\ (mod\ q-1)$.
- Cambio de base: supuesto δ otro generador de $\mathbb{F}_q$ tal que $\log_g \delta = h$ se tiene que

$$\log_g y = \log_g \delta \log_\delta y\ (mod\ q-1).$$

Un método obvio para encontrar el logaritmo discreto de y es simplemente ir probando potencias hasta encontrar un exponente x tal que $g^x = y$ en $\mathbb{F}_q$. Sin embargo, si q es muy grande, este método es intratable con los algoritmos y los medios computacionales actuales. La intratabilidad del DLP hace que se emplee en aplicaciones criptográficas, muchas de las cuales han sido desarrolladas e implementadas en las tres últimas décadas. Estos esquemas criptográficos han hecho que muchos investigadores se centren en el estudio del logaritmo discreto intentando crear algoritmos rápidos para solventar este problema.

El logaritmo discreto en cuerpos finitos grandes $\mathbb{F}_q$ se utiliza en criptografía debido a que tiene las características de una función unidireccional, esto es, mientras que el problema del logaritmo discreto parece intratable, su inverso (exponenciación discreta) es relativamente fácil de computar.

Dado $\mathbb{F}_q$ un cuerpo finito y $x \in \mathbb{F}_q$ tal que $0 \leq x \leq q-2$, se puede computar $g^x \in \mathbb{F}_q$ con $2\log_2 q$ multiplicaciones modulares. Por ejemplo, en

$$g^{19} = g^{10011_{(2}} = g^{10000_{(2}+10_{(2}+1_{(2}} = (((g^2)^2)^2 g)^2 g,$$

se puede evitar carga computacional tomando una reducción modular después de cada multiplicación.

Actualmente, un valor de q que es considerado como seguro (computacionalmente hablando) es el comprendido entre 10^{150} y 10^{350}, dependiendo de la importancia del

mensaje cifrado, del tiempo que deba permanecer secreto y de los recursos computacionales a disposición de los que deseen comprometer ese secreto. Además, hay otro aspecto del cuerpo que puede afectar a la facilidad de encontrar logaritmos discretos y es que $q-1$ tenga, al menos, un factor primo grande.

2.2.1 Esquemas basados en el DLP

Se han escrito gran cantidad de artículos y libros sobre criptosistemas basados en la intratabilidad del DLP (véanse [55, 62, 72]). A continuación se presentan algunos de los esquemas criptográficos más extendidos, cuya seguridad esta basada en la dificultad de cómputo del logaritmo discreto bajo ciertas condiciones.

2.2.1.1 Intercambio de clave de Diffie-Hellman

Al establecer una comunicación privada, entre dos interlocutores, en un canal inseguro, la primera necesidad es acordar una clave secreta, que es usada por ambos interlocutores para cifrar y descifrar la información. Si se tiene un grupo de n usuarios en el que cada pareja debe compartir una clave, en total son necesarias $n(n-1)/2$ claves diferentes (véase [12, 98]), hecho que plantea un problema en su distribución de forma segura. Con el objetivo de evitar este problema, Diffie y Hellman [35] describieron un protocolo por medio del cual dos personas pueden intercambiar información secreta sin compartir previamente una clave, este protocolo se conoce como "Intercambio de clave de Diffie-Hellman" y consiste en:

(a) Los usuarios U y V seleccionan públicamente un grupo multiplicativo finito G de orden n y generador $g \in G$.
(b) El usuario U genera un número aleatorio a, $1 \leq a \leq n-1$, calcula $g^a \in G$ y transmite este elemento a V, manteniendo secreto a.
(c) El usuario V genera un número aleatorio b, $1 \leq b \leq n-1$, calcula $g^b \in G$ y transmite este elemento a U, manteniendo secreto b.
(d) El usuario U recibe g^b y calcula $(g^b)^a \in G$.
(e) El usuario V recibe g^a y calcula $(g^a)^b \in G$.

Ahora U y V poseen un elemento común y secreto del grupo, $g^{ab} \in G$. Un atacante, S, puede conocer G, n, g^a y g^b pero calcular el secreto compartido g^{ab} es un problema intratable que se conoce como Problema de Diffie-Hellman Generalizado (GDHP). Si el grupo G es el grupo multiplicativo de los enteros módulo un número primo, se le denomina simplemente Problema de Diffie-Hellman (DHP). El DHP tiene a lo sumo una

complejidad similar al DLP y bajo ciertas condiciones sobre el grupo, son equivalentes.

Ejemplo.

Sea $p = 53$ un número primo, $G = \mathbb{Z}_{53}^*$ y $g = 2$ un generador de G. El protocolo de Diffie-Hellman es el siguiente

(a) El usuario U genera $a = 29$, calcula $g^a \ (mod\ 53) = 2^{29} \ (mod\ 53) = 45$ y se lo envía a V, manteniendo secreto $a = 29$.
(b) El usuario V genera $b = 19$, calcula $g^b \ (mod\ 53) = 2^{19} \ (mod\ 53) = 12$ y se lo envía a U, manteniendo secreto $b = 19$.
(c) El usuario U recibe 12 y calcula $12^{29} \ (mod\ 53) = 21$.
(d) El usuario V recibe 45 y calcula $45^{19} \ (mod\ 53) = 21$.

La clave privada o la información secreta que comparten ahora U y V es 21.

Para un primo p suficientemente grande es computacionalmente muy costoso que pueda averiguar la información secreta compartida por U y V.

2.2.1.2 El sistema de Massey-Omura

Sea $\mathbb{F}_q$ un cuerpo finito de orden $q = p^n$ (con p primo). Los usuarios U y V eligen aleatoriamente como claves públicas, sendos enteros e_U y e_V menores que $p-1$ y primos con él. Con la ayuda del algoritmo de Euclides extendido cada usuario calcula su clave privada d_U, d_V tal que

$$d_U e_U \ (mod\ q-1) = 1 \ (mod\ q-1) \text{ y } d_V e_V \ (mod\ q-1) = 1 \ (mod\ q-1).$$

Se supone ahora que se tiene una correspondencia entre los mensajes M y los elementos de $\mathbb{F}_q$. Si el usuario U quiere enviar al usuario V el mensaje $m \in M$ procede así:

(a) Calcula $c = m^{e_U} \ (mod\ q-1)$ y se lo envía a V.
(b) El usuario V calcula $c^{e_V} \ (mod\ q-1)$ y se lo devuelve a U.
(c) El usuario U calcula

$$f = (c^{e_V})^{d_U} \ (mod\ q-1) = ((m^{e_U})^{e_V})^{d_U} \ (mod\ q-1) = m^{e_V} \ (mod\ q-1).$$

(d) El usuario U envía f a V.
(e) El usuario V recupera el mensaje calculando

$$f^{d_V} \ (mod\ q-1) = (m^{e_V})^{d_V} \ (mod\ q-1) = m.$$

Este sistema realiza una doble ida y vuelta de mensajes entre ambos usuarios además de cuatro exponenciaciones, lo que provoca que sea lento y que exista cierta vulnerabilidad. Sin embargo el principal problema es que necesita un buen sistema adicional de identificación.

Ejemplo.

Sea el cuerpo finito $\mathbb{F}_{63476311}$

- El usuario U elige como clave pública $e_U = 343457$ y por tanto su clave privada es $d_U = 42217493$.
- El usuario V elige como clave pública $e_V = 454621$ y por tanto su clave privada es $d_V = 19954321$.
  El usuario U envía a V el mensaje "VEN HOY POR LA TARDE", cuyo equivalente numérico es:

$$(22, 05, 14, 28, 08, 15, 25, 28, 16, 15, 18, 28, 12, 01, 28, 20, 01, 18, 04, 05).$$

  Como $63476311 > 3 \cdot 10^8$ se puede tomar bloques de 4 letras, y entonces

$$m = (22051428, 08152528, 16151828, 12012820, 01180405).$$

- El usuario U envía a V los enteros

$$(22051428, 08152528, 16151828, 12012820, 01180405)^{343457} =$$
$$(36042738, 49108552, 52398562, 13357833, 48656509).$$

- El usuario V envía a U

$$(36042738, 49108552, 52398562, 13357833, 48656509)^{454621} =$$
$$(24264800, 31727189, 32852597, 5580293, 3764961).$$

- El usuario U envía ahora a V

$$(24264800, 31727189, 32852597, 5580293, 3764961)^{42217493} =$$
$$(36042738, 49108552, 52398562, 13357833, 48656509).$$

- Finalmente V calcula

$$(36042738, 49108552, 52398562, 13357833, 48656509)^{19954321} =$$
$$(22051428, 08152528, 16151828, 12012820, 01180405).$$

Al descomponer en caracteres individuales se recupera el mensaje original

$$(22, 05, 14, 28, 08, 15, 25, 28, 16, 15, 18, 28, 12, 01, 28, 20, 01, 18, 04, 05)$$
"VEN HOY POR LA TARDE".

2.2.1.3 Criptosistema de ElGamal

En 1985, ElGamal [37] propuso un criptosistema basado también en la exponenciación discreta sobre un cuerpo finito $\mathbb{Z}_p$. No obstante, aquí se presenta un protocolo más general sobre un grupo cíclico finito G de orden p, con p primo.

Considerando que los mensajes son elementos de un grupo finito G, si los usuarios U y V desean intercambiar un mensaje $m \in G$, utilizan el siguiente protocolo:

(a) Acuerdan un grupo finito G y un generador α de G.
(b) El usuario U elige $a \in G$, que es su clave privada, y calcula α^a en G, que es su clave pública.
(c) El usuario V elige $b \in G$, que es su clave privada, y calcula α^b en G, que es su clave pública.
(d) El usuario U elige un $v \in G$ y calcula α^v en G.
(e) El usuario U, con la clave pública de V, calcula $(\alpha^b)^v$ y $m\alpha^{bv}$ en G.
(f) Envía la pareja $(\alpha^v, m\alpha^{bv})$ a V.

Para recuperar el mensaje original, el usuario V calcula

$$(\alpha^{vb}) \text{ y } (\alpha^{vb})^{-1}$$

y obtiene m mediante

$$(m\alpha^{vb})(\alpha^{vb})^{-1}.$$

Por seguridad y eficacia, el grupo G y el elemento α se deben elegir de modo que verifiquen las siguientes condiciones:

- La operación en G debe ser fácil de aplicar.
- El problema del logaritmo discreto en el subgrupo cíclico de G generado por α, $\langle\alpha\rangle$, debe ser computacionalmente intratable.

Para simplificar el protocolo anterior se puede suponer, tal y como fue descrito por ElGamal [37], que el grupo sobre el que se llevan a cabo las operaciones es el grupo multiplicativo del cuerpo $\mathbb{Z}_p$; de esta forma las potencias y productos anteriores se efectúan módulo un número primo p.

Ejemplo.

Se considera el grupo $\mathbb{Z}^*_{15485863}$, con $p = 15485863$ primo y un generador $\alpha = 7$ de $\mathbb{Z}^*_{15485863}$.

Los usuarios U y V eligen las claves privadas $a = 28236$ y $b = 21702$ y calculan las

claves públicas

$$\alpha^a = 7^{28236} \ (mod\ 15485863) = 12506884,$$

$$\alpha^b = 7^{21702} \ (mod\ 15485863) = 8890431.$$

Si el usuario U desea enviar a V el mensaje $m =$"HIJO", en primer lugar codifica el mensaje expresándolo en base 26 (suponiendo un alfabeto de 26 letras) mediante la tabla

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

$$m = HIJO = 7 \cdot 26^3 + 8 \cdot 26^2 + 9 \cdot 26 + 14 = 128688.$$

Elige el número $v = 489$ y calcula $\alpha^v = 7^{489} \ (mod\ 15485863) = 12001315$.

Calcula también los valores

$$(\alpha^b)^v = 8890431^{489} \ (mod\ 15485863) = 9846598,$$

$$m\alpha^{vb} = 128688 \cdot 9846598 \ (mod\ 15485863) = 8263449$$

y decodifica el par $(\alpha^v, m\alpha^{vb}) = (12001315, 8263449)$, es decir,

$$\alpha^v = 12001315 = 1 \cdot 26^5 + 0 \cdot 26^4 + 6 \cdot 26^3 + 21 \cdot 26^2 + 11 \cdot 26 + 1 = BAGVLB,$$

$$m\alpha^{vb} = 8263449 = 18 \cdot 26^4 + 2 \cdot 26^3 + 4 \cdot 26^2 + 0 \cdot 26 + 25 = SCEAZ.$$

Por tanto, el mensaje que el usuario U envía a V es: $(BAGVLB, SCEAZ)$.

Si el usuario V desea descifrar el mensaje, en primer lugar codifica en base 26 la pareja recibida

$$BAGVLB = 1 \cdot 26^5 + 0 \cdot 26^4 + 6 \cdot 26^3 + 21 \cdot 26^2 + 11 \cdot 26 + 1 = 12001315 = \alpha^v,$$

$$SCEAZ = 18 \cdot 26^4 + 2 \cdot 26^3 + 4 \cdot 26^2 + 0 \cdot 26 + 25 = 8263449 = m\alpha^{vb}.$$

A continuación calcula

$$(\alpha^v)^b = 12001315^{21702} \ (mod\ 15485863) = 9846598 \ \text{ y } \ (m\alpha^{vb})(\alpha^{vb})^{-1}.$$

Para ello debe determinar el inverso de α^{vb} módulo 15485863 mediante el algoritmo de Euclides extendido

$$-662582 \cdot \alpha^{vb} + 421299 \cdot p = 1,$$

$$(\alpha^{vb})^{-1} = -662582 \ (mod\ 15485863) = 14823281.$$

Se tiene que

$$m = (m\alpha^{vb})(\alpha^{vb})^{-1} = (8263449) \cdot (14823281) \ (mod\ 15485863) = 128688$$

y el mensaje original se recupera codificando en base 26

$$m = 128688 = 7 \cdot 26^3 + 8 \cdot 26^2 + 9 \cdot 26 + 14 = HIJO.$$

La firma digital de ElGamal es explícita, pública e irrevocable y consta de dos partes: la firma del mensaje y el proceso de verificación. Para implementar el esquema de firma digital de ElGamal, se debe restringir a un cuerpo de orden primo p ($\mathbb{F}_p$) aunque actualmente puede ser extendido a cuerpos de orden p^n con p primo, usando una representación entera de los elementos del cuerpo que se necesiten (véase [119]).

Para firmar digitalmente un mensaje $m \in \mathbb{F}_q$ tal que $1 \leq m \leq p-1$, el usuario U procede así:

(a) Genera un número aleatorio h tal que $1 \leq h \leq p-1$ y $mcd(h, p-1) = 1$.
(b) Calcula el elemento $\alpha^h \ (mod\ p) = r$.
(c) Resuelve la congruencia $m = ar + hs \ (mod\ p-1)$.

La firma digital de U para el mensaje m es el par (r, s).

Para comprobar la firma, el receptor del mensaje realiza lo siguiente:

(a) Calcula $r^s \ (mod\ p) = (\alpha^h)^s \ (mod\ p)$.
(b) Calcula $(\alpha^a)^r(\alpha^h)^s \ (mod\ p)$ y comprueba que es igual a $\alpha^m \ (mod\ p)$.

Para conseguir la falsificación de la firma de U en el mensaje m, un atacante tiene que resolver la ecuación

$$\alpha^m = (\alpha^a)^r r^s,$$

con incógnitas r y s.

Si el atacante fija r y trata de resolver la ecuación en s, se encuentra con un problema del logaritmo discreto; mientras que si fija s e intenta resolver la ecuación para r, se encuentra ante una congruencia exponencial mixta, para la que no hay algoritmo conocido. Este problema se conoce como el Problema de la firma digital de ElGamal (ESP).

Ejemplo.

Se ve a continuación el modo en que el usuario U envía a V su firma digital para el mensaje m =“HIJO” (con los mismos datos que en el cifrado del mensaje).

Se considera $\mathbb{Z}^*_{15485863}$, con $p = 15485863$ primo y un generador $\alpha = 7$. Las claves privadas de U y V son, respectivamente, $a = 28236$ y $b = 21702$, y las claves públicas 12506884 y 8890431.

El usuario U calcula su firma digital para el mensaje m como sigue:

(a) Elige un número aleatorio $h = 90725$, que es primo con el orden del grupo

$$mcd(90725, 15485862) = 1.$$

(b) Calcula $r = 7^{90725}\ (mod\ 15485863) = 7635256$.

(c) Resuelve la congruencia $m = ar + hs\ (mod\ p-1)$, es decir

$$128688 = 28236 \cdot 7635256 + 90725 \cdot s\ (mod\ 15485862).$$

La ecuación anterior se resuelve despejando s

$$\begin{aligned} s &= (128688 - 28236 \cdot 7635256)(90725)^{-1}\ (mod\ 154858662) \\ &= (5211036)(90725)^{-1}\ (mod\ 154858662) \\ &= (5211036 \cdot 11031191)\ (mod\ 154858662) = 11047464. \end{aligned}$$

(d) La firma de U para el mensaje anterior es

$$(r, s) = (7635256, 11047464).$$

Para comprobar la firma del usuario U, V procede así:

(a) Calcula

$$(\alpha^h)^s = r^s = 7635256^{11047464}\ (mod\ 15485863) = 8799713,$$

$$(\alpha^a)^r = 12506884^{7635256}\ (mod\ 15485863) = 1260686,$$

y

$$\alpha^m = 7^{128688}\ (mod\ 15485863) = 5362356.$$

(b) Verifica que

$$r^s(\alpha^a)^r = (8799713 \cdot 1260686)\ (mod\ 15485863) = 5362356 = \alpha^m.$$

En 1994, el Instituto Nacional de Estándares y Tecnología de Estados Unidos (véase [89]), adoptó una versión ligeramente modificada de este esquema (DSS-Digital Signature Standard) para ser usada por las agencias gubernamentales.

2.2.2 Algoritmos de computación

2.2.2.1 Fórmulas explícitas

A continuación se describen fórmulas explícitas para el cálculo de logaritmos discretos en cuerpos finitos.

Wells [120] proporciona una forma polinómica explícita para el cálculo de logaritmos discretos en cuerpos primos y Mullen y White ([87]) proporcionan otra forma explícita para el cálculo de logaritmos discretos en cuerpos finitos cuyo orden es potencia de un número primo. Posteriormente, Niederreiter [90] proporciona demostraciones simples de estas fórmulas explícitas, ademas de generalizar la fórmula de Wells a cuerpos de orden primo.

Teorema 2.1: *(Fórmula de Wells).*

Sea $q = p^n$, $n \geq 1$, el orden de un cuerpo finito $\mathbb{F}_q$ e $y \in \mathbb{F}_q$. Se tiene que

$$\log_g y = \sum_{i=0}^{n-1} x_i p^i,$$

donde $x_i \in \{0, 1, 2, \ldots, p-1\}$ $\forall i = 0, 1, \ldots, n-1$.

Teorema 2.2: *Dado $\mathbb{F}_q$ un cuerpo finito con $q \geq 3$ y $k \in \mathbb{N}$ tal que $0 \leq k \leq q-1$, entonces*

$$\sum_{c \in \mathbb{F}_q,\ c \neq 1} \frac{c^k}{1-c} \in \mathbb{F}_p,$$

y la suma es congruente con k módulo p.

Teorema 2.3: *(Fórmula de Mullen y White).*

Dado $y \in \mathbb{F}_q^*$, con $q \geq 3$, se tiene que

$$\log_g y = -1 + \sum_{j=1}^{q-2} \frac{y^j}{g^{-j} - 1} \ (mod\ p) \ \in \mathbb{F}_q^*.$$

Si se define el logaritmo discreto de forma diferente, se consigue otra formula explí-

cita, que es una generalización de la de Wells aplicada a los cuerpos de orden potencia de un número primo.

Teorema 2.4: *(Generalización de la fórmula de Wells).*

Dado $y \in \mathbb{F}_q^*$, *con* $q \geq 3$, *se tiene que*

$$\log_g y = \sum_{j=1}^{q-2} \frac{y^j}{1-g^j} \ (mod\ p) \in \mathbb{F}_q^*.$$

2.2.2.2 Algoritmos de raíz cuadrada

En esta sección se presentan algoritmos para el cálculo de logaritmos discretos, cada uno de los cuales puede alcanzar un orden de complejidad $\sqrt{\overline{p}}$, donde $\overline{p}$ es el mayor factor primo de $q-1$. Esto no es bastante rápido para usarse en cuerpos finitos grandes y arbitrarios, pero si $q-1$ no tiene factores primos grandes, estos algoritmos pueden ser muy prácticos.

Varios de los ejemplos que se verán usan el cuerpo $\mathbb{F}_{37}^*$, generado por $g = 5$ y cuya tabla es

x	5^x	x	5^x	x	5^x
0	1	12	10	24	26
1	5	13	13	25	19
2	25	14	28	26	21
3	14	15	29	27	31
4	33	16	34	28	7
5	17	17	22	29	35
6	11	18	36	30	27
7	18	19	32	31	24
8	16	20	12	32	9
9	6	21	23	33	8
10	30	22	4	34	3
11	2	23	20	35	15

Método de la búsqueda exhaustiva

Dado un grupo G de orden n con generador g, el método de búsqueda exhaustiva consiste en calcular $g^0, g^1, g^2, \ldots$, hasta encontrar el valor buscado. Este método es intratable computacionalmente, pues requiere $O(n)$ multiplicaciones.

Método paso gigante paso enano

Este algoritmo fue creado por Shanks en 1981 (véase [110]) y consiste en calcular $x \ (mod\ q-1)$ dado un grupo G de orden q con generador g, conocido $y = g^x \ (mod\ q)$.

Si $m = \left\lceil (q-1)^{\frac{1}{2}} \right\rceil$, en la división euclidiana $x = i + jm$ se tiene

$$g^x = g^{jm} g^i \quad \longmapsto \quad y(g^{-m})^j = g^i,$$

con $0 \leq i, j \leq m$. Con esto, el algoritmo para el cálculo de i, j es el siguiente:

(a) Se construye una tabla con las parejas (i, g^i), para $0 \leq i \leq m$, y se ordena la lista de acuerdo con la segunda componente.
(b) Se calcula g^{-m}.
(c) Para $0 \leq j \leq m$, se calcula yg^{-jm} hasta encontrar un j tal que $yg^{-jm} = g^i$ para algún i de la lista.
(d) Para esta pareja i, j, se calcula $y = g^{i+jm}$.

Ejemplo.

Dado el grupo $\mathbb{Z}_{37}$ con generador $g = 5$, se tiene que $m = \left\lceil (37-1)^{\frac{1}{2}} \right\rceil = 6$. Se construye y ordena la lista siguiente por su segunda componente

$$(i, g^i) \to (0,1)(1,5)(3,14)(5,17)(2,25)(4,33).$$

Para encontrar $\log_5 2$, se procede así:

- Si $j = 0$, se calcula $2 \cdot 5^{-0 \cdot 6} = 2$.
- Si $j = 1$, se calcula $2 \cdot 5^{-1 \cdot 6} = 2 \cdot 5^{30} = 17$.

El número 17 aparece como segunda componente en la lista ordenada y corresponde a $i = 5$, luego se tiene

$$\log_5 2 = 5 + 1 \cdot 6 = 11.$$

Este método requiere una tabla de m entradas, ordenar la tabla y realizar la comparación. En total son $O(m \log m)$ operaciones bit.

Para alcanzar un orden de complejidad de alrededor $\sqrt{\overline{p}}$ (con $\overline{p}$ = mayor factor primo de $q-1$), se hace uso del algoritmo dividido en diversas etapas, donde en cada etapa se calcula un logaritmo en un subgrupo de $\mathbb{F}_q^*$ de orden no mayor que $\overline{p}$. Pollard [97] fue el primero en proponer esta idea multietapa.

Ejemplo.

Para mostrar un ejemplo se va a calcular $\log_5 2$ en $\mathbb{F}_{37}$ en tres etapas (nótese que $37-1=3^2\cdot 2^2$).

ETAPA 1, en un subgrupo de orden 3.

Se usa el algoritmo del paso gigante paso enano para encontrar el logaritmo de y^{12}, que para la base g^{12} es 2, esto es $y^{12}=(g^{12})^2$.

Si se toma la doceava raíz, se tiene $y=g^2(g^3)^{k_1},\ 0\le k_1\le 11$ y, puesto que g^3 es la doceava raíz de la unidad,

$$x=2+3k_1.$$

ETAPA 2, en un subgrupo de orden 3.

Para encontrar k_1, dado $y_2=yg^{-2}=(g^3)^{k_1}$, se utiliza el algoritmo del paso gigante paso enano para encontrar el logaritmo de y_2^4 en base $(g^3)^4$. Puesto que el logaritmo es cero, tenemos que $(g^3)^{4\cdot 0}=y_2^4$, así pues, tomando su cuarta raíz,

$$y_2=(g^3)^0(g^9)^{k_2}=(g^3)^0(g^3)^{3k_2},\ 0\le k_2\le 3,\ k_1=0+3k_2.$$

ETAPA 3 en un subgrupo de orden 4.

Se utiliza el algoritmo para encontrar el logaritmo de y_2 $(=y_2\cdot(g^3)^{-0})$ para la base g^9, que es 1. Así pues, $k_2=1,\to k_1=3,\to x=11$.

Método ρ de Pollard para logaritmos discretos

Un inconveniente del algoritmo de Shanks consiste en la necesidad de ordenar y almacenar una lista de tamaño $\sqrt{p}$. Para solventarlo, Pollard [97] introduce un algoritmo probabilístico que elimina la necesidad de tal almacenamiento.

Para encontrar $\log_g y$, en primer lugar se divide $\mathbb{F}_q^*$ en tres conjuntos S_1, S_2 y S_3, aproximadamente del mismo tamaño, mediante una función

$$H:\{1,2,3,\ldots,q-1\}\longmapsto\{S_1,S_2,S_3\}.$$

Se define una secuencia r_i de $\mathbb{F}_q^*$ para $i\ge 0$,

$$r_{i+1}=\begin{cases} yr_i & \text{si} \quad r_i\in S_1,\\ r_i^2 & \text{si} \quad r_i\in S_2,\\ gr_i & \text{si} \quad r_i\in S_3\ . \end{cases}$$

Así pues, cada elemento en la secuencia tiene la forma $r_i=y^{a_i}g^{b_i}$, donde $a_0=b_0$, para $i\ge 0$,

$$a_{i+1} = \begin{cases} a_i + 1 \ (mod\ q-1) & \text{si} \quad r_i \in S_1, \\ 2a_i \ (mod\ q-1) & \text{si} \quad r_i \in S_2, \\ a_i & \text{si} \quad r_i \in S_3, \end{cases}$$

y

$$b_{i+1} = \begin{cases} b_i & \text{si} \quad r_i \in S_1, \\ 2a_i \ (mod\ q-1) & \text{si} \quad r_i \in S_2, \\ b_i + 1 \ (mod\ q-1) & \text{si} \quad r_i \in S_3\ . \end{cases}$$

La secuencia r_i se comporta como una secuencia aleatoria en $\mathbb{F}_q^*$, luego se busca i de tamaño aproximado $\sqrt{q}$, tal que $r_i = r_{2i}$ (véase [97] para un análisis más detallado sobre el valor esperado de i).

Para encontrar $\log_g y$ se usa una secuencia $(r_i, a_i, b_i, r_{2i}, a_{2i}, b_{2i})$, donde cada elemento se calcula a partir del anterior hasta encontrar un caso en el que $r_i = r_{2i}$ (i.e. $y^{a_i} g^{b_i} = y^{a_{2i}} g^{b_{2i}}$).

Sea $m = a_i - a_{2i}\ (mod\ q-1)$ y $n = b_{2i} - b_i\ (mod\ q-1)$, entonces

$$y^m = g^n \text{ en } \mathbb{F}_q. \tag{2.1}$$

Se utiliza el algoritmo de Euclides extendido para encontrar un λ, μ tales que

$$d = mcd(m, q-1) = \lambda m + \mu(q-1).$$

Así pues, $\lambda m = d\ (mod\ q-1)$ y, si se usa la ecuación (2.1), se tiene $y^d = g^{\lambda n}$, donde λn debe ser de la forma dk (puesto que el miembro de la izquierda es una potencia d-ésima).

Tomando raíz d-ésima se tiene que

$$y = g^k (g^{\frac{q-1}{d}})^j, \quad 0 \le j \le d-1.$$

Se intenta para cada valor de j hasta encontrar una igualdad (para un m aleatorio).

Ejemplo.

Para el cálculo de $\log_5 17$ en $\mathbb{F}_{37}$, se definen los conjuntos

$$S_1 = \{1, 2, \ldots, 12\},$$

$$S_2 = \{13, 14, \ldots, 24\}$$

y

$$S_3 = \{25, 26, \ldots, 36\}.$$

Se calcula, pero no almacena:

$$(r_1 = 17,\ a_1 = 1, b_1 = 0, r_2 = 30, a_2 = 2, b_2 = 0),$$

$$(r_2 = 30,\ a_2 = 2, b_2 = 0, r_4 = 34, a_4 = 3, b_4 = 1),$$

$$(r_3 = 2,\ a_3 = 2, b_3 = 1, r_6 = 3, a_6 = 6, b_6 = 4),$$

$$(r_4 = 34,\ a_4 = 3, b_4 = 1, r_8 = 11, a_8 = 14, b_8 = 8),$$

$$(r_5 = 22,\ a_5 = 3, b_5 = 2, r_{10} = 34, a_{10} = 16, b_{10} = 8),$$

$$(r_6 = 3,\ a_6 = 6, b_6 = 4, r_{12} = 3, a_{12} = 32, b_{12} = 18).$$

Entonces se tiene

$$3 = 5^4 17^6 = 5^{18} 17^{32} \text{ y } 17^{26} = 5^{-14} = 5^{22},$$

$$d = mcd(26, 36) = 2 = 7 \cdot 26 - 5 \cdot 36,$$

y elevando a la séptima potencia

$$17^2 = 5^{22 \cdot 7} = 5^{10}.$$

Tomando raíces cuadradas $17 = 5^5 (5^{18})^j$, $j = 0$ ó $j = 1$. Si se toma $j = 0$ se obtiene la igualdad $17 = 17$, luego $\log_5 17 = 5$.

El método ρ de Pollard se puede usar en varias etapas, en subgrupos de $\mathbb{F}_q^*$, para obtener una complejidad de orden $\sqrt{p}$. Es posible, especialmente en un subgrupos del tamaño pequeño, que el algoritmo nos de una ecuación inservible ($y^a g^b = y^{a+q-1} g^b$), si esto sucede, se pueden cambiar simplemente las condiciones iniciales a_0 y b_0 para intentar el otra vez el algoritmo con un valor de r diferente.

Método de Pohlig-Hellman

Aunque este método es conocido como el algoritmo de Pohlig-Hellman [96], estos autores acreditan a Roland Silver como co-descubridor independiente.

Dado un grupo G de orden q, con generador g, conocidos los factores primos del orden del grupo

$$q - 1 = \prod_{i=1}^{k} p_i^{n_i},$$

el problema del cálculo de $x = \log_g y$ se reduce a k subproblemas, donde cada uno de ellos se resuelve encontrando n_i logaritmos discretos en un subgrupo de tamaño p_i, con $1 \leq i \leq k$.

Para cada i, se halla $x_i = x \ (mod \ p_i^{n_i})$ y se usa el teorema chino del resto para calcular x.

Para cada i, se tiene

$$x_i = \sum_{j=0}^{n_i-1} x_{i,j} p_i^j, \quad 0 \leq x_{i,j} \leq p_i - 1,$$

ya que $\forall i,$ existe algún $c_i \in \mathbb{Z}$ tal que $x = x_i + c_i p_i^{n_i}$.

