Estudio y Aplicaciones de Esquemas Criptográficos

Tlatoani de Jesús Reyes Bermejo

22 de noviembre de 2007

Índice general

AGRADECIMIENTOS

Por lo general, cuando alguien se titula sólo le reconocemos a esa persona, no vemos quien está por atrás. En mi caso muy particular, están todas las personas que me quieren como mi familia, la cual agradezco todo este cuarto de siglo que ha mantenido y alimentado mi gusto por las matemáticas, a mis amigos, profesores y novia, que han hecho que llegue a una de tantas metas en mi vida.

En especial quiero agradecer a mis padres, Leobardo Reyes Márquez y Felícitas Bermejo Rojas, a mis tíos, Luz Bermejo Rojas y Fernando Bermejo Rojas, y a mi abuela materna, Ana Rojas de Bermejo, entre muchos, mis amigos Luis Miguel Hernández Pérez, Demetrio Quintero Correa, Luis Manuel González Rosas y Cristhian Emmanuel Garay López, y a mis profesores Dr. Carlos Rentería Márquez, que con su sabiduría y paciencia me guió para hacer esta tesis, y al M. en C. Germán Tellez Castillo, quien me ha ayudado tanto en mi vida profesional.

Como podrán darse cuenta es imposible agradecer a todas las personas que me han ayudado y motivado a seguir adelante. A todos ellos GRACIAS.

Prefacio

Desde la época de los griegos, el hombre ha intentado ocultar información, ya sea para la guerra, ya sea personal, y por lo mismo hemos desarrollado métodos para hacerlo. Esos métodos han sido estudiado a lo largo de los años, para así dar paso a dos grandes ciencias: la *criptografía* y el *criptoanálisis*. La primera trata de estudiar y hacer métodos seguros para ocultar información; la segunda trata de encontrar las debilidades de los esquemas criptográficos.

Entre las motivaciones que tengo para estudiar criptografía y hacer esta tesis, "Estudio y Aplicaciones de Esquemas criptográficos", estan el tiempo que muchos matemáticos brillantes como Neal Koblitz y Menezes, entre otros, han dado para el desarrollo de esta ciencia, y sobre todo, la aplicación de varias de las ramas de las matemáticas como el álgebra y la teoría de números, que a lo largo de los años se cría que nunca se podrían aplicar.

Así también, puedo mencionar que la criptografía también ha dado a las matemáticas un avance al desarrollar algoritmos cada vez más rápidos para generar números primos, factorizar números compuestos en sus factores primos, y por lo mismo se le ha dado su lugar en este trabajo.

El trabajo se dividió en 5 capítulos: En la introducción mencionamos algo de historia de la criptografía, principios de Kerckhoffs, conceptos básicos de criptología. En el primer capítulo se mencionan los conceptos básicos de la criptografía, tipos de criptoanálisis, y la definición de criptosistema o esquema criptográfico. En el segundo capítulo se desarrollan algunos de los tantos criptosistemas simétricos, haciendo énfasis en el DES. En el tercer capítulo se mencionan los criptosistemas de llave pública mas importantes dando una explicación formal de porqué sirven esos criptosistemas. En el último capítulo se mencionan varios algoritmos que resuelven 2 grandes problemas matemáticos pero que siguen siendo muy costosos computacionalmente: La facotización en números primos y los tests de primalidad, lo cual nos va a ayudar para la generación de números primos.

Introducción

0.1. Breve historia de la criptografía

El primer indicio de criptografía se encuentra con los egipcios y sus geroglíficos, así como otras culturas como los hebreos, babilonios y asirios. Algunos llaman a estos inicios como sistemas protocriptográficos. En el 400 a. C., con los espartanos encontramos el primer sistema criptográfico archivado, la escitala, el cual era usado para comunicaciones secretas entre comandantes militares. Los griegos no se quedaron atrás, y usaron lo que hoy en día conocemos como cifrados de transposición. En el Siglo IV a. C. encontramos el primer tratado de criptografía: "En Defensa de las Fortificaciones". Este tratado fue escrito por el griego Aeneas Tacticus. Otro griego en el Siglo II a. C., Polybius desarrollo una máquina con el proposito de encriptar una letra en un par de símbolos. Dicha máquina es llamada "Cifrador de Polybius". Para el Siglo I a. C., Julio Cesar tenía ya su propio criptosistema el cual se explica mas adelante. Aquí aparecen los criptosistemas por substitución. En 1379, aparece el primer manual europeo en criptogrfía; éste es un compendio de varios criptosistemas hecho por Gabriele de Lavinde de Parma. Actualmente se encuentra en los archivos del Vaticano. En 1466, Leon Battista Alberti crea el disco que hoy lleva su nombre y además publica "Trattati in cifra", en el cual lo describe. Giambattista della Porta provee el primer ejemplo de los cifrados digráficos en "De furtivis literarum notis" en 1563.

En 1586 Blaise de Vigenère publicó "Traicté des chiffres", el cual contiene una tabla llamada *tabla de Vigenère*. Este criptosistema fue utilizado entre 1861 y 1865, en la guerra civil estadounidense los ejércitos federal y el confederado usaban el cifrado por transposición y el criptosistema de Vigenère, respectivamente. En 1929, el matemático Lester S. Hill propone el criptosistema que lleva su nombre, el cual utiliza las reglas del álgebra de matrices para encriptar. En la primera guerra mundial se empezaron a crear máquinas de rotor para cifrar, dando también un crecimiento en el criptoanálisis de éstas. Las principales fueron creadas en E. U. A., por Edwuard H. Herbern, en Holanda, por Hugo A. Kock, y en Alemania, por Arthur Scherbins; estos dos últimos fueron precursores de la máquina *Enygma* utilizada por los alemanes en la segunda guerra mundial. Además de la máquina alemana Enygma, hubo otras dos máquinas: Una de la Gran Bretaña llamada TYPEX, y la otra por E. U. A. llamada M-134-C *Sigaba*. En los 70ś, el Dr. Horst Feistel estableció la familia de criptosistemas "Cifrados de Feistel" los cuales son los precursores de lo que hoy conocemos como Data Encryption Standard o DES.

En la Universidad de Stanford, en 1976 se publicó un artículo titulado "New Directions in Criptography" el cual daba un giro a la criptografía. Este artículo, cuyos autores son Whitfield Diffie y Martin E. Hellman, propone una forma para minimizar la necesidad de canales seguros para distribuir las llaves. Este fue el primer paso para lo que hoy en día se conoce como *criptosistemas de llave pública*. Sin embargo, no fue hasta 1978 que se publicó el primer criptosistema de llave pública: RSA.

El RSA fue llamado así por las tres primera letras de cada apellido de los autores: **R**ivest, **S**hamir y **A**dleman. El artículo se llama "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". En los 80ś, Xuejia Lai y James Massey propusieron un nuevo y mas fuerte criptosistema cuyas llaves ocupan 128bits cada una. Dicho criptosistema es llamado *International Data Encryption Algorithm*, de ahí el nombre de *IDEA*. En 1984, el artículo titulado "A Public Key Criptosystem and a Signature Scheme Based on Discrete Logarithms" fue publicado por Taher El-Gamal, empleado de Hewlett-Packard Labs. En 1985, dos matemáticos estadounidenses, Dr. Victor Miller y Dr. Neal Koblitz se propusieron buscar criptosistemas con curvas elípticas, a partir de la idea de Lenstra que aplicó las curvas elípticas para atacar algunos criposistemas. En 1994, Ronald Rivest publicó un nuevo algoritmo para encriptar llamado RC-5. También se ha estado estudiando los criptosistemas hiperelípticos [1].

0.2. Criptología, criptografía y criptoanálisis

0.2.1. Criptografía

La criptografía es una rama que conforma a la criptología. Ésta última agrupa a la criptografía y a su contraparte, el criptoanálisis. Formalmente, basándonos en la página 15 de [Mz96]:

Definición: La *criptografía* es el estudio de técnicas matemáticas relacionadas con garantizar la seguridad de la información.

0.2.2. Criptoanálisis

El criptoanálisis consiste en comprometer la seguridad de un criptosistema. Esta ciencia sirve para saber información que quisieramos leer, pero que no estamos autorizados. Algunos la consideran un arte y otros una ciencia. Por ejemplo para la Real Academia Española el término criptoanálisis significa el arte de descifrar criptogramas. Antes de la definición formal de lo que es criptoanálisis, veamos la siguiente definición de [Mz96]:

Definición: Decimos que un criptosistema es *rompible* si un atacante, sin conocimiento del par de llaves (e, d), puede sistemáticamente recobrar el mensaje de su criptograma correspondiente en cierto tiempo.

El concepto de rompible sólo da la idea de si es confiable o no un criptosistema, pues al haberse obtenido sistemáticamente la llave, a lo cual nos referiremos con *romper el sistema*, lo único que pueda ayudar quizá sería la longitud de la llave, o implementar otro criptosistema basándose en el roto.

Definición: El *criptoanálisis*es la ciencia que estudia y analiza los criptosistemas con el objetivo de romperlos. Este análisis se hace normalmente usando técnicas matemáticas basadas en los conceptos con los cuales dicho criptosistema funciona, así como también conceptos pertenecientes al área de probabilidad. Esta ciencia también atenta en contra de los servicios para la seguridad de la información.

[1] Ver [Kz98] pp. 148-154

0.2.3. Criptología

Definición: La *criptología* es el estudio de la criptografía y el criptoanálisis.

En la siguiente sección se daran las partes de un esquema criptológico; ésto se hará para tener claro los conceptos que involucra.

0.3. Partes de un esquema criptológico

Como la criptología se encarga de enviar y recibir información, podemos identificar las siguientes partes en un sistema criptológico:

1. *Entidad*: es cualquier persona o cosa que envía, recibe o manipula el mensaje, ya sea plano o cifrado.

2. *Emisor*: es aquella entidad que envía el mensaje, y así mismo quien lo encripta.

3. *Receptor*: es aquella entidad a quien es enviado el mensaje, obviamente cifrado.

4. *Adversario*: es una entidad distita de las dos anteriores, pues ésta trata de romper la seguridad dada a la información, para así hacer uso de ella. Existen dos tipos de adversarios: Los *pasivos* y los *activos*. El adversario pasivo es aquel que puede leer la información mas sin embargo no la roba ni la modifica; al contrario de los adversarios pasivos, los activos sí modifican o roban la información. Estos últimos también son llamados *crackers*. Otra forma de llamarlo es *atacante* y *oponente*. Debido a este último para referirnos en las figuras a un adversario, lo haremos con la letra O o con el nombre *Oscar*.

5. *Canal*: es el medio que se ocupa para ser enviado el mensaje.

6. *Canal seguro*: es un canal en el cual el adversario no puede leer, borrar, reordenar o añadir algo al mensaje cifrado.

7. *Canal inseguro*: es cualquier canal en el cual cualquier entidad, además del remitente y el receptor, puede leer, borrar, reordenar o añadir algo al mensaje cifrado.

0.4. ¿Qué es lo que la criptografía debe cuidar?

En esta sección se menciona las cosas que uno le pide a un criptosistema.

1. *Autentificación*: Esta cualidad trata sobre todo de que las dos partes, el emisor y el receptor, se identifiquen uno al otro. Involucra también el origen del mensaje, la fecha en que fue enviado, el contenido, la hora de envio, etc. Según Menezes [Mz96] esta propiedad se subdivide en dos: *Autentificación de entidades* y *autentificación de origen de los datos*.

2. *Confidencialidad*: Esta cualidad es la que mas se busca y la que en un principio uno cree que es la única que se debe buscar. Ésta se refiere únicamente a que las personas autorizadas a leer el mensaje lo lean. Otras formas de llamar a esta cualidad son *privacidad* y *seguridad*.

3. *Negación de actos* (non-repudiation): Esta cualidad se explicará con un ejemplo.

1 Ejemplo: Imaginemos que Benito le manda un criptograma a Ana, pero después de esto, Benito se arrepiente de haberlo enviado y no quiere que Ana lea el mensaje original de este criptograma. Uno podría pensar que es muy fácil negar haberlo enviado. Sin embargo, en un buen criptosistema esto es una de las cosas primordiales que se cuidan.

El no rechazo o no repudio se refiere precisamente a la imposibilidad de no poder negar que uno envió algo, cuando uno sí lo hizo.

4. *Integridad de datos*: Esta cualidad se refiere en sí a la obviedad de querer recibir el mensaje exacto que se quiere transmitir. Esto es que se debe cuidar el que no se modifique el mensaje, ya sea borrarlo parcial o totalmente, cambiarlo por otro mensaje o insertar oraciones al mensaje. También trata de que se detecte alguna manipulación como las mencionadas.

0.5. Principios de Kerckhoffs

En esta sección se menciona los principios de Kerckhoffs mencionados en su artículo "*La Cryptographie Militaire*" del "`Journal des Sciences Militaires`" [Krf1883] página 12. En este artículo se mencionan 6 requerimientos que un criptografo debe tener en cuenta para basar la seguridad de su criptosistema. Estos 6 requerimientos los tradujo de manera casi literal Menezes en [Mz96] en la página 14.

Los principios de Kerckhoffs son:

1. El sistema debe ser, sino teóricamente irrompible, irrompible en la práctica.
2. El comprometer los detalles del sistema no debe inconvenir los correspondientes.
3. La llave debe poder ser recordada sin notas y debe ser fácil de cambiarla.
4. El criptograma debe poder ser transmitido por telegrama
5. El aparato de encriptación debe poderse portar y debe poder operarla una sola persona.
6. El sistema debe ser fácil, no debe requerir el conocimiento de una larga lista de reglas ni notas mentales.

Los 6 requerimientos que mencionados aquí son solo la traducción al español, basándonos en [Mz96], que a su vez son traducción del francés de [Krf1883].

El primer requisito nos dice que aunque se conozca un algoritmo, éste no se puede ejecutar en tiempo polinomial. Por ejemplo, en RSA, la seguridad está basada en que no se puede factorizar números muy grandes en tiempo polinomial, sin embargo, si existen algoritmos para factorizar.

El segundo requisito habla de que no debemos basar la seguridad de un criptosistema en la suposición de que el adversario no conoce que algoritmo para encriptar estoy usando; más aún debemos suponer que al adversario lo único que quiere hacer es encontrar la llave para poder desencriptar varios criptogramas con la misma llave, o que quiere encontrar el mensaje asociado a un criptograma.

El tercer requisito es claro; por ejemplo en el caso de los correos electrónicos, en los cuales se tiene acceso mediante una contraseña, el dueño de cada correo debe poder leer su correo sin ningún problema, y por ende debe de memorizar su contraseña. Para el cuarto requisito hay que recordar que el texto original fue escrito en el año 1883, y por tanto se pide ser transmitido por telegrama. Una actualización de este requisito sería:

- El criptosistema debe poder ser transmitido sin dificultad [2]

En el quinto y sexto requisitos, se enuncian, quizá de manera un poco antigua, dos de los objetivos de la programación: la *portabilidad* y la *legibilidad*, respectivamente.

[2] Ya sea enviado por correo electrónico, bajado de algún dominio de internet, en papel, etc.