Puesto que $g^{q-1} = 1$, se tienen las siguientes igualdades

$$(y)^{\frac{q-1}{p_i}} = (g^x)^{\frac{q-1}{p_i}} = g^{(\sum_{j=0}^{n_i-1} x_{i,j} p_i^j + c_i p_i^{n_i}) \frac{q-1}{p_i}} = (g^{\frac{q-1}{p_i}})^{x_{i,0}}, \tag{2.2}$$

con $0 \leq x_{i,0} \leq p_i - 1$;

$$(y \cdot g^{-x_{i,0}})^{\frac{q-1}{p_i^2}} = g^{(\sum_{j=1}^{n_i-1} x_{i,j} p_i^j + c_i p_i^{n_i}) \frac{q-1}{p_i^2}} = (g^{\frac{q-1}{p_i}})^{x_{i,1}}, \tag{2.3}$$

con $0 \leq x_{i,1} \leq p_i - 1$;

$$(y \cdot g^{-x_{i,0} - x_{i,1} p_i})^{\frac{q-1}{p_i^3}} = g^{(\sum_{j=2}^{n_i-1} x_{i,j} p_i^j + c_i p_i^{n_i}) \frac{q-1}{p_i^3}} = (g^{\frac{q-1}{p_i}})^{x_{i,2}}, \tag{2.4}$$

con $0 \leq x_{i,2} \leq p_i - 1$;

$$\cdots$$

$$(y \cdot g^{-(\sum_{j=0}^{n_i-2} x_{i,j} p_i^j)})^{\frac{q-1}{p_i^{n_i}}} = g^{(x_{i,n_i-1} p_i^{n_i-1} + c_i p_i^{n_i}) \frac{q-1}{p_i^{n_i}}} = (g^{\frac{q-1}{p_i}})^{x_{i,n_i-1}}, \tag{2.5}$$

con $0 \leq x_{i,n_i-1} \leq p_i - 1$.

Estas igualdades describen básicamente el algoritmo. Para encontrar x_i, se utilizan las ecuaciones (2.2) – (2.5), primero se calcula la parte izquierda de una ecuación (utilizando los resultados hallados previamente) y, a continuación se calcula $x_{i,j}$ como sus logaritmos discretos de la base $g^{\frac{q-1}{p_i}}$. Para valores pequeños de p_i, se utiliza uno de los algoritmos previamente mencionados en esta sección para calcular logaritmos discretos.

El orden de complejidad de este algoritmo es:

$$O(\sum_{i=1}^{r} e_i(\log n + \sqrt{p_i} \log p_i)).$$

Ejemplo.

Se va a calcular $\log_5 17$ en $\mathbb{F}_{37}$. La descomposición en factores del orden del grupo es

$$q - 1 = 2^2 \cdot 3^2.$$

Para $p_1 = 2$ se tiene

$$-1 = 17^{\frac{36}{2}} = (5^{\frac{36}{2}})^{x_{1,0}} = -1^{x_{1,0}} \rightarrow x_{1,0} = 1,$$

$$1 = (17 \cdot 5^{-1})^{\frac{36}{4}} = (5^{\frac{36}{2}})^{x_{1,1}} = -1^{x_{1,1}} \rightarrow x_{1,1} = 0,$$

luego

$$x_1 = x_{1,0} + x_{1,1} \cdot 2 = 1 + 0 \cdot 2 = 1 \Leftrightarrow x = 1 \ (mod\ 4).$$

Para $p_2 = 3$

$$26 = 17^{\frac{36}{2}} = (5^{\frac{36}{3}})^{x_{2,0}} = 10^{x_{2,0}} \rightarrow x_{2,0} = 2,$$

$$10 = (17 \cdot 5^{-2})^{\frac{36}{9}} = (5^{\frac{36}{3}})^{x_{2,1}} = 10^{x_{2,1}} \rightarrow x_{2,1} = 1,$$

luego

$$x_2 = x_{2,0} + x_{2,1} \cdot 3 = 2 + 1 \cdot 3 = 5 \Leftrightarrow x = 5 \ (mod\ 9).$$

Finalmente, se usa el teorema chino del resto para encontrar $x = 5$.

Método Index Calculus

El método más rápido para computar logaritmos discretos es conocido como el método Index Calculus. La idea básica del algoritmo apareció en 1922 en el trabajo de Kraitchik [69]. En 1979 Adleman [1] analiza el orden del algoritmo para el caso de que q sea un número primo y en 1983 Hellman y Reyneri [60] extienden el algoritmo para el caso $q = p^n$, para un p fijo y $n \rightarrow \infty$.

El orden de complejidad del algoritmo es

$$L[q, \alpha, c] = \exp((c + o(1))(\ln q)^{\alpha}(\ln \ln q)^{1-\alpha}), \quad 0 < \alpha < 1,$$

donde c es constante y $o(1) \rightarrow 0$, cuando $q \rightarrow \infty$, es decir, es un orden subexponencial.

El algoritmo implica operaciones en un anillo inmerso en un cuerpo finito, y su descripción es diferente para diferentes tipos de cuerpos. Se va a definir la versión básica del algoritmo, que tiene un orden de complejidad $O(L[q, \frac{1}{2}, c])$.

Sea $\mathbb{F}_p$, con p primo, cuyos elementos están representados por $\{0, 1, \ldots, p-1\}$ y sea S el conjunto de todos los primos menores o iguales que un entero b. Un elemento de

$\mathbb{F}_p^*$ se dice que es liso respecto a b si, en el anillo $\mathbb{Z}$, todos sus factores están contenidos en S.

El algoritmo tiene tres etapas.

(a) En la primera, se toma un entero aleatorio z en $[1, p-2]$, se calcula $g^z \ (mod\ p)$ y se ve si $g^z \ (mod\ p)$ es liso. En el caso de ser liso, sea

$$g^z \ (mod\ p) = \prod_{i=1}^{|S|} p_i^{\alpha_i}, \quad p_i \in S,$$

entonces se toma una ecuación en el logaritmo discreto de base g

$$z = \sum_{i=1}^{|S|} \alpha_i \log_g p_i \ (mod\ p-1), \tag{2.6}$$

donde z y todos los α_i son conocidos. Se continua este proceso hasta tener generadas más de $|S|$ ecuaciones.

(b) El la segunda etapa del algoritmo, se resuelve el sistema de ecuaciones (2.6) encontrando una única solución para $\log_g p_i$.

(c) En la etapa tres, se calcula el logaritmo discreto de un $y \in \mathbb{F}_p^*$. Para esto se toma z aleatoriamente, hasta encontrar uno tal que $yg^z \ (mod\ p)$ sea liso. Cuando se encuentre este z, se resuelve la ecuación

$$\log_g y = -z + \sum_{i=1}^{|S|} \alpha_i \log_g p_i \ (mod\ p-1).$$

Ejemplo.

ETAPA 1

Se desea calcular $\log_5 17$.

Se intenta $z = 7$,

$$5^7 \ (mod\ 37) = 18 = 2 \cdot 3^2 \Rightarrow \log_5 2 + 2 \cdot \log_5 3 \ (mod\ 36) = 7.$$

Se intenta $z = 6$

$$5^6 \ (mod\ 37) = 11 \text{ (no es liso)}.$$

Se intenta $z = 14$

$$5^{14} \ (mod\ 37) = 28 \text{ (no es liso)}.$$

Se intenta $z = 31$

$$5^{31} \ (mod\ 37) = 24 = 2^3 \cdot 3 \Rightarrow 3 \cdot \log_5 2 + 2 \cdot \log_5 3 \ (mod\ 36) = 31.$$

ETAPA 2

Se resta tres veces la primera ecuación de la segunda y se usa

$$5^{-1} \ (mod\ 36) = 29$$

para obtener

$$\log_5 2 = 11, \ \log_5 3 = 34.$$

ETAPA 3

Se intenta $z = 24$

$$17 \cdot 5^{24} \ (mod\ 37) = 35 \text{ (no es liso)}.$$

Se intenta $z = 15$

$$17 \cdot 5^{15} \ (mod\ 37) = 12 = 2^2 \cdot 3 \Rightarrow \log_5 17 = -15 + 2 \cdot 11 + 1 \cdot 34 \ (mod\ 36) = 5.$$

El algoritmo es básicamente el mismo para el caso $\mathbb{F}_{p^n}$, $n \ggg p$, cuyos elemento son el conjunto de todos los polinomios sobre $\mathbb{F}_p$ de grado menor o igual que n, con multiplicación módulo un polinomio fijo irreducible $f(x)$ de grado n.

Si S es el conjunto de los polinomios irreducibles en el anillo $\mathbb{F}_p[x]$ cuyo grado no excede de algún límite b, se llama liso a un elemento de $\mathbb{F}^*_{p^n}$ cuyos factores están en S. Esta simple extensión del algoritmo no se ejecuta en un tiempo subexponencial para valores de p y n arbitrarios.

Incrementando el tamaño de S aumenta la probabilidad de que un elemento aleatorio del cuerpo sea liso, es decir, la tarea de generar ecuaciones en la etapa uno puede lograrse mas rápidamente. Sin embargo, aumenta el número de desconocidos en la etapa dos, haciendo más difícil la solución del sistema de ecuaciones. Por lo tanto, hay que intentar limitar $|S|$, al tamaño que equilibre estas consideraciones.

En cuerpos finitos $\mathbb{F}_p$ con p primo y tomando como límite

$$b = L[p, \frac{1}{2}, \beta] = \exp((\beta + o(1))(\ln p)^{\frac{1}{2}}(\ln \ln p)^{\frac{1}{2}}),$$

el trabajo de Canfield, Erdos y Pomerance [26] muestra que se tienen alrededor de

$$\exp((\frac{\alpha}{2\beta} + o(1))(\ln p)^{\frac{1}{2}}(\ln \ln p)^{\frac{1}{2}})$$

enteros aleatorios entre 1 y p^α, entre uno liso.

En cuerpos $\mathbb{F}_{2^n}$, Odlyzko [91] muestra que se tiene que testear del orden de

$$\exp((1 + o(1))\frac{k}{b}\ln(\frac{k}{b}))$$

polinomios de grado k para encontrar uno que tenga todos su factores irreducibles de grado b o menor. En general, cuanto más grande es la magnitud del elemento del campo que se está probando, menos probable es que sea liso. Por tanto, una manera eficaz de acelerar el algoritmo es reducir la magnitud de los elementos del cuerpo que se prueba para ver si son lisos, pero esto debe ser hecho de manera que se pueda conseguir una ecuación entre los logaritmos discretos de los irreductibles en S.

2.2.3 Algoritmos más recientes y tendencias

Aún emergen ideas que podrían hacer el problema del cómputo del logaritmo discreto más fácil. Considerar, por ejemplo, que un grupo cíclico de orden $q-1$ es isomorfo con el grupo cíclico aditivo $\mathbb{Z}_{q-1}$. El cálculo del logaritmo discreto en $(\mathbb{Z}_{q-1}, +)$ se convierte en encontrar un x tal que $xq \ (mod\ q-1) = y$, donde g es un generador, lo que implica que $(g, q-1) = 1$; así pues, se puede computar fácilmente $g^{-1} \ (mod\ q-1)$ usando el Algoritmo de Euclides extendido.

Coppersmith, Odlyzko y Schroeppel [32] proponen tres algoritmos (con tres etapas cada uno) para el cálculo de logaritmos discretos en cuerpos de orden primo $(\mathbb{F}_p)$, cada uno de los cuales tiene una complejidad $L[p, \frac{1}{2}, c = 1]$ para la primera y segunda etapa y una complejidad $L[p, \frac{1}{2}, c = \frac{1}{2}]$ para la tercera etapa. Uno de estos algoritmos, llamado por Coppersmith "Método de los enteros Gaussianos", ha demostrado ser el más práctico y el más importante de los tres, de manera que está estimulando el desarrollo de otros algoritmos para el cálculo de logaritmos discretos. LaMacchia y Odlyzko [71] han implementado satisfactoriamente este método de los enteros Gaussianos en un cuerpo primo de tamaño $\approx 10^{67}$.

Sorenson [116] propone un algoritmo paralelo para el cómputo de logaritmos discretos en cuerpos de orden primo $\mathbb{F}_p$ y $\mathbb{F}_{p^n}$, usando circuitos booleanos probabilísticos de profundidad polinomial en $\ln p$ y un tamaño subexponencial.

Sin embargo, la principal amenaza a largo plazo para los logaritmos discretos viene de los computadores cuánticos, Short [113] propone un algoritmo probabilístico para calcular, en este tipo de computadores, logaritmos discretos en cuerpos finitos $\mathbb{F}_p$, con un orden de complejidad polinomial en $\ln p$. Mientras exista debate sobre si las computadoras cuánticas son factibles o no, los criptosistemas cuya seguridad se basa en el DLP están a salvo.

Capítulo 3

Criptosistemas asimétricos basados en matrices triangulares superiores por bloques

3.1 Propiedades generales

Todos los criptosistemas que se proponen basan su seguridad en el problema matemático del Logaritmo Discreto (DLP), aplicado a un determinado grupo de matrices triangulares superiores por bloques con elementos en $\mathbb{Z}_p$.

Dado p un número primo y $r, s \in \mathbb{N}$, se denota por $\mathrm{Mat}_r(\mathbb{Z}_p)$, $\mathrm{Mat}_s(\mathbb{Z}_p)$, $\mathrm{Mat}_{r\times s}(\mathbb{Z}_p)$ a las matrices de tamaño $r \times r$, $s \times s$ y $r \times s$, respectivamente, con elementos en $\mathbb{Z}_p$ y por $\mathrm{GL}_r(\mathbb{Z}_p)$, $\mathrm{GL}_s(\mathbb{Z}_p)$ a las matrices invertibles de tamaño $r \times r$, $s \times s$, también con elementos en $\mathbb{Z}_p$.

Se define el conjunto de matrices

$$\Omega = \left\{ \begin{bmatrix} A & X \\ \mathbf{0} & B \end{bmatrix}, A \in \mathrm{Mat}_r(\mathbb{Z}_p), B \in \mathrm{Mat}_s(\mathbb{Z}_p), X \in \mathrm{Mat}_{r\times s}(\mathbb{Z}_p) \right\},$$

y se considera el subconjunto

$$\Theta = \left\{ \begin{bmatrix} A & X \\ \mathbf{0} & B \end{bmatrix}, A \in \mathrm{GL}_r(\mathbb{Z}_p), B \in \mathrm{GL}_s(\mathbb{Z}_p), X \in \mathrm{Mat}_{r\times s}(\mathbb{Z}_p) \right\}.$$

A continuación se enuncian y demuestran una serie de teoremas que caracterizan estas matrices triangulares superiores por bloques.

Teorema 3.1: *El conjunto* Θ *tiene estructura de grupo no abeliano para el producto de matrices.*

DEMOSTRACIÓN: Por definición de Θ, es evidente que la operación producto es cerrada.

El neutro es $I = \begin{bmatrix} \mathrm{I}_r & \mathbf{0} \\ \mathbf{0} & \mathrm{I}_s \end{bmatrix}$, donde I_r e I_s son las matrices identidad $r \times r$ y $s \times s$ en $\mathrm{GL}_r(\mathbb{Z}_p)$ y $\mathrm{GL}_s(\mathbb{Z}_p)$, respectivamente.

El simétrico de un elemento cualquiera $M = \begin{bmatrix} A & X \\ \mathbf{0} & B \end{bmatrix}$ es

$$M^{-1} = \begin{bmatrix} A^{-1} & -A^{-1}XB^{-1} \\ \mathbf{0} & B^{-1} \end{bmatrix}.$$

La asociatividad es obvia por tratarse de matrices cuadradas. □

Teorema 3.2: *Dado* $M = \begin{bmatrix} A & X \\ \boldsymbol{0} & B \end{bmatrix}$ *un elemento cualquiera del grupo* Θ*, se considera el subgrupo generado por las sucesivas potencias de* M*.*

Tomando h *como un entero no negativo entonces*

$$M^h = \begin{bmatrix} A^h & X^{(h)} \\ \boldsymbol{0} & B^h \end{bmatrix}, \tag{3.1}$$

donde $X^{(h)} = \boldsymbol{0}$*, si* $h = 0$ *y si* $h \geq 1$

$$X^{(h)} = \sum_{i=1}^{h} A^{h-i} X B^{i-1}. \tag{3.2}$$

Además, si $0 \leq t \leq h$ *entonces*

$$X^{(h)} = A^t X^{(h-t)} + X^{(t)} B^{h-t}, \tag{3.3}$$

$$X^{(h)} = A^{(h-t)} X^{(h)} + X^{(h-t)} B^t\,. \tag{3.4}$$

DEMOSTRACIÓN: Se demuestra la ecuación (3.1) usando inducción sobre h. Para $h = 0$ y $h = 1$ el resultado es obvio. Se supone que es cierto para $h-1$ y se demuestra que es cierto para h.

$$\begin{aligned} M^h &= MM^{h-1} \\ &= \begin{bmatrix} A & X \\ \mathbf{0} & B \end{bmatrix} \begin{bmatrix} A^{h-1} & X^{(h-1)} \\ \mathbf{0} & B^{h-1} \end{bmatrix} \\ &= \begin{bmatrix} A^h & AX^{(h-1)} + XB^{h-1} \\ \mathbf{0} & B^h \end{bmatrix}. \end{aligned}$$

Por hipótesis de inducción, aplicando (3.2), se tiene

$$\begin{aligned} X^{(h)} &= AX^{(h-1)} + XB^{h-1} \\ &= A\sum_{i=1}^{h-1} A^{h-1-i}XB^{i-1} + XB^{h-1} \\ &= \sum_{i=1}^{h-1} A^{h-i}XB^{i-1} + XB^{h-1} \\ &= \sum_{i=1}^{h} A^{h-i}XB^{i-1}, \end{aligned}$$

obteniéndose la misma expresión que en (3.2).

También, si $0 \leq t \leq h$, se tiene

$$\begin{aligned} M^h &= M^t M^{h-t} \\ &= \begin{bmatrix} A^t & X^{(t)} \\ \mathbf{0} & B^t \end{bmatrix} \begin{bmatrix} A^{h-t} & X^{(h-t)} \\ \mathbf{0} & B^{h-t} \end{bmatrix} \\ &= \begin{bmatrix} A^h & A^t X^{(h-t)} + X^{(t)} B^{h-t} \\ \mathbf{0} & B^h \end{bmatrix}. \end{aligned}$$

Comparando estos resultados con (3.1) se obtiene (3.3). De la misma forma se demuestra (3.4) □

Como consecuencia, en el caso $t = 1$, se tiene

$$X^{(h)} = AX^{(h-1)} + XB^{h-1},$$

$$X^{(h)} = A^{h-1}X + X^{(h-1)}B.$$

Si a y b son enteros tales que $a + b \geq 0$, se tiene

$$X^{(a+b)} = A^a X^{(b)} + X^{(a)} B^b. \tag{3.5}$$

3.2 Orden de los elementos

En los esquemas que se presentan, el espacio de claves está íntimamente ligado al orden del grupo generado por la matriz

$$M = \begin{bmatrix} A & X \\ \mathbf{0} & B \end{bmatrix} \in \Theta,$$

por ello se presenta a continuación la forma de garantizar que este orden sea suficientemente grande.

Sea $f(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} + x^n$ un polinomio mónico en $\mathbb{Z}_p[x]$, cuya matriz $n \times n$ asociada es

$$\overline{A} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-2} & -a_{n-1} \end{bmatrix}$$

Si f es irreducible en $\mathbb{Z}_p[x]$, entonces el orden de $\overline{A}$ es igual al orden de cualquier raíz de f en $\mathbb{F}_{p^n}$ y el orden de $\overline{A}$ divide a $p^n - 1$ (véase [92]). Además, asumiendo que f es un polinomio primitivo en $\mathbb{Z}_p[x]$, el orden de $\overline{A}$ es exactamente $p^n - 1$. En consecuencia, si se trabaja en $\mathbb{Z}_p[x]$, es posible construir fácilmente matrices cuyo orden sea máximo.

Odoni, Varadharajan y Sanders [92], proponen un esquema extendido basado en la

construcción de la matriz por bloques

$$\overline{A} = \begin{bmatrix} \overline{A_1} & 0 & \dots & 0 \\ 0 & \overline{A_2} & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \overline{A_k} \end{bmatrix}$$

donde $\overline{A_i}$ es la matriz asociada a f_i siendo f_i, para $i = 1, 2, \dots, k$, polinomios primitivos distintos en $\mathbb{Z}_p[x]$ de grado n_i, para $i = 1, 2, \dots, k$, respectivamente. El orden de $\overline{A_i}$ es $p^{n_i} - 1$, para $i = 1, 2, \dots, k$, por tanto, el orden de $\overline{A}$ viene dado por

$$mcm(p^{n_1} - 1, p^{n_2} - 1, \dots, p^{n_k} - 1).$$

Con el objeto de usar una matriz de este tipo en un sistema de clave pública, los citados autores conjugan esta matriz $\overline{A}$ con una matriz invertible P de tamaño $n \times n$, con $n = n_1 + n_2 + \dots + n_k$, obteniendo una nueva matriz $A = P\overline{A}P^{-1}$ que tiene el mismo orden que $\overline{A}$.

De esta forma, construyendo

$$M = \begin{bmatrix} A & X \\ \mathbf{0} & B \end{bmatrix} \in \Theta,$$

utilizando polinomios primitivos se puede garantizar un orden determinado. Sean

$$f(x) = a_0 + a_1 x + \cdots + a_{r-1} x^{r-1} + x^r$$

y

$$g(x) = b_0 + b_1 x + \cdots + b_{s-1} x^{s-1} + x^s$$

dos polinomios primitivos en $\mathbb{Z}_p[x]$ y $\overline{A}$, $\overline{B}$ las matrices asociadas correspondientes. Sean P y Q dos matrices invertibles, tales que, $A = P\overline{A}P^{-1}$ y $B = Q\overline{B}Q^{-1}$. Con esta construcción, el orden de M es $mcm(p^r - 1, p^s - 1)$.

Es sabido por teoría de cuerpos ciclotómicos y raíces de la unidad, que el polinomio $x^n - 1$ es divisible por $x^d - 1$ si $d|n$; por tanto, si se elige r y s de manera que sean primos entre sí, se minimiza el número de divisores comunes y el $mcm(p^r - 1, p^s - 1)$ será máximo.

La tabla 3.1 muestra una comparativa de la variación del orden de M dependiendo del tamaño de p y del tamaño de los bloques superior izquierda (r) e inferior derecha

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^{1200}
3	5	2^{700}	2^{1120}	2^{1400}
4	5	2^{800}	2^{1280}	2^{1600}
5	6	2^{1000}	2^{1600}	2^{2000}

Tabla 3.1: *Orden de M, para p grandes*

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^{1264}
216	217	2^{1004}	2^{1599}	2^{2099}

Tabla 3.2: *Orden de M, para p pequeños*

(s) de la matriz M. Así, por ejemplo, para tamaños de p del orden de 160 bits y para valores de $r = 3$ y $s = 5$, se obtiene un orden de M, expresado en binario, de aproximadamente 1120 bits.

Más aún, también es posible alcanzar grandes tamaños del orden de M con valores muy pequeños de p, y valores de r y s no muy grandes. Como puede apreciarse en la tabla 3.2; por ejemplo, tomando unos tamaños de los bloques superior izquierda e inferior derecha de $r = 130, s = 131$ y un valor de $p = 13$ se obtiene un orden de alrededor de 963 bits. La elección de estos parámetros p, r y s debe hacerse siempre sin comprometer la seguridad del criptosistema.

Para elegir correctamente el tamaño de p, se deben tener en cuenta los ataques basados en algoritmos de raíz cuadrada [96], vistos en el capítulo 2, que establecen básicamente la posibilidad de reducir el problema del logaritmo discreto definido sobre un grupo finito a varios problemas del logaritmo discreto de tamaño menor, determinados

por los factores que componen el orden del grupo original. Por ello, los órdenes de los grupos usados en estos criptosistemas contienen un factor primo grande.

En los criptosistemas que se plantean, el orden de M no es primo, ya que viene determinado por un mínimo común múltiplo. El factor primo más grande que se puede obtener es $p^r - 1$ ó $p^s - 1$ puesto que es posible hacer que estos elementos sean primos (véase [101]).

Como la complejidad de la resolución del problema viene determinada por el factor primo más grande del orden del grupo, se debe optimizar la elección de p, r y s de forma que uno de sus factores sea lo más grande posible y el otro (que ya no debe ser necesariamente primo) lo menor posible. De esta forma se minimiza el número de bits usados, manteniendo el máximo nivel de seguridad (véase [38]).

3.3 Criptosistemas propuestos

En [4] se realiza un estudio sobre las aplicaciones de las matrices triangulares superiores por bloques a los criptosistemas simétricos de cifrado en flujo. En este trabajo se estudian las aplicaciones de este tipo de matrices a la criptografía de clave pública y se presentan nuevas primitivas criptográficas implementadas en grupos, cuya seguridad se basa en el problema del logaritmo discreto. Ejemplos numéricos de estos esquemas criptográficos pueden consultarse en el anexo A.

El primer esquema propuesto, es una adaptación del intercambio de clave de Diffie-Hellman a las matrices triangulares superiores por bloques, añadiendo cifrado de la información y firma digital.

3.3.1 Diffie-Hellman para matrices triangulares superiores por bloques

Sea la matriz

$$M = \begin{bmatrix} A & X \\ \mathbf{0} & B \end{bmatrix} \in \Theta, \tag{3.6}$$

y el subgrupo de orden m generado por dicha matriz

$$H = \{I, M, M^2, M^3, \ldots, M^{m-1}\}. \tag{3.7}$$

Eligiendo los bloques superior derecha de las matrices de H, se obtiene el conjunto

$$G = \{X^{(0)}, X^{(1)}, X^{(2)}, X^{(3)}, \ldots, X^{(m-1)}\}, \tag{3.8}$$

donde $X^{(0)} = \mathbf{0}$ y $X^{(1)} = X$.

3.3.1.1 Intercambio de clave

Sean U y V dos interlocutores que desean intercambiar una clave. Para ello ejecutan el siguiente protocolo:

(a) Acuerdan $p \in \mathbb{N}$ y $M \in \Theta$, siendo m el orden de M.

(b) El usuario U genera una clave privada $k_1 \in \mathbb{N}$, con $1 \leq k_1 \leq m-1$, de forma aleatoria y calcula

$$N_1 = M^{k_1} = \begin{bmatrix} A^{k_1} & X^{(k_1)} \\ \mathbf{0} & B^{k_1} \end{bmatrix} = \begin{bmatrix} A_1 & Y \\ \mathbf{0} & B_1 \end{bmatrix}.$$

(c) El usuario V genera una clave privada $k_2 \in \mathbb{N}$, con $1 \leq k_2 \leq m-1$, de forma aleatoria y calcula

$$N_2 = M^{k_2} = \begin{bmatrix} A^{k_2} & X^{(k_2)} \\ \mathbf{0} & B^{k_2} \end{bmatrix} = \begin{bmatrix} A_2 & Z \\ \mathbf{0} & B_2 \end{bmatrix}.$$

(d) Las claves públicas de los usuarios U y V son respectivamente las matrices N_1 y N_2.

(e) El usuario U calcula

$$N_2^{k_1} = \begin{bmatrix} A_2^{k_1} & Z^{(k_1)} \\ \mathbf{0} & B_2^{k_1} \end{bmatrix}.$$

(f) El usuario V calcula

$$N_1^{k_2} = \begin{bmatrix} A_1^{k_2} & Y^{(k_2)} \\ \mathbf{0} & B_1^{k_2} \end{bmatrix}.$$

Ahora tanto U como V poseen un elemento común y secreto del conjunto H, $M^{k_1 k_2} = N_1^{k_2} = N_2^{k_1} = M^{k_2 k_1}$. Podemos considerar como clave compartida parte de esta matriz $P = Z^{(k_1)} = Y^{(k_2)}$.

3.3.1.2 Esquema de cifrado

Se consideran los mismos elementos públicos y privados del protocolo de intercambio de clave visto anteriormente en 3.3.1.1, que se supone previamente realizado.

Si el interlocutor U desea enviar un mensaje $\mu \in G$ de forma confidencial al usuario V, ejecuta el siguiente protocolo:

(a) Genera una clave privada $k_3 \in \mathbb{N}$, con $1 \leq k_3 \leq m-1$, de forma aleatoria y calcula $T_1 = \begin{bmatrix} A^{k_3} & \mu \\ \mathbf{0} & B^{k_3} \end{bmatrix}$ y $T_2 = N_2^{k_1} = \begin{bmatrix} A_2^{k_1} & P \\ \mathbf{0} & B_2^{k_1} \end{bmatrix}$.

(b) Calcula la matriz $C = T_1 T_2$ y envía al usuario V el mensaje cifrado C.

Para recuperar el mensaje, V procede así:

(a) Genera la matriz $T_2 = N_1^{k_2} = \begin{bmatrix} A_1^{k_2} & P \\ \mathbf{0} & B_1^{k_2} \end{bmatrix}$ y calcula su inversa T_2^{-1}, matriz invertible por serlo A_1 y B_1.

(b) Obtiene la matriz T_1 efectuando el producto CT_2^{-1}.

(c) Recupera el mensaje μ seleccionando el correspondiente bloque de T_1.

Con esto, las funciones de cifrado y descifrado del usuario V, son respectivamente:

$$E_{N_2}(\mu) = T_1 T_2 \quad \text{y} \quad D_{k_2}(C) = CT_2^{-1}.$$

3.3.1.3 Firma digital

El esquema de firma digital que se propone, requiere el mensaje original como entrada para la verificación de la firma. Para esto se supone que los usuarios U y V han intercambiado la clave P (véase 3.3.1.1) y que U ha hecho llegar a V el mensaje $\mu \in G$, según el protocolo visto anteriormente en 3.3.1.2.

Si el emisor U desea firmar digitalmente el mensaje μ procede así:

(a) Genera un número aleatorio $r \in \mathbb{N}$.

(b) Partiendo de

$$T_2 = N_2^{k_1} = \begin{bmatrix} A_2^{k_1} & P \\ \mathbf{0} & B_2^{k_1} \end{bmatrix} = \begin{bmatrix} A_3 & P \\ \mathbf{0} & B_3 \end{bmatrix}$$

calcula

$$T_2^r = \begin{bmatrix} A_3^r & P^{(r)} \\ \mathbf{0} & B_3^r \end{bmatrix}.$$

(c) Calcula Q

$$Q = T_1 - T_2^r.$$

(d) La firma digital es la dupla (r, Q).

Si el receptor desea verificar la firma digital del usuario U, procede así:

(a) Partiendo de

$$T_2 = N_1^{k_2} = \begin{bmatrix} A_1^{k_2} & P \\ \mathbf{0} & B_1^{k_2} \end{bmatrix} = \begin{bmatrix} A_4 & P \\ \mathbf{0} & B_4 \end{bmatrix}$$

calcula

$$T_2^r = \begin{bmatrix} A_4^r & P^{(r)} \\ \mathbf{0} & B_4^r \end{bmatrix}.$$

(b) Calcula $R = Q + T_2^r$.

(c) Extrae el bloque superior derecho de la matriz R, al que se denota por W.

(d) Compara μ y W, resultando que la firma es correcta (auténtica) si

$$\mu = W$$

e incorrecta si

$$\mu \neq W.$$

3.3.1.4 Análisis de seguridad

La seguridad del protocolo de intercambio de clave de Diffie-Hellman depende de la dificultad del cálculo del logaritmo discreto y el criptosistema presentado es una adaptación de este protocolo a las matrices triangulares superiores por bloques, por lo que es de suponer que tiene una seguridad similar, tomando el orden de M lo suficientemente grande. Sin embargo, si se aplica el algoritmo de reducción de Menezes y Wu [82], que se presenta a continuación, la seguridad queda comprometida.

Dadas dos matrices $A, B \in GL_n(\mathbb{Z}_p)$ con $B = A^l$, $l \in \mathbb{N}$; el problema del logaritmo discreto en $GL_n(\mathbb{Z}_p)$ consiste en averiguar el valor de l. Este problema puede reducirse (véase [82]) al cálculo de logaritmos discretos en cuerpos finitos de pequeño tamaño $F_{q^{m_i}}$, donde m_i es el grado de los polinomios irreducibles en los que se puede factorizar el polinomio característico de la matriz A.

Algoritmo de reducción de Menezes y Wu

Entrada: Matrices $A, B \in GL_n(\mathbb{Z}_p)$ con $B = A^l$.

Salida: El entero l.

(a) Se usa el algoritmo de Hessenberg para encontrar el polinomio característico $p_A(x)$ de la matriz A.

(b) Se obtiene la factorización de $p_A(x)$ sobre F_p:

$$p_A(x) = f_1^{e_1} f_2^{e_2} \cdots f_s^{e_s}$$

donde cada f_i con $i = 1 \ldots s$, es un polinomio irreducible de grado m_i. Sean α_{ij}, las raíces de f_i en $F_{p^{m_i}}$, con $1 \leq j \leq m_i$.

(c) Desde $i = 1$ hasta s:

(1) Se computa $(A - \alpha_{i1} I)^l$ y $r_l = r(A - \alpha_{i1} I)^l$, para $l = 1, 2, \ldots, c, c+1$, donde c es el entero positivo mas pequeño tal que $r_c = r_{c+1}$.

(2) Se obtiene un valor propio μ_i correspondiente a α_{i1} resolviendo la ecuación $(A - \alpha_{i1} I)y = 0$.

(3) Se construye una matriz $Q_i \in GL_n(\mathbb{Z}_{p^{m_i}})$ cuya primera columna sea μ_i.

(4) Se calcula $D_i = Q_i^{-1} B Q_i$.

(5) La entrada $(1,1)$ de la matriz D_i, es α_{i1}^l, así pues, se puede encontrar l módulo $ord(\alpha_{i1})$, resolviendo un logaritmo discreto en $F_{p^{m_i}}$.

(d) Sea t el máximo de los valores encontrados en la etapa c.(1). Si $t > 1$ entonces se realiza

(1) Sea $\lambda \in F_{p^m}$ un valor propio que tiene el correspondiente bloque de Jordan de tamaño t.

(2) Se obtiene una base B_1 para $N[(A - \lambda I)^{t-1}]$.

(3) Se obtiene una base B_2 para $N[(A - \lambda I)^t])$.

(4) Se obtiene un vector $u \in B_2$, el cual no está en el subespacio generado por B_1 (u es un valor propio generalizado de rango t).

(5) Sea $u_t = u$ y $u_j = (A - \lambda I)u_{j+1}$ para $j = t-1, t-2, \ldots, 2, 1$.

(6) Se construye una matriz $Q \in GL_n(\mathbb{Z}_{p^m})$, cuyas primeras t columnas sean $u_1, u_2, \ldots u_t$.

(7) Se calcula $D = Q^{-1} B Q$.

(8) La entrada $(1,1)$ de la matriz D, es λ^l y (1,2) es $l\lambda^{l-1}$. Si $p\{t\} \geq p$, entonces primero se computa λ^{l-1} como λ^l/λ, y entonces se divide $l\lambda^{l-1}$ entre λ^{l-1} para obtener $l \ mod \ p$.

(9) Si $p\{t\} \geq p^2$, entonces sea J el bloque de Jordan de tamaño $t \times t$ en la esquina superior izquierda de $Q^{-1} A Q$. Sea $s = ord(\lambda), l' = l \ mod \ s$ (el cual se obtuvo

en la etapa 3), y se calculan las matrices $J^{l'}, J^{l'+s}, J^{l'+2s}, \ldots$ hasta que $J^{l'+js}$ sea igual a la esquina superior izquierda de D. Entonces $l \ mod \ p\{t\} = j$.

(e) Se obtiene $l \ mod \ o(A)$ usando la generalización del teorema chino del resto.

En el intercambio de clave (véase 3.3.1.1), las claves públicas compartidas por los interlocutores son:

$$N_1 = M^{k_1} = \begin{bmatrix} A^{k_1} & X^{(k_1)} \\ \mathbf{0} & B^{k_1} \end{bmatrix}, \quad N_2 = M^{k_2} = \begin{bmatrix} A^{k_2} & X^{(k_2)} \\ \mathbf{0} & B^{k_2} \end{bmatrix},$$

con lo que se puede aplicar el algoritmo mencionado tomando el bloque superior izquierda o inferior derecha de dichas matrices N_1 o N_2. Con esto, se reduce el problema del logaritmo discreto a computar logaritmos discretos en pequeñas extensiones de $F_{q^{m_i}}$, donde m_i es el grado de los polinomios irreducibles en los que se puede factorizar el polinomio característico de la matriz A o B. Aplicando este algoritmo, se tiene como entrada la matrices A^{k_1} (B^{k_2}) y como salida, el número k_1 (k_2).

3.3.2 Esquema aditivo

Paralelamente al esquema presentado anteriormente 3.3.1 se planteó este criptosistema (véase [5, 6]), basado en una propiedad aditiva de las matrices triangulares superiores por bloques.

Dada $M = \begin{bmatrix} A & X \\ \mathbf{0} & B \end{bmatrix} \in \Theta$ de orden $m \in \mathbb{N}$ y los conjuntos

$$H = \{I, M^1, M^2, \ldots, M^{m-1}\}, \ G = \{X^{(0)}, X^{(1)}, X^{(2)}, \ldots, X^{(m-1)}\},$$

se define, para un par de elementos $X^{(a)}, X^{(b)} \in G$, el operador $\otimes$ como

$$X^{(a)} \otimes X^{(b)} = X^{(j)}, \quad \text{con } j = a + b \ (mod \ m).$$

Teorema 3.3: *$(G, \otimes)$ es un grupo abeliano.*

DEMOSTRACIÓN: Dada la definición de G y del operador $\otimes$ es evidente que se trata de una operación interna.

Por otro lado, si se toma a como un entero no negativo, el neutro es $X^{(0)}$ ya que

$$X^{(a)} \otimes X^{(0)} = X^{(a)} \text{ y } X^{(0)} \otimes X^{(a)} = X^{(a)}.$$

Nótese que $X^{(0)} = \mathbf{0}$ y $B^0 = A^0 = I$.

El simétrico de un elemento $X^{(a)}$ cualquiera es $X^{(m-a)}$, ya que por definición

$$X^{(a)} \otimes X^{(m-a)} = X^{(0)},$$

$$X^{(m-a)} \otimes X^{(a)} = X^{(0)},$$

y el simétrico de $X^{(0)}$ es el mismo.

Verifica la propiedad asociativa, ya que

$$\begin{aligned}(X^{(a)} \otimes X^{(b)}) \otimes X^{(c)} &= X^{(j_1)} \otimes X^{(c)} \\ &= X^{(j_2)} \\ &= X^{(a)} \otimes X^{(j_3)} \\ &= X^{(a)} \otimes (X^{(b)} \otimes X^{(c)}),\end{aligned}$$

con

$$j_1 = a + b \ (mod\ m),$$

$$j_2 = j_1 + c \ (mod\ m) = a + b + c \ (mod\ m),$$

$$j_3 = b + c \ (mod\ m).$$

La conmutatividad es obvia. □

Teorema 3.4: *Dada la matriz $M \in \Theta$, con orden m,*

$$X^{(h+m)} = X^{(h)}, \text{ con } 0 \leq h \leq m-1.$$

Demostración: Dado que M es de orden m, se tiene que $M^m = I_n$, con lo que

$$X^{(m)} = X^{(0)} = \mathbf{0},$$

$$A^m = I_r,$$

$$B^m = I_s$$

y

$$M^{m+1} = \begin{bmatrix} A^{m+1} & X^{(m+1)} \\ \mathbf{0} & B^{m+1} \end{bmatrix} = M.$$

Se demuestra usando inducción sobre h. En primer lugar, para $h = 0$ se tiene

$$X^{(m)} = X^{(0)}.$$

Se supone cierto para $h-1$,

$$X^{(h-1+m)} = X^{(h-1)}$$

y se demuestra que es cierto para h.

$$\begin{aligned} X^{(h+m)} &= X^{(1+h-1+m)} \\ &= X^{[1+(h-1+m)]} \\ &= AX^{(h-1+m)} + X^{(1)}B^{h-1+m} \\ &= AX^{(h-1)} + X^{(1)}B^{h-1}B^{m} \\ &= AX^{(h-1)} + X^{(1)}B^{h-1} \\ &= X^{(1+h-1)} \\ &= X^{(h)}. \end{aligned}$$

3.3.2.1 Intercambio de clave

Sean U y V dos interlocutores que desean intercambiar una clave en el grupo G, para ello ejecutan el siguiente protocolo:

(a) Acuerdan $p \in \mathbb{N}$ y $M \in \Theta$, siendo m el orden de M.
(b) El usuario U genera una clave privada $k_1 \in \mathbb{N}$, con $1 \leq k_1 \leq m-1$, de forma aleatoria y calcula $X^{(k_1)}, B^{k_1}$ siendo estas sus claves públicas.
(c) El usuario V genera una clave privada $k_2 \in \mathbb{N}$, con $1 \leq k_2 \leq m-1$, de forma aleatoria y calcula $X^{(k_2)}, B^{k_2}$ siendo estas sus claves públicas.
(d) U calcula $X^{(k_1)} \otimes X^{(k_2)} = X^{(j_1)}$ con $j_1 = k_1 + k_2 \ (mod\ m)$.
(e) V calcula $X^{(k_2)} \otimes X^{(k_1)} = X^{(j_2)}$ con $j_2 = k_2 + k_1 \ (mod\ m)$.

La clave compartida por los usuarios U y V es $P = X^{(k_2+k_1)} = X^{(k_1+k_2)}$, ahora los dos interlocutores, poseen un elemento común y secreto del grupo G. Nótese que aplicando

el teorema 3.4

$$P = X^{(j_1)} = X^{(k_1+k_2)} = X^{(k_2+k_1)} = X^{(j_2)} \in G.$$

3.3.2.2 Esquema de cifrado

Se parte de los mismos elementos públicos y privados del protocolo de intercambio de clave visto en la sección (3.3.2.1) que se supone realizado.

Si el interlocutor U desea enviar un mensaje $\mu \in G$ de forma confidencial a V, realiza el producto matricial

$$AP + \mu B = C$$

y le envía este criptograma a V.

Para recuperar el mensaje, el usuario V procede así:

(a) Calcula el producto AP.
(b) Calcula el $C - AP = C'$.
(c) Calcula B^{-1} (matriz invertible por pertenecer a $GL_s(\mathbb{Z}_p)$) y realiza el producto $C'B^{-1}$, obteniendo μ.

Con esto, las funciones de cifrado y descifrado del usuario V son respectivamente:

$$E_{X^{(k_2)},B^{k_2}}(\mu) = AP + \mu B = C \quad \text{y} \quad D_{k_2}(C) = (C - AP)B^{-1}.$$

3.3.2.3 Firma digital

El esquema de firma digital mostrado a continuación, requiere el mensaje original como entrada para la verificación de la firma. Para ello se supone que U y V han hecho un intercambio de clave P (véase 3.3.2.1) y que U ha hecho llegar a V el mensaje $\mu \in G$, según el protocolo visto en 3.3.2.2. Si el emisor U desea firmar digitalmente el mensaje μ procede así:

(a) Genera un número aleatorio $w \in \mathbb{N}$, con $1 \leq w \leq m-1$.
(b) Calcula $H = B^w, A^w$ y $X^{(w)}$.
(c) Calcula $J = X^{(k_1+w)} = A^{k_1}X^{(w)} + X^{(k_1)}B^w$.
(d) Calcula $T = \mu - X^{((k_1+w)+k_2)}$, donde

$$\begin{aligned} X^{((k_1+w)+k_2)} &= A^{k_1+w}X^{(k_2)} + X^{(k_1+w)}B^{k_2} \\ &= A^{k_1}A^wX^{(k_2)} + JB^{k_2}. \end{aligned}$$

(e) La firma digital es la terna (H, J, T).

Obsérvese que en el paso (d), para calcular T hace falta A^{k_1} y A^w que son claves privadas del emisor U, también hace falta $X^{(k_2)}$ y B^{k_2}, que son claves públicas del receptor V y $X^{(k_1+w)}$, que se puede calcular con los datos accesibles por U.

Si el receptor V desea verificar la firma digital de U procede así:

(a) Calcula

$$\begin{aligned} X^{((k_1+w)+k_2)} = X^{(k_2+(k_1+w))} &= A^{k_2}X^{(k_1+w)} + X^{(k_2)}B^{k_1+w} \\ &= A^{k_2}J + X^{(k_2)}B^{k_1}H. \end{aligned}$$

(b) Calcula $R = T + X^{(k_2+(k_1+w))}$.

(c) Compara μ y R, resultando que la firma es auténtica si $\mu = R$ e incorrecta si $\mu \neq R$.

Nótese que todos los elementos necesarios para el cálculo de $X^{((k_1+w)+k_2)}$ en el paso (a), son claves públicas, elementos de la firma digital o claves privadas de V.

3.3.2.4 Análisis de seguridad

En el intercambio de clave visto en 3.3.2.1, las claves públicas compartidas por los interlocutores son $(X^{(k_1)}, B^{k_1})$ y $(X^{(k_2)}, B^{k_2})$, con lo que se puede aplicar el algoritmo de reducción de Menezes y Wu descrito en 3.3.1.4 para reducir el problema del logaritmo discreto al calculo de logaritmos discretos en pequeñas extensiones de $F_{q^{m_i}}$, donde m_i es el grado de los polinomios irreducibles en los que se puede factorizar el polinomio característico de la matriz B. Aplicando este algoritmo, se tiene como entrada la matriz B^{k_1} (B^{k_2}) y como salida el número natural k_1 (k_2).

3.3.3 Diffie-Hellman para matrices triangulares superiores por bloques modificado

Debido a la vulnerabilidad de los dos criptosistemas presentados en 3.3.1 y 3.3.2, se propone a continuación un esquema de cifrado público, que evita el ataque mediante el algoritmo de reducción de Menezes y Wu, ya que no se hacen públicas las potencias de A y B. Previamente, se ve un teorema que garantiza el secreto compartido (véase [38]).

Teorema 3.5: *Dada* $M = \begin{bmatrix} A & X \\ \boldsymbol{0} & B \end{bmatrix} \in \Theta$ *de orden* m *y* $k_1, k_2 \in \mathbb{N}$, *es conocido que*

$$M^{k_1} = \begin{bmatrix} A^{k_1} & X^{(k_1)} \\ \boldsymbol{0} & B^{k_1} \end{bmatrix}, \quad M^{k_2} = \begin{bmatrix} A^{k_2} & X^{(k_2)} \\ \boldsymbol{0} & B^{k_2} \end{bmatrix}.$$

Si se denota por

$$Y = X^{(k_1)},\ Z = X^{(k_2)},\ N_1 = \begin{bmatrix} A & Z \\ \boldsymbol{0} & B \end{bmatrix}, \quad N_2 = \begin{bmatrix} A & Y \\ \boldsymbol{0} & B \end{bmatrix}$$

y se calcula

$$N_1^{k_1} = \begin{bmatrix} A^{k_1} & Z^{(k_1)} \\ \boldsymbol{0} & B^{k_1} \end{bmatrix}, \quad N_2^{k_2} = \begin{bmatrix} A^{k_2} & Y^{(k_2)} \\ \boldsymbol{0} & B^{k_2} \end{bmatrix}$$

se tiene que $Z^{(k_1)} = Y^{(k_2)}$.

Demostración: Aplicando el teorema 3.2 se tiene

$$Y = X^{(k_1)} = \sum_{i=0}^{k_1-1} A^{k_1-1-i} X B^i \text{ y } Z = X^{(k_2)} = \sum_{i=0}^{k_2-1} A^{k_2-1-i} X B^i$$

por tanto

$$\begin{aligned} Z^{(k_1)} &= \sum_{i=0}^{k_1-1} A^{k_1-1-i} Z B^i \\ &= A^{k_1-1}Z + A^{k_1-2}ZB + \ldots + AZB^{k_1-2} + ZB^{k_1-1} \\ &= A^{k_1-1}(\sum_{i=0}^{k_2-1} A^{k_2-1-i} X B^i) \\ &+ A^{k_1-2}(\sum_{i=0}^{k_2-1} A^{k_2-1-i} X B^i)B \\ &+ \ldots + A(\sum_{i=0}^{k_2-1} A^{k_2-1-i} X B^i)B^{k_1-2} \\ &+ (\sum_{i=0}^{k_2-1} A^{k_2-1-i} X B^i)B^{k_1-1} \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=0}^{k_2-1} A^{k_1+k_2-2-i} X B^i \\
&+ \sum_{i=0}^{k_2-1} A^{k_1+k_2-3-i} X B^{i+1} \\
&+ \ldots + \sum_{i=0}^{k_2-1} A^{k_2-i} X B^{i+k_1-2} \\
&+ \sum_{i=0}^{k_2-1} A^{k_2-1-i} X B^{i+k_1-1}
\end{aligned}
$$

$$
\begin{aligned}
= \; & (A^{k_1+k_2-2} X + A^{k_1+k_2-3} X B \\
+ \; & \ldots + A^{k_1} X B^{k_2-2} + A^{k_1-1} X B^{k_2-1}) \\
+ \; & (A^{k_1+k_2-3} X B + A^{k_1+k_2-4} X B^2 + \ldots \\
+ \; & A^{k_1-1} X B^{k_2-1} + A^{k_1-2} X B^{k_2}) + \ldots \\
+ \; & (A^{k_2} X B^{k_1-2} + A^{k_2-1} X B^{k_1-1} + \ldots + A X B^{k_1+k_2-3}) \\
+ \; & (A^{k_2-1} X B^{k_1-1} + A^{k_2-2} X B^{k_1} + \ldots + X B^{k_1+k_2-2}) \\
= \; & A^{k_1+k_2-2} X + 2A^{k_1+k_2-3} X B + 3A^{k_1+k_2-4} X B^2 \\
+ \; & \ldots + 3A^2 X B^{k_1+k_2-4} + 2A X B^{k_1+k_2-3} + X B^{k_1+k_2-2}.
\end{aligned}
$$

La distribución de los coeficientes que aparecen como resultado de esta expresión puede obtenerse de la forma siguiente:

Si $k_1 < k_2$ tenemos la serie de coeficientes

$$\underbrace{1, 2, 3, 4, \ldots k_1 - 2, k_1 - 1}_{k_1-1}, \underbrace{k_1, k_1, \ldots, k_1}_{k_2-k_1+1}, \underbrace{k_1 - 1, k_1 - 2, \ldots, 4, 3, 2, 1}_{k_1-1}.$$

Si $k_1 > k_2$ tenemos la serie de coeficientes

$$\underbrace{1, 2, 3, 4, \ldots k_2 - 2, k_2 - 1}_{k_2-1}, \underbrace{k_2, k_2, \ldots, k_2}_{k_1-k_2+1}, \underbrace{k_2 - 1, k_2 - 2, \ldots, 4, 3, 2, 1}_{k_2-1}.$$

Si $k_1 = k_2$ tenemos la serie de coeficientes

$$\underbrace{1, 2, 3, 4, \ldots k_1 - 2, k_1 - 1}_{k_1-1}, \underbrace{k_1}_{1}, \underbrace{k_1 - 1, k_1 - 2, \ldots, 4, 3, 2, 1}_{k_1-1}.$$

Como las series son iguales, intercambiando k_1 por k_2, se puede suponer sin pérdida de generalidad que $k_1 \leq k_2$ y se puede simplificar la expresión anterior obteniendo

$$\begin{aligned} Z^{(k_1)} &= (X^{(k_2)})^{(k_1)} \\ &= \sum_{i=0}^{k_1-2}(i+1)A^{k_1+k_2-2-i}XB^i + k_1A^{k_1-1}X^{(k_1+1)}B^{k_1-1} \\ &+ \sum_{i=0}^{k_1-2}(i+1)A^iXB^{k_1+k_2-2-i} \end{aligned}$$

y por tanto

$$\begin{aligned} Z^{(k_1)} &= (X^{(k_2)})^{(k_1)} \\ &= \sum_{i=0}^{k_1-2}(i+1)(A^{k_1+k_2-2-i}XB^i + A^iXB^{k_1+k_2-2-i}) \\ &+ k_1A^{k_1-1}X^{(k_1+1)}B^{k_1-1}. \end{aligned}$$

Al desarrollar $Y^{(k_2)} = (X^{(k_1)})^{(k_2)}$ se llega a la misma expresión, tanto si $k_1 \leq k_2$ como si $k_1 > k_2$ ya que ambos términos conmutan respecto a la suma, de ahí que $Y^{(k_2)} = Z^{(k_1)}$. □

3.3.3.1 Intercambio de clave

Se consideran M, H y G con las mismas características utilizadas en la sección 3.3.1, expresiones (3.6), (3.7) y (3.8). Si U y V desean intercambiar una clave de forma confidencial, ejecutan el protocolo siguiente:

(a) Acuerdan $p \in \mathbb{N}, M \in \Theta$, siendo m el orden de M.

(b) El usuario U genera una clave privada $k_1 \in \mathbb{N}$, con $1 \leq k_1 \leq m-1$, de forma aleatoria y calcula

$$M^{k_1} = \begin{bmatrix} A^{k_1} & X^{(k_1)} \\ \mathbf{0} & B^{k_1} \end{bmatrix} = \begin{bmatrix} A^{k_1} & Y \\ \mathbf{0} & B^{k_1} \end{bmatrix}.$$

(c) El usuario V genera una clave privada $k_2 \in \mathbb{N}$, con $1 \leq k_2 \leq m-1$, de forma

aleatoria y calcula

$$M^{k_2} = \begin{bmatrix} A^{k_2} & X^{(k_2)} \\ \mathbf{0} & B^{k_2} \end{bmatrix} = \begin{bmatrix} A^{k_2} & Z \\ \mathbf{0} & B^{k_2} \end{bmatrix}.$$

(d) Las claves públicas de U y V son respectivamente Y y Z.

(e) El usuario U compone

$$N_1 = \begin{bmatrix} A & Z \\ \mathbf{0} & B \end{bmatrix}$$

y calcula

$$N_1^{k_1} = \begin{bmatrix} A^{k_1} & Z^{(k_1)} \\ \mathbf{0} & B^{k_1} \end{bmatrix}.$$

(f) V compone

$$N_2 = \begin{bmatrix} A & Y \\ \mathbf{0} & B \end{bmatrix}$$

y calcula

$$N_2^{k_2} = \begin{bmatrix} A^{k_2} & Y^{(k_2)} \\ \mathbf{0} & B^{k_2} \end{bmatrix}.$$

La clave compartida por U y V es, atendiendo al teorema 3.5,

$$P = Z^{(k_1)} = Y^{(k_2)},$$

ahora los dos interlocutores, poseen un elemento común y secreto.

3.3.3.2 Esquema de cifrado

Se parte de los mismos elementos públicos y privados del protocolo de intercambio de clave visto en la sección 3.3.3.1 que se supone realizado.

El interlocutor U desea enviar un mensaje $\mu \in G$ de forma privada a V, para ello sigue el protocolo:

(a) Construye las matrices $T_1 = \begin{bmatrix} A & \mu \\ \mathbf{0} & B \end{bmatrix}$ y $T_2 = \begin{bmatrix} A & Z^{(k_1)} \\ \mathbf{0} & B \end{bmatrix}$.

(b) Calcula la matriz $C = T_1T_2$ y envía a V el mensaje cifrado C.

Para recuperar el mensaje el usuario V procede así

(a) Genera la matriz $T_2 = \begin{bmatrix} A & Y^{(k_2)} \\ \mathbf{0} & B \end{bmatrix}$ y calcula su inversa T_2^{-1}, matriz invertible por serlo A y B.

(b) Obtiene la matriz T_1 efectuando el producto CT_2^{-1}.

(c) Recupera el mensaje μ seleccionando el bloque superior derecha de T_1.

Con esto las funciones de cifrado y descifrado del usuario v son respectivamente:

$$E_Z(\mu) = T_1T_2 \quad \text{y} \quad D_{k_2}(C) = CT_2^{-1}.$$

3.3.3.3 Firma digital

Se muestra a continuación un esquema de firma digital que requiere el mensaje original como entrada para la verificación de la firma. Para ello se supone que los usuarios U y V han intercambiado la clave P (véase 3.3.3.1) y U ha hecho llegar a V el mensaje $\mu \in G$ según el protocolo de la sección 3.3.3.2.

Si el emisor U desea firmar digitalmente el mensaje μ procede así:

(a) Genera un número aleatorio $r \in \mathbb{N}$.

(b) Partiendo de

$$N_3 = \begin{bmatrix} A & Z^{(k_1)} \\ \mathbf{0} & B \end{bmatrix} = \begin{bmatrix} A & P \\ \mathbf{0} & B \end{bmatrix},$$

calcula N_3^r y extrae $P^{(r)}$.

(c) Calcula $Q = \mu - P^{(r)}$.

(d) La firma digital es la dupla (r, Q).

Si el receptor V desea verificar la firma digital de U, procede así:

(a) Partiendo de

$$N_3 = \begin{bmatrix} A & Y^{(k_2)} \\ \mathbf{0} & B \end{bmatrix} = \begin{bmatrix} A & P \\ \mathbf{0} & B \end{bmatrix}$$

calcula N_3^r y extrae $P^{(r)}$.

(b) Calcula $W = Q + P^{(r)}$.

(c) Compara μ y W, resultando que la firma es correcta si $\mu = W$ e incorrecta si $\mu \neq W$.

3.3.3.4 Análisis de seguridad

En el análisis para establecer las debilidades de este esquema se han de asumir las denominadas condiciones del peor caso, es decir, que el criptoanalista tiene acceso completo al algoritmo de cifrado, posee una cantidad considerable de texto cifrado y conoce el texto en claro de parte de ese texto cifrado.

No se puede aplicar el algoritmo de reducción de Menezes puesto que las claves públicas de los usuarios U y V ($X^{(k_1)}$ y $X^{(k_2)}$) no son potencias de matrices.

La técnica de criptoanálisis utilizada en este esquema no es muy común en el análisis de sistemas criptográficos de clave pública, ha sido desarrollada en diciembre del 2006 por Climent, Gorla y Rosenthal [30] basándose en el teorema de Cayley-Hamilton.

Teorema 3.6: *(Teorema de Cayley-Hamilton).*

Dada una matriz $M \in GL_n(\mathbb{Z}_p)$ y su polinomio característico

$$q_M(\lambda) = \det(\lambda I_n - M) = a_0 + a_1\lambda + a_2\lambda^2 + \ldots + a_{n-1}\lambda^{n-1} + \lambda^n$$

se tiene

$$q_M(M) = a_0 + a_1 M + a_2 M^2 + \ldots + a_{n-1}M^{n-1} + M^n = \mathbf{0}_n.$$

Siendo I_n la matriz identidad de tamaño n y $\mathbf{0}_n$ la matriz nula del mismo tamaño. Aplicando este teorema, dadas las matrices

$$M = \begin{bmatrix} A & X \\ \mathbf{0} & B \end{bmatrix} \in \Theta \; N_1 = \begin{bmatrix} A & Z \\ \mathbf{0} & B \end{bmatrix} \in \Theta \text{ y } N_2 = \begin{bmatrix} A & Y \\ \mathbf{0} & B \end{bmatrix} \in \Theta,$$

de tamaños $n = r + s$, se tiene que los polinomios característicos son

$$\begin{aligned} q_M(\lambda) &= \det\left(\begin{bmatrix} \lambda I - A & -X \\ \mathbf{0} & \lambda I - B \end{bmatrix}\right) \\ &= \det(\lambda I - A) \cdot det(\lambda I - B) \\ &= q_A(\lambda) \cdot q_B(\lambda) \\ &= a_0 + a_1\lambda + a_2\lambda^2 + \ldots + a_{n-1}\lambda^{n-1} + \lambda^n \end{aligned}$$

supuesto que $\det(\lambda I - M) \neq 0$.

$$\begin{aligned} q_{N_1}(\lambda) &= \det\left(\begin{bmatrix} \lambda I - A & -Z \\ \mathbf{0} & \lambda I - B \end{bmatrix}\right) \\ &= \det(\lambda I - A) \cdot det(\lambda I - B) \\ &= q_A(\lambda) \cdot q_B(\lambda) \\ &= a_0 + a_1\lambda + a_2\lambda^2 + \ldots + a_{n-1}\lambda^{n-1} + \lambda^n. \end{aligned}$$

$$\begin{aligned} q_{N_2}(\lambda) &= \det\left(\begin{bmatrix} \lambda I - A & -Y \\ \mathbf{0} & \lambda I - B \end{bmatrix}\right) \\ &= \det(\lambda I - A) \cdot det(\lambda I - B) \\ &= q_A(\lambda) \cdot q_B(\lambda) \\ &= a_0 + a_1\lambda + a_2\lambda^2 + \ldots + a_{n-1}\lambda^{n-1} + \lambda^n. \end{aligned}$$

Las tres matrices tienen el mismo polinomio característico

$$q_M(\lambda) = q_{N_1}(\lambda) = q_{N_2}(\lambda)$$

y el Teorema de Cayley-Hamilton, garantiza que

$$q_M(M) = q_{N_1}(N_1) = q_{N_2}(N_2) = 0,$$

con lo que se tienen las siguientes expresiones en M

$$a_0 I + a_1 M + a_2 M^2 + \ldots + a_{n-1} M^{n-1} + M^n = \mathbf{0},$$

$$a_0 I + a_1 M + a_2 M^2 + \ldots + a_{n-1} M^{n-1} = -M^n,$$

multiplicando por M

$$a_0 M + a_1 M^2 + a_2 M^3 + \ldots + a_{n-1} M^n = -M^{n+1}.$$

Sustituyendo el valor de M^n

$$a_0 M + a_1 M^2 + a_2 M^3 + \ldots + a_{n-1}(-a_0 I - a_1 M - a_2 M^2 + \ldots - a_{n-1} M^{n-1}) = -M^{n+1},$$

agrupando términos se obtiene

$$M^{n+1} = b_0 I + b_1 M + b_2 M^2 + \ldots + b_{n-1} M^{n-1}.$$

Siguiendo este proceso para un cierto $p \geq n$, se tiene

$$M^p = m_0 I + m_1 M + m_2 M^2 + \ldots + m_{n-1} M^{n-1}, \tag{3.9}$$

luego

$$\begin{aligned} M^p &= \begin{bmatrix} A^p & X^{(p)} \\ \mathbf{0} & B^p \end{bmatrix} \\ &= m_0 \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & I \end{bmatrix} + m_1 \begin{bmatrix} A & X \\ \mathbf{0} & B \end{bmatrix} + \ldots + m_{n-1} \begin{bmatrix} A^{n-1} & X^{(n-1)} \\ \mathbf{0} & B^{n-1} \end{bmatrix}. \end{aligned}$$

En consecuencia, se tienen las igualdades

$$A^p = m_0 I + m_1 A + m_2 A^2 + \ldots + m_{n-1} A^{n-1},$$

$$X^{(p)} = m_1 X + m_2 X^{(2)} + \ldots + m_{n-1} X^{(n-1)},$$

$$B^p = m_0 I + m_1 B + m_2 B^2 + \ldots + m_{n-1} B^{n-1}.$$

Actuando como en la expresión (3.9), se obtiene

$$N_1^p = m_0 I + m_1 N_1 + m_2 N_2^2 + \ldots + m_{n-1} N_1^{n-1},$$

$$N_2^p = m_0 I + m_1 N_2 + m_2 N_2^2 + \ldots + m_{n-1} N_2^{n-1},$$

donde los coeficientes $m_0, m_1, m_2, \ldots, m_{n-1}$ son coincidentes en todos los casos.