Capítulo 1

Conceptos básicos y algo mas abstracto

Mathematics is not a deductive science
- - that is a cliche.
When you try to prove a theorem,
you don't just list the hypotheses,
and then start to reason.
What you do is trial and error,
experimentation, guesswork.
I Want to be a Mathematician, Washington : MAA Spectrum, 1985.
Paul Richard Halmos

1.1. Conceptos básicos

En esta primera sección de este capítulo se verán algunos tecnicismos, los cuales son importantes para el resto del desarrollo. Antes de empezar con la definición de criptosistema, debemos saber algunos conceptos básicos, y además se hará distinción de cómo se les llamarán mas adelante en la tesis en algunos de estos términos cuando se utilice varios nombres.

Definición:

- **Desencriptar**: es el proceso de obtención del mensaje original a partir de la llave y de la función correspondientes. También es llamado *descifrar*, mas sin embargo, me he inclinado mas por llamar a este proceso desencriptar, pues la palabra descifrar da idea mas bien de hacer criptoanálisis.

- **Encriptar**: es el proceso para obtener el criptograma y, utilizando una función para encriptar. También es llamado *cifrar*, mas sin embargo, me he inclinado mas por llamar a este proceso encriptar, siguiendo así con el proceso anteriormente mencionado en esta misma definición.

- **Mensaje plano**: es en sí el mensaje que queremos que el receptor lea, es decir, el mensaje que queremos comunicar. Otras formas de llamarlo: *Mensaje original*, *mensaje llano* o simplemente *mensaje*.

- **Criptograma**: es en sí lo que se envía, es el resultado de aplicar una función para encriptar a un mensaje. Otras formas de llamarlo: *mensaje cifrado*, *mensaje encriptado* o *criptomensaje*.

- **Espacio de mensajes**: consiste de todos aquellos mensajes que pueden ser enviados escritos en un alfabeto Σ.
- **Espacio de criptogramas**: consiste de palabras escritas en un alfabeto, que puede no ser el mismo que el de los mensajes, y que al parecer no tienen sentido.

Al traducir los textos en inglés uno encuentra palabras como *ciphertext* y *plaintext*, que en español querrían decir mensaje cifrado y mensaje plano, respectivamente. Aquí en lugar de tomar estas traducciones utilizaremos los términos criptograma y mensaje, respectivamente.

1.2. Tipos de criptoanálisis

En esta sección sólo vamos a ver los ataques a criptosistemas, y dejaremos de lado aquellos ataques a los protocolos de seguridad [1]. Como base además se tendrán los Principios de Kerckhoffs, mencionados en la sección 0.5.

Muchos textos clasifican los distintos tipos de ataque, por ejemplo Stinson [St95] en 4, Buchmann [Bch02] [2] en 5, y Menezes [Mz96] en 6 [3]. Aquí se optó por seguir con la postura de Stinson en las páginas 24 y 25, pues el objetivo principal es entender y analizar los criptosistemas.

Los tipos de criptoanálisis son:

- **AMC** (*Ataque con el Mensaje Conocido*) Se conoce un criptograma, y se trata de encontrar el mensaje correspondiente.
- **ASCC** (*Ataque con solo el Conocimiento del Criptograma*) Se conoce un mensaje y su criptograma correspondiente. Puede ser que se conozcan varios pares de éstos. Se trata de desencriptar otros criptogramas.
- **AME** (*Ataque con Mensaje Elegido*) De algún modo se conoce la función para encriptar y la llave con la que se encripta (como por ejemplo en el caso de la llave pública). Se trata de desencriptar otros criptogramas.
- **ACE** (*Ataque con Criptograma Elegido*) Se puede desencriptar pero no se conoce la llave. Se quiere encontrar la llave para poder desencriptar más rápido y que sirva de mejor manera el mensaje original.

En todos los anteriores siempre se quiere obtener la llave.

1.3. Criptosistema

En esta sección vamos a definir lo que es un criptosistema de forma matemática y rigurosa, así como también el cómo los podemos clasificar de acuerdo a la distribución de las llaves.

Definición: : Un *criptosistema* es una quintupla (**P**, **C**, **K**, **E**, **D**) donde:

- **P** y **C** son dos conjuntos, llamados el *espacio de mensajes* y el *espacio de criptogramas*, y cuyos elementos son llamados *mensajes* y *criptogramas*, respectivamente.
- **K** es un conjunto llamado el *espacio de llaves*, y sus elementos son llamados *llaves*.

[1] Para estos ataques puede ver [Mz96] página 42
[2] Sólo se agrega el tipo Apaptativo del AME
[3] Se le agrega el tipo Adaptativo del AME y del ACE

- **E** y **D** son dos familias de funciones, tales que para cada llave $k \in \mathbf{K}$ existen dos funciones $e_k \in \mathbf{E}$ y $d_k \in \mathbf{D}$ tales que:

  1. $e_k : \mathbf{P} \longrightarrow \mathbf{C}$ y $d_k : \mathbf{C} \longrightarrow \mathbf{P}$
  2. e_k es biyectiva $\forall k \in \mathbf{K}$
  3. $\forall e_k$ existe una $k' \in \mathbf{K}$, tal que $d_{k'} \circ e_k = 1$
  4. Las funciones $e_k \in \mathbf{E}$ son llamadas funciones de encriptación. Y las funciones $d_k \in \mathbf{D}$

  Esta quintupla a veces también es llamada *esquema criptográfico.*

En cualquier criptosistema tenemos que dada una llave $k \in \mathbf{K}$, ésta tiene asociadas dos funciones, e_k y d_k; sin embargo, no se dice nada respecto a ellas, i. e., puede ser que para algunas llaves k, se cumpla $d_k \circ e_k = \mathbf{1}$.

2 Ejemplo (Cifrado de César): Este criptosistema es muy simple y es también uno de los primeros documentados históricamente. [4] Fue usado tanto por el emprador romano Julio César, debido al cual el nombre del criptosistema, en las guerras contra los galos, en la guerra de secesión y también por la armada rusa en 1915.

El cifrado de César se explica fácilmente con aritmética modular en [Kb98] en las páginas 55 y 56. Sin embargo, aquí se va a explicar con el grupo $(\mathbf{Z}_n, +)$[5].

Cifrado de César en abstracto El criptosistema (**P**, **C**, **K**, **E**, **D**) está dado por

- **P**= **C**= Σ^*
- **K**= $\Sigma = \mathbf{Z}_{27} = \{0, 1, ..., 26\}$
- **E**= $\{\hat{e}_k : \Sigma^* \longrightarrow \Sigma^* \mid k \in \mathbf{K}\}$, donde la función para encriptar $\hat{e}_k$ es la extensión natural de $e_k : \Sigma \longrightarrow \Sigma$ la cual está dada por $e_k(\sigma) = \sigma + k$
- **D**= $\left\{\hat{d}_k : \Sigma^* \longrightarrow \Sigma^* \mid k \in \mathbf{K}\right\}$, donde la función para desencriptar $\hat{d}_k$ es la extensión natural de $d_k : \Sigma \longrightarrow \Sigma$ la cual está dada por $d_k(\sigma) = \sigma - k$.

Así pues, si el mensaje plano es $m = \sigma_1\sigma_2 \cdots \sigma_s$, entonces el criptograma c es $c = \hat{e}_k(\sigma_1\sigma_2 \cdots \sigma_s) = \rho_1\rho_2 \cdots \rho_s$, donde $\rho_i = e_k(\sigma_i)$, $\forall i \in \{1, ..., s\}$.

Particularmente, en la antigua Roma, Julio César utilizaba como llave $k = 3$.

Criptoanalizándolo... Este criptosistema tiene 27 llaves, por lo que se puede romper muy fácilmente por fuerza bruta. Otra forma de analizarlo es estadísticamente debido a que cifrar una letra siempre da como resultado una misma letra, podemos decir que la letra que más encontremos en el criptograma corresponderá a la letra utilizada con más frecuencia en el idioma en que esté escrito el mensaje.

[4] Se sabe que César lo usó en las guerras contra los galos por que Suetonio lo nombra en su libro "Vida de los 12 Césares". Esta obra la podemos encontrar en francés como "La vie des 12 Césars" en cualquiera de las dos páginas web siguientes:
http://remacle.org/bloodwolf/historiens/suetone/table.html
http://bcs.fltr.ud.ac.be/SUET/accueil.html

[5] Para el estudio de este grupo puede ver [Gril99] o [Hers88]

1.4. Llaves privada y pública

Los criptosistemas se clasifican en criptosistemas de llave privada y criptosistemas de llave pública. Esta clasificación está basada en cómo se distribuyen las llaves, lo cual está completamente relacionado con la facilidad de obtener la función o la llave para desencriptar. Aparte de la diferencia arriba mencionada, se ha identificado que el algoritmo para hallar el criptograma es más complejo y costoso en los de llave pública que en los de llave privada, por lo que el criptoanálisis se dificulta también, y, aunque la seguridad es muy importante, debemos cuidar también el tiempo que tarda la computadora al encriptar y al desencriptar sabiendo la llave, así como también el uso de la memoria, pues un algoritmo muy complejo aplicado a un mensaje tardaría mucho tiempo.

Definición:

- Los *criptosistemas de llave privada* son aquellos en los cuales para cada pareja de funciones de encripción y de desencriptación, (e, d), dada es computacionalmente fácil obtener d de e, y viceversa. También son llamados criptosistemas clásicos, criptosistemas simétricos y criptosistemas convencionales.
- Los*criptosistemas de llave pública* son aquellos en los cuales dada la función de encripción e_k, es computacionalmente imposible saber o computar la función de decriptación d_k a partir de e_k o sin conocer la llave $k \in \mathbf{K}$.

Capítulo 2

Criptosistemas Simétricos

Estos criptosistemas son llamados simétricos pues en la mayoría de los casos prácticos, se tiene que $d_k \circ e_k = \mathbf{1}$ para todas las llaves $k \in \mathbf{K}$, o se puede obtener de manera sencilla una llave k' tal que cumpla $d_{k'} \circ e_k = \mathbf{1}$.

2.1. Manejo de las llaves

En este caso, el manejo de llaves es algo anticuado, y se asemeja mucho al de un cerrajero, en el sentido de que el cerrajero solo da las llaves a las personas que están autorizadas a abrir cierta puerta. En particular para enviar mensajes, uno debe de dar una llave distinta a cada persona de quien uno quiera recibir algún mensaje. Esto quiere decir que en un conjunto de x personas se necesitan $x(x-1)$ par de llaves para que se comuniquen todas con todas.

2.2. Técnicas para encriptar en un criptosistema clásico

2.2.1. Cífrado afín

Definición: Una *función afín de anillos* es una función $f : R_1 \longrightarrow R_2$ de la forma $f(r) = \varphi(r) + s$, donde $s \in R_2$ y $\varphi : R_1 \longrightarrow R_2$ es un homomorfismo de anillos.

Este criptosistema va a tener como funciones para encriptar y para desencriptar funciones afines, de ahí su nombre.

Definamos lo que necesitamos para nuestro criptosistema:

El criptosistema

Definición: Un *cifrado afín* es aquel cuyas funciones para encriptar son funciones afines.

Con la definición anterior tenemos: El espacio de mensajes y el espacio de criptogramas resultan ser el mismo conjunto: $\mathbf{P} = \mathbf{C} = \Sigma^n$.

El espacio de llaves es: $\mathbf{K} = \mathbf{Z}_n^2$.

Si la llave es $k = (a,\, b)$ y si el mensaje es $m = \sigma_1\sigma_2 \cdots \sigma_k$, entonces las funciones para encriptar y desencriptar son:

$$e_k : \Sigma \longrightarrow \Sigma; e_k(\sigma) = \varphi(\sigma) + b$$

y

$$d_k : \Sigma \longrightarrow \Sigma; d_k(c) = \varphi^{-1}(c - b)$$

De esta forma tenemos

Teorema 2.1. *Las funciones para desencriptar en un cifrado afín son afines.*

Dem:
Tenemos que dichas funciones son de la forma $d_k(c) = \varphi^{-1}(c - b)$
Debido a que φ es un homomorfismo de anillos tenemos que φ^{-1} lo es, y así:
$\varphi^{-1}(c - b) = \varphi^{-1}(c) - \varphi(b)$ De esta forma, $d_k(c) = \varphi^{-1}(c) - \varphi^{-1}(b)$ □

Cifrado de Hill

Este es el caso mas general de los cifrados en bloque. Fue inventado por Lester S. Hill en 1929. Para explicar este criptosistema, necesitamos un resultado de teoría de anillos, que tiene que ver con el anillo de matrices sobre un anillo R.

Teorema 2.2. *Sea R un anillo y sea $M(R)$ el anillo de matrices sobre R. Sea $A \in M(R)$. A es invertible si y sólo si $det(A)$ es unidad de R.*

En nuestro caso particular, $\mathbf{Z}_m$ jugará como el anillo R, y $M(\mathbf{Z}_m)$ será el anillo de matrices con entradas en $\mathbf{Z}_m$. Como sabemos, las unidades de $\mathbf{Z}_m$ son aquellas clases $\bar{r}$ tales que $gcd(r, m) = 1$. De aquí que las matrices $A \in M(\mathbf{Z}_m)$ invertibles son aquellas que cumplen $gcd(det(A), m) = 1$

El criptosistema

$\mathbf{P} = \mathbf{C} = \mathbf{Z}_m^n$
$\mathbf{K} = \{A \in M(\mathbf{Z}_m) \| A\, es\, invertible\}$
Si el mensaje es $m = \sigma_1\sigma_2\cdots\sigma_n$ entonces, viéndolo como vector vertical tendríamos:

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_n \end{bmatrix}$$

Luego, si

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \cdots & \cdots & \ddots & \cdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix}$$

Entonces tenemos que

$$e_A(m) = A \cdot m$$

Similarmente, tomando el criptograma c como vector columna tenemos:

$$d_A(c) = A^{-1} \cdot c$$

2.2.2. Cifrado por permutación

Como su nombre lo indica, las funciones de encriptación serán permutaciones. Éstas se aplicarán a palabras de n-letras.

Definición: Sea X un conjunto. Una *permutación* es una función $X \xrightarrow{f} X$ biyectiva. Al conjunto de permutaciones en X lo denotamos por S_X. En particular, si $n \in \mathbf{N}$ entonces denotaremos al conjunto de permutaciones en $\{1, 2, \ldots, n\}$ por S_n.

Teorema 2.3. *Sea $n \in \mathbb{N}$. $(S_n, \circ)$ es un grupo con $n!$ elementos.*

Teorema 2.4. *Sea G un grupo finito con n elementos, y sea $g \in G$. Entonces $o(g) \,\|\, |G|$*

Obsérvese que en general el factorial de n se va haciendo grande, mientras n crece, por ejemplo: $10! = 3628800$, $20! = 2432902008176640000$ y $27! = 10888869450418352160768000000$. Esto no quiere decir que este criptosistema sea seguro. De hecho es muy inseguro debido a los teoremas 2.3 y 2.4.

Para la siguiente notación el lector puede consultar [Hers88] en la página 111.

Sean $n \in \mathbf{N}$ y $\varphi \in S_n$ una permutación. Entonces, φ la representaremos por:

$$\varphi = \begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ \varphi(1) & \varphi(2) & \varphi(3) & \cdots & \varphi(n) \end{pmatrix}$$

El criptosistema

$\mathbf{P} = \mathbf{C} = \Sigma^n$
$\mathbf{K} = S_n$
Si $m = \sigma_1\sigma_2\cdots\sigma_n$ es el mensaje, entonces lo encriptamos de la siguiente manera: $c = \sigma_{\pi(1)}\sigma_{\pi(2)}\cdots\sigma_{\pi(n)}$

2.2.3. Cifrado por substitución

Como en la subsección 2.2.2, las funciones para encriptar son permutaciones, pero aplicadas de distinta manera. Mientras que en la subsección 2.2.2 las permutaciones se aplicaban a los subíndices de los símbolos del mensaje a encriptar, en este caso se aplica directamente a los símbolos. De esta manera, en el cifrado por substitución las letras no cambian de posición, sino mas bien cambian por otra letra o símbolo que quizá no estaba en el mensaje.