En el esquema que se está analizando, las claves públicas son

$$Y = X^{(k_1)}, Z = X^{(k_2)},$$

es decir, se conoce M, N_1 y N_2, y se puede plantear, para un $k_1 < n$, el sistema lineal

$$Y = X^{(k_1)} = m_1 X + m_2 X^{(2)} + \ldots + m_{n-1} X^{(n-1)},$$

en el que todos las matrices son conocidas. En el caso de que el sistema tenga solución se calculan los coeficientes $m_1, m_2, \ldots, m_{n-1}$ y se está en condiciones de averiguar la clave secreta compartida $Z^{(k_1)}$ mediante la ecuación

$$Z^{(k_1)} = m_1 Z + m_2 Z^{(2)} + \ldots + m_{n-1} Z^{(n-1)},$$

puesto que las matrices $Z, Z^{(2)}, \ldots, Z^{(n-1)}$ y los coeficientes $m_1, m_2, \ldots m_{n-1}$ son conocidos.

Ejemplo.

Dada la matriz

$$M = \begin{bmatrix} 0 & 1 & 1 & 1 & 2 \\ 4 & 4 & 2 & 4 & 3 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 3 & 4 & 2 \end{bmatrix}$$

con elementos en $\mathbb{Z}_5$.

Si $k \geq 5$ se puede escribir

$$M^k = m_0 I + m_1 M + m_2 M^2 + m_3 M^3 + m_4 M^4,$$

$$X^{(k)} = m_1 X + m_2 X^{(2)} + m_3 X^{(3)} + m_4 X^{(4)},$$

$$Z^{(k)} = m_1 Z + m_2 Z^{(2)} + m_3 Z^{(3)} + m_4 Z^{(4)}.$$

Si las claves secretas son $k_1 = 700$ y $k_2 = 400$, se tiene que

$$Y = X^{(700)} = \begin{bmatrix} 3 & 2 & 3 \\ 1 & 2 & 1 \end{bmatrix},$$

$$Z = X^{(400)} = \begin{bmatrix} 3 & 4 & 0 \\ 3 & 2 & 3 \end{bmatrix}.$$

Al ser

$$X^{(700)} = m_1 X + m_2 X^{(2)} + m_3 X^{(3)} + m_4 X^{(4)},$$

se tiene

$$X^{(700)} = m_1 \begin{bmatrix} 1 & 1 & 2 \\ 2 & 4 & 3 \end{bmatrix} + m_2 \begin{bmatrix} 3 & 3 & 3 \\ 1 & 4 & 0 \end{bmatrix} + m_3 \begin{bmatrix} 1 & 0 & 4 \\ 1 & 2 & 1 \end{bmatrix} + m_4 \begin{bmatrix} 3 & 3 & 0 \\ 0 & 4 & 2 \end{bmatrix}.$$

Resolviendo el sistema, se tiene $m_1 = 4, m_2 = 2, m_3 = 1$ y $m_4 = 4$. Como se conoce

$$Z^{(1)} = \begin{bmatrix} 3 & 4 & 0 \\ 3 & 2 & 3 \end{bmatrix}, Z^{(2)} = \begin{bmatrix} 3 & 0 & 2 \\ 3 & 4 & 0 \end{bmatrix},$$

$$Z^{(3)} = \begin{bmatrix} 0 & 0 & 1 \\ 3 & 2 & 4 \end{bmatrix} \text{ y } Z^{(4)} = \begin{bmatrix} 1 & 3 & 2 \\ 0 & 1 & 3 \end{bmatrix},$$

se está en condiciones de averiguar la clave secreta $Z^{(700)}$

$$Z^{(700)} = 4Z^{(1)} + 2Z^{(2)} + 1Z^{(3)} + 4Z^{(4)} = \begin{bmatrix} 2 & 3 & 3 \\ 1 & 2 & 3 \end{bmatrix}.$$

Se concluye que es posible un ataque con éxito, simplemente con las claves públicas del criptosistema, si se hace uso del teorema de Cayley-Hamilton.

3.3.4 Esquema multiplicativo.

Tras el análisis realizado en cada uno de los esquemas anteriores, se presenta a continuación un sistema criptográfico de clave pública, que proporciona unos tiempos de ejecución sensiblemente menores al criptosistema de ElGamal y equiparable a RSA, con unos niveles de seguridad similares.

Sean

$$M_1 = \begin{bmatrix} A_1 & X_1 \\ \mathbf{0} & B_1 \end{bmatrix} \text{ y } M_2 = \begin{bmatrix} A_2 & X_2 \\ \mathbf{0} & B_2 \end{bmatrix}$$

elementos del conjunto Θ con órdenes m_1 y m_2, respectivamente, y los conjuntos

$$G_1 = \{X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, \ldots, X_1^{(m_1-1)}\},$$

$$G_2 = \{X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, \ldots, X_2^{(m_2-1)}\}.$$

Se define, para un par de números $x, y \in \mathbb{N}$, la siguiente notación:

$$A_{xy} = A_1^x A_2^y,$$

$$B_{xy} = B_1^x B_2^y$$

y

$$C_{xy} = A_1^x X_2^{(y)} + X_1^{(x)} B_2^y.$$

3.3.4.1 Intercambio de clave

Sean U y V dos interlocutores que desean intercambiar una clave. Para ello ejecutan el protocolo siguiente:

(a) Acuerdan $p \in \mathbb{N}$ y $M_1, M_2 \in \Theta$, siendo m_1 y m_2 los órdenes de M_1 y M_2 respectivamente.

(b) El usuario U genera dos números aleatorios $r, s \in \mathbb{N}$ tales que

$$1 \leq r \leq m_1 - 1,\ 1 \leq s \leq m_2 - 1,$$

calcula A_{rs}, B_{rs}, C_{rs} y agrupa estos valores como bloques en una nueva matriz

$$C = \begin{bmatrix} A_{rs} & C_{rs} \\ \mathbf{0} & B_{rs} \end{bmatrix}.$$

(c) El usuario U envía la matriz C a V.

(d) V genera dos números aleatorios $v, w \in \mathbb{N}$ tales que

$$1 \leq v \leq m_1 - 1,\ 1 \leq w \leq m_2 - 1,$$

calcula A_{vw}, B_{vw} y C_{vw} y agrupa estos valores como bloques en una nueva matriz

$$D = \begin{bmatrix} A_{vw} & C_{vw} \\ \mathbf{0} & B_{vw} \end{bmatrix}.$$

(e) El usuario V envía la matriz D a U.

(f) Las claves públicas de los usuarios U y V son las matrices C y D respectivamente.

(g) El usuario U calcula

$$K_u = A_1^r A_{vw} X_2^{(s)} + A_1^r C_{vw} B_2^s + X_1^{(r)} B_{vw} B_2^s.$$

(h) El usuario V calcula

$$K_v = A_1^v A_{rs} X_2^{(w)} + A_1^v C_{rs} B_2^w + X_1^{(v)} B_{rs} B_2^w.$$

En el siguiente teorema se demuestra que $K_u = K_v$, siendo este valor la clave compartida.

Teorema 3.7: *Con los valores descritos anteriormente,* $K_u = K_v$.

DEMOSTRACIÓN: Se tiene que

$$C = \begin{bmatrix} A_{rs} & C_{rs} \\ \mathbf{0} & B_{rs} \end{bmatrix} = M_1^r M_2^s,\ D = \begin{bmatrix} A_{vw} & C_{vw} \\ \mathbf{0} & B_{vw} \end{bmatrix} = M_1^v M_2^w,$$

$$M_1^r = \begin{bmatrix} A_1^r & X_1^{(r)} \\ \mathbf{0} & B_1^r \end{bmatrix},\ M_1^v = \begin{bmatrix} A_1^v & X_1^{(v)} \\ \mathbf{0} & B_1^v \end{bmatrix},$$

$$M_2^s = \begin{bmatrix} A_2^s & X_2^{(s)} \\ \mathbf{0} & B_2^s \end{bmatrix} \text{ y } M_2^w = \begin{bmatrix} A_2^w & X_2^{(w)} \\ \mathbf{0} & B_2^w \end{bmatrix}.$$

Sean

$$M_u = M_1^r D M_2^s = \begin{bmatrix} A_u & K_u \\ \mathbf{0} & B_u \end{bmatrix},$$

$$M_v = M_1^v C M_2^w = \begin{bmatrix} A_v & K_v \\ \mathbf{0} & B_v \end{bmatrix}.$$

Con esto se tiene que

$$M_u = M_1^r D M_2^s = M_1^r M_1^v M_2^w M_2^s = M_1^v M_1^r M_2^s M_2^w = M_1^v C M_2^w = M_v$$

y, consecuentemente, $K_u = K_v$. □

Como ha quedado demostrado en este teorema, los usuarios U y V tienen ahora un secreto compartido,

$$K_u = K_v = P.$$

Las claves privadas son respectivamente r, s, v y w, que no tienen por qué ser números primos, con lo que se evitan los tests de primalidad.

3.3.4.2 Esquema de cifrado

Se parte de los mismos elementos públicos y privados del protocolo de intercambio de clave visto anteriormente en 3.3.4.1, que se supone previamente realizado.

Si el interlocutor U desea enviar un mensaje $\mu \in G_1$ de forma privada al interlocutor V, procede así:

(a) Construye la matriz

$$T_1 = \begin{bmatrix} A_1^r & \mu \\ \mathbf{0} & B_1^r \end{bmatrix}$$

y calcula

$$M_u = M_1^r D M_2^s.$$

(b) Realiza el producto matricial

$$H = T_1 M_u$$

y le envía al usuario V la matriz H.

Para recuperar el mensaje enviado por U, el usuario V procede así:

(a) Construye la matriz

$$M_v = M_1^v C M_2^w.$$

(b) Calcula la inversa M_v^{-1}, matriz invertible por serlo A_1, A_2, B_1 y B_2.
(c) Calcula el producto HM_v^{-1}, que es igual a T_1.
(d) Recupera el mensaje μ seleccionando el correspondiente bloque de T_1.

Con esto las funciones de cifrado y descifrado del usuario V son respectivamente:

$$E_D(\mu) = T_1 M_u \text{ y } D_{r,s}(H) = HM_v^{-1}.$$

3.3.4.3 Firma digital

A continuación se describe un esquema de firma digital que requiere el mensaje original como entrada para su verificación. Para ello se supone que los usuarios U y V han hecho un intercambio de clave P (véase 3.3.4.1) y que el usuario U ha enviado a V el mensaje μ, según el protocolo visto en 3.3.4.2.

Si el emisor U desea firmar digitalmente el mensaje μ procede así:

(a) Genera un número aleatorio $r \in \mathbb{N}$ y calcula $Q = \mu - P^{(r)}$.
(b) La firma digital es la dupla (r, Q).

Si el receptor V desea verificar la firma digital de U, procede así:

(a) Calcula $P^{(r)}$.
(b) Calcula $R = Q + P^{(r)}$.
(c) Compara μ y R, resultando que la firma es correcta (auténtica) si $\mu = R$ e incorrecta si $\mu \neq R$.

3.3.4.4 Análisis de seguridad

Los ataques por fuerza bruta no son viables si se elige un orden de M suficientemente alto, por ejemplo 1024 bits. No es posible realizar un criptoanálisis de este sistema utilizando el Teorema de Cayley-Hamilton puesto que, además de tener polinomios característicos diferentes, no se publican potencias de matrices, sólo son accesibles p, M_1, M_2, C y D.

Por otro lado, aplicar la reducción propuesta por Menezes y Wu es inviable, puesto que como se ha comentado anteriormente, no se publican potencias de matrices.

En definitiva, para criptoanalizar este esquema se tiene que aplicar algoritmos de raíz cuadrada para el cálculo de logaritmos discretos como los vistos en la sección 2.2.2. Estos algoritmos alcanzan un orden de complejidad $\sqrt{\overline{p}}$, donde $\overline{p}$ es el mayor factor primo del orden del grupo generado por la matriz M.

Con la elección que se va ha hacer en la sección 3.3.5 de p y de los tamaños de los bloques A y B se garantiza la ineficacia de este tipo de ataques, ya que el factor primo más grande que se puede obtener es $p^r - 1$ ó $p^s - 1$ que, como se verá se puede hacer eficientemente tan grande como se quiera (por ejemplo del orden de 1024 bits).

3.3.5 Versión optimizada del esquema multiplicativo

En el esquema propuesto, dada la matriz

$$M = \begin{bmatrix} A & X \\ \mathbf{0} & B \end{bmatrix} \in \Theta,$$

con los bloques $A \in \mathrm{GL}_r(\mathbb{Z}_p)$ y $B \in \mathrm{GL}_s(\mathbb{Z}_p)$, el orden del subgrupo generado por M es $mcm(p^r - 1, p^s - 1)$, valor que se puede hacer tan grande como se quiera eligiendo los valores de p, r y s de forma apropiada.

Los algoritmos más rápidos para el cálculo de logaritmos discretos son los de raíz cuadrada, vistos en el capítulo 2 sección 2.2.2, que pueden alcanzar un orden de complejidad de $\sqrt{\overline{q}}$, donde $\overline{q}$ es el mayor factor primo de orden del grupo. Puesto que en

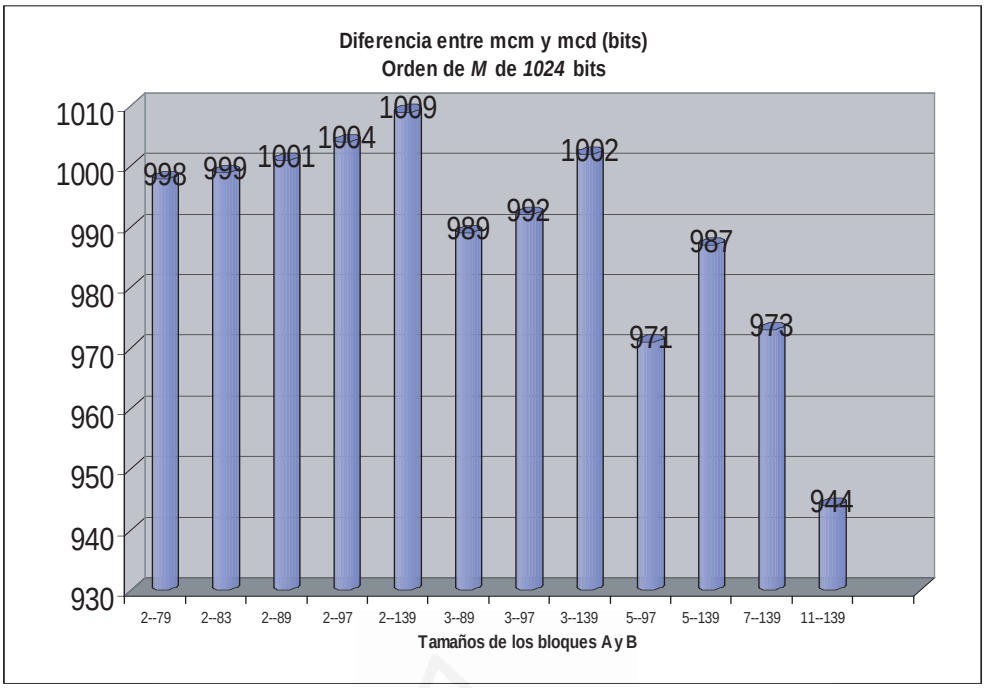

Figura 3.1: *Diferencia en bits entre el mcm y el mcd*

los criptosistemas presentados el orden del grupo es $mcm(p^r - 1, p^s - 1)$, hay que intentar hacer este número lo suficientemente grande para evitar ataques basados en los algoritmos de raíz cuadrada, por ejemplo 1024 bits. Además, para eludir estos ataques, se debe evitar que $p^r - 1$ y $p^s - 1$ tengan divisores comunes grandes, es decir, hay que intentar minimizar el $mcd(p^r - 1, p^s - 1)$. Estos dos parámetros (mcm y mcd), junto como el tiempo de ejecución, permiten estudiar diferentes opciones a la hora de seleccionar un tamaño de matriz óptimo. Se debe maximizar la diferencia entre el mcm y el mcd, con el menor tiempo de ejecución posible.

En el anexo B se muestran las tablas realizadas para el cálculo del mcm y mcd. Resumiendo estas tablas en el gráfico de barras de la figura 3.1, se observa que la diferencia en bits entre el mcd y el mcm es mayor cuanto mayor es la diferencia entre r y s. Por consiguiente, un buen tamaño para el bloque A de la matriz M es $r = 2$.

Para obtener el tamaño óptimo de s se estudian los tiempos de ejecución del intercambio de clave del esquema multiplicativo. En el anexo C se adjuntan tablas con algunos de estos tiempos de las que se ha realizado un resumen en el gráfico de la figura 3.2.

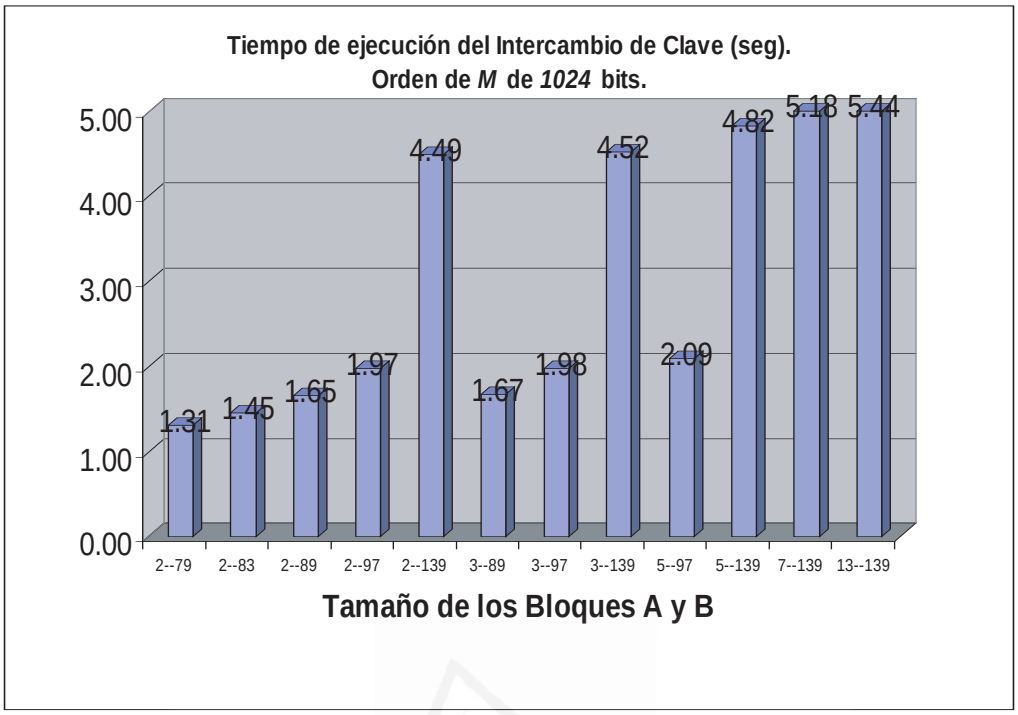

Figura 3.2: *Tiempo de ejecución del esquema multiplicativo*

Analizando las diversas posibilidades, se concluye que al menos los tamaños mostrados en la tabla siguiente optimizan el esquema de cifrado propuesto.

r	s	p	Orden M (bits)	Tiempo ejecución (seg.)
2	83	5167	1024	1,44820
2	89	2903	1024	1,65156

Para la realización de las comparaciones con otros criptosistemas existentes, se ha tomado como valores óptimos $r = 2, s = 89$ y $p = 2903$. Por otro lado, los exponentes de las matrices, que aparecen en el esquema multiplicativo no tienen por que ser primos ni grandes números ya que la pieza fundamental del criptosistema es el orden de la matriz M.

3.3.5.1 Comparativa con Diffie-Hellman, ElGamal y RSA

En este apartado se realiza una comparativa entre en intercambio de clave Diffie-Hellman y el intercambio de clave del sistema multiplicativo en su versión optimizada,

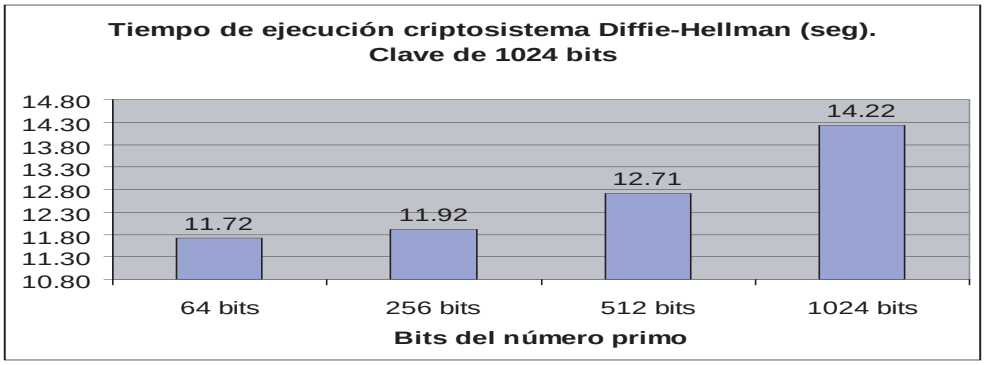

Figura 3.3: *Tiempo de ejecución Diffie-Hellman. Clave 1024 bits*

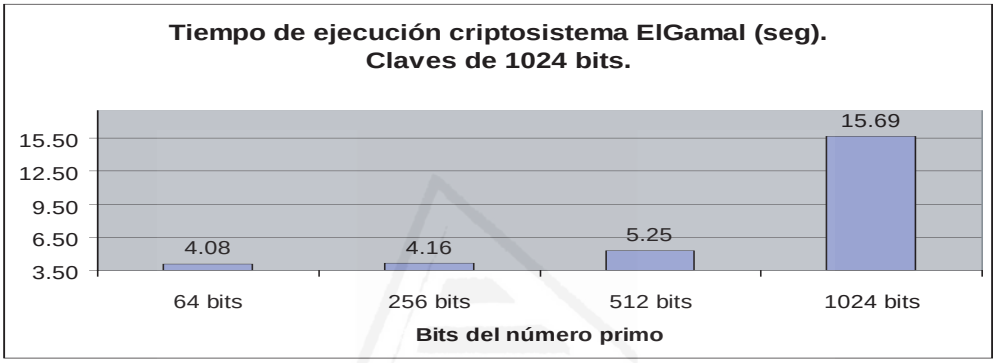

Figura 3.4: *Tiempo de ejecución ElGamal. Clave 1024 bits*

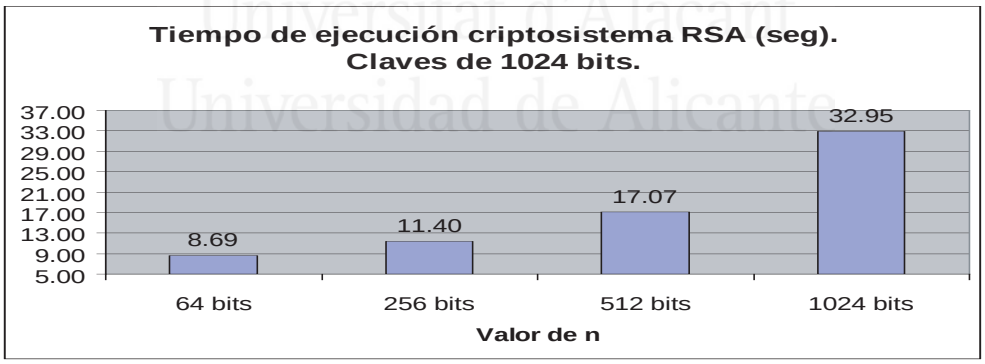

Figura 3.5: *Tiempo de ejecución RSA. Clave 1024 bits*

así como entre el cifrado de la información hecha con ElGamal y RSA y el cifrado hecho con este mismo criptosistema. Para ello, se han realizado múltiples pruebas cuyos tiempos se adjuntan en los anexos C y D. En las figuras 3.3, 3.4 y 3.5 se muestra un resumen gráfico de los mismos.

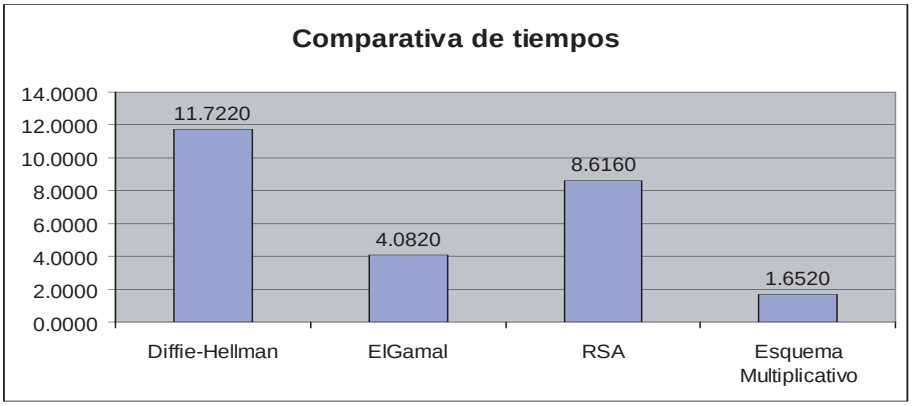

Figura 3.6: *Comparativa de los tiempos de ejecución*

Por otro lado, se han seleccionado distintos tamaños de matrices y se ha adecuado el tamaño de p para obtener un orden de 1024 bits. Todas las pruebas se han realizado bajo las mismas condiciones iniciales, empleando un microprocesador Intel Pentium 4 con una frecuencia de reloj de 3.06 GHz, 512KB de caché, 400 MHz de front-side-bus y 1024 MB de RAM. La implementación se ha realizado en MATLAB versión 7.0.1.24704 (R14) Service Pack 1.

Conviene hacer una serie de aclaraciones sobre los tiempos de ejecución mostrados en la figura 3.6:

- Se ha utilizado el programa MATLAB que optimiza operaciones con matrices.
- Las pruebas de tiempos de los criptosistemas Diffie-Hellman, RSA y el ElGamal se han realizado también en MATLAB que no optimiza la generación de grandes números primos.
- Se deja como línea futura de investigación la implementación en otras aplicaciones.

Se puede concluir que existe una aplicación (MATLAB) sobre la que el esquema multiplicativo propuesto es claramente superior a otros criptosistemas existentes como Diffie-Hellman, RSA o el ElGamal.

Para la implementación del esquema multiplicativo se ha utilizado el algoritmo de exponenciación rápida para matrices triangulares superiores por bloques que se propone en la sección 3.4.2.

3.4 Exponenciación rápida

Muchos algoritmos de clave pública utilizan exponenciación dentro de grupos finitos, siendo las bases y los exponentes números muy grandes. Realizar exponenciaciones mediante multiplicación de la base por si misma tantas veces como indique el exponente es inviable para números grandes, por lo que es necesario buscar algoritmos de exponenciación eficientes.

En los esquemas propuestos se realizan grandes potencias de matrices por lo que es necesario un algoritmo de exponenciación rápida, aplicado a matrices triangulares por bloques, que haga eficiente el calculo de estas potencias, se aporta una propuesta en la sección 3.4.2.

3.4.1 Exponenciación rápida para números

Dados dos números naturales a y b, para calcular $c = a^b \ (mod\ N)$ se considera la representación binaria de b,

$$b = 2^0 b_0 + 2^1 b_1 + 2^2 b_2 + \ldots + 2^n b_n,$$

y se expresa la potencia que se quiere calcular en función de dicha representación,

$$a^b = a^{2^0 b_0 + 2^1 b_1 + 2^2 b_2 + \ldots + 2^n b_n} = \prod_{i=0}^{n} a^{2^i b_i}.$$

Nótese que los coeficientes b_i sólo pueden valer 0 ó 1, por tanto, para calcular a^b solo se deben multiplicar los elementos a^{2^i} correspondientes a los dígitos binarios de b que valen 1.

Nótese, además, que $a^{2^i} = (a^{2^{i-1}})^2$, por lo que, partiendo de a, se puede calcular el siguiente valor de esta serie elevando al cuadrado el anterior.

El algoritmo de exponenciación rápida para números naturales queda como sigue

(a) Se hace $c := 1,\ a := a\ (mod\ N),\ j := 0$;

(b) Mientras $j < n - 1 = [\log_2 b] + 1$ hacer

- Si $b_j = 1$ entonces efectuar $c := c * a$
- $a := a^2\ (mod\ N)$
- $j := j + 1$, y volver a (b)

(c) Fin.

3.4.2 Exponenciación rápida para matrices triangulares superiores por bloques

A continuación se propone un algoritmo de exponenciación rápida para matrices triangulares superiores por bloques que es una adaptación del algoritmo de exponenciación rápida para números enteros (véase [7]).

Sea

$$M = \begin{bmatrix} A & X \\ \mathbf{0} & B \end{bmatrix} \in \Theta$$

y $n \in \mathbb{N}$, existe un conjunto de índices ordenado

$$I = \{i_1,\ i_2,\ i_3,\ i_4,\ \ldots, i_q\} \subset \mathbb{N},$$

tal que el número n se puede escribir como

$$n = 2^{i_1} + 2^{i_2} + 2^{i_3} + \ldots + 2^{i_q} = \sum_{j=1}^{q} 2^{i_j}.$$

Para el cálculo de potencias de los bloques A y B de la matriz M,

$$A^n = A^{2^{i_1}+2^{i_2}+\ldots+2^{i_q}} = A^{2^{i_1}} A^{2^{i_2}} \ldots A^{2^{i_q}},$$

$$B^n = B^{2^{i_1}+2^{i_2}+\ldots+2^{i_q}} = B^{2^{i_1}} B^{2^{i_2}} \ldots B^{2^{i_q}},$$

se utiliza

$A^{2^0} = A = A_0,$

$A^{2^1} = AA = A_1,$

$A^{2^2} = A^2 A^2 = A_2,$

$A^{2^3} = A^{2^2} A^{2^2} = A_3,$

$A^{2^4} = A^{2^3} A^{2^3} = A_4,$

.....

$A^{2^e} = A^{2^{e-1}} A^{2^{e-1}} = A_e.$

En consecuencia, el cálculo de potencias grandes de matrices del tipo A o B, se reduce a multiplicar matrices, por ejemplo:

$$A^{1234} = A^{2^{10}+2^7+2^6+2^4+2^1} = A^{2^{10}} A^{2^7} A^{2^6} A^{2^4} A^{2^1}.$$

Para el bloque X de M, se tiene

$X^{(2^0)} = X$,

$X^{(2^1)} = X^{(1+1)} = AX^{(1)} + X^{(1)}B = AX + XB = X_1$,

$X^{(2^2)} = X^{(2+2)} = A^2X^{(2)} + X^{(2)}B^2 = A_1X_1 + X_1B_1 = X_2$,

$X^{(2^3)} = X^{(2^2+2^2)} = A^{2^2}X^{(2^2)} + X^{(2^2)}B^{2^2} = A_2X_2 + X_2B_2 = X_3$,

$X^{(2^4)} = X^{(2^3+2^3)} = A^{2^3}X^{(2^3)} + X^{(2^3)}B^{2^3} = A_3X_3 + X_3B_3 = X_4$,

.....

$X^{(2^e)} = X^{(2^{e-1}+2^{e-1})} = A_{e-1}X_{e-1} + X_{e-1}B_{e-1} = X_e$.

El siguiente teorema garantiza el cálculo rápido de potencias de los bloques superior derecha de las matrices triangulares por bloques.

Teorema 3.8: *Dado un entero n*

$$n = \sum_{j=1}^{q} 2^{i_j},$$

y dado un conjunto ordenado de índices

$$I = \{i_1,\ i_2,\ i_3,\ i_4,\ \ldots, i_q\} \subset \mathbb{N},$$

se tiene que

$$X^{(n)} = \sum_{k=1}^{q} A^{n_1^{(k)}} X^{(n_2^{(k)})} B^{n_3^{(k)}}. \tag{3.10}$$

Los exponentes toman los valores

$$n_1^{(k)} = \begin{cases} 0 & \textit{para } k = q, \\ \sum_{j=1}^{q-k} 2^{i_j} & \textit{para } k\textit{=1, 2, \ldots,q-1,} \end{cases}$$

$$n_2^{(k)} = 2^{i_{q-k+1}},\ \textit{para } k = 1, 2, 3, \ldots q,$$

y

$$n_3^{(k)} = \begin{cases} 0 & \textit{para } k = 1, \\ \sum_{j=q-k+2}^{q} 2^{i_j} & \textit{para } k\textit{=2, 3, \ldots,q.} \end{cases}$$

DEMOSTRACIÓN: Se demuestra la expresión (3.10) por inducción sobre q.

Para $q = 1$ se tiene $I = \{i_1\}$,

$$\begin{aligned}\sum_{k=1}^{1} A^{n_1^{(k)}} X^{(n_2^{(k)})} B^{n_3^{(k)}} &= A^{n_1^{(1)}} X^{(n_2^{(1)})} B^{n_3^{(1)}} \\ &= A^0 X^{(2^{i_1})} B^0 \\ &= X^{(2^{i_1})} \\ &= X^{(n)}.\end{aligned}$$

Se supone que la expresión (3.10) es cierta para un número $n \in \mathbb{N}$, cuyo conjunto de índices I tiene $q-1$ elementos y se demuestra que es cierta para $n \in \mathbb{N}$ cuyo conjunto de índices tiene q elementos.

$$\begin{aligned}X^{(n)} &= X^{((2^{i_1}+2^{i_2}+2^{i_3}+\ldots+2^{i_{q-1}})+2^{i_q})} \\ &= X^{(n'+2^{i_q})} \\ &= A^{n'} X^{(2^{i_q})} + X^{(n')} B^{2^{i_q}} \\ &= A^{2^{i_1}+2^{i_2}+2^{i_3}+\ldots+2^{i_{q-1}}} X^{(2^{i_q})} + X^{(n')} B^{2^{i_q}} \\ &= A^{2^{i_1}+2^{i_2}+2^{i_3}+\ldots+2^{i_{q-1}}} X^{(2^{i_q})} + (\sum_{k=1}^{q-1} A^{n_1^{(k)}} X^{(n_2^{(k)})} B^{n_3^{(k)}}) B^{2^{i_q}} \\ &= A^{2^{i_1}+2^{i_2}+2^{i_3}+\ldots+2^{i_{q-1}}} X^{(2^{i_q})} + \sum_{k=1}^{q-1} A^{n_1^{(k)}} X^{(n_2^{(k)})} B^{n_3^{(k)}+2^{i_q}} \\ &= \sum_{k=1}^{q} A^{n_1^{(k)}} X^{(n_2^{(k)})} B^{n_3^{(k)}}.\end{aligned}$$

Ejemplo.

Dado $M \in \Theta$, y $n = 1234 = 2^{10+7+6+4+1}$, cuyo conjunto ordenado de índices es

$$I = \{10,\ 7,\ 6,\ 4,\ 1\} \subset \mathbb{N},$$

por el teorema 3.2 del capítulo 3 (página 92), se tiene

$$\begin{aligned} X^{(1234)} &= A^{2^{10}} A^{2^7} A^{2^6} A^{2^4} X^{(2^1)} + A^{2^{10}} A^{2^7} A^{2^6} X^{(2^4)} B^{2^1} \\ &+ A^{2^{10}} A^{2^7} X^{(2^6)} B^{2^4} B^{2^1} + A^{2^{10}} X^{(2^7)} B^{2^6} B^{2^4} B^{2^1} \\ &+ X^{(2^{10})} B^{2^7} B^{2^6} B^{2^4} B^{2^1}. \end{aligned}$$

Aplicando el teorema 3.8,

$$X^{(1234)} = \sum_{k=1}^{5} A^{n_1^{(k)}} X^{(n_2^{(k)})} B^{n_3^{(k)}},$$

con lo que:

(a) Primer índice

$$n_1^{(k)} = \sum_{j=1}^{5-k} 2^{i_j}.$$

Para $k = 1$, se tiene

$$n_1^{(1)} = \sum_{j=1}^{4} 2^{i_j} = 2^{i_1} + 2^{i_2} + 2^{i_3} + 2^{i_4} = 2^{10} + 2^7 + 2^6 + 2^4,$$

si $k = 2$

$$n_1^{(2)} = \sum_{j=1}^{3} 2^{i_j} = 2^{i_1} + 2^{i_2} + 2^{i_3} = 2^{10} + 2^7 + 2^6,$$

si $k = 3$

$$n_1^{(3)} = \sum_{j=1}^{2} 2^{i_j} = 2^{i_1} + 2^{i_2} = 2^{10} + 2^7,$$

si $k = 4$

$$n_1^{(4)} = \sum_{j=1}^{1} 2^{i_j} = 2^{i_1} = 2^{10},$$

y si $k = 5$

$$n_1^{(5)} = 0.$$

(b) Segundo índice

$$n_2^{(k)} = 2^{i_{5-k+1}} = 2^{i_{6-k}}.$$

Para $k = 1$, se tiene

$$n_2^{(1)} = 2^{i_5} = 2^1,$$

si $k = 2$

$$n_2^{(2)} = 2^{i_4} = 2^4,$$

si $k = 3$

$$n_2^{(3)} = 2^{i_3} = 2^6,$$

si $k = 4$

$$n_2^{(4)} = 2^{i_2} = 2^7,$$

y si $k = 5$

$$n_2^{(5)} = 2^{i_1} = 2^{10}.$$

(c) Tercer índice

$$n_3^{(k)} = \sum_{j=5-k+2}^{5} 2^{i_j} = \sum_{j=7-k}^{5} 2^{i_j}.$$

Para $k = 1$

$$n_3^{(1)} = 0,$$

si $k = 2$

$$n_3^{(2)} = \sum_{j=5}^{5} 2^{i_j} = 2^{i_5} = 2^1,$$

si $k = 3$

$$n_3^{(3)} = \sum_{j=4}^{5} 2^{i_j} = 2^{i_4} + 2^{i_5} = 2^4 + 2^1,$$

si $k = 4$

$$n_3^{(4)} = \sum_{j=3}^{5} 2^{i_j} = 2^{i_3} + 2^{i_4} + 2^{i_5} = 2^6 + 2^4 + 2^1,$$

si $k = 5$

$$n_3^{(5)} = \sum_{j=3}^{5} 2^{i_j} = 2^{i_2} + 2^{i_3} + 2^{i_4} + 2^{i_5} = 2^7 + 2^6 + 2^4 + 2^1.$$

En resumen, se tiene

$$\begin{aligned}
X^{(1234)} &= \sum_{k=1}^{5} A^{n_1^{(k)}} X^{(n_2^{(k)})} B^{n_3^{(k)}} \\
&= A^{n_1^{(1)}} X^{(n_2^{(1)})} B^{n_3^{(1)}} + A^{n_1^{(2)}} X^{(n_2^{(2)})} B^{n_3^{(2)}} \\
&+ A^{n_1^{(3)}} X^{(n_3^{(k)})} B^{n_3^{(3)}} + A^{n_1^{(4)}} X^{(n_2^{(4)})} B^{n_3^{(4)}} \\
&+ A^{n_1^{(5)}} X^{(n_2^{(5)})} B^{n_3^{(5)}} \\
&= A^{2^{10}+2^7+2^6+2^4} X^{(2^1)} B^0 + A^{2^{10}+2^7+2^6} X^{(2^4)} B^{2^1} \\
&+ A^{2^{10}+2^7} X^{(2^6)} B^{2^4+2^1} + A^{2^{10}} X^{(2^7)} B^{2^6+2^4+2^1} \\
&+ A^0 X^{(2^{10})} B^{2^7+2^6+2^4+2^1}.
\end{aligned}$$

Capítulo 4

Conclusiones y lineas futuras de investigación

4.1 Conclusiones

Como consecuencia del crecimiento de Internet a nivel mundial y de las comunicaciones en general, la seguridad se ha convertido en un aspecto de gran relevancia, hasta el punto que se considera un requisito en los planes de diseño e instrumentación de nuevas redes.

El cifrado de la información es el presente y futuro inmediato de la protección de la información. La criptografía asegura la confidencialidad de la información allá donde esté. El inconveniente es la sobrecarga de los ordenadores y la falta de un estándar que sea aceptado globalmente.

Los métodos de cifrado clásicos han demostrado su ineficacia ante la potencia de cálculo de los ordenadores modernos, por lo que no se usan en aplicaciones que requieren una mínima seguridad.

El hecho de que los sistemas de cifrado de clave pública sean más modernos y ofrezcan más posibilidades que los de clave privada, no supone que vayan a desplazarlos, porque cada uno es el más adecuado en cada caso particular. Los de clave privada se emplean cuando se quiere cifrar información que no se va a transmitir a una gran cantidad de usuarios mientras que los de clave pública son los más adecuados para el intercambio de información. Además, éstos deben ser necesariamente empleados en los casos donde los de clave privada no son válidos, como para la creación de firmas digitales, el intercambio de claves o la autenticación de usuarios.

En definitiva, los criptosistemas de clave pública son importantes en todas las áreas

del comercio electrónico. Los métodos existentes confían en la dificultad de resolver ciertos problemas. Puesto que el poder computacional aumenta permanentemente, la longitud de la clave requerida para mantener el nivel de seguridad necesario debe ser mayor. Es por lo tanto deseable encontrar técnicas con una mayor complejidad algebraica.

Después de estudiarse las aplicaciones de las matrices triangulares superiores por bloques al cifrado simétrico de flujo (véase [4]), así como su aplicación a las curvas elípticas (véase [29, 38]), en este trabajo se plantea el uso de las matrices triangulares por bloques a los criptosistemas asimétricos o de clave pública. Primero se plantean por orden cronológico tres esquemas criptográficos y debido a las debilidades de seguridad de dichos criptosistemas, se ha diseñado e implementado otro cuya idea básica es estudiar el comportamiento de los productos matriciales del tipo

$$M_1^v M_2^w,$$

con M_1, M_2 elementos de un grupo de matrices triangulares superiores por bloques con orden muy elevado, y v, w enteros que no tienen porque ser de gran tamaño. Para el criptoanálisis de dicho esquema criptográfico no son aplicables técnicas de fuerza bruta puesto que el orden del grupo es de 1024 bits aproximadamente. Además, como se ha comprobado, también es seguro ante ataques basados en valores y vectores propios, con lo que el único ataque posible es mediante algoritmos de raíz cuadrada que también quedan inutilizados eligiendo convenientemente un grupo con orden muy elevado (por ejemplo de alrededor de 1024 bits) y con divisores comunes relativamente pequeños.

Se ha realizado un estudio para determinar el tamaño óptimo de dichas matrices triangulares superiores por bloques, llegando a la conclusión de que los bloques de tamaño $r = 2$, $s = 89$ y $p = 2903$, obtienen unos tiempos de ejecución buenos, perfectamente comparables o incluso mejores que RSA. Puesto que no se requieren números primos elevados, se evita el problema de los tests de primalidad.

Una de las líneas futuras de investigación sería completar un criptoanálisis detallado de este tipo de criptosistemas, así como su implementación en otras aplicaciones.

4.2 Líneas futuras de investigación

Durante el estudio de las matrices triangulares por bloques a los criptosistemas de clave pública, han aparecido varias líneas futuras de investigación. A continuación, se describe una serie de aplicaciones que resultan de interés en criptografía.

4.2.1 Modificaciones

Una posibilidad consiste en modificar la matriz del criptosistema, de forma que en lugar de ser de tamaño 2×2 bloques, pase a ser de 3×3 bloques, de la forma

$$M = \begin{bmatrix} A & Y & X \\ \mathbf{0} & B & Z \\ \mathbf{0} & \mathbf{0} & C \end{bmatrix}.$$

En esta construcción los bloques A, B y C son matrices asociadas a polinomios primitivos, donde r, s y t son los tamaños respectivos de dichos bloques. En este caso el orden de la matriz M sería

$$mcm(p^r - 1, p^s - 1, p^t - 1).$$

4.2.2 Propuesta de nuevos esquemas

Sean

$$M_1 = \begin{bmatrix} A_1 & X_1 \\ \mathbf{O} & B_1 \end{bmatrix},$$

y

$$M_2 = \begin{bmatrix} A_2 & X_2 \\ \mathbf{O} & B_2 \end{bmatrix}$$

elementos del conjunto Θ, con órdenes m_1 y m_2 respectivamente.

Un nuevo esquema de intercambio de clave para dos interlocutores U y V es:

(a) Acuerdan $p \in \mathbb{N}$, $M_1, M_2 \in \Theta$ con ordenes m_1 y m_2 respectivamente.

(b) El usuario U genera dos números aleatorios $r, s \in \mathbb{N}$, tal que

$$1 \leq r \leq m_1 \text{ y } 1 \leq s \leq m_2,$$

calcula

$$C = M_1^r M_2^s$$

y le envía este valor a V.

(c) El usuario V genera dos números aleatorios $v, w \in \mathbb{N}$, tal que

$$1 \leq v \leq m_1 \text{ y } 1 \leq w \leq m_2,$$

calcula

$$F = M_1^v M_2^w,$$

$$\begin{aligned} D &= M_1^v C M_2^w \\ &= M_1^v M_1^r M_2^s M_2^w \\ &= M_1^{v+r} M_2^{s+w} \\ &= M_1^{r+v} M_2^{w+s} \\ &= M_1^r M_1^v M_2^w M_2^s \end{aligned}$$

y le envía D a U.

(d) El usuario U calcula

$$M_1^{-r}, M_2^{-s}$$

y

$$F = M_1^{-r} D M_2^{-s} = M_1^{-r} M_1^r M_1^v M_2^w M_2^s M_2^{-s} = M_1^v M_2^w.$$

El secreto compartido es F, y las claves privadas son respectivamente r, s, v y w. Además, estas claves privadas no tienen por que ser números primos, con lo que se evitarían los tests de primalidad.

Es muy interesante realizar un estudio sobre la viabilidad y seguridad del esquema sugerido anteriormente, así como otros que se puedan constituir con técnicas parecidas.

Bibliografía

[1] Adleman, L.M.: A subexponential algorithm for the discrete logarithm problem with applications cryptography. Proc. of the 20th Annual IEEE Symposium on Foundations of Computer Science. IEEE Press, 55-60, (1979).

[2] Agraval, M., Kayal, N., Saxena, N.: Primes is in $\mathbb{P}$. Annals of Mathematics, Vol. 160(2), 781-793, (2004)
Web http://cse.iitk.ac.in/news/primality.html, (2003).

[3] Alexi, W., Chor, B., Goldreich, O., Schnorr, C.P.: RSA and Rabin functions:certeins parts are hard as the whole. SIAM Journal on Computing, Vol. 17, 194-209, (1988).

[4] Álvarez, R.: Aplicaciones de las matrices por bloques a los criptosistemas de cifrado en flujo. Tesis Doctoral (2005).

[5] Álvarez, R., Tortosa, L., Vicent, J.F., Zamora, A.: A public key cryptosystem based on block upper triangular matrices. 4ht WSEAS, Proceedings of ISCOCO 2005, 163-168. Tenerife (2005).

[6] Álvarez, R., Tortosa, L., Vicent, J.F., Zamora, A.: An internal security kernel. WSEAS, Proceedings of Transaction on Business and economics. Venecia (2004).

[7] Álvarez, R., Ferrández, F., Vicent, J.F., Zamora, A.: Applying quick exponentiation for block upper triangular matrices. Applied Mathematics and Computation, Vol. 183, 729-737 (2006).

[8] Anshel, I., Anshel, M., Goldfeld, D.: An algebraic method for public-key cryptography. Mathematical Research Letters, Vol. 6, 287-291, (1999).

[9] Bach, E., Shallit, J.: Algorithmic number theory. MIT Press. Cambridge, MA, (1996).

[10] Beckman, B.: Arne Beurling and the Swedish crypto program during World War II, Providence (2002).

[11] Beckman, B.: Códigos secretos. Piramide, Madrid (1990).

[12] Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. Advances

in Cryptology - CRYPTO 1993– Lecture Notes in Computer Science, Vol 773, (1994).

[13] Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations Among Notions os Security for Public-key Encryption Schemes. Advances in Cryptology –CRYPTO 1998– Lecture Notes in Computer Science, Vol. 1462, 26-46, (1998).

[14] Bennet, C., Brassard, G.: The dawn of a new era for quantum cryptography: the experimental prototype is working. SICACT News, Vol. 20, 78, (1989).

[15] Ben-Or, M., Chor, B., Shamir, A.: On the cryptographic security of single RSA bits. Proceedings of the 15th ACM STOC, 421-430, (1983).

[16] Beth, T., González, S., González, M.I., Martínez, C., Steinwandt, R.: Cryptographic Shelter for the Information Society: Modeling and Fighting Novel Attacks on Cryptographic Primitives. Techno-Legal Aspects of Information Society and New Economy, (2003).

[17] Beth, T.: El estado de la Criptografía de Clave Pública a la vista de las posibilidades de la Computación Cuántica. Actas de la VI Reunión Española sobre Criptología y Seguridad de la Información, VI RECSI, 39-50, (2001).

[18] Beth, T., Frisch, M., Simmons, G.J.: Public-Key Cryptography: State of the Art and Future Directions. Lecture Notes in Computer Science, Springer Verlag, (1992).

[19] Beth, T.: The State of Public-Key Cryptography - Not Only After the Invention of Quantum Computers. Information Security Technical Report, (1999).

[20] Blackley, G.R., Borosh, I.: RSA public key cryptosystems do not always concel messages. Comput. Math. Appl., Vol. 5, 169-178, (1979).

[21] Blum, M., Miscali, S.: How to generate cryptographically strong sequences of pseudo-random bits. SIAM Journal on Computing, Vol. 13, 850-864, (1986).

[22] Boneh, D., Venkatesan, R.: Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related scheme. Advances in Cryptology –CRYPTO 1996– Lecture Notes in Computer Science, Vol. 921, 129-142, (2001).

[23] Boneh, D., Shparlinski, I.E.: On the unpredictability of bits of the elliptic curve Diffie-Hellman scheme. Advances in Cryptology –CRYPTO 2001– Lecture Notes in Computer Science, Vol. 2139, 201-212, (2001).

[24] Brillhart, J., Lehmer, D.H., Selfridge, J.L.: New primality criteria and factorizations of $2^m + 1$. Math. Comp., Vol. 29, 620-647, (1975).

[25] Buchmann, J.A., Williams, H. C.: A Key Exchange System Based on Imaginary Quadratic Fields. Journal of Cryptology Vol. 1, (1988).

[26] Canfield, E.R., Erdos, P., Pomerance, C.: On a problem of Oppenheim concerning

Factorisation Numerorum. J. Number Theory, Vol. 17, 1-28, (1983).

[27] Chor, B., Goldreich, O.: RSA/Rabin least significant bit are $\frac{1}{2} + \frac{1}{poly(\log n)}$ secure. Advances in Cryptology –CRYPTO 1984– Lecture Notes in Computer Science, Vol. 196, 303-313, (1984).

[28] Churchouse, R.F.: A classical Cipher Machine: The ENIGMA, some Aspects, Its history and Solution. Institute of Mathematics and its Applications, Vol. 27, (1991).

[29] Climent, J.J., Ferrandiz, F., Vicent, J.F., Zamora, A.: A non linear elliptic curve cryptosystem based on matrices. Applied Mathematics and Computation, Vol. 74, 150-164 (2005).

[30] Climent, J.J., Gorla, E., Rosenthal, J.: Cryptanalysis of the CFVZ cryptosystem. Advances in Mathematics of Communications (AMC) Vol. 1, 1-11 (2006).

[31] Cohen, H.: A course in computational algebraic number theory. GTM 138, Springer. Berlin (1995).

[32] Coppersmith, D., Odlyzko, A., and Schroeppel, R.: Discrete logarithms in $GF(p)$. Algorithmica, 1-15, (1986).

[33] Coppersmith, D.: Cryptography. IBM J. Res. Develop. Vol. 31, No. 2, (1987).

[34] Deavours, C.A., Kruh, L.: Machine Cryptography and Modern cryptanalysis. Artech House Inc., (1985).

[35] Diffie, W., Hellman, M.: New directions In Cryptography. IEEE Trans. Information Theory, Vol. 22, 644-654, (1976).

[36] Diffie, W.: The First Ten Years of Public Key Cryptography. Proceedings of the IEEE Vol. 76, No. 5, (1988).

[37] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Information Theory, Vol. 31, 469-472, (1985).

[38] Ferrández, F.: Sistemas criptográficos de curva elíptica basados en matrices. Tesis Doctoral, 102-103, (2004).

[39] Fischlin, R., Schnorr, C.P.: Stronger security proofs for RSA and Rabin bitsStronger security proofs for RSA and Rabin bits. Journal of Cryptology, Vol. 13, 221-244, (2000).

[40] Garzon, M., Zalcstein, Y.: The Complexity of Grigorchuk groups with application to cryptography. Theoretical Computer Science, Vol. 88, 83-98, (1991).

[41] Gauss, K.F.: Disquisitiones arithmeticae. Fleischer, Leipzig (1801). Translation into English by Arthur A. Clarke Springer-Verlag, New York (1986).

[42] Goldreich, O., Levin, L.A.: A hard core predicate for any one way function. Proceedings of the 21st ACM STOC, 25-22, (1989).

[43] Goldwasser, S., Bellare, M.: Lecture Notes on Cryptography, (2001).

[44] Goldwasser, S., Miscali, S.: Probabilistic encryption. Journal of Computer and Systrem Science, Vol. 28, 270-299, (1984).

[45] González, M.I., Naslund, M.: A survey of hard core functions. Proc. Workshop on Cryptography and Computational Number Theory Singapure 1999. Birkaüser, 227-255, (2000).

[46] González, M.I., Steinwandt, R.: Obstacles in Two Public-Key Cryptosystems Based on Group Factorizations. Cryptology. Editors: K. Nemoga, O. Grosek, Tatra Mountains Math. Publications, 23-37, (2002).

[47] González, M.I., Sparlinski, I.E.: On the security of Diffie-Hellman bits. Proc. Workshop on Cryptography and Computational Number Theory Singapure 1999. Birkaüser, 256-268, (2001).

[48] González, M.I., Sparlinski, I.E.: Security of the most significant bits of the Shamir message passing scheme. Mathematics of Computation, Vol. 71, 333-342, (2002).

[49] González, M.I., Martínez, C., Steinwandt, R.: Un Marco Común para Varios Esquemas de Clave Pública Basados en Grupos. Actas de la VII Reunión Española sobre Criptología y Seguridad de la Información, VII RECSI, 353-364, (2002).

[50] González, M.I., Naslund, M., Sparlinski, I.E.: The hidden number problem inextension fields and its applications. Proc. the 3rd Latin American Theoretical Informatics Conference, Cancun 2002. Lecture Notes on Computer Science, 105-117, Berlin (2001).

[51] Gordon, D. M.: A Survey of Fast Exponentiation Methods. Journal of Algorithms, Vol. 27, 129-146, (1998).

[52] Gorenstein, D., Lyons, R., Solomon, R.: The Classification of the Finite Simple Groups, Volume 40 of Mathematical Surveys and Monogrpahs. AMS, Providence, (1998).

[53] Gruska, J.: Quantum Computing. MacGraw-Hill, (1999).

[54] Hahn, S.G., Lee, E., Park, J.H.: Complexity of the Generalized Conjugacy Problem. Discrete Applied Mathematics, (2003).

[55] Hall, C., Goldberg, I., Schneider, B.: Reaction Attacks Against Several Public-Key Cryptosystems. Vijay Varadharajan and Yi Mu, editors. Information and Communication Security, Second International Conference, ICICS'99, Lecture Notes in Computer Science, pages 2-12. Springer, Vol. 1726, (1999).

[56] Hastad, J.: On using RSA with low exponent in a public key network. Advanced in Criptology-Proc. of Crypto'85, LNCS, Vol. 218, 403-408, (1986).

[57] Hastad, J., Impagliazzo, R., Levin, L.A., Luby, M.: Pseudo random number ge-

nerators from any one way function. SIAM Journal on Computing, Vol. 28, 1364-1396, (1999).

[58] Hastad, J.: Solving simultaneous modular equations of low degree. SIAM J. Comput., Vol. 17, 336-341, (1988).

[59] Hastad, J., Naslund, M.: The securyty of individual RSA and discrete log bits. Journal of the AMS (2002).

[60] Hellman, M.E., Reyneri, J.M.: Fast computation of discrete logarithm in $GF(p)$. Advances in Cryptology: Proceedings of CRYPTO'82 Plenum Press, 3-13, (1983).

[61] Hoffman, K., Kunze, R.: Linear Algebra. Prentice-Hall, New Jersey (1971).

[62] Horng, G., Yang, C.S.: Key authentication scheme for cryptosystems based on discrete logarithms. Computer Communications Vol. 19, 848-850, (1996).

[63] IEEE Std 1363.: IEEE Standard especifications for public key cryptography (2000).

[64] Kahn, D.: The Codebreakers, the Story of Secret Writing. Macmillan Publishing Co. Inc., (1967).

[65] Kitaev, A.Y., Shen, A.H., Vyalyi, M.N.: Classical and Quantum Computation. Graduate Studies in Mathematics. AMS Press, Vol. 47, (2002).

[66] Ko, K.H., Lee, S.L., Cheon, J.H., Han, J.W., Kang, J., Park, C.: New Public-Key Cryptosystem using Braid Groups. In Advances in Cryptology. Proceedings of CRYPTO 2000, Lecture Notes in Computer Science, pages 166-183, Santa Barbara, California, USA, (2000).

[67] Koblitz, N.: A Course in Number Theory and Cryptography, GTM 114, Springer Verlag, (1987).

[68] Koblitz, N.: Hyperelliptic Cryptosystems. Journal of Cryptology Vol. 1, (1989).

[69] Kraitchik, M.: Théorie des nombres. Gauthier-Villars. Vol. 1, 119-123, Paris (1922).

[70] Kurosawa, K., Takeuchi, M.: Public key cryptosystem using a reciprocal number with the same intractability as factoring a large number. Cryptology, Vol. 1, 225-233, (1988).

[71] LaMacchia, B.A., Odlyzko, A.M.: Computation of discrete logarithm problem in prime fields. Designs, Codes and Cryptography, Vol. 1, 46-62, (2001).

[72] Lee, C.C., Hwang, M.S., Li, L.H.: A new key authentication scheme based on discrete logarithms. Applied Mathematics and Computation, 343-349, (2003).

[73] Lidl, R., Müller,W.B.: Permutation polynomials in RSA cryptosystems. Advances in Cryptology Proceedings of Cryptology 83, 293-301, (1984).

[74] Lidl, R., Niederreiter, H.: Introduction to Finite Fields and Their Applications.

Cambridge University Press, (1994).

[75] Loxton, J.H., Khoo, D.S.P., Bird, G.J.: A cubic RSA code equivalent to factorization. Cryptology, Vol. 5, 139-150, (1992).

[76] Magliveras, S.S.: A cryptosystem from logarithmic signatures of finite groups. Proceedings of the 29'th Midwest Symposium on Circuits and Systems, pages 972-975. Elsevier Publishing Company, (1986).

[77] McCurley, K.: The discret logarithm problem. Crytology and Computational Number Theory, Proceedings of Symposia in Applied Mathematics, Vol. 42, 49-74, (1990).

[78] McEliece, R. J.: A Public Key Cryptosystem Based on Algebraic Coding Theory. DSN Progress Report 42-44, Jet Propulsion Laboratory, (1978).

[79] McEliece, R. J.: Finite Fields for Computer Scientists and Engineers. Kluwer Academic Publishers, (1987).

[80] Menezes, A., Wu, Yi-Hong: A polinomial representation for logaritms in $GF(q)$. Acta arithmetica, Vol. 47, 255-261, (1986).

[81] Menezes, A., Van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Florida (2001).

[82] Menezes, A., Wu, Yi-Hong.: The Discrete Logarithm Problem in $GL(n, q)$. Ars Combinatoria, Vol. 47, 22-32, (1997).

[83] Miller, G.: Riemann's hypothesis and test for primality. J. Comput. and Sistem Sci., Vol. 13, 300-317, (1976).

[84] Miller, V. S.: Use of Elliptic Curves in Cryptography. Advances in Cryptology Crypto'85, Lecture Notes in Computer Science, Vol. 218, (1985).

[85] Mollin, R.A..: RSA and public-key cryptography. Chapman and Hall /CRC Boca Raton. Florida (2003).

[86] Mosca, M., Ekert, A.: The Hidden Subgroup Problem and Eigenvalue Estimation on a Quantum Computer. Proceedings of the 1st NASA International Conference on Quantum Computing and Quantum Comunication, (1999).

[87] Mullen, G., White, D.: A polynomial representation for logarithms in GF(q). Acta Arithmetica, Vol. 47, 255-261, (1986).

[88] National Bureau of Standards. Data Encryption Standard, FIPS-Pub.46. National Bureau of Standards, U.S. Department of Commerce. Washington D.C., (1977).

[89] National Institute for Standards and Technology. Digital Signature Standard. Computer System Laboratory, FIPS PUB 186, Gaithersburg, (1994).

[90] Niederreiter, H.: A short proof for explicit formulas for discrete logarithms in

finite fields. Applicable Algebra in Eng., Comm., and Computer, Vol. 1, 55-57, (1990).

[91] Odlyzko, A.M.: Discrete logarithm and smooth polynomils in finite fields. Theory, Applications and Algorithms. G.L. Mullen and P. Shiue, eds., American Math. Society, Contemporary Math Series, 269-278, (1994).

[92] Odoni, R. W. K., Varadharajan, V., Sanders, P. W.: Public Key Distribution in Matrix Rings. Electronic Letters, Vol. 20, 386-387, (1984).

[93] Paeng, S.H., Kwon, D., Ha, K.C., Kim, J.H.: Improved public key cryptosystem using finite non-abelian groups, IACR eprint (2001).

[94] Paeng, S.H., Ha, K.C., Kim, J.H., Chee, S., Park, C.: New Public Key Cryptosystem Using Finite Non Abelian Groups. J. Kilian. Advances in cryptology - CRYPTO 2001, Lecture Notes in Computer Science, Springer, Vol. 2139, 470-485, (2001).

[95] Pocklington, H.C.: The determination of the prime or composite nature of large numbers by Fermat's theorem. Math. Porc. Cambridge Philos. Soc., Vol. 18, 29-30, (1914).

[96] Pohlig, S.C., Hellman, M.E.: An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. IEEE Trans. Info. Theory, Vol. 24, 106-110, (1978).

[97] Pollard, J.M.: Monte Carlo methods for index computation (mod p). Math. Computation, Vol. 32, 918-924, (1978).

[98] Popek, G., Kline, C.: Encryption and security computer networks. ACM Computing Survey, Vol. 11, 331-356, (1979).

[99] Rabin, M.O.: Digitalized signartures and public key functions as intractable as factorization. MIT Laboratory for Computer Science, Technical Report MIT/LCS/TM-212, (1979).

[100] Rabin, M.O.: Probabilistic algorithms for testing primality. J. Number Theory, Vol. 12, 128-138, (1980).

[101] Reisel, H.: Mersenne numbers. MTAC, Vol. 12, 207-213, (1958).

[102] Riesel, H.: Prime numbers and computer methods for factorization. Birhäuser Boston (1994).

[103] Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, Vol. 21, 120-126, (1978).

[104] Rossen, K.: Elementary Number Theory International Edition. (2004).

[105] Rotman, J.J.: An Introduction to the Theory of Groups. Graduate Texts in Mat-

hematics. Springer, Vol. 148, (1999).

[106] Salomaa, A.: Public Key Cryptology. Monographs on Theoretical Computer Science, Vol.23, Springer Verlag, (1990).