El criptosistema

$\mathbf{P} = \mathbf{C} = \mathbf{Z}_{27}^n$
$\mathbf{K} = S_{27}$
Las funciones para encriptar:
$e_\pi : \mathbf{Z}_{27}^n \longrightarrow \mathbf{Z}_{27}^n$ donde $e_\pi(m) = e_\pi(\sigma_1\sigma_2\cdots\sigma_n) = \pi(\sigma_1)\,\pi(\sigma_2)\cdots\pi(\sigma_n)$
Las funciones para desencriptar:
$d_\pi : \mathbf{Z}_{27}^n \longrightarrow \mathbf{Z}_{27}^n$ donde $d_\pi(c) = d_\pi(\rho_1\rho_2\cdots\rho_n) = \pi^{-1}(\rho_1)\,\pi^{-1}(\rho_2)\cdots\pi^{-1}(\rho_n)$

Un poco más de notación y un ejemplo

En este criptosistema utilizamos como funciones base a las permutaciones. Debido a que estamos utilizando un conjunto finito sobre el cual estamos aplicando las permutaciones, la manera común en representar cada función π es en forma matricial[1]. i.e.

$$\begin{pmatrix} a_1 & a_2 & \cdots & a_k \\ \pi(a_1) & \pi(a_2) & \cdots & \pi(a_k) \end{pmatrix}$$

Un ejemplo muy particular es el cifrado de César revisado en 2. Otro ejemplo es el siguiente.

3 Ejemplo: La permutación que utilizaremos para encriptar será

$$\pi = \begin{pmatrix} a & b & c & d & e & f & g & h & i & j & k & l & m & n & \tilde{n} & o & p & q & r & s \\ B & X & Y & C & F & E & A & M & J & L & K & Z & P & W & N & G & I & O & Q & \tilde{N} \\ & t & u & v & w & x & y & z & & & & & & & & & & & & \\ & T & D & H & V & S & U & R & & & & & & & & & & & & \end{pmatrix}_2$$

Y así tenemos que

$$m = latareaestahecha, c = zbtbqfbf\tilde{n}tbmfymb$$

Obsérvese que la representación de π es mas extensa que la representación de una permutación en el cifrado de César. Esto es debido a que el conjunto que se toma para el cifrado de César es un subconjunto muy particular de las permutaciones. Así pues, tenemos: que el cifra de César es un ejemplo de cifrado por substitución, pues las funciones de encriptación del cifrado de César las podemos ver como

$$\begin{pmatrix} 1 & & 2 & & \cdots & 27 & \\ (1+k)mod & 27 & (2+k)mod & 27 & \cdots & (27+k)mod & 27 \end{pmatrix}$$

y así, las funciones son biyectivas, es decir, son permutaciones sobre Z_27.

2.2.4. Cifrado en bloque

Definición: Un criptosistema es llamado *criptosistema de cifrado en bloque* si el espacio de los mensajes y el espacio de los criptogramas son Σ^n, donde Σ es un alfabeto. A n se le llama la *longitud de bloque.*

Teorema 2.5. *En un criptosistema de cifrado en bloque, las funciones de encriptación son permutaciones.*

Dem:

Por definición de un criptosistema tenemos que cualquier función de encriptación es inyectiva. Y como Σ es un conjunto finito, así lo es también Σ^n. Además, sabemos que una función inyectiva que tenga como dominio y contradominio el mismo conjunto, en particular Σ^n, es biyectiva. □

Tenemos el mensaje $m = \sigma_1\sigma_2\cdots\sigma_n$ que queremos transmitir. Éste lo dividiremos en k bloques de r letras cada uno, de la siguiente manera: Por el algoritmo de la división sabemos que existe q, $b \in \mathbf{N}$ tal que $n = rq + b$, con $b = 0$ ó $0 < b < r$. Si $b = 0$ entonces hemos terminado y elegiremos

[1] Ver [Hers88] un libro de teoría de grupos.

[2] El uso de mayúsculas en el segundo renglón sólo es para identificar el criptograma.

$k = q$. Si $b > 0$ entonces tomamos $k = q + 1$; elegiremos al azar $r - b$ letras en Σ. Digamos $\sigma_{k,b+1}\sigma_{k,b+2}\cdots\sigma_{k,r}$. Luego definamos $m' = \sigma_1\sigma_2\cdots\sigma_n\sigma_{k,b+1}\cdots\sigma_{k,r}$

Ahora hagamos una notación mas funcional.

$$\begin{array}{rcl}
\sigma_{1,1} & = & \sigma_1 \\
\sigma_{1,2} & = & \sigma_2 \\
 & \vdots & \\
\sigma_{1,r} & = & \sigma_r \\
\sigma_{2,1} & = & \sigma_r + 1 \\
\sigma_{2,2} & = & \sigma_r + 2 \\
 & \vdots & \\
\sigma_{2,r} & = & \sigma_2 r \\
 & \vdots & \\
\sigma_{k-1,1} & = & \sigma_{(k-2)r+1} \\
\sigma_{k-1,2} & = & \sigma_{(k-2)r+2} \\
 & \vdots & \\
\sigma_{k-1,r} & = & \sigma_{(k-1)r} \\
\sigma_{k,1} & = & \sigma_{(k-1)r+1} \\
\sigma_{k,2} & = & \sigma_{(k-1)r+2} \\
 & \vdots & \\
\sigma_{k,b} & = & \sigma_{(k-1)r+b} \\
\sigma_{k,b+1} & & \\
 & \vdots & \\
\sigma_{k,r} & &
\end{array}$$

Ahora definimos $m_i = \sigma_{i,1}\sigma_{i,2}\cdots\sigma_{i,r}$ para $i = 1, 2, \ldots, k$.

Dado lo que acabamos de hacer, podemos decir que cualquier mensaje puede completarse de tal forma que se pueda dividir en bloques de r letras. Dicho en otra forma, podemos fijar primero la longitud del bloque y luego completar el mensaje como ya lo hemos visto cuando sea necesario.

Obsérvese que las letras elegidas aleatoriamente son añadidas al final del mensaje, puede también añadirse una cantidad de éstas a cada bloque.

Modos de operación para algoritmos de cifrado en bloque

En la práctica muchas veces se utiliza sólo un subconjunto de S_{Σ^n}. Los modos que presentamos aquí son ejemplos de ésto.

En las siguientes 4 subsecciones veremos modos de operación para algoritmos de cifrado bloque. Estos modos seguirán la siguiente notación a menos que se diga lo contrario:

Para referirnos a una letra del alfabeto Σ que es parte de un mensaje la denotaremos como σ. En caso que pertenezca a un criptograma la denotaremos como ρ.

El mensaje será $m = m_1m_2m_3\cdots m_k$ con $m_i = \sigma_{i,1}\sigma_{i,2}\cdots\sigma_{i,r}$ para cada $i \in \{1, \ldots, k\}$.

El criptograma será $c = c_1c_2c_3\cdots c_k$ con $c_i = \rho_{i,1}\rho_{i,2}\cdots\rho_{i,r}$ para cada $i \in \{1, \ldots, k\}$.

$\Sigma = \mathbf{Z}_2 = \{0, 1\}$

$\oplus : \mathbf{Z}_2 \times \mathbf{Z}_2 \longrightarrow \mathbf{Z}_2$ es la suma usual. Extendiendo esta operación a $\mathbf{Z}_2^n$, para cada $n \in \mathbf{N}$, tenemos:

$$\oplus : \mathbf{Z}_2^n \times \mathbf{Z}_2^n \longrightarrow \mathbf{Z}_2^n, \quad (\alpha_1, \ldots, \alpha_n) \oplus (\beta_1, \ldots, \beta_n) = (\alpha_1 \oplus \beta_1, \ldots, \alpha_n \oplus \beta_n)$$

Así pues si $\alpha \in \mathbf{Z}_2^n$ entonces $\alpha = -\alpha$
El espacio de llaves: $\mathbf{K} = S_n$
Las funciones para encriptar:

$$\mathbf{E}_k = \{e_\pi : \Sigma^n \longrightarrow \Sigma^n \| \pi \in S_n\}$$

donde $e_\pi \left(\sigma_1\sigma_2 \cdots \sigma_n = \sigma_{\pi(1)}\sigma_{\pi(2)} \cdots \sigma_{\pi(n)}\right)$
Las funciones para desencriptar:

$$\mathbf{D}_k = \{d_\pi : \Sigma^n \longrightarrow \Sigma^n \| \pi \in S_n\}$$

donde $d_\pi \left(\rho_1\rho_2 \cdots \rho_n = \rho_{\pi^{-1}(1)}\rho_{\pi^{-1}(2)} \cdots \rho_{\pi^{-1}(n)}\right)$.

ECB: *E*lectronic *C*ode*b*ook mode

En este modo, la forma de encriptar es muy sencilla, solo hay que utilizar la extensión $\check{e}$ de e_k, vista en 2.2.2, que a continuación se describe:

$$\check{e}_\pi : \Sigma^n \longrightarrow \Sigma^n \check{e}_\pi (m_1 m_2 \cdots m_k) = e_\pi (m_1)\, e_\pi (m_2) \cdots e_\pi (m_k)$$

De este modo tenemos $c_1 = e_\pi (m_1)$ Para desencriptar, la función también es muy sencilla:

$$\check{d}_\pi : \Sigma^n \longrightarrow \Sigma^n \check{d}_\pi (c_1 c_2 \cdots c_k) = d_\pi (c_1)\, d_\pi (c_2) \cdots d_\pi (c_k)$$

CBC: *C*ipher*B*lock *C*haining mode

Para este modo vamos a necesitar un *vector* o *n-palabra de inicialización* $IV \in \Sigma^n$, el cual puede hacerse público. Para obtener el criptograma c es necesario hacer los siguientes pasos:

1. $c_0 = IV$
2. $\forall j \in \{1, \ldots, k\}$, $c_j = e_\pi (c_{j-1} \oplus m_j)$.

Para desencriptar sólo tenemos que usar el siguiente teorema:

Teorema 2.6.

$$\forall i = 1, \ldots, k,\ m_i = c_{i-1} \oplus d_\pi (c_j)$$

Dem: Sea $j \in \{1, \ldots, k\}$
Luego

$$\begin{aligned} c_{j-1} \oplus d_\pi (c_j) &= c_{j-1} \oplus d_\pi (c_{j-1} \oplus m_j) \\ &= c_{j-1} \oplus (c_{j-1} \oplus m_j) \\ &= (c_{j-1} \oplus c_{j-1}) \oplus m_j \\ &= m_j \end{aligned}$$

Luego obtenemos el resultado□.

Observación 1: *La encriptación de un bloque del mensaje depende de los bloques que lo anteceden. Así, un bloque de mensaje puede ser encriptado en bloques de criptograma distintos en mensajes distintos.*

CFB: *C*ipher *F*eed*b*ack mode

En este modo también vamos a necesitar el vector de inicialización $IV \in \Sigma^n$. Para encriptar hacemos lo siguiente:

$I_1 = IV$; y para cada $j \in \{1, \ldots, k\}$

1. $\mathbf{O}_j = e_\pi(I_j)$
2. t_j será las primeras r letras de $\mathbf{O}_j$[3]
3. $c_j = m_j \oplus t_j$
4. I_{j+1} será igual a la parte sobrante al extraer t_j añadiendo al final c_j.[4]

Para desencriptar tenemos que hacer lo siguiente:

$IV = I_1$; y para cada $j \in \{1, \ldots, k\}$

1. $\mathbf{O}_j = e_\pi(I_j)$
2. t_j será las primeras r letras de
3. $m_j = c_j \oplus t_j$
4. $I_{j+1} = 2^r I_j + c_j mod\, 2^n$

OFB: *O*utput *F*eed*b*ack mode

Como en los dos modos anteriores, aquí también necesitamos un vector de inicialización $IV \in \Sigma^n$. La encriptación es muy similar al modo anterior. De hecho sólo cambia el último renglón:

De $I_{j+1} = 2^r I_j + c_j mod\, 2^n$ a $I_{j+1} = \mathbf{O}_j$

Observación 2: $\forall j \in \{1, \ldots, k\}$, $\mathbf{O}_j = e_\pi^j(IV)$

Dem: La prueba se hará por inducción sobre j.

Para cuando $j = 1$, tenemos por definición de $\mathbf{O}_1$ que ya se cumple.

Supongámoslo cierto para todo $j \leq l$, y probémoslo para $l + 1$. Tenemos pues que $\mathbf{O}_{l+1} = e_\pi(I_{l+1})$, y además $I_{l+1} = \mathbf{O}_l$. Luego, por hipótesis de inducción, tenemos $I_{l+1} = e_\pi^l(IV)$. Y así $\mathbf{O}_{l+1} = e_\pi^{l+1}(IV)$

Criptoanálisis a los criptosistemas de cifrado en bloque

2.2.5. Cifrado en flujo

Este criptosistema, en inglés "stream cipher", se puede pensar como una generalización del cifrado en bloque, equivalentemente, que el cifrado en bloque es un caso particular del cifrado en flujo con la longitud de los mensajes fija. La idea de este criptosistema es encriptar letra por letra el mensaje.

[3]Empezando de izquierda a derecha.

[4]O como está indicado en [Bch02] $I_{j+1} = 2^r I_j + c_j mod\, 2^n$

El criptosistema

Definición: Un *cifrado en flujo* es un 6-tupla (**P**, **C**, **K**, **L**, f, **E**, **D**) donde **P**, **C**, **K**, **E**, **D** tienen el mismo significado que en la definición de criptosistema, y además:

- $\mathbf{K} = L^m$ para algún $m \in \mathbf{N}$, donde L es un alfabeto, llamado *alfabeto de llaves.*
- $f : \mathbf{N} \times \mathbf{K} \longrightarrow L$ es la función generadora de la sucesión $z = (z_1, z_2, \ldots)$.
- Para cada $z = (z_1, z_2, \ldots)$ en $Z = \{z \| z \, es \, generada \, por \, k \, con \, f, \, para \, \text{algún} \, k \in K\}$ existen $e_z^i : \Sigma \longrightarrow \Sigma^5$ funciones invertibles.

  Definimos las funciones para encriptar como $e_z = (e_z^1, e_z^2, \ldots)$ que serán los elementos de **E**.

  Análogamente, definimos las funciones para desencripar: $d_z^i \in \mathbf{D}$ tal que $d_z \circ e_z = 1_{\mathbf{P}}$ y $d_z = (d_z^1, d_z^2, \ldots)$

  Donde $\hat{e}_z(\sigma_1\sigma_2 \cdots \sigma_n) = e_{z_1}(\sigma_1)e_{z_2}(\sigma_2) \cdots e_{z_n}(\sigma_n)$

En este caso **K** se puede ver como un conjunto de semillas que generarán una sucesión $z = (z_i)_{i \in \mathbf{N}}$, que será la llave completa para encriptar.