[107] Scheidler, R., Buchmann, J.A., Williams, H.C.: A Key Exchange Protocol Using Real Quadratic Fields. Journal of Cryptology, Vol. 7, 192-198, (1994).

[108] Scheidler, R., Williams, H.C.: A Public Key Cryptosystem Utilizing Cyclotomic Fields. Designs, Codes and Cryptography, Vol. 6, 237-242, (1995).

[109] Schneier, B.: Applied Cryptography Second Edition: protocols, algorithms and source code in C. John Wiley and Sons, New York (1996).

[110] Shanks, D.: Class number, a theory of factorization, and genera. Number Theory Institute, Proc. Symposium pure Mathematics, American Mathematical Society, Vol. 20, 415-440, (1981).

[111] Shannon, C.: A Mathematical Theory of Communication. Bell System Thechnical Journal, (1948).

[112] Shannon, C.: Communications Theory of Secrecy Systems. Bell System Thechnical Journal, (1949).

[113] Short, P.W.: Algorithms for quantum computation: discrete logarithm and factoring. Proc. of the 35th Annual IEEE Symposium on Foundations of Computer Science. IEEE Press, 124-134, (1994).

[114] Singh, S.: Los códigos secretos: El arte de la ciencia de la criptografía desde el antiguo Egipto a la era de Internet, Debate. Madrid (2000).

[115] Solovay, R., Strassen, V.: A fast Monte-Carlo test for primality. SIAM J. Comput., Vol. 6, 84-85, (1975).

[116] Sorenson, J.P.: Polylog depth circuits for integer factoring and discrete logarithms. Information and Computation, Vol. 110, 1-18, (1994).

[117] Stallings, W.: Cryptography and Network Security: Principles and Practice. Third Edition. Prentice Hall, New Jersey, (2003).

[118] Teske, E.: Squar-root algorithms for the discrete logarithm problem (a survey). Center for Applied Cryptographic Research. University of Waterloo, Technical Report CORR 2001-07, (2001).

[119] Van Oorschot, P.: A comparison of practical public key cryptosystems based on integer factorization and discrete logarithms. Contemporary Cryptology: The Science of information Integrity. IEEE Press, 289-322, New York, (1992).

[120] Wells, A.L.: A polynomial form for logarithms modulo a prime. IEEE Trans. Information Theory, Vol. 30, 845-846, (1984).

[121] Wiener, M.J.: Cryptoanalysis of short RSA secret exponents. IEEE Trans. In-

form. Theory IEEE Press, Vol. 36, 553-558, (1990).

[122] Williams, H.C.: A modification of RSA public key encryption procedure. IEEE Trans. Inform. Theory IEEE Press,Vol. 26, 726-729, (1980).

Apéndice A

Ejemplos

A.1 Diffie-Hellman para matrices triangulares superiores por bloques.

A.1.1 Intercambio de clave.

(a) Los usuarios U y V acuerdan $p = 127$ y

$$M = \begin{bmatrix} A & X \\ \mathbf{0} & B \end{bmatrix} \in \Theta \text{ con elementos en } \mathbb{Z}_p$$

$$M = \begin{bmatrix} 93 & 5 & 122 & 86 & 107 & 122 & 60 & 1 & 15 & 6 \\ 120 & 15 & 6 & 124 & 1 & 38 & 75 & 90 & 19 & 30 \\ 123 & 15 & 17 & 122 & 10 & 8 & 125 & 108 & 0 & 103 \\ 0 & 21 & 13 & 3 & 12 & 78 & 121 & 61 & 9 & 5 \\ 96 & 9 & 122 & 91 & 110 & 13 & 38 & 51 & 61 & 126 \\ 0 & 0 & 0 & 0 & 0 & 118 & 110 & 119 & 114 & 113 \\ 0 & 0 & 0 & 0 & 0 & 16 & 23 & 24 & 27 & 11 \\ 0 & 0 & 0 & 0 & 0 & 10 & 126 & 28 & 107 & 121 \\ 0 & 0 & 0 & 0 & 0 & 33 & 43 & 37 & 45 & 23 \\ 0 & 0 & 0 & 0 & 0 & 90 & 79 & 90 & 77 & 61 \end{bmatrix}.$$

(b) U genera una clave privada $k_1 = 123456789$, de forma aleatoria y calcula

$$N_1 = M^{k_1} = \begin{bmatrix} A^{k_1} & Y \\ \mathbf{0} & B^{k_1} \end{bmatrix} =$$

$$= \begin{bmatrix} 102 & 117 & 116 & 67 & 125 & 107 & 32 & 31 & 91 & 85 \\ 58 & 4 & 80 & 66 & 45 & 68 & 96 & 119 & 6 & 100 \\ 118 & 110 & 37 & 6 & 74 & 9 & 50 & 10 & 27 & 70 \\ 52 & 106 & 94 & 51 & 34 & 87 & 19 & 123 & 57 & 36 \\ 22 & 75 & 2 & 83 & 37 & 32 & 45 & 11 & 105 & 77 \\ 0 & 0 & 0 & 0 & 0 & 42 & 88 & 0 & 60 & 84 \\ 0 & 0 & 0 & 0 & 0 & 58 & 36 & 116 & 17 & 25 \\ 0 & 0 & 0 & 0 & 0 & 74 & 50 & 77 & 120 & 113 \\ 0 & 0 & 0 & 0 & 0 & 42 & 65 & 15 & 114 & 47 \\ 0 & 0 & 0 & 0 & 0 & 46 & 84 & 19 & 24 & 114 \end{bmatrix}.$$

(c) V genera una clave privada $k_2 = 987654321$, de forma aleatoria y calcula

$$N_2 = M^{k_2} = \begin{bmatrix} A^{k_2} & Z \\ \mathbf{0} & B^{k_2} \end{bmatrix} =$$

$$= \begin{bmatrix} 21 & 20 & 63 & 74 & 25 & 78 & 7 & 19 & 52 & 108 \\ 44 & 62 & 54 & 124 & 100 & 13 & 103 & 116 & 73 & 6 \\ 9 & 18 & 29 & 37 & 15 & 0 & 72 & 53 & 74 & 69 \\ 0 & 109 & 113 & 77 & 43 & 107 & 29 & 44 & 31 & 93 \\ 29 & 67 & 73 & 37 & 2 & 68 & 28 & 77 & 10 & 19 \\ 0 & 0 & 0 & 0 & 0 & 13 & 58 & 22 & 51 & 100 \\ 0 & 0 & 0 & 0 & 0 & 30 & 46 & 51 & 71 & 62 \\ 0 & 0 & 0 & 0 & 0 & 115 & 1 & 28 & 8 & 13 \\ 0 & 0 & 0 & 0 & 0 & 98 & 80 & 22 & 25 & 47 \\ 0 & 0 & 0 & 0 & 0 & 5 & 31 & 8 & 96 & 65 \end{bmatrix}.$$

(d) Las claves públicas de U y V son respectivamente N_1 y N_2.

(e) U calcula

$$N_2^{k_1} = \begin{bmatrix} 93 & 5 & 122 & 86 & 107 & 75 & 86 & 125 & 4 & 44 \\ 120 & 15 & 6 & 124 & 1 & 9 & 58 & 44 & 14 & 66 \\ 123 & 15 & 17 & 122 & 10 & 75 & 80 & 103 & 122 & 101 \\ 0 & 21 & 13 & 3 & 12 & 52 & 123 & 12 & 54 & 85 \\ 96 & 9 & 122 & 91 & 110 & 39 & 103 & 105 & 90 & 45 \\ 0 & 0 & 0 & 0 & 0 & 118 & 110 & 119 & 114 & 113 \\ 0 & 0 & 0 & 0 & 0 & 16 & 23 & 24 & 27 & 11 \\ 0 & 0 & 0 & 0 & 0 & 10 & 126 & 28 & 107 & 121 \\ 0 & 0 & 0 & 0 & 0 & 33 & 43 & 37 & 45 & 23 \\ 0 & 0 & 0 & 0 & 0 & 90 & 79 & 90 & 77 & 61 \end{bmatrix}.$$

(f) V calcula

$$N_1^{k_2} = \begin{bmatrix} 93 & 5 & 122 & 86 & 107 & 75 & 86 & 125 & 4 & 44 \\ 120 & 15 & 6 & 124 & 1 & 9 & 58 & 44 & 14 & 66 \\ 123 & 15 & 17 & 122 & 10 & 75 & 80 & 103 & 122 & 101 \\ 0 & 21 & 13 & 3 & 12 & 52 & 123 & 12 & 54 & 85 \\ 96 & 9 & 122 & 91 & 110 & 39 & 103 & 105 & 90 & 45 \\ 0 & 0 & 0 & 0 & 0 & 118 & 110 & 119 & 114 & 113 \\ 0 & 0 & 0 & 0 & 0 & 16 & 23 & 24 & 27 & 11 \\ 0 & 0 & 0 & 0 & 0 & 10 & 126 & 28 & 107 & 121 \\ 0 & 0 & 0 & 0 & 0 & 33 & 43 & 37 & 45 & 23 \\ 0 & 0 & 0 & 0 & 0 & 90 & 79 & 90 & 77 & 61 \end{bmatrix},$$

La clave compartida por U y V es el bloque superior derecha que se denomina

$$P = Z^{(k_1)} = Y^{(k_2)}$$

$$P = \begin{bmatrix} 75 & 86 & 125 & 4 & 44 \\ 9 & 58 & 44 & 14 & 66 \\ 75 & 80 & 103 & 122 & 101 \\ 52 & 123 & 12 & 54 & 85 \\ 39 & 103 & 105 & 90 & 45 \end{bmatrix}.$$

A.1.2 Esquema de cifrado.

Sea

$$\mu = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix} \in G,$$

un mensaje codificado que un interlocutor U quiere enviar a otro V, para ello sigue el esquema:.

(a) Genera una clave privada $k_3 = 5678901234$ y construye las matrices

$$T_1 = \begin{bmatrix} A^{k_3} & \mu \\ \mathbf{0} & B^{k_3} \end{bmatrix} =$$

$$= \begin{bmatrix} 93 & 5 & 122 & 86 & 107 & 1 & 2 & 3 & 4 & 5 \\ 120 & 15 & 6 & 124 & 1 & 6 & 7 & 8 & 9 & 10 \\ 123 & 15 & 17 & 122 & 10 & 11 & 12 & 13 & 14 & 15 \\ 0 & 21 & 13 & 3 & 12 & 16 & 17 & 18 & 19 & 20 \\ 96 & 9 & 122 & 91 & 110 & 21 & 22 & 23 & 24 & 25 \\ 0 & 0 & 0 & 0 & 0 & 118 & 110 & 119 & 114 & 113 \\ 0 & 0 & 0 & 0 & 0 & 16 & 23 & 24 & 27 & 11 \\ 0 & 0 & 0 & 0 & 0 & 10 & 126 & 28 & 107 & 121 \\ 0 & 0 & 0 & 0 & 0 & 33 & 43 & 37 & 45 & 23 \\ 0 & 0 & 0 & 0 & 0 & 90 & 79 & 90 & 77 & 61 \end{bmatrix}$$

y

$$T_2 = N_2^{k_1} = \begin{bmatrix} A_2^{k_1} & P \\ \mathbf{0} & B_2^{k_1} \end{bmatrix} =$$

$$= \begin{bmatrix} 77 & 21 & 21 & 38 & 6 & 101 & 40 & 12 & 126 & 8 \\ 73 & 62 & 122 & 40 & 34 & 112 & 79 & 38 & 117 & 62 \\ 118 & 88 & 81 & 23 & 58 & 65 & 121 & 51 & 91 & 61 \\ 124 & 61 & 62 & 8 & 98 & 69 & 66 & 65 & 48 & 50 \\ 111 & 95 & 67 & 120 & 91 & 115 & 84 & 28 & 103 & 48 \\ 0 & 0 & 0 & 0 & 0 & 79 & 15 & 6 & 123 & 87 \\ 0 & 0 & 0 & 0 & 0 & 96 & 6 & 50 & 57 & 91 \\ 0 & 0 & 0 & 0 & 0 & 83 & 104 & 38 & 3 & 35 \\ 0 & 0 & 0 & 0 & 0 & 54 & 89 & 47 & 93 & 1 \\ 0 & 0 & 0 & 0 & 0 & 31 & 60 & 118 & 113 & 27 \end{bmatrix},$$

(b) Calcula la matriz $C = T_1 T_2 =$

$$= \begin{bmatrix} 83 & 50 & 44 & 98 & 16 & 12 & 104 & 67 & 39 & 112 \\ 53 & 25 & 17 & 81 & 21 & 76 & 58 & 2 & 3 & 111 \\ 112 & 86 & 121 & 71 & 15 & 122 & 106 & 91 & 5 & 48 \\ 123 & 102 & 78 & 70 & 51 & 91 & 85 & 56 & 47 & 35 \\ 89 & 10 & 48 & 92 & 118 & 84 & 114 & 103 & 12 & 27 \\ 0 & 0 & 0 & 0 & 0 & 125 & 15 & 82 & 118 & 105 \\ 0 & 0 & 0 & 0 & 0 & 76 & 99 & 6 & 9 & 13 \\ 0 & 0 & 0 & 0 & 0 & 15 & 85 & 93 & 15 & 117 \\ 0 & 0 & 0 & 0 & 0 & 83 & 77 & 10 & 107 & 58 \\ 0 & 0 & 0 & 0 & 0 & 36 & 53 & 115 & 96 & 55 \end{bmatrix},$$

y envía al usuario V el mensaje cifrado C.

Para recuperar el mensaje:

(a) El usuario V genera la matriz $T_2 = N_1^{k_2} = \begin{bmatrix} A_1^{k_2} & P \\ \mathbf{0} & B_1^{k_2} \end{bmatrix} =$

$$= \begin{bmatrix} 93 & 5 & 122 & 86 & 107 & 75 & 86 & 125 & 4 & 44 \\ 120 & 15 & 6 & 124 & 1 & 9 & 58 & 44 & 14 & 66 \\ 123 & 15 & 17 & 122 & 10 & 75 & 80 & 103 & 122 & 101 \\ 0 & 21 & 13 & 3 & 12 & 52 & 123 & 12 & 54 & 85 \\ 96 & 9 & 122 & 91 & 110 & 39 & 103 & 105 & 90 & 45 \\ 0 & 0 & 0 & 0 & 0 & 118 & 110 & 119 & 114 & 113 \\ 0 & 0 & 0 & 0 & 0 & 16 & 23 & 24 & 27 & 11 \\ 0 & 0 & 0 & 0 & 0 & 10 & 126 & 28 & 107 & 121 \\ 0 & 0 & 0 & 0 & 0 & 33 & 43 & 37 & 45 & 23 \\ 0 & 0 & 0 & 0 & 0 & 90 & 79 & 90 & 77 & 61 \end{bmatrix}$$

y calcula su inversa

$$T_2^{-1} = \begin{bmatrix} 124 & 114 & 52 & 64 & 60 & 1 & 124 & 3 & 0 & 2 \\ 65 & 60 & 3 & 124 & 51 & 0 & 0 & 0 & 0 & 125 \\ 116 & 67 & 62 & 98 & 31 & 97 & 60 & 0 & 0 & 97 \\ 2 & 12 & 64 & 61 & 3 & 61 & 0 & 0 & 2 & 124 \\ 62 & 22 & 61 & 67 & 61 & 32 & 61 & 112 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 125 & 1 & 125 & 3 & 125 \\ 0 & 0 & 0 & 0 & 0 & 3 & 125 & 1 & 63 & 64 \\ 0 & 0 & 0 & 0 & 0 & 126 & 2 & 28 & 51 & 125 \\ 0 & 0 & 0 & 0 & 0 & 66 & 47 & 98 & 124 & 10 \\ 0 & 0 & 0 & 0 & 0 & 126 & 0 & 126 & 0 & 0 \end{bmatrix}.$$

(b) El usuario V obtiene T_1, efectuando el producto $T_1 = CT_2^{-1} =$

$$= \begin{bmatrix} 93 & 5 & 122 & 86 & 107 & 1 & 2 & 3 & 4 & 5 \\ 120 & 15 & 6 & 124 & 1 & 6 & 7 & 8 & 9 & 10 \\ 123 & 15 & 17 & 122 & 10 & 11 & 12 & 13 & 14 & 15 \\ 0 & 21 & 13 & 3 & 12 & 16 & 17 & 18 & 19 & 20 \\ 96 & 9 & 122 & 91 & 110 & 21 & 22 & 23 & 24 & 25 \\ 0 & 0 & 0 & 0 & 0 & 118 & 110 & 119 & 114 & 113 \\ 0 & 0 & 0 & 0 & 0 & 16 & 23 & 24 & 27 & 11 \\ 0 & 0 & 0 & 0 & 0 & 10 & 126 & 28 & 107 & 121 \\ 0 & 0 & 0 & 0 & 0 & 33 & 43 & 37 & 45 & 23 \\ 0 & 0 & 0 & 0 & 0 & 90 & 79 & 90 & 77 & 61 \end{bmatrix}.$$

(c) V recupera el mensaje μ seleccionando el bloque superior derecha de T_1.

A.1.3 Firma digital.

Se recuerda que las claves públicas de los usuarios U y V son respectivamente N_1 y N_2,se supone que han hecho un intercambio de clave P, y que el usuario U ha hecho llegar a V el mensaje

$$\mu = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix},$$

Si el usuario U desea firmar digitalmente el mensaje μ procede así:

(a) Genera un número aleatorio $r = 890987654321$.

(b) Con $T_2 = N_2^{k_1} =$

$$= \begin{bmatrix} 77 & 21 & 21 & 38 & 6 & 62 & 103 & 66 & 121 & 117 \\ 73 & 62 & 122 & 40 & 34 & 64 & 40 & 105 & 25 & 42 \\ 118 & 88 & 81 & 23 & 58 & 11 & 121 & 43 & 68 & 101 \\ 124 & 61 & 62 & 8 & 98 & 119 & 122 & 79 & 35 & 9 \\ 111 & 95 & 67 & 120 & 91 & 104 & 83 & 86 & 71 & 75 \\ 0 & 0 & 0 & 0 & 0 & 79 & 15 & 6 & 123 & 87 \\ 0 & 0 & 0 & 0 & 0 & 96 & 6 & 50 & 57 & 91 \\ 0 & 0 & 0 & 0 & 0 & 83 & 104 & 38 & 3 & 35 \\ 0 & 0 & 0 & 0 & 0 & 54 & 89 & 47 & 93 & 1 \\ 0 & 0 & 0 & 0 & 0 & 31 & 60 & 118 & 113 & 27 \end{bmatrix},$$

calcula

$$T_2^r = \begin{bmatrix} 33 & 19 & 62 & 77 & 104 & 25 & 107 & 108 & 101 & 103 \\ 48 & 115 & 26 & 104 & 32 & 3 & 37 & 122 & 44 & 115 \\ 22 & 9 & 10 & 73 & 126 & 61 & 6 & 44 & 0 & 14 \\ 46 & 48 & 26 & 46 & 100 & 4 & 83 & 25 & 51 & 114 \\ 48 & 23 & 114 & 95 & 38 & 1 & 29 & 88 & 64 & 114 \\ 0 & 0 & 0 & 0 & 0 & 68 & 33 & 110 & 51 & 65 \\ 0 & 0 & 0 & 0 & 0 & 115 & 121 & 55 & 22 & 63 \\ 0 & 0 & 0 & 0 & 0 & 17 & 18 & 109 & 60 & 27 \\ 0 & 0 & 0 & 0 & 0 & 46 & 89 & 120 & 23 & 17 \\ 0 & 0 & 0 & 0 & 0 & 97 & 50 & 114 & 19 & 117 \end{bmatrix},$$

y $Q = T_1 - T_2^r =$

$$= \begin{bmatrix} 58 & 68 & 35 & 18 & 87 & 103 & 22 & 22 & 30 & 29 \\ 122 & 79 & 62 & 16 & 94 & 3 & 97 & 13 & 92 & 22 \\ 108 & 31 & 51 & 30 & 91 & 77 & 6 & 96 & 14 & 1 \\ 126 & 50 & 45 & 24 & 57 & 12 & 61 & 120 & 95 & 33 \\ 95 & 114 & 54 & 37 & 122 & 20 & 120 & 62 & 87 & 38 \\ 0 & 0 & 0 & 0 & 0 & 23 & 54 & 114 & 44 & 126 \\ 0 & 0 & 0 & 0 & 0 & 55 & 73 & 33 & 98 & 63 \\ 0 & 0 & 0 & 0 & 0 & 113 & 22 & 79 & 43 & 63 \\ 0 & 0 & 0 & 0 & 0 & 126 & 9 & 78 & 47 & 13 \\ 0 & 0 & 0 & 0 & 0 & 46 & 87 & 54 & 113 & 43 \end{bmatrix}.$$

(c) La firma digital es (r, Q).

Si el usuario V desea verificar la firma digital de U, procede así:

(a) Con $T_2 = N_1^{k_2} =$

$$= \begin{bmatrix} 77 & 21 & 21 & 38 & 6 & 62 & 103 & 66 & 121 & 117 \\ 73 & 62 & 122 & 40 & 34 & 64 & 40 & 105 & 25 & 42 \\ 118 & 88 & 81 & 23 & 58 & 11 & 121 & 43 & 68 & 101 \\ 124 & 61 & 62 & 8 & 98 & 119 & 122 & 79 & 35 & 9 \\ 111 & 95 & 67 & 120 & 91 & 104 & 83 & 86 & 71 & 75 \\ 0 & 0 & 0 & 0 & 0 & 79 & 15 & 6 & 123 & 87 \\ 0 & 0 & 0 & 0 & 0 & 96 & 6 & 50 & 57 & 91 \\ 0 & 0 & 0 & 0 & 0 & 83 & 104 & 38 & 3 & 35 \\ 0 & 0 & 0 & 0 & 0 & 54 & 89 & 47 & 93 & 1 \\ 0 & 0 & 0 & 0 & 0 & 31 & 60 & 118 & 113 & 27 \end{bmatrix},$$

calcula

$$T_2^r = \begin{bmatrix} 33 & 19 & 62 & 77 & 104 & 25 & 107 & 108 & 101 & 103 \\ 48 & 115 & 26 & 104 & 32 & 3 & 37 & 122 & 44 & 115 \\ 22 & 9 & 10 & 73 & 126 & 61 & 6 & 44 & 0 & 14 \\ 46 & 48 & 26 & 46 & 100 & 4 & 83 & 25 & 51 & 114 \\ 48 & 23 & 114 & 95 & 38 & 1 & 29 & 88 & 64 & 114 \\ 0 & 0 & 0 & 0 & 0 & 68 & 33 & 110 & 51 & 65 \\ 0 & 0 & 0 & 0 & 0 & 115 & 121 & 55 & 22 & 63 \\ 0 & 0 & 0 & 0 & 0 & 17 & 18 & 109 & 60 & 27 \\ 0 & 0 & 0 & 0 & 0 & 46 & 89 & 120 & 23 & 17 \\ 0 & 0 & 0 & 0 & 0 & 97 & 50 & 114 & 19 & 117 \end{bmatrix},$$

y $R = Q + T_2^r =$

$$= \begin{bmatrix} 91 & 87 & 97 & 95 & 64 & 1 & 2 & 3 & 4 & 5 \\ 43 & 67 & 88 & 120 & 126 & 6 & 7 & 8 & 9 & 10 \\ 3 & 40 & 61 & 103 & 90 & 11 & 12 & 13 & 14 & 15 \\ 45 & 98 & 71 & 70 & 30 & 16 & 17 & 18 & 19 & 20 \\ 16 & 10 & 41 & 5 & 33 & 21 & 22 & 23 & 24 & 25 \\ 0 & 0 & 0 & 0 & 0 & 91 & 87 & 97 & 95 & 64 \\ 0 & 0 & 0 & 0 & 0 & 43 & 67 & 88 & 120 & 126 \\ 0 & 0 & 0 & 0 & 0 & 3 & 40 & 61 & 103 & 90 \\ 0 & 0 & 0 & 0 & 0 & 45 & 98 & 71 & 70 & 30 \\ 0 & 0 & 0 & 0 & 0 & 16 & 10 & 41 & 5 & 33 \end{bmatrix}.$$

(b) V compara el bloque correspondiente de la matriz R con μ, resultando que la firma es auténtica puesto que $\mu = R$.

A.2 Esquema aditivo.

A.2.1 Intercambio de clave.

Sean U y V dos interlocutores que quieren intercambiar una clave de forma secreta, para ello acuerdan $p = 127$ y

$$M = \begin{bmatrix} A & X \\ \mathbf{0} & B \end{bmatrix} \in \Theta \text{ con elementos en } \mathbb{Z}_p$$

$$A = \begin{bmatrix} 93 & 5 & 122 & 86 & 107 \\ 120 & 15 & 6 & 124 & 1 \\ 123 & 15 & 17 & 122 & 10 \\ 0 & 21 & 13 & 3 & 12 \\ 96 & 9 & 122 & 91 & 110 \end{bmatrix},$$

$$B = \begin{bmatrix} 118 & 110 & 119 & 114 & 113 \\ 16 & 23 & 24 & 27 & 11 \\ 10 & 126 & 28 & 107 & 121 \\ 33 & 43 & 37 & 45 & 23 \\ 90 & 79 & 90 & 77 & 61 \end{bmatrix},$$

$$X = \begin{bmatrix} 78 & 112 & 57 & 66 & 110 \\ 96 & 80 & 53 & 115 & 92 \\ 44 & 78 & 61 & 76 & 38 \\ 19 & 113 & 101 & 56 & 39 \\ 69 & 20 & 72 & 80 & 45 \end{bmatrix}.$$

(a) El usuario U genera una clave privada $k_1 = 123456789$, de forma aleatoria y

calcula

$$A^{k_1} = \begin{bmatrix} 102 & 117 & 116 & 67 & 125 \\ 58 & 4 & 80 & 66 & 45 \\ 118 & 110 & 37 & 6 & 74 \\ 52 & 106 & 94 & 51 & 34 \\ 22 & 75 & 2 & 83 & 37 \end{bmatrix},$$

$$B^{k_1} = \begin{bmatrix} 42 & 88 & 0 & 60 & 84 \\ 58 & 36 & 116 & 17 & 25 \\ 74 & 50 & 77 & 120 & 113 \\ 42 & 65 & 15 & 114 & 47 \\ 46 & 84 & 19 & 24 & 114 \end{bmatrix},$$

y

$$X^{(k_1)} = \begin{bmatrix} 29 & 2 & 55 & 52 & 74 \\ 84 & 40 & 83 & 70 & 80 \\ 88 & 7 & 28 & 126 & 90 \\ 120 & 110 & 14 & 105 & 38 \\ 3 & 39 & 59 & 76 & 51 \end{bmatrix}.$$

Es decir, calcula M^{k_1}.

(b) El usuario V genera una clave privada $k_2 = 987654321$, de forma aleatoria y calcula

$$A^{k_2} = \begin{bmatrix} 21 & 20 & 63 & 74 & 25 \\ 44 & 62 & 54 & 124 & 100 \\ 9 & 18 & 29 & 37 & 15 \\ 0 & 109 & 113 & 77 & 43 \\ 29 & 67 & 73 & 37 & 2 \end{bmatrix},$$

$$B^{k_2} = \begin{bmatrix} 13 & 58 & 22 & 51 & 100 \\ 30 & 46 & 51 & 71 & 62 \\ 115 & 1 & 28 & 8 & 13 \\ 98 & 80 & 22 & 25 & 47 \\ 5 & 31 & 8 & 96 & 65 \end{bmatrix},$$

y

$$X^{(k_2)} = \begin{bmatrix} 41 & 26 & 84 & 84 & 56 \\ 84 & 119 & 38 & 17 & 37 \\ 81 & 14 & 14 & 113 & 3 \\ 40 & 17 & 112 & 125 & 54 \\ 67 & 110 & 74 & 77 & 118 \end{bmatrix}.$$

Es decir, calcula M^{k_2}.

(c) Las claves públicas de U y V son respectivamente

$$(X^{(k_1)}, B^{k_1}) \text{ y } (X^{(k_2)}, B^{k_2}).$$

(d) El usuario U calcula

$$X^{(k_1)} \otimes X^{(k_2)} = X^{(k_1+k_2)} = A^{k_1} X^{(k_2)} + X^{(k_1)} B^{k_2}$$

$$= \begin{bmatrix} 80 & 96 & 102 & 60 & 126 \\ 37 & 48 & 112 & 114 & 66 \\ 21 & 62 & 22 & 40 & 67 \\ 40 & 99 & 113 & 126 & 30 \\ 31 & 56 & 68 & 93 & 119 \end{bmatrix}.$$

(e) El usuario V calcula

$$X^{(k_2)} \otimes X^{(k_1)} = X^{(k_2+k_1)} = A^{k_2} X^{(k_1)} + X^{(k_2)} B^{k_1}$$

$$= \begin{bmatrix} 80 & 96 & 102 & 60 & 126 \\ 37 & 48 & 112 & 114 & 66 \\ 21 & 62 & 22 & 40 & 67 \\ 40 & 99 & 113 & 126 & 30 \\ 31 & 56 & 68 & 93 & 119 \end{bmatrix}.$$

La clave compartida por U y V es por tanto

$$P = X^{(k_2+k_1)} = X^{(k_1+k_2)} = \begin{bmatrix} 80 & 96 & 102 & 60 & 126 \\ 37 & 48 & 112 & 114 & 66 \\ 21 & 62 & 22 & 40 & 67 \\ 40 & 99 & 113 & 126 & 30 \\ 31 & 56 & 68 & 93 & 119 \end{bmatrix}.$$

A.2.2 Esquema de cifrado.

Sea

$$\mu = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix} \in G,$$

un mensaje que el usuario U quiere enviar a V. Para ello realiza el producto matricial $AP + \mu B = C$,

$$C = \begin{bmatrix} 121 & 7 & 83 & 90 & 84 \\ 93 & 39 & 18 & 74 & 113 \\ 66 & 23 & 59 & 11 & 114 \\ 22 & 79 & 21 & 125 & 88 \\ 86 & 37 & 62 & 28 & 24 \end{bmatrix}$$

y envía al usuario V el criptograma C.

Para recuperar el mensaje, el usuario V procede así:

(a) El usuario V realiza el producto AP

$$AP = \begin{bmatrix} 121 & 49 & 123 & 52 & 78 \\ 28 & 81 & 92 & 91 & 113 \\ 63 & 65 & 40 & 83 & 120 \\ 81 & 121 & 36 & 125 & 100 \\ 80 & 79 & 111 & 83 & 42 \end{bmatrix}$$

(b) Calcula $C - AP = C'$

$$C' = \begin{bmatrix} 0 & 85 & 87 & 38 & 6 \\ 65 & 85 & 53 & 110 & 0 \\ 3 & 85 & 19 & 55 & 121 \\ 68 & 85 & 112 & 0 & 115 \\ 6 & 85 & 78 & 72 & 109 \end{bmatrix}.$$

(c) Calcula B^{-1}

$$B^{-1} = \begin{bmatrix} 122 & 98 & 97 & 67 & 30 \\ 105 & 95 & 98 & 73 & 114 \\ 9 & 121 & 46 & 57 & 7 \\ 48 & 91 & 70 & 126 & 56 \\ 89 & 49 & 63 & 9 & 4 \end{bmatrix}.$$

(d) Realiza el producto $C'B^{-1}$ recuperando el mensaje μ.

A.2.3 Firma digital.