$\mathbf{P} = \mathbf{C} = \Sigma^*$
$\mathbf{K} = \Sigma^m$ para algún $m \in \mathbf{N}$

Sean $k = (k_1, k_2, \ldots, k_m) \in \mathbf{K}$ y $w = \sigma_1, \sigma_2, \ldots, \sigma_n \in \mathbf{P}$ la llave para encriptar y el mensaje a encriptar, respectivamente. Definimos z_i de la siguiente manera:
Si $i \leq m$ entonces $z_i = k_i$

Un ejemplo: Cifrado de Vigenère

Este criptosistema es de construcción tan sencilla cómo el de César, pero un poco más difícil de criptoanalizar. Fue inventado por un criptógrafo francés del siglo XVI, llamado Blaise Vigenére.

Este criptosistema se explicará de manera sencilla con algo de teoría de grupos aplicado al grupo aditivo $\mathbf{Z}_m$, con m un natural (normalmente mayor a 1).

El criptosistema

Fijamos una $r \in \mathbf{N}$ que será el tamaño de la llave, y tenemos el mensaje $m = \sigma_1\sigma_2 \cdots \sigma_n$, digamos que $n \geq r$, pues si $n \leq r$, en sí la llave podríamos decir que sólo sería un n-vector perteneciente al conjunto $\mathbf{Z}_{27}^n$. Tomando en cuenta la sección 2.2.4, el mensaje m podemos escribirlo de la siguiente manera: $m = m_1^1 \cdots m_r^1 \cdots m_1^t \cdots m_r^t$. El espacio de mensajes y el espacio de criptogramas resultan ser el mismo conjunto: $\mathbf{P} = \mathbf{C} = \mathbf{Z}_{27}^*$
El espacio de llaves es: $\mathbf{K} = \mathbf{Z}_{27}^r$
Si la llave $k = k_1k_2 \cdots k_r$, entonces las funciones para encriptar y para desencriptar son:

$$e_k(m) = (m_1^1 + k_1) \cdots (m_r^1 + k_r) \cdots (m_1^t + k_1) \cdots (m_r^t + k_r), \; y$$

$$d_k(m) = (m_1^1 - k_1) \cdots (m_r^1 - k_r) \cdots (m_1^t - k_1) \cdots (m_r^t - k_r)$$

[5] La i indica que va a encriptar el i-ésimo símbolo del mensaje ω y la z índica que cadena de llave se usa.

Criptoanalizándolo...

En este caso el tratar por fuerza bruta es una mala idea, pues simplemente para cuando $r = 5$, tenemos mas de $1{,}1 \times 10^7$. [6] Es más una mejor manera es proceder por el análisis estadístico y el índice de coincidencia. En este criptosistema se debe primero saber la longitud de la llave. Sabiendo ésta, el criptoanálisis es mucho mas sencillo pues se procede como si fueran varios criptogramas encriptados con el cifrado de César. Para encontrar la longitud de la llave, se puede usar el *test de Kasiski*.

Implementando mejorías

Lo que hace a este criptosistema vulnerable es en sí la longitud de la llave, pues si es lo bastante chica, uno puede aplicar como ya se dijo el análisis estadístico.

Sin embargo, una mejoría es usar llaves exactamente del tamaño del mensaje. Así pues, será imposible usar un análisis estadístico. Obviamente, cada mensaje que uno quiera enviar, deberá ser encriptado con una llave distinta. De aquí el nombre de *sistema de llave desechable*[7]

2.3. DES: Data Encryption Standard

2.3.1. Un poco de historia

El 15 de mayo de 1973, la NBS (siglas en inglés de National Bureau of Standards)[8] publicó una solicitud de un criptosistema para encriptar información no clasificada. El algoritmo debía ser seguro y barato, computacionalmente hablando. Primero fue propuesto el criptosistema Lucifer por IBM en 1974. Lucifer utilizaba una llave de 128 bits, la cual fue cambiada por una de 56 bits para dar paso al DES. Quien hizo los tests al Lucifer fue la National Security Agency. El 17 de marzo de 1975 fue publicado el algoritmo del DES en el Registro Federal, y el 15 de enero de 1977 fue adoptado por la NBS como estándar para aplicaciones no clasificadas.

Desde entonces, el DES se ha revisado cada 5 años aproximadamente, hasta 1998, año en que fue dado de baja como estándar. Una de las razones por las que se dió de baja fue el gran incremento en velocidad de las PC's, lo que hizo que pudiese romperse inclusive por fuerza bruta. La NIST empezó a trabajar en un sustituto al cual llamó Advanced Encription Standar, AES.

En 1998, Electronic Frontier Foundation gano la competencia RSA DES Challenge II-2, rompiendo el DES en menos de 3 días. La computadora de la EFF fue llamada DES cracker, la cual fue desarrollada con un presupuesto de $250 000US. Al siguiente año, Net construyó una red de 100 000 computadoras y con un DES cracker, lograron ganar el concurso RSA DES Challenge III, rompiendo el DES en 22 horas 15 minutos. Se ha logrado romper el DES con un proyecto que costó aproximadamente $1000 000 US en 3.5 horas.

2.3.2. Cifrados tipo Feistel

Definición: r es el número de rondas
$\mathbf{C} = \mathbf{P} = \{0, 1\}^{2t}$
$\mathbf{K}$ es el espacio de llaves.

[6] Dato sacado de Stinson [St95] en la página 13.

[7] En los textos en inglés se le dice "one-time-pad". Ver [Mz96] en la página 21, [St95] en la página 50, o [Kz98] en la página 2. A esta implementación se le llama *cifrado de Vernam*.

[8] Ahora ya no existe como tal la NBS, y pasó a ser National Institute of Standards and Tecnology

A la función $g : \mathbf{K} \longrightarrow \mathbf{K}^r$ la llamaremos *función generadora de las llaves de ronda*. Sea $p \in \{0, 1\}^{2t}$, escribiremos $p = (L, R)$ para indicar que p ha sido dividido a la mitad, esto es, si $p = p_1p_2 \cdots p_tp_{t+1} \cdots p_{2t}$, entonces, $L = p_1p_2 \cdots p_t$ y $R = p_{t+1} \cdots p_{2t}$. Con la notación anterior, la función para encriptar está dada por:

Para cada $1 \leq i \leq r$, $(L_i, R_i) = (R_{i-1}, L_{i-1} \oplus f_{k_i}(R_{i-1}))$ y el criptograma será $c = E_k = (R_r, L_r)$

2.3.3. El algoritmo

Usaremos $\Sigma = \{0, 1\}$ como alfabeto, $k' = r_0r_1 \cdots r_{63}$, donde $r_i \in \Sigma$ para toda $i \in \{0, 1, \ldots, 63\}$.

La llave

Las llaves del criptosistema DES constan de 8 bytes. Los 7 bits mas significativos de cada byte serán tomados en cuenta para encriptar y desencriptar, y el menos significativo será únicamente utilizado como bit de paridad, obligando a cada byte tener un número impar de unos.

Primero eliminamos los 8 bits de paridad y obtenemos $K = s_0s_1 \cdots s_{56}$, luego debemos sacar 16 subllaves, las cuales serán utilizadas en cada ronda de la encriptación. Para obtener estas 16 subllaves se necesitan dos permutaciones:

$$P_{56} : \Sigma^{56} \longrightarrow \Sigma^{56} P_{48} : \Sigma^{56} \longrightarrow \Sigma^{48}$$

Ya con las permutaciones, las subllaves se calculan de la siguiente manera:

1. Se calcula $P_{56}(K) = t_0t_1 \cdots t_{55}$.
2. Dividimos exactamente a la mitad los 56 bits: $(t_0t_1 \cdots t_{27}, t_{28}t_{29} \cdots t_{55})$
3. Se aplica la función corrimiento a la izquierda ($<<_n$: $\Sigma^m \longrightarrow \Sigma^m$, donde n indica cuantos bits a la izquierda se mueven) a ambas cadenas de 28 bits: $(z_0^iz_1^i \cdots z_{27}^i, z_{28}^iz_{29}^i \cdots z_{55}^i) = (<<_{n(i)} (t_0t_1 \cdots t_{27}), <<_{n(i)} (t_{28}t_{29} \cdots t_{55}))$, donde $i = 1, 2, 3, \ldots, 16$ indica el índice de la subllave k_i que se va a calcular. La n depende de la subllave y se obtiene de la sequencia de corrimiento (1, 2, 2, 2, 2, 2, 2, 1, 1, 1, 2, 2, 2, 2, 2, 2)
4. $k_i = P_{48}(z_0^iz_1^i \cdots z_{27}^i, z_{28}^iz_{29}^i \cdots z_{55}^i)$
5. Para calcular la siguiente subllave, en caso que se tenga que hacer, se debe tomar la cadena de 56 bits, $(z_0^iz_1^i \cdots z_{27}^i, z_{28}^iz_{29}^i \cdots z_{55}^i)$, y pasar a (3).

Con lo anterior, hemos obtenido las 16 subllaves $k_1, k_2, \ldots, k_{16}$, las cuales tienen una forma extendida:

$$k_i = k_{i,1}k_{i,2}k_{i,3} \cdots k_{i,48}$$

Encriptando

La función de encriptación es:

$$IP^{-1} \circ \Pi_{T_{16}} \circ \Theta(16) \circ \Pi_{T_{15}} \circ \cdots \circ \Theta(2) \circ \Pi_{T_1} \circ IP$$

donde:

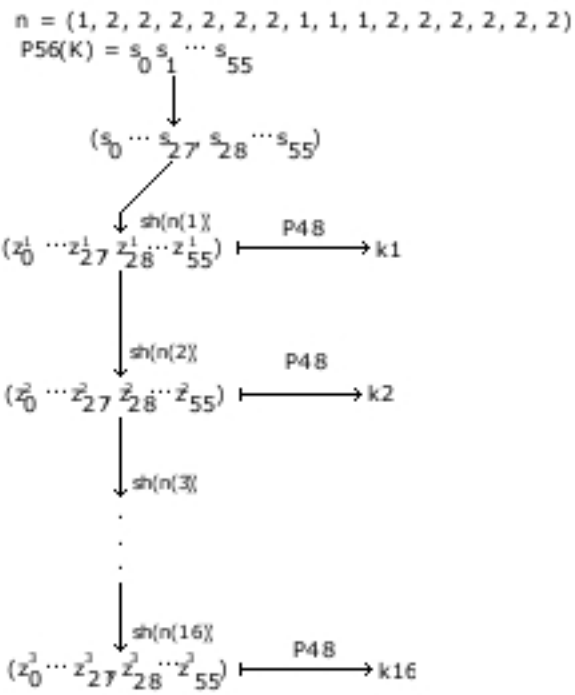

Figura 2.1: Esquema para la obtención de las 16 subllaves.

- $IP : \Sigma^{64} \longrightarrow \Sigma^{64}$ es una permutación, llamada *permutación inicial*.
- Π_{T_i} con $i = 1, 2, \ldots, 16$, es una función resultado de la composición de 2 funciones: $T_i : \Sigma^{32} \longrightarrow \Sigma^{32}$, y la suma con L_{i-1}; y
- $\Theta : \Sigma^{32} \times \Sigma^{32} \longrightarrow \Sigma^{32} \times \Sigma^{32}$

La función $T_i = S \circ Sum_{k_j} \circ E$. En algunos textos, E es llamada la función de expansión de 32 a 48 bits, luego se suma la respectiva subllave, k_j, y por último el resultado se mete a las 8 S-cajas, las cuales comprimen de nuevo a 32 bits. Ver figura 2.2 y figura 2.3 que es un ejemplo.

Las S-cajas se representan con 4 filas y 16 columnas. Cada fila tiene una sóla vez cada número del conjunto {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16}. Después de sumar la llave a $E(q_{32} \cdots q_{63})$ obtenemos 8 renglones con 6 bits cada renglón. De los 6 bits del primer renglón vamos a formar dos números binarios y meterlos a la S-caja S_0, de la siguiente forma:

- El primero con los 4 dígitos de en medio. Este número nos va a da la columna, digamos r.
- El segundo con los dos números de los extremos concatenando el de la derecha después del bit más a la izquierda. Este número nos va dar el renglón, digamos s.

Ahora, teniendo la pareja (r, s), buscamos el elemento (r, s) de la S-caja correspondiente, i.e., el que está en la columna r, contando del 0 al 15, y en el renglón s, contando de 0 al 3. Esté número se representa en binario, digamos $r_{1,1}r_{1,2}r_{1,3}r_{1,4}$.

Análogamente procedemos con los renglones 2, 3, 4, 5, 6, 7 y 8, los cuales interactuaran con las S-cajas S_1, S_2, S_3, S_4, S_5, S_6 y S_7, respectivamente.

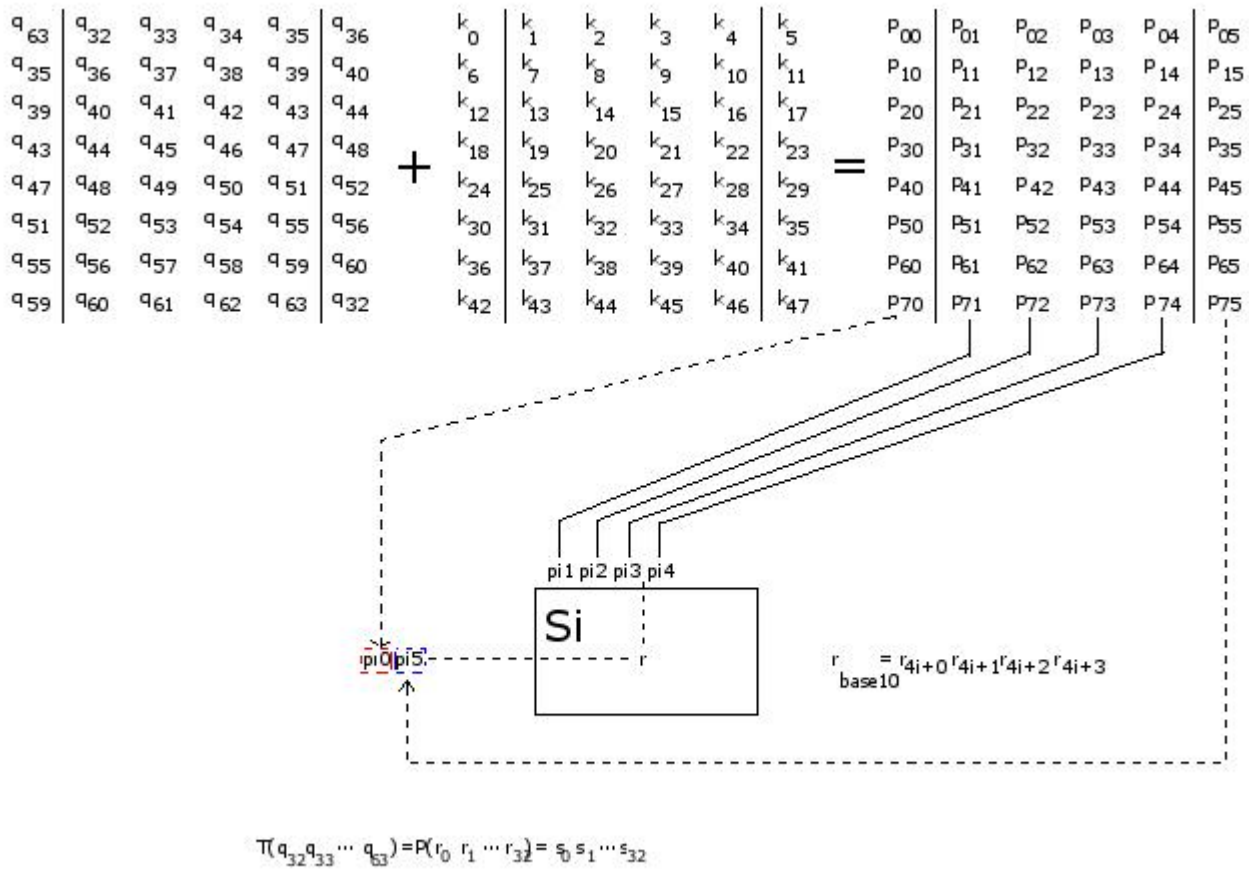

Figura 2.2: Esquema de como trabaja T_i

Ahora, juntamos las expresiones en binario que salieron de las S-cajas para obtener $r_{1,1} \cdots r_{1,4} \cdots r_{i,3} \cdots r_{i,4} \cdots r_{8,}$ Luego aplicamos una permutación P: $T_j = P(r_{1,1} \cdots r_{1,4} \cdots r_{i,3} \cdots r_{i,4} \cdots r_{8,4} \cdots r_{8,4})$, donde j es la ronda que se está calculando.