Se supone que los usuarios U y V han hecho un intercambio de clave P, y que U ha hecho llegar a V el mensaje

$$\mu = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix},$$

según el protocolo anterior. Las claves públicas de U y V son respectivamente

$$(X^{(k_1)}, B^{k_1}) \text{ y } (X^{(k_2)}, B^{k_2}).$$

Si U desea firmar digitalmente el mensaje μ procede así:

(a) Genera un número aleatorio $w = 890987654321$

(b) Calcula

$$H = B^w = \begin{bmatrix} 57 & 70 & 113 & 65 & 105 \\ 82 & 66 & 92 & 22 & 125 \\ 89 & 46 & 50 & 123 & 15 \\ 98 & 115 & 71 & 113 & 3 \\ 85 & 64 & 94 & 38 & 81 \end{bmatrix},$$

$$A^w = \begin{bmatrix} 110 & 55 & 121 & 123 & 108 \\ 17 & 55 & 99 & 30 & 66 \\ 64 & 74 & 37 & 6 & 97 \\ 54 & 104 & 41 & 54 & 12 \\ 63 & 126 & 108 & 69 & 52 \end{bmatrix},$$

$$X^{(w)} = \begin{bmatrix} 9 & 61 & 109 & 25 & 88 \\ 121 & 81 & 123 & 66 & 35 \\ 46 & 9 & 120 & 126 & 113 \\ 33 & 0 & 82 & 27 & 43 \\ 126 & 55 & 50 & 116 & 41 \end{bmatrix},$$

$$A^{k_1+w} = \begin{bmatrix} 122 & 40 & 14 & 39 & 83 \\ 60 & 20 & 33 & 52 & 21 \\ 106 & 80 & 104 & 98 & 81 \\ 19 & 88 & 119 & 0 & 107 \\ 95 & 108 & 34 & 65 & 26 \end{bmatrix}$$

y

$$\begin{aligned} J &= X^{(k_1+w)} \\ &= A^{k_1}X^{(w)} + X^{(k_1)}B^w \\ &= \begin{bmatrix} 84 & 5 & 40 & 15 & 20 \\ 19 & 110 & 107 & 120 & 4 \\ 71 & 101 & 108 & 113 & 5 \\ 38 & 13 & 68 & 116 & 75 \\ 89 & 58 & 84 & 3 & 48 \end{bmatrix}. \end{aligned}$$

(c) U calcula

$$\begin{aligned} X^{((k_1+w)+k_2)} &= A^{k_1+w}X^{(k_2)} + X^{(k_1+w)}B^{k_2} \\ &= \begin{bmatrix} 41 & 16 & 89 & 35 & 87 \\ 72 & 85 & 109 & 19 & 24 \\ 87 & 23 & 121 & 11 & 121 \\ 112 & 57 & 47 & 59 & 66 \\ 111 & 76 & 13 & 4 & 66 \end{bmatrix}, \end{aligned}$$

y

$$T = \mu - X^{((k_1+w)+k_2)} = \begin{bmatrix} 87 & 113 & 41 & 96 & 45 \\ 61 & 49 & 26 & 117 & 113 \\ 51 & 116 & 19 & 3 & 21 \\ 31 & 87 & 98 & 87 & 81 \\ 37 & 73 & 10 & 20 & 86 \end{bmatrix}.$$

(d) La firma digital es la terna (H, J, T).

Si V desea verificar la firma digital de U procede así:

(a) Calcula

$$B^{k_1+w} = B^{k_1}B^w = B^{k_1}H = \begin{bmatrix} 24 & 69 & 106 & 33 & 42 \\ 53 & 87 & 46 & 111 & 55 \\ 87 & 34 & 13 & 88 & 50 \\ 96 & 35 & 112 & 99 & 18 \\ 2 & 9 & 7 & 122 & 29 \end{bmatrix},$$

$$
\begin{aligned}
X^{((k_1+w)+k_2)} &= X^{(k_2+(k_1+w))} \\
&= A^{k_2}X^{(k_1+w)} + X^{(k_2)}B^{k_1+w} \\
&= A^{k_2}J + X^{(k_2)}B^{k_1+w} \\
&= \begin{bmatrix} 41 & 16 & 89 & 35 & 87 \\ 72 & 85 & 109 & 19 & 24 \\ 87 & 23 & 121 & 11 & 121 \\ 112 & 57 & 47 & 59 & 66 \\ 111 & 76 & 13 & 4 & 66 \end{bmatrix},
\end{aligned}
$$

y

$$
\begin{aligned}
R &= T + X^{(k_2+(k_1+w))} \\
&= \begin{bmatrix} 87 & 113 & 41 & 96 & 45 \\ 61 & 49 & 26 & 117 & 113 \\ 51 & 116 & 19 & 3 & 21 \\ 31 & 87 & 98 & 87 & 81 \\ 37 & 73 & 10 & 20 & 86 \end{bmatrix} + \begin{bmatrix} 41 & 16 & 89 & 35 & 87 \\ 72 & 85 & 109 & 19 & 24 \\ 87 & 23 & 121 & 11 & 121 \\ 112 & 57 & 47 & 59 & 66 \\ 111 & 76 & 13 & 4 & 66 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix}.
\end{aligned}
$$

(b) V compara μ y R, resultando que la firma es auténtica puesto que $\mu = R$.

A.3 Diffie-Hellman para matrices triangulares superiores por bloques modificado.

A.3.1 Intercambio de clave.

Sean U y V dos interlocutores que quieren intercambiar una clave de forma secreta, para ello siguen el protocolo:

Acuerdan $p = 127$ y

$$M = \begin{bmatrix} A & X \\ \mathbf{0} & B \end{bmatrix} \in \Theta \text{ con elementos en } \mathbb{Z}_p$$

$$A = \begin{bmatrix} 93 & 5 & 122 & 86 & 107 \\ 120 & 15 & 6 & 124 & 1 \\ 123 & 15 & 17 & 122 & 10 \\ 0 & 21 & 13 & 3 & 12 \\ 96 & 9 & 122 & 91 & 110 \end{bmatrix},$$

$$B = \begin{bmatrix} 118 & 110 & 119 & 114 & 113 \\ 16 & 23 & 24 & 27 & 11 \\ 10 & 126 & 28 & 107 & 121 \\ 33 & 43 & 37 & 45 & 23 \\ 90 & 79 & 90 & 77 & 61 \end{bmatrix},$$

$$X = \begin{bmatrix} 112 & 105 & 100 & 29 & 25 \\ 25 & 58 & 106 & 23 & 50 \\ 84 & 8 & 16 & 86 & 112 \\ 27 & 72 & 22 & 86 & 4 \\ 46 & 94 & 37 & 45 & 60 \end{bmatrix}.$$

(a) El usuario U genera una clave privada $k_1 = 123456789$, de forma aleatoria y calcula

$$A^{k_1} = \begin{bmatrix} 102 & 117 & 116 & 67 & 125 \\ 58 & 4 & 80 & 66 & 45 \\ 118 & 110 & 37 & 6 & 74 \\ 52 & 106 & 94 & 51 & 34 \\ 22 & 75 & 2 & 83 & 37 \end{bmatrix},$$

$$B^{k_1} = \begin{bmatrix} 42 & 88 & 0 & 60 & 84 \\ 58 & 36 & 116 & 17 & 25 \\ 74 & 50 & 77 & 120 & 113 \\ 42 & 65 & 15 & 114 & 47 \\ 46 & 84 & 19 & 24 & 114 \end{bmatrix},$$

y

$$X^{(k_1)} = \begin{bmatrix} 105 & 32 & 63 & 70 & 116 \\ 58 & 101 & 67 & 82 & 25 \\ 38 & 3 & 48 & 0 & 54 \\ 125 & 110 & 121 & 67 & 57 \\ 121 & 103 & 19 & 115 & 64 \end{bmatrix} = Y .$$

Es decir, calcula M^{k_1}.

(b) El usuario V genera una clave privada $k_2 = 987654321$, de forma aleatoria y calcula

$$A^{k_2} = \begin{bmatrix} 21 & 20 & 63 & 74 & 25 \\ 44 & 62 & 54 & 124 & 100 \\ 9 & 18 & 29 & 37 & 15 \\ 0 & 109 & 113 & 77 & 43 \\ 29 & 67 & 73 & 37 & 2 \end{bmatrix},$$

$$B^{k_2} = \begin{bmatrix} 13 & 58 & 22 & 51 & 100 \\ 30 & 46 & 51 & 71 & 62 \\ 115 & 1 & 28 & 8 & 13 \\ 98 & 80 & 22 & 25 & 47 \\ 5 & 31 & 8 & 96 & 65 \end{bmatrix},$$

y

$$X^{(k_2)} = \begin{bmatrix} 97 & 83 & 98 & 45 & 116 \\ 8 & 126 & 105 & 50 & 3 \\ 16 & 105 & 21 & 61 & 11 \\ 24 & 33 & 70 & 26 & 69 \\ 32 & 87 & 63 & 7 & 77 \end{bmatrix} = Z\ .$$

Es decir, calcula M^{k_2}.

(c) Las claves públicas de U y V son respectivamente Y y Z.

(d) El usuario U calcula

$$Z^{(k_1)} = \begin{bmatrix} 99 & 62 & 46 & 117 & 65 \\ 17 & 27 & 72 & 16 & 89 \\ 62 & 4 & 72 & 88 & 111 \\ 17 & 13 & 63 & 103 & 14 \\ 59 & 126 & 82 & 46 & 37 \end{bmatrix}.$$

(e) El usuario V calcula

$$Y^{(k_2)} = \begin{bmatrix} 99 & 62 & 46 & 117 & 65 \\ 17 & 27 & 72 & 16 & 89 \\ 62 & 4 & 72 & 88 & 111 \\ 17 & 13 & 63 & 103 & 14 \\ 59 & 126 & 82 & 46 & 37 \end{bmatrix}.$$

La clave compartida por U y V es por tanto

$$P = Z^{(k_1)} = Y^{(k_2)}.$$

A.3.2 Esquema de cifrado.

Sea

$$\mu = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix} \in G,$$

un mensaje codificado que el usuario U quiere enviar a V, para ello, sigue el protocolo:

(a) Construye las matrices

$$T_1 = \begin{bmatrix} A & \mu \\ \mathbf{0} & B \end{bmatrix} =$$

$$= \begin{bmatrix} 93 & 5 & 122 & 86 & 107 & 1 & 2 & 3 & 4 & 5 \\ 120 & 15 & 6 & 124 & 1 & 6 & 7 & 8 & 9 & 10 \\ 123 & 15 & 17 & 122 & 10 & 11 & 12 & 13 & 14 & 15 \\ 0 & 21 & 13 & 3 & 12 & 16 & 17 & 18 & 19 & 20 \\ 96 & 9 & 122 & 91 & 110 & 21 & 22 & 23 & 24 & 25 \\ 0 & 0 & 0 & 0 & 0 & 118 & 110 & 119 & 114 & 113 \\ 0 & 0 & 0 & 0 & 0 & 16 & 23 & 24 & 27 & 11 \\ 0 & 0 & 0 & 0 & 0 & 10 & 126 & 28 & 107 & 121 \\ 0 & 0 & 0 & 0 & 0 & 33 & 43 & 37 & 45 & 23 \\ 0 & 0 & 0 & 0 & 0 & 90 & 79 & 90 & 77 & 61 \end{bmatrix}$$

y

$$T_2 = \begin{bmatrix} A & P \\ \mathbf{0} & B \end{bmatrix} =$$

$$= \begin{bmatrix} 93 & 5 & 122 & 86 & 107 & 75 & 86 & 125 & 4 & 44 \\ 120 & 15 & 6 & 124 & 1 & 9 & 58 & 44 & 14 & 66 \\ 123 & 15 & 17 & 122 & 10 & 75 & 80 & 103 & 122 & 101 \\ 0 & 21 & 13 & 3 & 12 & 52 & 123 & 12 & 54 & 85 \\ 96 & 9 & 122 & 91 & 110 & 39 & 103 & 105 & 90 & 45 \\ 0 & 0 & 0 & 0 & 0 & 118 & 110 & 119 & 114 & 113 \\ 0 & 0 & 0 & 0 & 0 & 16 & 23 & 24 & 27 & 11 \\ 0 & 0 & 0 & 0 & 0 & 10 & 126 & 28 & 107 & 121 \\ 0 & 0 & 0 & 0 & 0 & 33 & 43 & 37 & 45 & 23 \\ 0 & 0 & 0 & 0 & 0 & 90 & 79 & 90 & 77 & 61 \end{bmatrix},$$

matriz invertible por serlo A y B.

(b) El usuario U calcula la matriz $C = T_1T_2 =$

$$= \begin{bmatrix} 110 & 59 & 63 & 96 & 102 & 50 & 108 & 62 & 47 & 46 \\ 78 & 99 & 56 & 40 & 35 & 8 & 59 & 17 & 63 & 62 \\ 34 & 64 & 30 & 40 & 35 & 100 & 100 & 126 & 32 & 10 \\ 64 & 46 & 72 & 84 & 110 & 78 & 11 & 115 & 74 & 53 \\ 14 & 12 & 122 & 122 & 55 & 59 & 125 & 90 & 54 & 89 \\ 0 & 0 & 0 & 0 & 0 & 72 & 10 & 112 & 60 & 104 \\ 0 & 0 & 0 & 0 & 0 & 59 & 104 & 37 & 90 & 34 \\ 0 & 0 & 0 & 0 & 0 & 117 & 96 & 35 & 80 & 125 \\ 0 & 0 & 0 & 0 & 0 & 125 & 79 & 78 & 105 & 68 \\ 0 & 0 & 0 & 0 & 0 & 114 & 72 & 97 & 86 & 116 \end{bmatrix},$$

y envía a V el mensaje cifrado C.

Para recuperar el mensaje, el usuario V realiza:

(a) Genera la matriz

$$T_2 = \begin{bmatrix} A & P \\ \mathbf{0} & B \end{bmatrix} =$$

$$= \begin{bmatrix} 93 & 5 & 122 & 86 & 107 & 75 & 86 & 125 & 4 & 44 \\ 120 & 15 & 6 & 124 & 1 & 9 & 58 & 44 & 14 & 66 \\ 123 & 15 & 17 & 122 & 10 & 75 & 80 & 103 & 122 & 101 \\ 0 & 21 & 13 & 3 & 12 & 52 & 123 & 12 & 54 & 85 \\ 96 & 9 & 122 & 91 & 110 & 39 & 103 & 105 & 90 & 45 \\ 0 & 0 & 0 & 0 & 0 & 118 & 110 & 119 & 114 & 113 \\ 0 & 0 & 0 & 0 & 0 & 16 & 23 & 24 & 27 & 11 \\ 0 & 0 & 0 & 0 & 0 & 10 & 126 & 28 & 107 & 121 \\ 0 & 0 & 0 & 0 & 0 & 33 & 43 & 37 & 45 & 23 \\ 0 & 0 & 0 & 0 & 0 & 90 & 79 & 90 & 77 & 61 \end{bmatrix}$$

y calcula su inversa

$$T_2^{-1} = \begin{bmatrix} 14 & 124 & 23 & 20 & 11 & 117 & 48 & 91 & 63 & 12 \\ 12 & 22 & 26 & 48 & 126 & 18 & 11 & 52 & 29 & 117 \\ 4 & 122 & 38 & 97 & 41 & 65 & 71 & 86 & 111 & 126 \\ 116 & 72 & 13 & 104 & 31 & 13 & 117 & 105 & 29 & 103 \\ 115 & 23 & 90 & 39 & 66 & 38 & 1 & 34 & 11 & 58 \\ 0 & 0 & 0 & 0 & 0 & 122 & 98 & 97 & 67 & 30 \\ 0 & 0 & 0 & 0 & 0 & 105 & 95 & 98 & 73 & 114 \\ 0 & 0 & 0 & 0 & 0 & 9 & 121 & 46 & 57 & 7 \\ 0 & 0 & 0 & 0 & 0 & 48 & 91 & 70 & 126 & 56 \\ 0 & 0 & 0 & 0 & 0 & 89 & 49 & 63 & 9 & 4 \end{bmatrix}.$$

(b) Obtiene T_1, efectuando el producto $T_1 = CT_2^{-1} =$

$$= \begin{bmatrix} 93 & 5 & 122 & 86 & 107 & 1 & 2 & 3 & 4 & 5 \\ 120 & 15 & 6 & 124 & 1 & 6 & 7 & 8 & 9 & 10 \\ 123 & 15 & 17 & 122 & 10 & 11 & 12 & 13 & 14 & 15 \\ 0 & 21 & 13 & 3 & 12 & 16 & 17 & 18 & 19 & 20 \\ 96 & 9 & 122 & 91 & 110 & 21 & 22 & 23 & 24 & 25 \\ 0 & 0 & 0 & 0 & 0 & 118 & 110 & 119 & 114 & 113 \\ 0 & 0 & 0 & 0 & 0 & 16 & 23 & 24 & 27 & 11 \\ 0 & 0 & 0 & 0 & 0 & 10 & 126 & 28 & 107 & 121 \\ 0 & 0 & 0 & 0 & 0 & 33 & 43 & 37 & 45 & 23 \\ 0 & 0 & 0 & 0 & 0 & 90 & 79 & 90 & 77 & 61 \end{bmatrix}.$$

(c) Recupera el mensaje μ seleccionando el correspondiente bloque de T_1.

A.3.3 Firma digital.

Se supone que U y V han intercambiado la clave P, y que U ha hecho llegar a V el mensaje

$$\mu = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix}.$$

Si el usuario U desea firmar digitalmente el mensaje μ, procede así:

(a) Genera un número aleatorio $r = 890987654321$.

(b) Calcula

$$P^{(r)} = \begin{bmatrix} 31 & 60 & 7 & 90 & 100 \\ 31 & 37 & 67 & 111 & 35 \\ 90 & 114 & 124 & 108 & 109 \\ 59 & 124 & 6 & 110 & 9 \\ 125 & 59 & 19 & 41 & 110 \end{bmatrix},$$

y

$$Q = \mu - P^{(r)} = \begin{bmatrix} 97 & 69 & 123 & 41 & 32 \\ 102 & 97 & 68 & 25 & 102 \\ 48 & 25 & 16 & 33 & 33 \\ 84 & 20 & 12 & 36 & 11 \\ 23 & 90 & 4 & 110 & 42 \end{bmatrix},$$

(c) La firma digital es (r, Q).

Si el usuario V desea verificar la firma digital de U, procede así:

(a) Calcula

$$P^{(r)} = \begin{bmatrix} 31 & 60 & 7 & 90 & 100 \\ 31 & 37 & 67 & 111 & 35 \\ 90 & 114 & 124 & 108 & 109 \\ 59 & 124 & 6 & 110 & 9 \\ 125 & 59 & 19 & 41 & 110 \end{bmatrix},$$

y

$$R = Q + P^{(r)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix},$$

(b) El usuario V compara μ y R, resultando que la firma es auténtica puesto que $\mu = R$.

A.4 Esquema multiplicativo.

A.4.1 Intercambio de clave.

Si dos interlocutores U y V desean intercambiar una clave, siguen el protocolo:

(a) Acuerdan valores para

$$p = 127,$$

$$M_1 = \begin{bmatrix} A_1 & X_1 \\ \mathbf{0} & B_1 \end{bmatrix} \in \Theta \text{ con elementos en } \mathbb{Z}_p$$

$$M_1 = \begin{bmatrix} 93 & 5 & 122 & 86 & 107 & 48 & 108 & 108 & 77 & 57 \\ 120 & 15 & 6 & 124 & 1 & 87 & 68 & 24 & 2 & 101 \\ 123 & 15 & 17 & 122 & 10 & 25 & 19 & 3 & 23 & 83 \\ 0 & 21 & 13 & 3 & 12 & 100 & 116 & 18 & 33 & 16 \\ 96 & 9 & 122 & 91 & 110 & 87 & 16 & 50 & 122 & 120 \\ 0 & 0 & 0 & 0 & 0 & 118 & 110 & 119 & 114 & 113 \\ 0 & 0 & 0 & 0 & 0 & 16 & 23 & 24 & 27 & 11 \\ 0 & 0 & 0 & 0 & 0 & 10 & 126 & 28 & 107 & 121 \\ 0 & 0 & 0 & 0 & 0 & 33 & 43 & 37 & 45 & 23 \\ 0 & 0 & 0 & 0 & 0 & 90 & 79 & 90 & 77 & 61 \end{bmatrix}$$

con orden m_1 y

$$M_2 = \begin{bmatrix} A_2 & X_2 \\ \mathbf{0} & B_2 \end{bmatrix} \in \Theta$$

$$M_2 = \begin{bmatrix} 99 & 112 & 94 & 84 & 95 & 29 & 44 & 49 & 86 & 73 \\ 123 & 5 & 119 & 123 & 122 & 21 & 21 & 89 & 2 & 115 \\ 126 & 5 & 3 & 121 & 4 & 46 & 48 & 87 & 37 & 109 \\ 3 & 11 & 126 & 2 & 6 & 93 & 9 & 47 & 30 & 65 \\ 102 & 116 & 94 & 89 & 98 & 22 & 116 & 93 & 117 & 44 \\ 0 & 0 & 0 & 0 & 0 & 92 & 81 & 97 & 85 & 102 \\ 0 & 0 & 0 & 0 & 0 & 16 & 23 & 24 & 27 & 11 \\ 0 & 0 & 0 & 0 & 0 & 33 & 10 & 67 & 118 & 77 \\ 0 & 0 & 0 & 0 & 0 & 33 & 43 & 37 & 45 & 23 \\ 0 & 0 & 0 & 0 & 0 & 38 & 21 & 46 & 19 & 39 \end{bmatrix}$$

con orden m_2.

(b) El usuario U genera dos números aleatorios

$$r = 11119999,\ s = 99990000$$

y calcula

$$A_{rs} = \begin{bmatrix} 57 & 6 & 112 & 66 & 99 \\ 12 & 12 & 96 & 111 & 100 \\ 99 & 75 & 48 & 17 & 92 \\ 75 & 83 & 63 & 62 & 52 \\ 79 & 110 & 93 & 8 & 30 \end{bmatrix},$$

$$B_{rs} = \begin{bmatrix} 60 & 45 & 35 & 69 & 41 \\ 13 & 50 & 53 & 103 & 30 \\ 40 & 44 & 56 & 2 & 57 \\ 40 & 18 & 71 & 97 & 98 \\ 60 & 21 & 68 & 107 & 72 \end{bmatrix},$$

y

$$C_{rs} = \begin{bmatrix} 108 & 2 & 106 & 95 & 52 \\ 21 & 91 & 10 & 28 & 33 \\ 22 & 111 & 56 & 44 & 93 \\ 125 & 55 & 33 & 63 & 49 \\ 6 & 111 & 75 & 75 & 44 \end{bmatrix}.$$

Compone la matriz

$$C = \begin{bmatrix} A_{rs} & C_{rs} \\ \mathbf{0} & B_{rs} \end{bmatrix}$$

y se la envía al usuario V.

(c) Igualmente, el usuario V elige de forma aleatoria dos números naturales

$$v = 11113333,\ w = 99998888$$

y calcula

$$A_{vw} = \begin{bmatrix} 17 & 20 & 74 & 19 & 64 \\ 9 & 98 & 68 & 85 & 76 \\ 19 & 26 & 13 & 37 & 17 \\ 116 & 51 & 56 & 100 & 76 \\ 0 & 43 & 93 & 112 & 95 \end{bmatrix},$$

$$B_{vw} = \begin{bmatrix} 18 & 54 & 16 & 56 & 118 \\ 120 & 11 & 120 & 53 & 91 \\ 28 & 55 & 82 & 0 & 126 \\ 17 & 118 & 96 & 47 & 107 \\ 55 & 18 & 25 & 90 & 107 \end{bmatrix},$$

y

$$C_{vw} = \begin{bmatrix} 125 & 126 & 0 & 45 & 36 \\ 70 & 93 & 125 & 34 & 54 \\ 42 & 32 & 95 & 4 & 24 \\ 52 & 72 & 125 & 120 & 104 \\ 69 & 21 & 121 & 5 & 115 \end{bmatrix}.$$

Compone la matriz

$$D = \begin{bmatrix} A_{vw} & C_{vw} \\ \mathbf{0} & B_{vw} \end{bmatrix}$$

y se la envía al usuario U.

(d) El usuario U calcula

$$K_u = A_1^r A_{vw} X_2^{(s)} + A_1^r C_{vw} B_2^s + X_1^{(r)} B_{vw} B_2^s =$$

$$= \begin{bmatrix} 7 & 119 & 26 & 38 & 50 \\ 122 & 44 & 110 & 86 & 81 \\ 97 & 0 & 21 & 112 & 120 \\ 7 & 60 & 15 & 118 & 68 \\ 97 & 40 & 78 & 85 & 4 \end{bmatrix}.$$

(e) El usuario V calcula

$$K_v = A_1^v A_{rs} X_2^{(w)} + A_1^v C_{rs} B_2^w + X_1^{(v)} B_{rs} B_2^w =$$

$$= \begin{bmatrix} 7 & 119 & 26 & 38 & 50 \\ 122 & 44 & 110 & 86 & 81 \\ 97 & 0 & 21 & 112 & 120 \\ 7 & 60 & 15 & 118 & 68 \\ 97 & 40 & 78 & 85 & 4 \end{bmatrix}.$$

El secreto compartido es $k_u = K_v = P$.

Los valores públicos son p, M_1, M_2, C y D y las claves privadas son r, s, v y w.

A.4.2 Esquema de cifrado.

Sea

$$\mu = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix}$$

un mensaje codificado que el usuario U quiere enviar a V, para ello sigue el protocolo:

(a) Construye las matrices

$$T_1 = \begin{bmatrix} A_1^r & \mu \\ \mathbf{0} & B_1^r \end{bmatrix} =$$

$$= \begin{bmatrix} 102 & 82 & 43 & 70 & 90 & 1 & 2 & 3 & 4 & 5 \\ 9 & 103 & 107 & 125 & 63 & 6 & 7 & 8 & 9 & 10 \\ 15 & 92 & 8 & 20 & 42 & 11 & 12 & 13 & 14 & 15 \\ 125 & 90 & 71 & 107 & 7 & 16 & 17 & 18 & 19 & 20 \\ 0 & 6 & 71 & 111 & 25 & 21 & 22 & 23 & 24 & 25 \\ 0 & 0 & 0 & 0 & 0 & 47 & 111 & 73 & 123 & 34 \\ 0 & 0 & 0 & 0 & 0 & 46 & 83 & 125 & 80 & 33 \\ 0 & 0 & 0 & 0 & 0 & 3 & 104 & 118 & 43 & 3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 35 & 0 & 73 & 68 \\ 0 & 0 & 0 & 0 & 0 & 24 & 38 & 42 & 54 & 22 \end{bmatrix}$$

y

$$M_u = M_1^r D M_2^s =$$

$$= \begin{bmatrix} 7 & 40 & 47 & 103 & 5 & 11 & 40 & 107 & 53 & 69 \\ 3 & 39 & 78 & 11 & 79 & 13 & 74 & 94 & 125 & 22 \\ 95 & 29 & 46 & 66 & 33 & 87 & 62 & 48 & 79 & 8 \\ 10 & 124 & 117 & 119 & 76 & 62 & 118 & 72 & 42 & 116 \\ 19 & 121 & 4 & 72 & 0 & 125 & 83 & 61 & 76 & 113 \\ 0 & 0 & 0 & 0 & 0 & 98 & 58 & 71 & 101 & 8 \\ 0 & 0 & 0 & 0 & 0 & 83 & 17 & 3 & 12 & 5 \\ 0 & 0 & 0 & 0 & 0 & 25 & 51 & 35 & 106 & 59 \\ 0 & 0 & 0 & 0 & 0 & 49 & 32 & 19 & 5 & 42 \\ 0 & 0 & 0 & 0 & 0 & 113 & 92 & 37 & 75 & 103 \end{bmatrix}.$$

(b) Calcula el producto $H = T_1 M_u =$

$$= \begin{bmatrix} 89 & 28 & 1 & 100 & 11 & 13 & 40 & 36 & 80 & 33 \\ 30 & 123 & 62 & 85 & 4 & 102 & 102 & 47 & 13 & 20 \\ 107 & 44 & 89 & 107 & 110 & 119 & 65 & 95 & 86 & 29 \\ 76 & 46 & 6 & 38 & 49 & 118 & 70 & 97 & 51 & 61 \\ 93 & 32 & 57 & 76 & 77 & 84 & 71 & 97 & 54 & 29 \\ 0 & 0 & 0 & 0 & 0 & 113 & 33 & 41 & 91 & 63 \\ 0 & 0 & 0 & 0 & 0 & 73 & 48 & 90 & 50 & 58 \\ 0 & 0 & 0 & 0 & 0 & 98 & 87 & 122 & 21 & 96 \\ 0 & 0 & 0 & 0 & 0 & 40 & 101 & 15 & 17 & 93 \\ 0 & 0 & 0 & 0 & 0 & 4 & 58 & 48 & 108 & 28 \end{bmatrix},$$

y envía al usuario V el mensaje cifrado H.

Para recuperar el mensaje, el usuario V procede así:

(a) Construye la matriz $M_v = M_1^v C M_2^w =$

$$= \begin{bmatrix} 7 & 40 & 47 & 103 & 5 & 11 & 40 & 107 & 53 & 69 \\ 3 & 39 & 78 & 11 & 79 & 13 & 74 & 94 & 125 & 22 \\ 95 & 29 & 46 & 66 & 33 & 87 & 62 & 48 & 79 & 8 \\ 10 & 124 & 117 & 119 & 76 & 62 & 118 & 72 & 42 & 116 \\ 19 & 121 & 4 & 72 & 0 & 125 & 83 & 61 & 76 & 113 \\ 0 & 0 & 0 & 0 & 0 & 98 & 58 & 71 & 101 & 8 \\ 0 & 0 & 0 & 0 & 0 & 83 & 17 & 3 & 12 & 5 \\ 0 & 0 & 0 & 0 & 0 & 25 & 51 & 35 & 106 & 59 \\ 0 & 0 & 0 & 0 & 0 & 49 & 32 & 19 & 5 & 42 \\ 0 & 0 & 0 & 0 & 0 & 113 & 92 & 37 & 75 & 103 \end{bmatrix}.$$

(b) Obtiene la matriz inversa $M_v^{-1} =$

$$= \begin{bmatrix} 103 & 12 & 121 & 22 & 80 & 54 & 18 & 30 & 73 & 122 \\ 90 & 70 & 23 & 112 & 124 & 59 & 86 & 76 & 110 & 48 \\ 8 & 35 & 107 & 31 & 111 & 19 & 52 & 111 & 33 & 31 \\ 126 & 93 & 72 & 20 & 94 & 2 & 87 & 56 & 63 & 113 \\ 97 & 39 & 72 & 15 & 120 & 62 & 79 & 62 & 33 & 31 \\ 0 & 0 & 0 & 0 & 0 & 113 & 105 & 50 & 61 & 8 \\ 0 & 0 & 0 & 0 & 0 & 33 & 43 & 70 & 16 & 3 \\ 0 & 0 & 0 & 0 & 0 & 112 & 76 & 50 & 70 & 56 \\ 0 & 0 & 0 & 0 & 0 & 85 & 42 & 52 & 43 & 24 \\ 0 & 0 & 0 & 0 & 0 & 97 & 49 & 71 & 86 & 13 \end{bmatrix}.$$

(c) Calcula el producto

$$HM_v^{-1} = T_1 =$$

$$= \begin{bmatrix} 102 & 82 & 43 & 70 & 90 & 1 & 2 & 3 & 4 & 5 \\ 9 & 103 & 107 & 125 & 63 & 6 & 7 & 8 & 9 & 10 \\ 15 & 92 & 8 & 20 & 42 & 11 & 12 & 13 & 14 & 15 \\ 125 & 90 & 71 & 107 & 7 & 16 & 17 & 18 & 19 & 20 \\ 0 & 6 & 71 & 111 & 25 & 21 & 22 & 23 & 24 & 25 \\ 0 & 0 & 0 & 0 & 0 & 47 & 111 & 73 & 123 & 34 \\ 0 & 0 & 0 & 0 & 0 & 46 & 83 & 125 & 80 & 33 \\ 0 & 0 & 0 & 0 & 0 & 3 & 104 & 118 & 43 & 3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 35 & 0 & 73 & 68 \\ 0 & 0 & 0 & 0 & 0 & 24 & 38 & 42 & 54 & 22 \end{bmatrix}.$$

(d) Recupera el mensaje μ seleccionando el bloque correspondiente de T_1.