2.3.4. Criptoanálisis

2.3.5. Triple DES

El Triple DES requiere una llave de 192 bits, el triple de la longitud del DES. Esta llave se divide en tres partes, dando así tres llaves auxiliares k_1, k_2 y k_3, las cuales se utilizan en las tres repeticiones para encriptar. La forma de encriptar es tomar la llave k_1 y encriptar el mensaje m con el DES, luego aplicar el algoritmo para desencriptar utilizando la llave k_2, y por último aplicar el algoritmo para encriptar usando la llave k_3. La forma de desencriptar el criptograma es análoga a la encriptación.

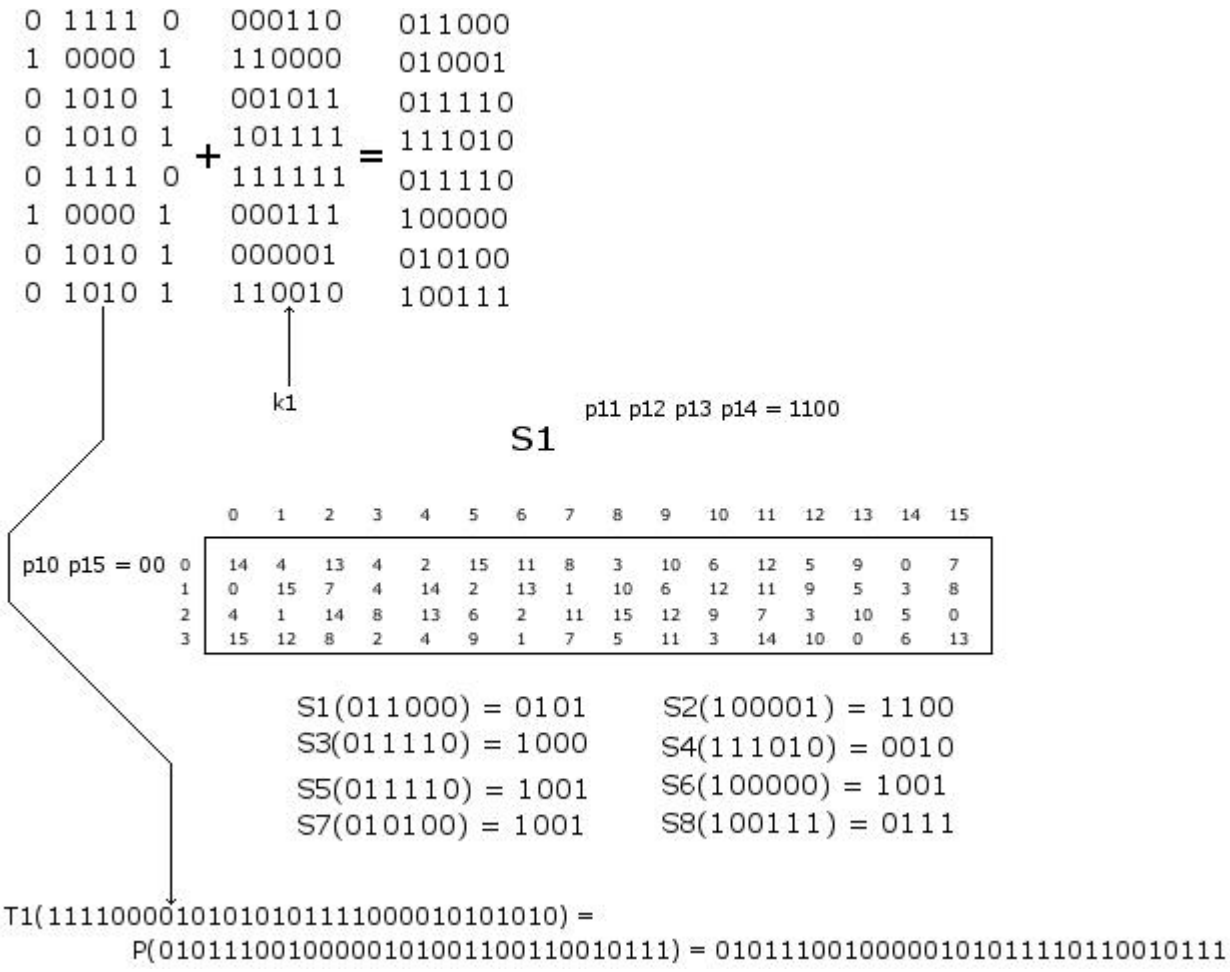

Figura 2.3: Ejemplo del cálculo de los Ti's

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	S1
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	S2
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	S3
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	S4
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	12	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	S5
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	S6
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	S7
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	S8
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

Figura 2.4: Ejemplo de S-Cajas.

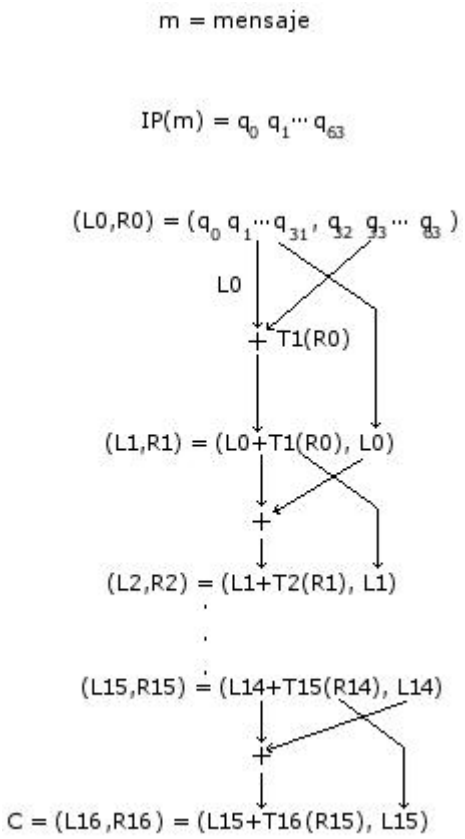

Figura 2.5: Esquema de las 16 rondas

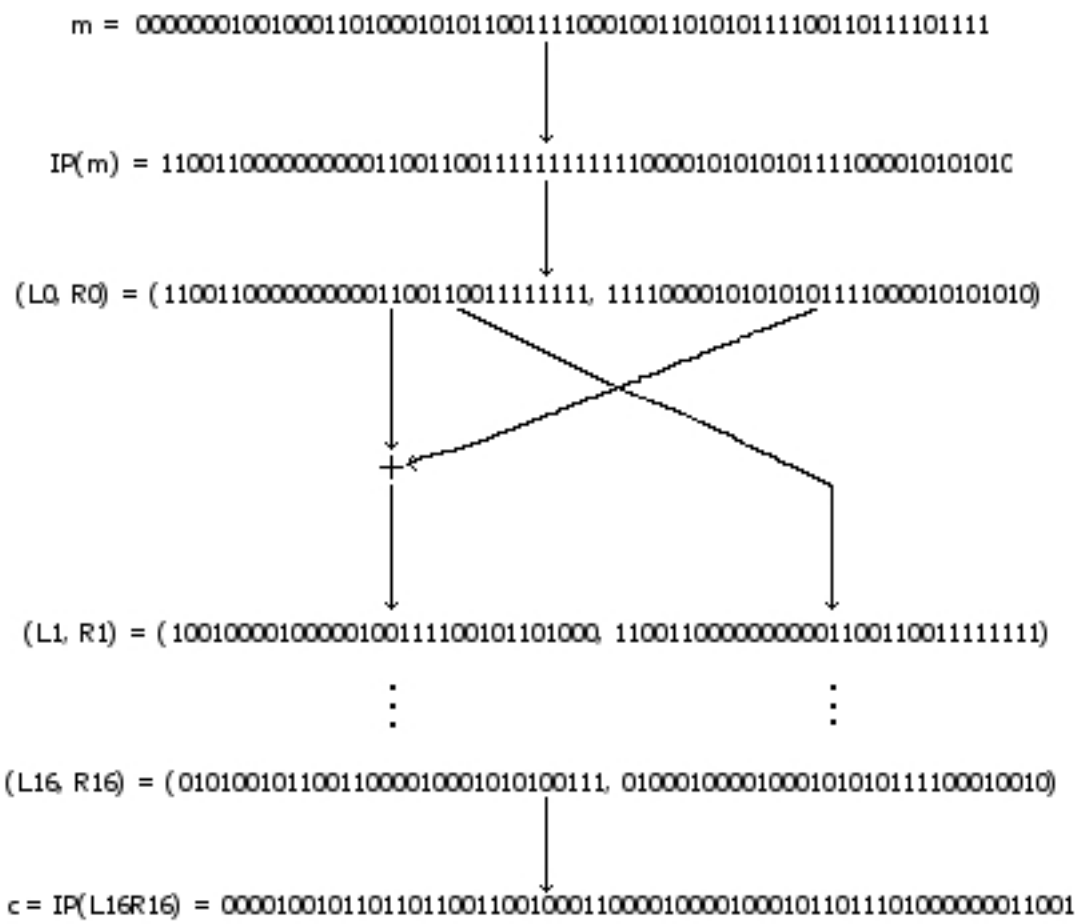

Figura 2.6: Ejemplo de cómo se calculan las rondas.

Capítulo 3

Criptosistemas de llave pública

En estos criptosistemas, la llave para encriptar es pública. Debido a ésto los criptosistemas de llave pública por sí sólos no proveen autentificación del origen de datos ni integridad de datos. Sin embargo, éstos se pueden lograr con el uso de técnicas como lo son los códigos de autentificación de mensaje y la firma digital. Una propiedad que caracteriza estos criptosistemas es que, en general, el

romper un criptosistema de llave pública es tan difícil como resolver un problema que se cree que es difícil. Por ejemplo, el problema de factorizar un número en sus factores primos y el problema del logaritmo discreto.

Cuando se criptoanaliza un criptosistema de llave pública, el criptoanálisis se hace para recobrar sistemáticamente dos cosas: Primero el mensaje a partir de un criptograma dado, y segundo la llave con la que se encriptan los mensajes. Cuando se ha obtenido un algoritmo eficiente que recobre sistemáticamente el mensaje se dice que el criptosistema ha sido roto y cuando se ha obtenido un algoritmo eficiente que recobre la llave se dice que el criptosistema ha sido roto completamente.

En el primer caso del párrafo anterior, se puede cambiar de llave. Sin embargo en el segundo ya no se puede hacer nada, ese criptosistema es inseguro de ahí en adelante.

En la práctica la encriptación con llave pública se hace para datos pequeños como números de tarjetas y PIN's, hacer firmas digitales, y para transportar llaves de criptosistemas de llave privada debido a su lenta encriptación.

3.1. Manejo de las llaves

El manejo de llaves en este tipo de criptosistemas es bastante más sencillo que en los de llave privada. Esta ganancia en la distribución tiene que pagarse de algún modo, y aquí el pago es en que se tiene que implementar un protocolo para saber quien es la persona que envia el mensaje [1]. Otro protocolo que se tiene, y quizá se deba implementar, es el de la obtención y protección de las llaves para poder encriptar.

Aquí en sí lo que se necesita es una caja, así como las de los ballet parking, en la cual se pondrán todas las llaves para poder encriptar el mensaje. Así, si Benito quiere transmitir un mensaje a Ana, lo que debe hacer es ir a la caja, extraer la llave de Ana, y por último utilizar dicha llave en la función para encriptar. Ver figura 3.1

[1] ver sección 3.3

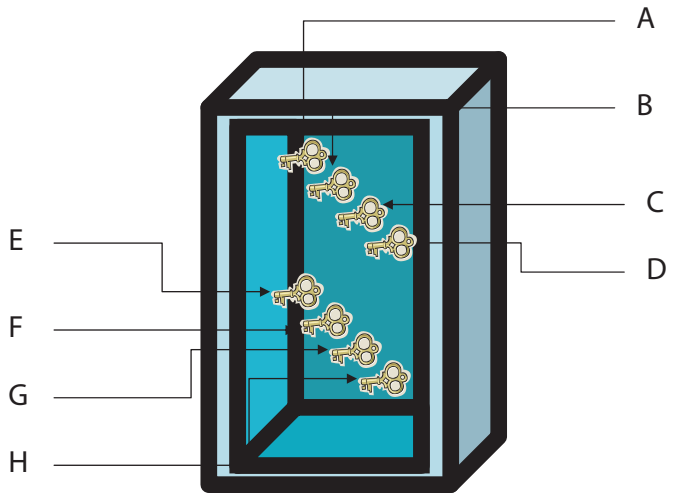

Figura 3.1: Caja de llaves

3.2. Criptosistemas de llave pública importantes

Los criptosistemas de llave pública normalmente basan su seguridad en problemas matemáticos aún no resueltos, o que son difícil de resolver, computacionalmente hablando.

3.2.1. RSA

El RSA es el primer criptosistema de llave pública, y fue inventado por Ron Rivest, Adi Shamir, Len Adleman. Este criptosistema se basa en la imposibilidad de factorizar un número grande (de 100 a 200 cifras decimales) en sus dos divisores primos.

Un poco de teoría

$=$e, $dondeeeslaidentidad.$

Teorema 3.1. *Sean G un grupo, $g \in G$ y $m \in \mathbf{Z}$. Entonces $g^m = 1$ si y solo si m es divisible por el orden de g en G.*

Teorema 3.2. *Sea $g \in G$ de orden finito k. Si $n \in \mathbf{Z}$, entonces $or(g^n) = or(g)/gcd(k, n)$*

El criptosistema

Para generar la llave, el receptor debe generar dos números primos grandes, p, $q \in \mathbf{N}$, aleatoria e independientemente. Luego calcula el producto de éstos, $n := pq$.

Para terminar sólo tiene que elegir un entero $e \in \mathbf{Z}$ tal que

$$1 < e < \varphi(n) = (p-1)(q-1)$$

y

$$gcd(e,\ (p-1)(q-1)) = 1$$

En pocas palabras, debe elegir un elemento de $\mathbf{Z}_{(p-1)(q-1)}$ que sea invertible. Ahora entonces, ha generado su llave pública: $(n,\ e)$ Para desencriptar, sólo tiene que encontrar el inverso en $\mathbf{Z}_{(p-1)(q-1)}$,

i. e., debe encontrar $d \in \mathbf{Z}$ tal que

$$1 < d < (p-1)\,(q-1)$$

y

$$de \equiv 1 mod \quad (p-1)\,(q-1)$$

Para encriptar debemos:

1. Obtener la llave.
2. Representar el mensaje $m \in \{0, 1, \ldots, n-1\}$.
3. Calcular $c = m^e \, mod \quad n$.
4. Enviar c.

Para recobrar el mensaje debemos usar la llave privada d y calcular $m = c^d \, mod \quad n$. El siguiente teorema demuestra el porque funciona este criptosistema.

Teorema 3.3. *Sea (n, e) una llave pública de RSA y d la correspondiente llave privada. Entonces*

$$(m^e)^d \, mod \, n = m$$

para cualquier entero m que cumpla $0 \leq m < n$

Demostración. Tenemos que $ed \equiv 1 \, mod \, (p-1)(q-1)$, luego existe un $a \in \mathbf{Z}$ tal que $ed = 1 + a(p-1)(q-1)$.

Luego $(m^e)^d = m^{ed} = m^{1+a(p-1)(q-1)} = m(m^{(p-1)(q-1)})^a$, o lo que es lo mismo $(m^e)^d = m(m^{(p-1)(q-1)})^a = m \, mod \, p$.

Por otro lado, si p no divide a m, entonces la congruencia se sigue por el teorema chico de Fermat. En otro caso, tendríamos que los dos números, $(m^e)^d$ y m, son divisibles por p.

Análogamente, obtenemos que $(m^e)^d = m \, mod \, q$. Luego al ser dos números diferentes p y q y al tener que $0 \leq m < n$ obtenemos lo que queríamos demostrar.□

El RSA y el problema de factorizar $n = pq$

Esta sección está dedicada a demostrar que el calcular la llave privada d sabiendo únicamente la llave pública (n, e) es tan difícil como factorizar n en sus dos factores primo p y q. No obstante sabemos que quizá pueda haber otra forma de obtener la preciada llave d.

Primeramente, tenemos que e y d son primos relativos a $\varphi(n) = (p-1)(q-1)$, luego $\varphi(n)$ es par y por lo tanto el producto ed es impar.

Sean

$$s = max\{t \in \mathbf{N} : 2^t | (ed-1)\}$$

y

$$k = (ed-1)/2^s$$

Ahora, siguiendo con la notación establecida en estos últimos renglones, tenemos:

Lema 3.4. *Sean $n, a \in \mathbf{Z}$ tal que $gcd(a, n) = 1$, y sea $k \in \mathbf{N}$, entonces el orden de la clase $a^k + n\mathbf{Z}$ en el grupo $(\mathbf{Z}/n\mathbf{Z})^*$ es un elemento del conjunto $\{2^i : 0 \leq i \leq s\}$.*

DEMOSTRACIÓN. Sea $a \in \mathbf{Z}$ primo relativo a n. Por el teorema anterior, $a^{ed-1} \equiv 1\, mod n$. Ahora, como $ed-1 = k2^s$, tenemos que $(a^k)^{2^s} \equiv 1\, mod\, n$. Luego, por el teorema 3.1, tenemos que el orden de $a^k + n\mathbf{Z}$ divide a 2^2, y de aquí el resultado.□

Teorema 3.5. *Sea $a \in \mathbf{Z}$ primo relativo a n. Si a tiene diferentes órdenes respecto a $\mathbf{Z}_p$ y a $\mathbf{Z}_q$, entonces $1 < gcd(a^{2^t k} - 1, n) < n$ para algún $t \in \{0, 1, \ldots, s-1\}$.*

DEMOSTRACIÓN. Por el lema tenemos que el orden de $a^k\, mod\, p$, $or_p(a^k)$, y el orden de $a^k\, mod\, q$, $or_q(a^k)$ están en el conjunto $\{2^i : 0 \leq i \leq s\}$. Sin perdida de la generalidad, podemos asumir que $or_p(a^k) > or_q(a^k)$.

Sea $or_q(a^k) = 2^t$. Luego $t < s$, $a^{2^t k} \equiv 1\, mod\, q$, pero $a^{2^t k} \equiv 1\, mod\, p$ por la suposición que hicimos. Por lo tanto $gcd(a^{2^t k}, n) = q$. □

Teorema 3.6. *El número de enteros a primos a n en el conjunto $\{1, 2, \ldots, n-1\}$ para los cuales a^k tiene distintos órdenes respecto a $\mathbf{Z}_p$ y a $\mathbf{Z}_q$ es de al menos $(p-1)(q-1)/2$.*

DEMOSTRACIÓN. Sea $g \in \mathbf{Z}$ tal que $g_p \in (\mathbf{Z}_p)^*$ y $g_q \in (\mathbf{Z}_q)^*$ son raíces primitivas, respectivamente. Este g exite por el teorema chino del residuo.

Primero trataremos el caso en el que $or_p(a^k) > or_q(a^k)$. Por el lema anterior tenemos que dichos órdenes son potencias de 2. Sean $x \in \{1, 2, \ldots, p-1\}$ y $y \in \{1, 2, \ldots, q-1\}$ dos números impares. Sea a el menor número no negativo solución de las congruencias:

$$a \equiv g^x\, mod\, p, \quad a \equiv g^y\, mod\, q$$

Luego $a \in \{1, 2, \ldots, n-1\}$. Ahora, por como se definió k arriba, tenemos que el $or_p(g^k)$ es potencia de 2; por otro lado, al tener que x es impar y $gcd(x, g^k) = 1$. Por el teorema 3.2, $or_p(a^k) = or_p(g^k)$. Además $or_q(a^k) \leq or_q(g^k) < or_p(a^k)$.

También tenemos que las soluciones del sistema congruencias son distintas a pares porque g es una raíz primitiva en $(\mathbf{Z}_p)^*$ y en $(\mathbf{Z}_q)^*$. Luego, hemos encontrado $(p-1)(q-1)/2$ enteros $a \in \{1, 2, \ldots, n-1\}$ que son distintos a pares, primos relativos a n y para los cuales el orden de $a^k\, mod\, p$ y $mod\, q$ son distintos.

Ahora bien, si $or_p(a^k) < or_q(a^k)$, la demostración es análoga.

Finalmente, supongamos que $or_p(a^k) = or_q(a^k)$

Elección sugerida de las llaves para encriptar y para desencriptar

Se recomienda que el módulo n sea de por lo menos 512 bits, aunque para evitar algoritmos como el NFS[2] y el QS [3] se recomienda usar n del orden de 768 bits.

Para evitar la factorización por curvas elìpticas, debe elegirse a los primos p y q del mismo orden, así, es recomendado que cada primo sea del orden de 512 bits y así $n = pq$ será de aproximadamente 1024 bits.

Es deseable elegir números pequeños para el exponente e, como por ejemplo $e = 3$, para hacer más eficiente el cálculo de c. Sin embargo, no se debe usar si se desea enviar un mismo mensaje a varias personas con el mismo exponente e y con módulos n_1, $n_2, \ldots$

También se sugiere no usar exponentes pequeños para e para cuando el mensaje $m < n^{1/e}$, pues puede ser recobrado al calcular la raíz e-ésima de $c = m^e$.

[2]Number Field Sieve

[3]Quadratic Sieve

3.2.2. ElGamal

Un poco de teoría

Para explicar este criptosistema utilizaremos algunos resultados conocidos en teoría de números, teoría de grupos y teoría de campos.

En lo siguiente, denotaremos a $\mathbf{F}_q$ como el campo con q elementos [4], de característica p. Teniendo también que $q = p^f$. Si $a \in \mathbf{F}_q^*$, denotaremos por $o(a) = min\{n \in \mathbf{N} \| a^n = 1 \in \mathbf{F}_q\}$ al orden de a.

Proposición 3.7. *El orden de cualquier elemento de* $\mathbf{F}_q^*$ *divide a* $q-1$.

Definición: Se dice que $g \in \mathbf{F}_q^*$ genera a (o es un generador de) $\mathbf{F}_q$ si $\mathbf{F}_q^* = \{g^i \| i \in \mathbf{N}\}$; equivalente, si $o(g) = q-1$.

Proposición 3.8. *Si* g *es un generador de* $\mathbf{F}_q^*$, *entonces* g^j *también lo es si y sólo si* $gcd(j,\ q-1) = 1$. *En particular,* $\mathbf{F}_q^*$ *tiene* $\varphi(q-1)$ *generadores.*

Demostración. Probaremos primeramente que si $a \in \mathbf{F}_q^*$ tiene orden d, entonces a^j tiene orden d si y sólo si $gcd(j,\ d) = 1$. Si $a \in \mathbf{F}_q^*$ tiene orden d, entonces, 1, a, a^2, ..., a^{d-1} son las d raíces del polinomio $x^d - 1$. Es decir, si hay algún elemento $b \in \mathbf{F}_q^*$ que tenga orden d, entonces está contenido en el subgrupo de $\mathbf{F}_q^*$ generado por a.

Ahora, si $gcd(j,\ d) = d' > 1$ entonces, $o\left(a^j\right) < d$ pues $\left(a^j\right)^{\frac{d}{d'}} = \left(a^d\right)^{\frac{j}{d'}} = 1$, donde $\frac{d}{d'},\ \frac{j}{d'} \in \mathbf{Z}$. Recíprocamente, si $gcd(j,\ d) = 1$ y $o\left(a^j\right) = d'$ entonces, $d' \leq d$, y $\left(a^{d'}\right)^j = 1$, $\left(a^{d'}\right)^d = 1$. Luego $o\left(a^{d'}\right) \| j$ y $o\left(a^{d'}\right) \| d$. Por tanto $o\left(a^{d'}\right) \| gcd(j,\ d) = 1$, y así $o\left(a^{d'}\right) = 1$. Es decir $d' \geq d$.

Por lo tanto $d' = d$ y de aquí el resultado al que queríamos llegar. De lo anterior podemos afirmar que si $d \| q-1$ entonces, existen 0 ó $\varphi(d)$ elementos con orden d. Ahora bien, si tomamos en cuenta el resultado de la página 79 de [Yan02], tenemos que

$$\sum_{d \| q-1} \varphi(d) = q-1$$

y así, $\forall d \| q-1$, existen $\varphi(d)$ elementos en $\mathbf{F}_q^*$ con orden d. En particular, para $d = q-1$, existen $\varphi(q-1)$ generadores de $\mathbf{F}_q^*$. Mas aún, si $g \in \mathbf{F}_q^*$ es un generador entonces, g^j es generador si y sólo si $gcd(j,\ q-1) = 1$.□

Proposición 3.9. *Todo campo finito tiene generadores.*

Definición: Sea G un grupo finito, y sean $b,\ y \in G$ tal que $y = b^x$ para algún $x \in \mathbf{Z}$. Definimos el *logaritmo discreto de* y *respecto a* b como cualquier $x \in \mathbf{Z}$ tal que $b^x = y$.

El criptosistema ElGamal y el intercambio de llave Diffie-Hellman utilizán la no existencia hasta el momento de un algoritmo que resuelva el problema del logaritmo discreto. Si bien, mas adelante se verá que el intercambio de llave Diffie-Hellman se basa en el problema del mismo nombre, está intimamente relacionado al problema del logaritmo discreto.

Definición: El *problema del logaritmo discreto* es encontrar la menor x no negativa de la definición de logaritmo discreto.

[4] Recuérdese que es único salvo isomorfismo.

Diffie-Hellman

Las dos partes, A, B, involucradas se ponen de acuerdo en un número primo grande p y en una raíz primitiva $g \in \mathbf{Z}_p$. A y B eligen un número aleatoriamente a, $b \in \{0, 1, \ldots, p-2\}$, respectivamente.

A calcula g^a, y envía el resultado a B. Se guarda a como llave privada.
B calcula g^b, y envía el resultado a A. Se guarda b como llave privada.
Ambos calculan $g^{ab} \in \mathbf{Z}_p$, el resultado es una llave común.

Definición: El *problema Diffie-Hellman* consiste en encontrar la llave común g^{ab} a partir de p, g, g^a, g^b

Si bien este problema se puede solucionar al encontrar el logaritmo discreto b de g^b, nadie ha demostrado que al resolver el problema Diffie-Hellman podemos resolver eficientemente el problema del logaritmo discreto.

El criptosistema

Para generar la llave, el receptor debe generar aleatoriamente un número primo p y un elemento $\alpha \in \mathbf{Z}_p^*$ que genere a este grupo multiplicativo. Luego debe elegir un entero $a \in \{0, 1, \ldots, p-2\}$ aleatoriamente, y calcular $\alpha^a \, mod \quad p$. La llave pública es (p, α, α^a).

La llave privada es a.
Para encriptar un mensaje debemos:

1. Obtener la llave pública.
2. Representar al mensaje $m \in \{0, 1, \ldots, p-1\}$
3. Elegir un entero $k \in \{0, 1, \ldots, p-2\}$ aleatoriamente.
4. Calcular $\gamma = \alpha^k$ y $\delta = m\,(\alpha^a)^k$
5. Enviar $c = (\gamma, \delta)$

Para recuperar el mensaje debemos:

1. Calcular $\gamma^{p-1-a} = \gamma^{-a} = \alpha^{-ak}$
2. Calcular $m = (\gamma^{-a} \cdot \delta)$

Seguridad del criptosistema ElGamal

Teorema 3.10. *Romper el criptosistema ElGamal es equivalente a resolver el problema Diffie-Hellman*

Demostración. Supongamos que tenemos un algoritmo A que resuelve el problema Diffie-Hellman, y tenemos un criptograma $\left(g^k, mg^{ak}\right)$ calculado con el criptosistema ElGamal. Recordemos que en este punto, el atacante conoce p, g, g^a y el criptograma $\left(g^k, mg^{ak}\right)$. Ahora, con el algoritmo A podemos conocer g^{ak}. Y así podemos calcular $m = \left(mg^{ak}\right)\left(g^{ak}\right)^{-1}$

Recíprocamente, si conocemos un algoritmo B tal que dado un criptograma cifrado con ElGamal, $\left(g^k, mg^{ak}\right)$, podemos saber m, es decir,

$$B\left(p,\, g,\, g^a,\, g^b,\, 1\right) = \left((g^a)^{log_g(g^b)}\right)^{-1}(1) \tag{3.2.1}$$

$$= \left((g^a)^b\right)^{-1} \tag{3.2.2}$$

$$= \left(g^{ab}\right)^{-1} \tag{3.2.3}$$

De aquí, obtenemos el resultado. □

3.3. Firma Digital

Las firmas digitales son usadas para firmar documentos electrónicos, y tienen propiedades similares a las manuales.

En la sección 3.1 hemos mencionado la necesidad de hacer un protocolo para saber quien es quien envía el mensaje. Precisamente, la firma digital es una opción para saber quien envia el mensaje, y así, no permitirá al emisor del mensaje negar que lo hizo.

3.3.1. ¿Para que sirve una firma digital?

Una firma digital nos va a servir para hacer el protocolo de autentificación, el cual se va requerir cuando tenemos un criptosistema de llave pública.

Para hacer este protocolo necesitamos un tipo de funciones que sea difícil calcular o computar su inversa sin saber algún tip —como por ejemplo una llave—, pero fácil de aplicar dicha función a todos los elementos del dominio. Este tipo de funciones son llamadas *hash*.