A.4.3 Firma digital.

Se supone que los usuarios U y V han intercambiado la clave P, y que U ha hecho llegar a V el mensaje

$$\mu = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix}.$$

Si el usuario U desea firmar digitalmente el mensaje μ, procede así:

(a) Genera un número aleatorio $r = 11118888$.

(b) Calcula

$$P^{(r)} = \begin{bmatrix} 15 & 52 & 31 & 71 & 101 \\ 84 & 87 & 63 & 30 & 56 \\ 59 & 64 & 120 & 99 & 38 \\ 68 & 98 & 92 & 125 & 48 \\ 1 & 65 & 71 & 66 & 103 \end{bmatrix},$$

y

$$Q = \mu - P^{(r)} = \begin{bmatrix} 113 & 77 & 99 & 60 & 31 \\ 49 & 47 & 72 & 106 & 81 \\ 79 & 75 & 20 & 42 & 104 \\ 75 & 46 & 53 & 21 & 99 \\ 20 & 84 & 79 & 85 & 49 \end{bmatrix},$$

(c) La firma digital es (r, Q).

Si el usuario V desea verificar la firma digital de U, procede así:

(a) Calcula

$$P^{(r)} = \begin{bmatrix} 15 & 52 & 31 & 71 & 101 \\ 84 & 87 & 63 & 30 & 56 \\ 59 & 64 & 120 & 99 & 38 \\ 68 & 98 & 92 & 125 & 48 \\ 1 & 65 & 71 & 66 & 103 \end{bmatrix},$$

y

$$R = Q + P^{(r)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix},$$

(b) Compara R con μ, resultando que la firma es auténtica puesto que $\mu = R$.

Apéndice B

Número de bits del mcd y mcm

Como se comentó en el capítulo 3 sección 3.3.5, en los criptosistemas presentados, el orden del grupo es $mcm(p^r-1, p^s-1)$ y hay que intentar minimizar el $mcd(p^r-1, p^s-1)$. Las tablas que se incluyen a continuación, muestran valores de estos mcm y mcd para distintos números primos r (tamaño del bloque A) y s (tamaño del bloque B). Para la realización de las pruebas, se ha adecuado el tamaño de p para obtener un tamaño fijo de clave cuya representación binaria es del orden de 1024 bits.

Las pruebas se han realizado en MATLAB ver. 7.0.1.24704 (R14) SP 1., con una representación binaria del orden de M aproximado de 1024 bits, con las mismas condiciones iniciales y empleando un microprocesador Intel Pentium 4 con una frecuencia de reloj de 3.06 GHz, 512KB de caché, 400 MHz de front-side-bus y 1024 MB de RAM.

r	s	mcm (bits)	mcd (bits)	p
2	37	1024	55	212900057
2	59	1024	36	159936
2	73	1024	28	16691
2	79	1024	26	7963
2	83	1024	25	5167
2	89	1024	23	2902
2	97	1024	20	2437
2	101	1024	20	1123
2	103	1024	20	983
2	107	1024	19	757
2	109	1024	19	673
2	113	1024	18	523
2	127	1024	16	263
2	131	1024	16	223
2	137	1024	15	173
2	139	1024	15	163
2	149	1024	14	113
2	153	1024	13	103
2	157	1024	13	97
2	163	1024	12	79
2	167	1024	12	73
2	173	1024	12	61
2	179	1024	11	53
2	181	1024	11	53
2	191	1024	9	43
2	197	1024	8	37

Tabla B.1: Número de bits para r=2

r	s	mcm (bits)	mcd (bits)	p
3	37	1024	83	210181297
3	59	1024	52	166909
3	73	1024	42	16631
3	83	1024	37	5101
3	89	1024	35	2903
3	97	1024	32	1459
3	101	1024	30	1117
3	107	1024	29	757
3	113	1024	27	523
3	127	1024	24	263
3	131	1024	23	223
3	139	1024	22	162

Tabla B.2: Número de bits para r=3

r	s	mcm (bits)	mcd (bits)	p
5	59	1024	86	166909
5	73	1024	64	16631
5	83	1024	57	4073
5	89	1024	53	2897
5	97	1024	37	1499
5	101	1024	51	1123
5	107	1024	48	757
5	113	1024	45	509
5	127	1024	40	263
5	131	1024	39	223
5	139	1024	35	167

Tabla B.3: Número de bits para r=5

r	s	mcm (bits)	mcd (bits)	p
7	59	1024	121	166013
7	73	1024	90	16603
7	83	1024	83	5023
7	89	1024	74	2903
7	97	1024	51	1499
7	101	1024	71	1123
7	107	1024	67	753
7	113	1024	63	509
7	127	1024	56	263
7	131	1024	55	223
7	139	1024	50	167

Tabla B.4: Número de bits para r=7

r	s	mcm (bits)	mcd (bits)	p
11	59	1024	190	166909
11	73	1024	154	16603
11	83	1024	135	5167
11	89	1024	126	2903
11	97	1024	116	1498
11	101	1024	111	1123
11	107	1024	105	757
11	113	1017	99	509
11	127	1021	88	263
11	131	1022	86	223
11	139	1024	81	163

Tabla B.5: Número de bits para r=11

r	s	mcm (bits)	mcd (bits)	p
13	59	1024	225	166909
13	73	1024	182	16603
13	83	1024	160	5167
13	89	1024	150	2903
13	97	1024	137	1498
13	101	1024	131	1123
13	107	1024	124	757
13	113	1017	117	509
13	127	1021	104	263
13	131	1022	101	223
13	139	1024	96	163

Tabla B.6: Número de bits para r=13

r	s	mcm (bits)	mcd (bits)	p
17	59	1024	295	166909
17	73	1024	238	16603
17	83	1024	209	5167
17	89	1024	196	2903
17	97	1024	179	1498
17	101	1024	172	1123
17	107	1024	163	757
17	113	1017	153	509
17	127	1021	137	263
17	131	1022	133	223
17	139	1024	125	163

Tabla B.7: Número de bits para r=17

r	s	mcm (bits)	mcd (bits)	p
19	59	1024	330	166909
19	73	1024	266	16603
19	83	1024	234	5167
19	89	1024	225	2903
19	97	1024	200	1498
19	101	1024	193	1123
19	107	1024	182	757
19	113	1017	171	509
19	127	1021	157	263
19	131	1022	148	223
19	139	1024	140	163

Tabla B.8: Número de bits para r=19

r	s	mcm (bits)	mcd (bits)	p
23	59	1024	400	166909
23	73	1024	322	16603
23	83	1024	281	5167
23	89	1024	265	2903
23	97	1024	243	1498
23	101	1024	233	1123
23	107	1024	220	757
23	113	1017	207	509
23	127	1021	185	263
23	131	1022	179	223
23	139	1024	169	163

Tabla B.9: Número de bits para r=23

r	s	mcm (bits)	mcd (bits)	p
29	59	1024	503	166909
29	73	1024	407	16603
29	83	1024	358	5167
29	89	1024	334	2903
29	97	1024	306	1498
29	101	1024	294	1123
29	107	1024	277	757
29	113	1017	261	509
29	127	1021	233	263
29	131	1022	226	223
29	139	1024	213	163

Tabla B.10: Número de bits para r=29

r	s	mcm (bits)	mcd (bits)	p
31	59	1024	538	166909
31	73	1024	435	16603
31	83	1024	382	5167
31	89	1024	357	2903
31	97	1024	327	1498
31	101	1024	314	1123
31	107	1024	296	757
31	113	1017	279	509
31	127	1021	250	263
31	131	1022	242	223
31	139	1024	228	163

Tabla B.11: Número de bits para r=31

r	s	mcm (bits)	mcd (bits)	p
37	59	1024	641	166909
37	73	1024	519	16603
37	83	1024	456	5167
37	89	1024	426	2903
37	97	1024	390	1498
37	101	1024	374	1123
37	107	1024	354	757
37	113	1017	333	509
37	127	1021	297	263
37	131	1022	288	223
37	139	1024	272	163

Tabla B.12: Número de bits para r=37

r	s	mcm (bits)	mcd (bits)	p
41	59	1024	711	166909
41	73	1024	557	16603
41	83	1024	506	5167
41	89	1024	472	2903
41	97	1024	433	1498
41	101	1024	436	1123
41	107	1024	392	757
41	113	1017	367	509
41	127	1021	330	263
41	131	1022	320	223
41	139	1024	301	163

Tabla B.13: Número de bits para r=41

r	s	mcm (bits)	mcd (bits)	p
43	59	1024	746	166909
43	73	1024	603	16603
43	83	1024	530	5167
43	89	1024	494	2903
43	97	1024	454	1498
43	101	1024	436	1123
43	107	1024	411	757
43	113	1017	387	509
43	127	1021	345	263
43	131	1022	335	223
43	139	1024	310	163

Tabla B.14: Número de bits para r=43

r	s	mcm (bits)	mcd (bits)	p
47	59	1024	815	166909
47	73	1024	659	16603
47	83	1024	580	5167
47	89	1024	541	2903
47	97	1024	496	1498
47	101	1024	476	1123
47	107	1024	450	757
47	113	1017	423	509
47	127	1021	376	263
47	131	1022	366	223
47	139	1024	345	163

Tabla B.15: Número de bits para r=47

r	s	mcm (bits)	mcd (bits)	p
53	59	1024	919	166909
53	73	1024	743	16603
53	83	1024	654	5167
53	89	1024	610	2903
53	97	1024	559	1498
53	101	1024	537	1123
53	107	1024	507	757
53	113	1017	476	509
53	127	1021	426	263
53	131	1022	413	223
53	139	1024	389	163

Tabla B.16: Número de bits para r=53

r	s	mcm (bits)	mcd (bits)	p
59	59	1024	1023	166909
59	73	1024	827	16603
59	83	1024	728	5167
59	89	1024	679	2903
59	97	1024	622	1498
59	101	1024	598	1123
59	107	1024	564	757
59	109	1024	554	673
59	113	1017	530	509
59	127	1021	474	263
59	131	1022	455	223
59	139	1024	433	163

Tabla B.17: Número de bits para r=59

Apéndice C

Tiempos de ejecución en el esquema multiplicativo

Se incluyen a continuación los tiempos de ejecución del intercambio de clave del esquema multiplicativo propuesto. Para un valor de r y s fijados (tamaños de los bloques A y B de la matriz M), se han realizado 100 pruebas obteniéndose los siguientes parámetros: valor de p necesario para obtener un orden de 1024 bits, máximo y mínimo tiempo de ejecución para realizar un intercambio de clave, media aritmética de los tiempos de las 100 pruebas y desviación estándar de dichas pruebas, todo ello para unos exponentes numéricos del orden de 512 bits. Se han seleccionado distintos tamaños de matrices y se ha adecuado el tamaño de p para obtener en todos los casos un tamaño de clave (mcm de $p^r - 1, p^s - 1$) de 1024 bits. Para la realización de cada prueba se han fijado unas condiciones iniciales, empleando un microprocesador Intel Pentium 4 con una frecuencia de reloj de 3.06 GHz, 512KB de caché, 400 MHz de front-side-bus y 1024 MB de RAM. y con una implementación realizada en MATLAB versión 7.0.1.24704 (R14) Service Pack 1.

s	p	Máximo	Mínimo	Promedio	Desviación típica
37	212900057	0.406250	0.328125	0.369945	0.022990
59	159936	0.796875	0.671875	0.741491	0.019203
73	16691	1.234375	1.000000	1.054920	0.024066
83	5167	1.421875	1.296875	1.361696	0.021806
89	2902	1.734375	1.484375	1.571627	0.025889
97	2437	2.031250	1.625000	1.674660	0.039305
101	1123	2.171875	2.000000	2.080755	0.025490
107	757	2.500000	2.375000	2.452351	0.025424
113	523	2.937500	2.703125	2.765625	0.047083
127	263	3.718750	3.515625	3.577506	0.038076
139	163	4.468750	4.140625	4.238397	0.066682
149	113	4.656250	4.250000	4.374072	0.067562

Tabla C.1: Tiempo de ejecución para r = 2

s	p	Máximo	Mínimo	Promedio	Desviación típica
59	166909	0.781250	0.687500	0.748621	0.022550
73	16631	1.156250	1.015625	1.085018	0.028317
83	5101	1.390625	1.296875	1.358456	0.024613
89	2903	1.812500	1.531250	1.587776	0.044696
97	1459	1.953125	1.828125	1.903493	0.025068
101	1117	2.093750	1.984375	2.040901	0.026654
107	757	2.484375	2.328125	2.385570	0.039175
113	523	2.734375	2.578125	2.648897	0.035094
127	263	3.578125	3.375000	3.453585	0.039507
139	162	4.468750	4.203125	4.279412	0.048468
149	113	5.109375	4.90625	4.994485	0.048026

Tabla C.2: Tiempo de ejecución para r = 3

s	p	Máximo	Mínimo	Promedio	Desviación típica
59	166909	0.828125	0.765625	0.796224	0.018269
73	16631	1.203125	1.078125	1.134651	0.029045
83	4073	1.593750	1.390625	1.437500	0.044026
89	2897	1.703125	1.578125	1.648438	0.026719
97	1499	2.046875	1.890625	1.994945	0.029476
101	1123	2.296875	2.109375	2.168658	0.032816
107	757	2.515625	2.343750	2.461857	0.037332
113	509	2.859375	2.656250	2.776654	0.035574
127	263	3.656250	3.437500	3.576746	0.041316
131	223	4.031250	3.781250	3.838235	0.047717
139	167	4.500000	4.250000	4.393842	0.054369

Tabla C.3: Tiempo de ejecución para r = 5

s	p	Máximo	Mínimo	Promedio	Desviación típica
59	166013	0.859375	0.781250	0.835938	0.020080
73	16603	1.234375	1.125000	1.184743	0.026222
83	5023	1.546875	1.453125	1.508732	0.019686
89	2903	1.796875	1.671875	1.740809	0.028558
97	1499	2.250000	2.015625	2.066176	0.052750
101	1123	2.375000	2.203125	2.285846	0.030173
107	753	2.656250	2.500000	2.577665	0.029916
113	509	2.906250	2.750000	2.863971	0.035156
127	263	3.574850	3.285450	3.417505	0.041787
131	223	3.889325	3.589725	3.737500	0.021937
139	167	4.715630	4.438305	4.572173	0.025812

Tabla C.4: Tiempo de ejecución para r = 7

s	p	Máximo	Mínimo	Promedio	Desviación típica
59	166909	0.953125	0.875000	0.907813	0.022643
73	16603	1.343750	1.281250	1.303125	0.018340
83	5167	1.687500	1.625000	1.651563	0.022158
89	2903	2.218750	1.828125	1.931250	0.109301
97	1498	2.421875	2.218750	2.300000	0.078229
101	1123	2.562500	2.406250	2.459375	0.046701
107	757	2.859375	2.718750	2.754688	0.042345
113	509	3.109375	3.031250	3.056250	0.025727
127	263	3.968750	3.812500	3.918750	0.047621
131	223	4.421875	4.171875	4.268750	0.074317
139	163	4.859375	4.718750	4.793750	0.042184

Tabla C.5: Tiempo de ejecución para r = 11

s	p	Máximo	Mínimo	Promedio	Desviación típica
59	166909	1.015625	0.921875	0.964063	0.028575
73	16603	1.421875	1.343750	1.370313	0.020898
83	5167	1.750000	1.703125	1.720313	0.017195
89	2903	2.046875	1.984375	2.004688	0.019557
97	1498	2.437500	2.343750	2.370313	0.029509
101	1123	2.609375	2.531250	2.559375	0.020571
107	757	2.906250	2.796875	2.848438	0.029509
113	509	3.281250	3.156250	3.209375	0.037702
127	263	4.140625	4.015625	4.065625	0.038836
131	223	4.593750	4.281250	4.431250	0.100778
139	163	5.140625	4.890625	4.978125	0.073362

Tabla C.6: Tiempo de ejecución para r = 13

s	p	Máximo	Mínimo	Promedio	Desviación típica
59	166909	1.078125	1.031250	1.054688	0.015185
73	16603	1.515625	1.421875	1.482813	0.024924
83	5167	2.062500	1.890625	1.943750	0.047278
89	2903	2.281250	2.125000	2.179688	0.046730
97	1498	2.687500	2.546875	2.625000	0.037558
101	1123	2.875000	2.796875	2.825000	0.023058
107	757	3.203125	3.078125	3.146875	0.041143
113	509	4.625000	4.312500	4.390625	0.088388
127	263	4.484375	4.328125	4.406250	0.048300
131	223	4.921875	4.609375	4.712500	0.102381
139	163	5.578125	5.250000	5.342188	0.114926

Tabla C.7: Tiempo de ejecución para r = 17

s	p	Máximo	Mínimo	Promedio	Desviación típica
59	166909	1.125000	1.093750	1.109375	0.010417
73	16603	1.609375	1.515625	1.567188	0.030414
83	5167	2.062500	1.953125	2.010938	0.033794
89	2903	2.468750	2.234375	2.293750	0.078229
97	1498	2.718750	2.640625	2.692188	0.026608
101	1123	2.921875	2.875000	2.896875	0.016796
107	757	3.390625	3.218750	3.267188	0.047990
113	509	3.781250	3.593750	3.646875	0.058091
127	263	4.765625	4.515625	4.581250	0.072842
131	223	4.875000	4.765625	4.826563	0.036421
139	163	5.765625	5.437500	5.559375	0.107952

Tabla C.8: Tiempo de ejecución para r = 19

s	p	Máximo	Mínimo	Promedio	Desviación típica
59	166909	1.296875	1.234375	1.264063	0.015538
73	16603	1.765625	1.703125	1.728125	0.019764
83	5167	2.296875	2.125000	2.179688	0.050631
89	2903	2.687500	2.453125	2.567188	0.077270
97	1498	3.125000	2.859375	2.996875	0.088327
101	1123	3.328125	3.125000	3.246875	0.059293
107	757	3.703125	3.484375	3.607813	0.088526
113	509	4.078125	3.843750	3.946875	0.079057
127	263	5.093750	4.765625	4.912500	0.113202
131	223	5.390625	5.109375	5.226563	0.096108
139	163	6.359375	5.781250	5.982813	0.182879

Tabla C.9: Tiempo de ejecución para r = 23

s	p	Máximo	Mínimo	Promedio	Desviación típica
59	166909	1.515625	1.390625	1.439063	0.037158
73	16603	2.109375	1.968750	2.026563	0.052627
83	5167	2.578125	2.437500	2.504688	0.055634
89	2903	3.125000	2.812500	2.939063	0.094169
97	1498	3.437500	3.265625	3.331250	0.066618
101	1123	3.656250	3.500000	3.532813	0.046263
107	757	4.093750	3.828125	3.953125	0.091998
113	509	4.500000	4.250000	4.351563	0.093533
127	263	5.578125	5.281250	5.389063	0.100576
131	223	5.953125	5.734375	5.854688	0.071811
139	163	6.593750	6.343750	6.445313	0.084385

Tabla C.10: Tiempo de ejecución para r = 29

s	p	Máximo	Mínimo	Promedio	Desviación típica
59	166909	1.609375	1.484375	1.528125	0.044683
73	16603	2.156250	2.015625	2.089063	0.044225
83	5167	2.765625	2.546875	2.659375	0.070956
89	2903	3.156250	2.859375	2.962500	0.098050
97	1498	3.562500	3.359375	3.467188	0.082827
101	1123	3.859375	3.609375	3.715625	0.094269
107	757	4.203125	3.984375	4.128125	0.075404
113	509	4.734375	4.359375	4.462500	0.123515
127	263	5.750000	5.453125	5.565625	0.106434
131	223	6.203125	5.781250	5.981250	0.131101
139	163	7.015625	6.562500	6.748438	0.137297

Tabla C.11: Tiempo de ejecución para r = 31

s	p	Máximo	Mínimo	Promedio	Desviación típica
59	166909	1.843750	1.734375	1.792188	0.046028
73	16603	1.843750	1.734375	1.792188	0.046028
83	5167	3.062500	2.875000	2.967188	0.080164
89	2903	3.421875	3.218750	3.284375	0.055021
97	1498	3.937500	3.718750	3.785938	0.075494
101	1123	4.234375	3.953125	4.051563	0.094628
107	757	4.609375	4.359375	4.456250	0.088327
113	509	5.031250	4.765625	4.829688	0.093880
127	263	6.281250	5.953125	6.104688	0.126521
131	223	6.875000	6.390625	6.554688	0.178038
139	163	7.406250	7.015625	7.203125	0.130935

Tabla C.12: Tiempo de ejecución para r = 37

s	p	Máximo	Mínimo	Promedio	Desviación típica
59	166909	1.984375	1.875000	1.920313	0.036421
73	16603	2.734375	2.593750	2.667188	0.053140
83	5167	3.296875	3.125000	3.242188	0.067608
89	2903	3.687500	3.453125	3.548438	0.091539
97	1498	4.265625	3.921875	4.106250	0.110436
101	1123	4.453125	4.343750	4.387500	0.031076
107	757	4.890625	4.578125	4.687500	0.105974
113	509	5.390625	5.125000	5.237500	0.100130
127	263	6.687500	6.312500	6.539063	0.134058
131	223	7.000000	6.640625	6.732813	0.101382
139	163	7.781250	7.453125	7.581250	0.128594

Tabla C.13: Tiempo de ejecución para r = 41

s	p	Máximo	Mínimo	Promedio	Desviación típica
59	166909	2.062500	1.984375	2.012500	0.025302
73	16603	2.859375	2.640625	2.756250	0.081423
83	5167	3.421875	3.234375	3.329688	0.071962
89	2903	3.937500	3.593750	3.687500	0.113632
97	1498	4.234375	4.046875	4.143750	0.051455
101	1123	4.625000	4.312500	4.437500	0.090211
107	757	4.890625	4.781250	4.826563	0.034114
113	509	5.390625	5.234375	5.287500	0.044925
127	263	6.734375	6.484375	6.576563	0.080502
131	223	7.390625	6.890625	7.100000	0.172615
139	163	8.046875	7.718750	7.840625	0.124522

Tabla C.14: Tiempo de ejecución para r = 43

s	p	Máximo	Mínimo	Promedio	Desviación típica
59	166909	2.406250	2.140625	2.190625	0.079263
73	16603	2.875000	2.796875	2.848438	0.030414
83	5167	3.484375	3.421875	3.453125	0.023292
89	2903	3.843750	3.781250	3.809375	0.023058
97	1498	4.500000	4.296875	4.376563	0.064822
101	1123	5.015625	4.593750	4.675000	0.122102
107	757	5.218750	5.062500	5.126563	0.057740
113	509	5.812500	5.578125	5.637500	0.070956
127	263	7.250000	6.843750	6.996875	0.112384
137	173	8.062500	7.828125	7.923438	0.075637
139	163	8.437500	8.078125	8.206250	0.113823

Tabla C.15: Tiempo de ejecución para r = 47

s	p	Máximo	Mínimo	Promedio	Desviación típica
59	166909	2.578125	2.437500	2.501563	0.037881
73	16603	3.359375	3.140625	3.218750	0.074754
83	5167	3.984375	3.781250	3.829688	0.058672
89	2903	4.468750	4.156250	4.278125	0.117344
97	1498	4.812500	4.718750	4.771875	0.033914
101	1123	5.453125	5.109375	5.200000	0.115714
107	757	6.125000	5.593750	5.825000	0.197478
113	509	6.421875	6.078125	6.192188	0.117170
127	263	8.140625	7.343750	7.517188	0.240989
131	223	8.218750	7.843750	8.012500	0.145610
139	163	9.046875	8.781250	8.890625	0.096319

Tabla C.16: Tiempo de ejecución para r = 53

s	p	Máximo	Mínimo	Promedio	Desviación típica
59	166909	2.890625	2.796875	2.834375	0.036234
73	16603	3.781250	3.515625	3.598438	0.088404
83	5167	4.500000	4.156250	4.351563	0.140962
89	2903	4.906250	4.578125	4.665625	0.105255
97	1498	5.609375	5.265625	5.390625	0.132787
101	1123	6.140625	5.578125	5.757813	0.176508
107	757	6.437500	6.062500	6.268750	0.137957
113	509	7.015625	6.562500	6.846875	0.168254
127	263	8.421875	8.125000	8.256250	0.112722
131	223	9.187500	8.500000	8.720313	0.245671
139	163	9.796875	9.406250	9.557813	0.129697

Tabla C.17: Tiempo de ejecución para r = 59

Apéndice D

Tiempos criptosistemas estándar

Se incluyen a continuación tablas con tiempos de ejecución, y con los parámetros más importantes para el intercambio de clave Diffie-Hellman y para el cifrado de información de ElGamal y RSA.

A la hora de computar tiempos se han realizado 100 pruebas, obteniendo posteriormente, el máximo, el mínimo, la media aritmética y la desviación estándar de los tiempos de las pruebas. Cada uno de los casos tenía las mismas condiciones iniciales, empleando un microprocesador Intel Pentium 4 con una frecuencia de reloj de 3.06 GHz, 512KB de caché, 400 MHz de front-side-bus y 1024 MB de RAM. La implementación se ha realizado en MATLAB versión 7.0.1.24704 (R14) Service Pack 1.

D.1 Diffie-Hellman. Intercambio de clave

Se han realizado pruebas para tamaños de clave de $64, 256, 512$ y 1024 bits y para cada uno de estos casos se ha variado el tamaño del número primo p para valores igualmente de $64, 256, 512$ y 1024 bits.

N. Primo	**Clave 1, 2**	**Máx.**	**Mín.**	**Media**	**D. Típica**
2^{64}	2^{64}	11.6719	11.6406	11.6564	0.0156
2^{256}	2^{64}	11.9219	11.6406	11.7500	0.1230
2^{512}	2^{64}	12.7656	12.6094	12.694	0.0581
2^{1024}	2^{64}	14.2500	14.0469	14.1500	0.0888

Tabla D.1: Diffie-Hellman para claves de 64 bits

N. Primo	Clave 1, 2	Máx.	Mín.	Media	D. Típica
2^{64}	2^{256}	11.8438	11.7031	11.8086	0.0411
2^{256}	2^{256}	12.0000	11.7813	11.8344	0.0955
2^{512}	2^{256}	12.8750	12.6250	12.7219	0.1171
2^{1024}	2^{256}	14.1094	14.0625	14.0775	0.0198

Tabla D.2: Diffie-Hellman para claves de 256 bits

N. Primo	Clave 1, 2	Máx.	Mín.	Media	D. Típica
2^{64}	2^{512}	11.6406	11.5781	11.5625	0.0827
2^{256}	2^{512}	11.9438	11.7562	11.7275	0.1494
2^{512}	2^{512}	12.5781	12.5000	12.5325	0.0306
2^{1024}	2^{512}	14.1841	14.0781	14.1087	0.0541

Tabla D.3: Diffie-Hellman para claves de 512 bits

N. Primo	Clave 1, 2	Máx.	Mín.	Media	D. Típica
2^{64}	2^{1024}	11.7500	11.6875	11.7219	0.0279
2^{256}	2^{1024}	12.4531	11.6562	11.9156	0.3249
2^{512}	2^{1024}	12.8750	12.6406	12.7065	0.0959
2^{1024}	2^{1024}	14.3438	14.1562	14.2219	0.0815

Tabla D.4: Diffie-Hellman para claves de 1024 bits

D.2 ElGamal. Cifrado de información

Se han realizado pruebas de cifrado de información para tamaños de clave de $64, 256, 512$ y 1024 bits y para cada uno de estos casos se ha variados el tamaño del número primo p para valores igualmente de $64, 256, 512$ y 1024 bits. En todos los casos la palabra la palabra cifrada era 'hola'.

N. Primo	**Clave privada**	**Máx.**	**Mín.**	**Media**	**D. Típica**
2^{64}	2^{64}	4.2975	4.1375	4.1533	0.0826
2^{256}	2^{64}	4.3125	4.1875	4.2615	0.0494
2^{512}	2^{64}	6.0625	5.4535	5.7785	0.3197
2^{1024}	2^{64}	22.1925	21.3825	21.742	0.7486

Tabla D.5: ElGamal para claves de 64 bits

N. Primo	**Clave privada**	**Máx.**	**Mín.**	**Media**	**D. Típica**
2^{64}	2^{256}	4.1250	4.0156	4.0437	0.0486
2^{256}	2^{256}	4.2656	4.0973	4.1412	0.0763
2^{512}	2^{256}	6.7656	6.3281	6.4481	0.2423
2^{1024}	2^{256}	23.3125	21.5313	22.275	0.9038

Tabla D.6: ElGamal para claves de 256 bits

N. Primo	**Clave privada**	**Máx.**	**Mín.**	**Media**	**D. Típica**
2^{64}	2^{512}	4.0625	4.0156	4.0312	0.0247
2^{256}	2^{512}	4.2188	4.1406	4.1781	0.0498
2^{512}	2^{512}	6.2031	5.6875	5.8031	0.4511
2^{1024}	2^{512}	22.5469	20.9531	21.2781	0.8227

Tabla D.7: ElGamal para claves de 512 bits

N. Primo	Clave privada	Máx.	Mín.	Media	D. Típica
2^{64}	2^{1024}	4.1318	4.0625	4.0826	0.0361
2^{256}	2^{1024}	4.2187	4.1406	4.1636	0.0556
2^{512}	2^{1024}	5.7218	5.0625	5.2537	0.3432
2^{1024}	2^{1024}	17.1500	15.1062	15.6906	0.9418

Tabla D.8: ElGamal para claves de 1024 bits

D.3 RSA. Cifrado de información

Se han realizado pruebas de cifrado de información para tamaños de e de $64, 256, 512$ y 1024 bits y para cada uno de estos casos se ha variados el tamaño del módulo RSA, para valores igualmente de $64, 256, 512$ y 1024 bits. En todos los casos la palabra la palabra cifrada era 'hola'.

RSA modulo n	e	Máx.	Mín.	Media	D. Típica
2^{64}	2^{64}	0.5938	0.4375	0.4750	0.0853
2^{256}	2^{64}	0.9219	0.8031	0.8813	0.1018
2^{512}	2^{64}	2.1094	1.9813	2.0688	0.1572
2^{1024}	2^{64}	18.1406	15.8063	17.1946	0.9419

Tabla D.9: RSA para claves de 64 bits

RSA modulo n	e	Máx.	Mín.	Media	D. Típica
2^{64}	2^{256}	0.6406	0.5556	0.6050	0.0853
2^{256}	2^{256}	1.1406	1.0781	1.1063	0.1855
2^{512}	2^{256}	3.7813	3.4219	3.6109	0.2572
2^{1024}	2^{256}	28.9688	25.2500	27.1946	0.9791

Tabla D.10: RSA para claves de 256 bits

RSA modulo n	**e**	**Máx.**	**Mín.**	**Media**	**D. Típica**
2^{64}	2^{512}	0.9688	0.9531	0.9612	0.0100
2^{256}	2^{512}	1.8113	1.6875	1.7418	0.1692
2^{512}	2^{512}	5.8750	5.5156	5.6438	0.6040
2^{1024}	2^{512}	31.6875	28.8063	30.1946	1.1419

Tabla D.11: RSA para claves de 512 bits

RSA modulo n	**e**	**Máx.**	**Mín.**	**Media**	**D. Típica**
2^{64}	2^{1024}	8.6875	8.3438	8.5438	0.0853
2^{256}	2^{1024}	11.6406	10.7969	11.3969	0.1018
2^{512}	2^{1024}	18.3594	15.7500	17.0688	0.8957
2^{1024}	2^{1024}	33.9531	30.1250	32.9531	1.9419

Tabla D.12: RSA para claves de 1024 bits