Definición: Por una *función hash* nos referimos a una función

$$h : \Sigma^* \longrightarrow \Sigma_n, \quad n \in \mathbf{N}$$

Una función hash, h, es llamada *función unidireccional* si es imposible obtener el inverso x tal que $h(x) = s$ para una imagen s dada.

Una *colisión* de h es una pareja $(x, x') \in D^2$, donde D es el dominio de h, tal que $x \neq x'$ y $h(x) = h(x')$

Una función hash, h, es llamada *resistente a colisión* si es prácticamente imposible calcular alguna colisión (x, x') de h.

En la siguiente sección se verá como se firma con distintos métodos o funciones.

3.3.2. Métodos para firmar

RSA

Algoritmo

Ana quiere mandar el documento m a Benito con firma digital para que Benito sepa quien se lo mandó o para que Benito no tenga duda que quien lo manda Es Ana

Ana tiene como llave secreta d.

El número primo que se utiliza en el criptosistema RSA es n.

La firma: $s = s(d, m) = m_d \, mod \, n$

Benito verifica que el documento fue enviado por Ana al calcular $s^e \, mod \, n = m^{ed} = m \, mod \, n$

ElGamal

Algoritmo

Ana genera un número primo largo y una raíz primitiva de $\mathbf{Z}/p\mathbf{Z}$.

Ana elige dos números $a, k \in \{1, 2, \ldots, p-2\}$, donde k es primo relativo a $p-1$ y calcula $A = g^a$. Su llave pública es (p, g, A), y la privada es a.

Sea $m \in \{0,1\}^*$ el documento a firmar. Sea $h : \{0,1\}^* \longrightarrow \{1, 2, p-2\}$ la función hash resistente a colisión, que además se conoce públicamente.

Ana calcula $r = g^k \in \mathbf{Z}/p\mathbf{Z}$, $s = k^{-1}(h(x) - ar) \in (\mathbf{Z}/p\mathbf{Z})^*$.

La firma para x de Ana es la pareja (r, s).

Algoritmo de firma digital

Este algoritmo mejor conocido como DSA [5] fue sugerido y estandarizado por la National Institute of Standards and Technology. Es una variante del método ElGamal.

Algoritmo

Ana elige un número primo $q \in [2^{159}, 2^{160}]$.

Ana elige un número primo largo p con las siguientes características:

- $2^{511} < p < 2^{512+64t}$ para alguna $t \in \{0, 1, \ldots, 8\}$
- q divide a $p-1$, esto implica que el grupo $\mathbf{Z}/p\mathbf{Z}$ tiene elementos de orden q.

Ana elige una raíz primitiva $x \, mod \, p$ y calcula $g = x^{(p-1)/q}$

Ana elige un número aleatoriamente $a \in \{1, 2, \ldots, q-1\}$ y calcula $A = g^a \in (\mathbf{Z}/p\mathbf{Z})^*$

La llave pública de Ana es (p, q, g, A) y su llave privada es a.

Si x es el documento a firmar, Ana usa una función hash resistente a colisión que es pública, $h : \{0,1\} \longrightarrow \{1, 2, \ldots, q-1\}$.

Ana elige un número aleatorio $k \in \{1, 2, \ldots, q-1\}$ y calcula $r = (g^k \, mod \, p) \, mod \, q$ y $s = k^{-1}(h(x) + ar) \, mod \, q$, donde k^{-1} es el inverso de k modulo q.

La firma es (r, s)

[5] Sigla en inglés de Digital Signature Algorithm

Capítulo 4

Generación de números primos

4.1. Números primos

Definición: : Un *número primo* es un número entero positivo mayor que 1 cuyos únicos divisores positivos son el 1 y él mismo.

Definición: : Un número es llamado *compuesto* si no es primo.

Teorema 4.1 (Fundamental de la Aritmética). [1] *Todo entero positivo n mayor que 1 puede escribirse de manera única, salvo permutaciones, como producto de números primos*

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$$

donde p_1, p_2, $\ldots p_k$ son primos distintos, y $\forall i = 1, \ldots, k$, α_i son enteros positivos.

4.2. Tests de primalidad y algoritmos que factorizan

Uno de los problemas en matemáticas mas estudiados históricamente hasta el momento es el saber si un número es primo o es compuesto. De la mano de este problema viene otro, para muchos mas interesante, y es el factorizar un número compuesto. Estos dos problemas se desarrollan al haber demostrado el Teorema 4.1.

Este último problema, si bien en secundaria podría decirse que es una tarea fácil, cuando hablamos de números grandes [2] el trabajo llega a ser tan arduo que ni una computadora puede factorizar algunos números en un tiempo relativamente pequeño; entendiendo ésto como "quisieramos tener la factorización en primos después de unos cuantos minutos y no depués de unos días". En esta sección se verán algunos tests de primalidad y algunos algoritmos para factorizar números[3].

4.2.1. Tests de primalidad

Estos tests tienen como objetivo decir si un número es primo o si es compuesto. En esta sección se explicarán 4 tests. El último, quizá el más importante de ellos, fue descubierto en 2002 por los indués Manindra Agrawal, Neeraj Kayal y Nitin Saxena. Formalmente:

[1] Ver demostración de este teorema en [Yan02]

[2] Ver sección 4.3

[3] En general, cada algoritmo para factorizar sólo factoriza bien a cierto tipo de números. Por ejemplo, el método $(p-1)$ es apropiado para cuando el número en cuestión es múltiplo de un número primo p, tal que $p-1$ tiene solamente divisores pequeños.

Definición: Un *test de primalidad* es un algoritmo que nos dice categóricamente si un número es primo o es compuesto.

Los tests de primalidad son algoritmos que ocupan mucho tiempo para decidir si un número es primo o no. Debido a esto, se han implementado otros algoritmos llamados *tests de primalidad probabilísticos*. Estos funcionan como a continuación se menciona:

1. Se construye un conjunto $W(n) \subseteq \mathbf{Z}n$, para cada n impar positivo al cual se le hace el test
2. Dado $\alpha \in \mathbf{Z}n$, se puede decir si $\alpha \in W(n)$ en tiempo polinomial
3. Si n es primo $W(n) = \emptyset$
4. Si n es compuesto, entonces $|W(n)| \geq \frac{n}{2}$

De esta forma, si elegimos $b \in \mathbf{Z}n$ al azar, por (4) tenemos que existe al menos un 50 % de probabilidad de que $b \in W(n)$. Así también, si el test se hace a b_1, b_2, ... $b_k \in \mathbf{Z}n$ y si resulta que para cada $i \in \{1, \ldots, k\}$, $b_i \notin W(n)$, entonces $P(n \text{ es primo}) \leq \frac{1}{2^k}$. En esta situación, diremos que b_i *pasó el test de primalidad*. Por otro lado, si para algún $i \in \{1, \ldots, k\}$ se cumple que $b_i \text{J} in W(n)$, entonces automáticamente n es compuesto. Debido a ésto a los $b \in W(n)$ se les llama *testigos de que n es compuesto*. Debido a la existencia de tests probabilísticos, nos vamos a referir como *tests de primalidad determinísticos* a aquellos definidos en 21. Muchos de estos últimos tests, sólo sirven para ciertos tipos de números, por lo cual, cuando un número pertenece a alguno de estos tipos quedará descartado.[4]

Test de Fermat

Teorema 4.2 (Pequeño teorema de Fermat). *Sea p un número primo, y sea $n \in \mathbf{Z}$ tal que $gcd(p, n)$. Entonces, $n^{p-1} \equiv 1 mod p$*

1. Input n y t.
2. Inicializamos $i = 1$
3. Generamos aleatoriamente un número $a_i \in \mathbf{Z}^+$ tal que $gcd(n, a_i) = 1$
4. Calcular $b = a_i^{n-1} \, mod \quad n$. Si $b = 1$ entonces es posible que sea primo y vaya al paso [5]; de otro modo, es compuesto y vaya al paso [6].
5. Actualizamos $i = i + 1$, si $i > t$ pasa a [6], si no pasa a [3].
6. Imprime el resultado y termina.

Este algoritmo en realidad no es un test de primalidad pues en la página 130 de [Bch02] se menciona que el número $561 = 3 \times 11 \times 17$, y además $a^{560} \equiv 1 mod 561$, para cada a como lo requiere el algoritmo.

[4] Ver 4.4 para mayor información.

Números de Carmichael

Para la definición de los números de Carmichael, necesitaremos la siguiente

Definición: Sean $n, a \in \mathbf{Z}$ tal que $1 < a < n$, n número compuesto y $gcd\,(a, n) = 1$. Decimos que n es *pseudoprimo a la base a* si $a^{n-1} \equiv 1 mod n$

Definición: Sea $n \in \mathbf{Z}^+$ un número compuesto. Decimos que n es un *número de Carmichael* si para cada $a \in \mathbf{Z}$ primo relativo a n, n es pseudoprimo a la base a.

El estudio formal de estos números empieza en el año de 1910, año en que Carmichael dió a conocer una conjetura[5], por la cual dichos números llevan su nombre:

"Hay una infinidad de estos números(de Carmichael)"

la cual fue demostrada en [AlGrPm]. Esto hace que el algoritmo de Fermat no sea en realidad un test de primalidad.

Proposición 4.3 (Caracterización de los números de Carmichael). *1. Si n es divisible por un cuadrado perfecto mayor que 1, entonces n no es un número de Carmichael.*

2. Sea $n \in \mathbf{N}$. Si n es un entero libre de cuadrados, entonces n es un número de Carmichael si y sólo si $p - 1 \| n - 1$ para cualquier primo $p \| n$.

Para la demostración de esta propocisión favor de remitirse a la página 128 de [Kb98].

Por último una proposición que, al igual que la anterior, la demostración puede encontrarse en [Kb98].

Proposición 4.4. *Un número de Carmichael es el producto de al menos tres primos distintos.*

Test de Solovay-Strassen

1. Input n que es el número que queremos saber si es primo o no, y t que es el número de iteraciones. Inicializamos $j = 1$
2. Elije un entero $1 < a < n - 1$
3. Si $\left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}} mod \quad n$ entonces es posible que sea primo y pasa a [4]. Si no es compuesto y pasa a [5].
4. Actualizamos $j = j + 1$. Si $j > t$ pasa a [5], de otro modo, pasa a [2]
5. Imprime el resultado y termina.

Test de Rabin-Miller

1. Inputs n y t.
2. Escriba a $n - 1 = 2^k m$ con m impar. Inicializamos $j = 1$
3. Elija un entero $1 \leq a \leq n - 1$ aleatoriamente
4. Calcule $b = a^m \, mod \quad n$

[5]Para mas información ver [Ca10] y [Ca12]

5. Si $b = 1$ entonces n es posiblemente primo y vaya al paso [6], de otra manera es compuesto y vaya al paso [7]

6. Actualizamos $j = j + 1$. Si $j > t$ entonces vaya al paso [7], de otro modo vaya al paso [3]

7. Imprima el resultado y termine.

En [Bch02] se prueba que hay a lo mas un 25 % de probabilidad de que encontremos una base b para la cual n sea pseudoprimo fuerte. Dicho de otra manera, si hacemos el test para t bases, y todas estas pasaron el test, entonces hay una probabilidad de $\frac{1}{4^t}$ de que sea primo.

Test AKS

[6]

Input: integer n ¿ 1.

1. If $n = ab$ para algunos $a \in \mathbf{N}$ y $b > 1$, output COMPOSITE.

2. Encontrar el menor r such that $or(n) > log2n$.

3. If $1 < gcd(a, n) < n$ for some $a \leq r$, output COMPOSITE.

4. If $n \leq r$, output PRIME.

5. For $a = 1$ to $\left\lfloor \sqrt{\varphi(r)} logn \right\rfloor$ do
   if $((X + a)^n \neq X^n + a\,(modX^r - 1,\, n))$, output COMPOSITE;

6. Output PRIME;

4.2.2. Algoritmos que factorizan

El plantear un algoritmo que factorice un número surge desde el momento en que es demostrado el Teorema Fundamental de la Aritmética [7]. En criptografia, en particular para el criptosistema RSA, es importante este problema pues la seguridad se basa en no tener (todavía) un algoritmo que factorice un número en tiempo polinomial. En los siguientes algoritmos n es el número que se quiere factorizar.

Algoritmo ρ de Pollard

En este algoritmo, B sólo es una cota superior para los exponentes.

1. Input: $n \in \mathbf{N}$

2. Inicializar $a = b = 2$.

3. Inicializar $i = 1$.

4. Calcular $a = a^2 + 1\, mod \quad n$, $b = b^2 + 1\, mod \quad n$ y $b = b^2 + 1\, mod \quad n$

5. Calcular $d = gcd\,(a - b,\, n)$

[6] Abreviación de Agrawal, Kayal y Saxena [AKS04]
[7] Ver 4.1

6. Si $1 < d < n$ entonces d es un divisor propio de n y termina el algoritmo, de otro modo vaya al paso [7].

7. Si $d = n$ terminar el algoritmo, de otro modo vaya al paso [2]

Método $p-1$ de Pollard

En este algoritmo se utiliza la variable B como una cota superior.

1. Input: n, $B \in \mathbf{N}$

2. Inicializamos las variables $j = a = 2$

3. Calculamos $a = a^j \, mod \quad n$

4. Calculamos $d = gcd\,(a-1,\, n)$

5. Si $1 < d < n$ entonces d es un factor de n.

6. Actualizamos $j = j + 1$. Si $j <= B$ ir a paso [3], en otro caso terminar.

Algoritmo de la criba cuadrática

1. Input n, $t \in \mathbf{N}$

2. Seleccionar el conjunto factor base $S = \{p_1, p_2, \ldots, p_t\}$, donde $p_1 = -1$ y p_j, $j \in \{2, \ldots, t\}$, es el $j-1$-ésimo número primo p para el cual n es un residuo cuadrático modulo p

3. Calcular $m = \sqrt{n}$

4. (Conjuntar $t+1$ pares (a_i, b_i). Los valores de x son elegidos en orden 0, ± 1, ± 2, ...)
   Inicialice $i = 1$. Mientras $i \leq t+1$ haga lo siguiente:

   a) Calcule $b = q\,(x) = (x+m)^2 - n$, y probar, por simple división por los elementos en S, cuando b es p_t-suave. Si no, elija un nuevo x y repita el paso 3.1.

   b) Si b es p_t-suave, y digamos que $b = \prod_{j=1}^{t} p_{j=1}^{e_{ij}}$, entonces calculemos $a_i = x + m$, $b_i = b$ y $v_i = (v_{i1}, v_{i2}, \ldots, v_{it})$, donde $v_{ij} = e_{ij} \, mod \quad n$ para $1 < j < t$

   c) $i = i + 1$

5. Usar algebra lineal sobre $\mathbf{Z_2}$ para encontrar un subconjunto no vacío $T \subseteq \{1, 2, \ldots, t+1\}$ tal que $\sum_{i \in T} v_i = 0$

6. Calcular $x = \prod_{i \in T} a_i \, mod \quad n$.

7. $\forall j \in \{1, \ldots t\}$, calcular $l_j = \left(\sum_{i \in T} e_{ij}\right)/2$.

8. Calcular $y = \prod_{j=1}^{t} p_j^{l_j} \, mod \quad n$.

9. Si $x \equiv \pm y \, mod \quad n$, entonces encontrar encontrar otro subconjunto no vacío $T \subseteq \{1, 2, \ldots, t+1\}$ tal que $\sum_{i \in T} v_i = 0$, e ir al paso [5]. (En el caso tal que dicho subconjunto T no exista, remplace algunos de los pares (a_i, b_i) con nuevos pares (paso [3]), y vaya al paso [4])

10. Calcule $d = gcd(x - y, n)$ y return(d).

NFS

[8]

1. Selecionar un campo de números. Se fabricará un polinomio $f(x)$ **Z**-irreducible, tal que $f(m) = 0\, mod \quad n$, para algún m conocido. Luego, tenemos un homomorphismo de

$$\mathbf{Z}[r] \xrightarrow{\varphi} \mathbf{Z}/n\mathbf{Z},$$

donde r es una raíz de f. En este momento, el mejor método conocido para elegir el polinomio es el dado por Brian Murphy y Peter montgomery en la tesis doctoral del primero.

2. Hacer la criba. En este paso debemos encontrar pares (a, b) de enteros pequeños tal que (1) $a - bm$ tenga sólo factores primos pequeños, y el ideal $\langle a - br \rangle$ tiene sólo factores primos de norma pequeña.

3. Construimos una matriz sobre F_2 tal que los vectores en el kernel correspondan a las parejas (a, b), cuyos productos darán dos cuadrados perfectos congruentes en $\mathbf{Z}/n\mathbf{Z}$. Este problema fue resuelto con un método iterativo por Peter Montgomery.

4. Dados dos cuadrados perfectos de los pasos anteriores y sus factorizaciones en primos, calcularemos sus raíces cuadradas para factorizar finalmente a n.

Algoritmo de factorización con curvas elípticas

Las implementaciones de estos algoritmos se pueden encontrar en:
www.alpertron.com.ar/ECM.html
http://www.loria.fr/ zimmerma/papers/ecm-entry.pdf
http://www.crypto-world.com/FactorPapers.html
http://www.fermigier.com/fermigier/elliptic.html.en
http://www.inria.fr/rrrt/rr-1547.html
http://www.rsasecurtiy.com/rsalabs/node.asp?id=2013
http://www.maplesoft.com/applications/app_center_view.aspx?AID=97
http://www.best.tuke.sk/simka/download/pub/sim_05.pdf

4.3. ¿Qué tan grandes son los números grandes?

El nombre de esta sección es una pregunta que todos los que han oido o leido acerca de números grandes nos hacemos. Y es que la verdad, aunque las matemáticas no lo quieran, la definición de un número grande depende de la evolución computacional, la cual hace posible calcular distintas funciones que tienen que ver con números enteros. Por ejemplo, hace 30 años un número grande podía haber consistido de 50 cifras decimales; sin embargo, con la evolución a grandes pasos de la computación, ahora necesitamos un número de 200 cifras decimales para que la computadora se tarde días enteros en hacer cómputo.

4.4. Generación de números primos

Este proceso por computadora se puede decir que es relativamente sencillo, pero algo costoso.

El problema es generar números con las siguientes propiedades:

[8] del inglés **N**umber **F**ield **S**ieve

- El número es primo.
- El número no es primo de Fermat ni primo de Mersenne.
- Debe generarse en sistema binario.
- Debe tener de 150 a 200 dígitos decimales, lo cual se traduce en aproximadamente 500 a 650 bits. A veces se puede pedir de más bits.

Cuadro 4.1: Tabla de requisitos para generar un número primo

Lema 4.5. *Para un $n \in \mathbf{N}$ k-bit, que se requiere generar para que se aplique algún test de primalidad, sólo hay que generar $k-2$ bits.*

Demostración. Para esto tenemos que tomar en cuenta que el bit más significativo y el bit menos significativo serán unos, pues un cero en el bit más significativo querría decir que no es k-bit, y un cero en el bit menos significativo querría decir que el número dado es par, y así también, que no es primo.

4.4.1. Algoritmos para generar números primos

En esta sección, k denotará el número de dígitos que deseamos que tengan los números aleatorios, y m será el número de números aleatorios que deseamos que se generen.

Método de Von Neumann

[9]

1. Se elige un número n_0 con k dígitos aleatoriamente.
2. Se eleva al cuadrado n_i para obtener un número M de $2k$ cifras.
3. Se actualiza, $i = i + 1$, y se toman los k dígitos de en medio de M como el nuevo número aleatorio n_i.
4. Si $i < m$ kpase a [3], de lo contrario pare.

Generador congruencial lineal

1. Input a, b, n, x_0, m; inicializar $j = 1$
2. Calcular $x_j = (ax_{j-1} + b)\ mod \quad n$
3. Incremente $j = j + 1$. Si $j \geq k$ entonces pare, de lo contrario, vaya al paso [2]

Generador de bits por RSA

1. Input x_0, $k \in \mathbf{N}$. Inicializar $j = 1$
2. Elegir aleatoriamente dos primos p y q en el rango $2^k \leq p$, $q < 2^{k+1}$ y calcular $n = pq$

[9] middle square method

3. Elegir $e \in [1,\, n]$ tal que $gcd\,(e,\, \varphi\,(n)) = 1$.
4. Calcular $x_j = (x_{j-1})^e \; mod \quad n$
5. Calcular $z_j = x_j \, mod \quad 2$
6. Actualizar $j = j + 1$. Si $j > k$ pare, de lo contrario vaya al paso [4]

Generador Blum-Blum-Shub

1. Elegir dos números primos p y q de $l/2$bits y tal que $p \equiv q \equiv 3 \, mod \quad 4$, y definir $n = pq$.
2. Sea s_0 un residuo cuadrático módulo n, e inicialicemos $j = 1$.
3. Calculamos $s_{j+1} = s_j^2 \, mod \quad n$
4. Calculamos $z_j = s_j \, mod \quad 2$
5. Actualizamos $j = j + 1$; si $j > k$ entonces pare, de lo contrario vaya al paso [3]

Generador por logaritmo discreto

1. Elegimos un número primo p con l bits y un elemento primitivo α
2. Elegimos un número aleatorio $x_0 \in \mathbf{Z}_p^*$ e inicializamos $i = 1$
3. Calculamos $x_{i+1} = \alpha^{x_i} \, mod \quad p$
4. Si $x_i < p/2$ entonces definimos $z_i = 0$. De lo contrario definimos $z_i = 1$
5. Actualizamos $i = i + 1$; si $i > k$ entonces pare, de lo contrario vaya al paso [3]

Capítulo 5

Conclusiones

El objetivo de este trabajo es estudiar los esquemas criptográficos de una manera formal y tratando de explicarlos de una manera fácil para aquellas personas interesadas en la criptografía. No nos extendimos mucho en el criptoanálisis pues no es el objetivo principal de este trabajo.

Dentro de los sistemas criptográficos podemos encontrar dos grupos: los de llave privada y los de llave pública. Dicha división se ha hecho por la facilidad para desencriptar en los criptosistemas de llave privada sabiendo la llave para encriptar, dándose en general, que se usa la misma llave para la desencripción.

Pudimos ver las ventajas de los criptosistemas de llave pública respecto a los de llave privada, y viceversa, de tal forma que podemos decir que para una mejor efectividad computacional, sin comprometer el mensaje, deben combinarse al menos un criptosistema de cada tipo, utilizando el de llave pública para encriptar la llave del de llave privada. De una manera mas sencilla, se transforma un canal inseguro a canal seguro en un intervalo de tiempo.

Los criptosistemas de llave privada pueden ser muy útiles cuando se enseña criptografía. Son fáciles de entender pues se usan matemáticas básicas (suma modular, permutaciones, etc.)y en general sus algoritmos son bastante mas sencillos que los de llave pública.

Los criptosistemas de llave pública se basan en problemas que hasta el momento son difíciles de calcular usando una computadora, esto nos puede dar una falsa idea de seguridad, pues nadie sabe si exista un algoritmo muy eficiente que resuelva algunos de los problemas mencionados. Sin embargo, mientras ese algoritmo no se dé a conocer o no se encuentre, los algoritmos de llave pública seguiran siendo seguros.

La siguiente tabla muestra los criptosistemas que estudiamos, que utilizan para encriptar y que tipo de criptosistema es:

* Según los textos consultados, para el cifrado en flujo se requieren únicamente funciones, es decir, nosotros pudieramos proponer un criptosistema que sea en flujo con, por dar un ejemplo, autómatas celulares.

En la comparación entre los criptosistemas de llave privada y los de llave pública pusimos especial atención en el manejo de llaves, teniendo como resultado que tienen mucha mas eficiencia el manejo en los criptosistemas de llave pública. Sin embargo, esa eficiencia se tiene que pagar con la firma digital y con un protocolo para evitar que alguien no autorizado cambie una llave pública de la caja de llaves.

Si bien se confía en que no se pueden resolver fácilmente los problemas base de los sistemas de llave pública, la verdadera confianza la da la práctica con números particulares, para así evitar los algoritmos que pueden resolver parcialmente el problema como es la factorización con curvas elípticas en el caso del RSA.

Nombre	Problema u objetos matemáticos base	Tipo de criptosistema
Cifrado afín	funciones afines en anillos	Llave privada
Cifrado de Hill	Anillo de matrices	Llave privada
Cifrado por permutación	Permutaciones	Llave privada
Cifrado por substitución	Permutaciones	Llave privada
Cifrado en bloque	Permutaciones	Llave privada
Cifrado en flujo	Funciones*	Llave privada
Cifrado de Vigenère	Suma modular	Llave privada
DES	Redes de Feistel	Llave privada
RSA	Factorización de un número en sus factores primos	Llave pública
ElGamal	Logaritmo discreto	Llave pública
Diffie-Hellman	Logaritmo discreto	Llave pública

El avance tecnológico ha hecho que dejemos de utilizar criptosistemas como el DES, cuya llave es lo suficientemente chica para poder desencriptar los mensajes con fuerza bruta, dando así, paso a criptosistemas mas complejos como el RSA, ElGamal, etc. Así también, los avances en el criptoanálisis y la necesisdad de minimizar el uso de canales seguros para la transmisión de datos y/o llaves para encriptar han hecho que prolifere el uso de los criptosistemas de llave pública, pues se confía, quizá en demasía, que el mensaje llegará a su destinatario aunque no se envie por un canal seguro.

Los adelantos matemáticos también han hecho que los esquemas criptográficos tengan que evolucionar con llaves más grandes como en el caso del cifrado de Vigenère, con restricciones en los números primos como en el RSA, o como los modos de encriptación para criptosistemas en bloque. Entre los adelantos matemáticos se encuentran los distintos algoritmos que se han desarrollado para factorizar números en sus factores primos, así como este se han desarrollado en otros problemas matemáticos, sin embargo, por la lentitud de los algoritmos no han hecho que criptosistemas como el RSA y ElGamal, entre otros, queden como inseguros.

Se puede plantear varios y distintos criptosistemas los cuales pueden caer en dos rubros. El primero es que sea muy fácil de desencriptar y no sea útil. El otro es que ademàs de ser muy difícil de desencriptar también utilice mucho tiempo para poder encriptar, lo cual lo hace impráctico.

Bibliografía

[AlGrPm] W. R. Alford, A. Granville y Carl Pomerance, *There are infinitely many Carmichael numbers*. Department of Mathematics, University of Georgia, Athens, Georgia 30602, U.S.A.

[Atk91] Atkin, A. O. L., Morain, Fl. *Finding Suitable Curves for the Elliptic Clurve Method of Factorization*. Rapport de recherche d'INRIA. November 1991.

[AKS04] Agrawal, Manindra, Neeraj Kayal, Nitin Saxena. *PRIMES is in P* Annals of Mathematics **160**(2): 781-793 (2004)..

[Bch02] BUCHMANN, Johannes A. *Introduction to Cryptography*- `Undergraduate texts in mathematics`. Springer-Verlag, E. U. A., New York,2002.

[Ca10] R. D. Carmichael, *Note on a new number theory function*, Bull. Amer. Math. Soc. **16** (1910), 323-238.

[Ca12] R. D. Carmichael, *On composite numbers P which satisfy the Fermat congruence* $a^{P-1} \equiv 1 \, mod \, P$, Amer. Math. Monthly **19** (1912), 22-27.

[Cert06] Certicom, *A Brief History: The origins of public key criptography and ECC*, http://download.certicom.com/index.php?action=res,cc&issue2-1&&article=4

[DH76] W. Diffie and M.E. Hellman, *New directions in cryptography*. IEEE Transactions on Information Theory, IT-22: 644-654, 1976. http://citeseer.ist.psu.edu/diffie76new.html

[ElG85] . ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions on Information Theory, IT-31: 469-472, 1985, http://crypto.csail.mit.edu/classes/6.857/papers/elgamal.pdf

[Gril99] GRILLET, Pierre A. *Algebra*. A Wiley-Intersciencie publication, JOHN WILEY & SONS, INC., E. U. A., New York, 1999.

[Hal85] HALMOS, Paul Richard. *I Want to Be a Mathematician*. MAA Spectrum, E. U. A., Washington, 1985.

[Hers88] HERSTEIN, I.N. *Álgebra Abstracta*. Grupo Editorial Iberoamérica, México, 1988.

[HpUll02] HOPCROFT, John E., Rajeev Motwani, Jeffrey D. Ullman. *Introducción a la Teoría de Autómatas, Lenguajes y Computación*. Addison Wesley, España. Segunda edición, 2002.

[Ka96] KAHN, David. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*.Scribner, U. S. A., New York. Segund edición, 1996.

[Krf1883] KERCKHOFFS. *La Cryptographie Militaire*.Journal des Sciences Militaires. 1883. http://www.petitcolas.net/fabien/kerckhoffs/crypto_militaire_1.pdf

[KgPk00] KERNIGHAN, Brian W. y Pike, Rob. *La práctica de la programación*. PEARSON EDUCACIÒN, México, 2000.

[Kb98] KOBLITZ, Neal. *A Course in Number Theory and Cryptography*- `Graduate Texts in Mathematics`. Springer-Verlag, U. S. A., New York. Segunda Edición, 1998.

[Kz98] KOBLITZ, Neal. *Algebraic Aspects of Cryptography/Neal Koblitz. With an appendix on hyperelliptic curves by Alfred J. Menezes* - `Graduate Texts in Mathematics`. Springer-Verlag, Alemania, Berlin. 1998 (Algorithms and computation in mathematics; Vol. 3)

[Lucpdf] LUCENA, Manuel J. *Criptografía y Seguridad en Computadores*. Creative Commons. España. Cuarta edición, versión 0.5.

Puede bajarse de: http://www.telefonica.net/web2/lcripto.html

[Mz96] MENEZES, Alfred, P. van Oorschot, S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[RSA78] R. Rivest, A. Shamir, and L. Adleman.*A method for obtaining digital signatures and public-key cryptosystems*, Comunications of the ACM **21** (1978), pp. 120-126

[SChKe] SCHNEIER, Bruce and John Kelsey. *Unbalanced Feistel Networks and Block-Cipher Design*.

[St95] STINSON. *Cryptography: Theory and Practice*. CRC Press LLC. E. U. A. 1995

[Sue] SUETONIO. *La vie des 12 Césars*. http://remacle.org/bloodwolf/historiens/suetone/table.html

[Us63] USPENSKY, James Victor. *Theory of Equations*. Mc Graw Hill, U. S. A., 1963.

[Yan02] YAN, Song Y. *Number Theory for Computing*. Springer, Germany, Berlin. Segunda edición, 2002.