Criptografía

José Angel de Bustos Pérez

<jadebustos@augcyl.org>

<joseangel.bustos@hispalinux.es>

Introducción

¿Que es la criptografía?

La principal aplicación de la criptografía es la de proteger información para evitar que sea accesible por observadores **NO** autorizados, proteger datos, pero también tiene otras aplicaciones.

Aplicaciones de la criptografía

- Modificar un mensaje de tal forma que sea completamente ilegible a no ser que se posea la clave para volver a ponerlo en su estado original.

- Verificar que un mensaje **NO** ha sido modificado **INTENCIONADAMENTE** por un tercero.

- Verificar que "alguien" es quien realmente dice ser.

Tipos de criptografía

- **Criptografía simétrica**, también conocida como "*criptografía clásica*" o de "*llave privada*" . Este tipo de criptografía es anterior al nacimiento de los ordenadores.

- **Criptografía asimétrica**, también conocida como "*criptografía moderna*" o de "*llave pública*" . Este tipo de criptografía se desarrollo en los años 70 y utiliza complicados algoritmos matemáticos relacionados con *números primos* y *curvas elípticas*.

- **Esteganografía**, se trata de ocultar información sensible a simple vista contenida en otro tipo de información. Por ejemplo en un archivo gráfico utilizar el bit menos significativo del color de todos y cada uno de los puntos de la imagen para transmitir una información. Alguien que vea la imagen no se dará cuenta de nada ya que el cambio que se produce en la imagen no es significativo.

Controversias

Controversias

Respecto al uso de la criptografía existe una gran controversia sobre si se debe o no utilizar "*critografía fuerte*". Este tipo de criptografía es muy segura y es practicamente imposible desencriptar mensajes encriptados con este tipo de criptografía.

La controversia surge ya que este tipo de criptografía podría ser utilizado por organizaciones criminales para asegurar sus comunicaciones y de esta forma poder cometer sus actividades criminales de una forma más segura.

Hay "interesados" en que este tipo de criptografía no se utilize dando argumentos como el anterior. Argumento muy contundente. Pero hay otro argumento también contundente y es el DERECHO A LA INTIMIDAD.

Con respecto a los algoritmos utilizados también existe controversia. Deben ser públicos o deben ser mantenidos en secreto? Hay opiniones para todos los gustos, pero la única forma de asegurar que un algoritmo criptográfico es fiable es siendo público y permitiendo a la comunidad científica examinarlo para buscar agujeros y arreglarlos cuando fuese posible. Tengase en cuenta que si se mantiene en secreto tarde o temprano será descubierto.

También se puede discutir sobre el software utilizado en criptografía:

- ¿Software propietario?

- ¿Software libre?

Teniendo en cuenta que el software que utilizamos conocerá nuestras claves, luego si tiene algún fallo o tiene alguna "funcionalidad" no documentada podría comprometer nuestra seguridad. Las ventajas del software libre son:

1. Al estar el código fuente disponible cualquiera, con los conocimientos necesarios, puede examinar el código y comprobar que hace lo que realmente debería hacer.

2. Al estar el código fuente disponible se pueden encontrar fallos y agujeros de seguridad que pueden ser resueltos en poco tiempo. Como ejemplo tenemos el nucleo de GNU/Linux.

3. Se puede adaptar según nuestras necesidades y/o manias.

En EE.UU. el software criptográfico, para exportación, estaba sometido a unas leyes muy estrictas, debido a lo cual la longitud de las claves distaba mucho de ser ideal para proteger las comunicaciones.

Si se van a utilizar programas relacionados con la criptográfia asegurese de que dichos programas soportan encriptación fuerte y en caso de no soportarla busque otro software que le de más garantías de seguridad.

La red Echelon

Esta red es, supuestamente, una red formada por EE.UU. e Inglaterra y cuenta con el apoyo de Nueva Zelanda y Australia. Su proposito es el de escuchar todas las comunicaciones electrónicas con el fin de luchar contra terroristas y criminales. Esto lo hacen mediante la busqueda de unas palabras clave en todas las comunicaciones.

Hay sospechas de que esta red podría estar siendo utilizada para otros fines menos altruistas como podría ser el espionaje industrial, . . .

La red Enfopol

La red Enfopol parece ser un proyecto de la Unión Europea como respuesta a la red Echelon. La Unión Europea tiene intenciones de crear su propia agencia de vigilancia electrónica. Su cometido el mismo que el de la red Echelon, pero con otros miembros, a excepción de los Britanicos, que estarian en ambos lados.

Conceptos básicos

- Llamaremos "*texto plano*" al texto que queremos proteger mediante el uso de técnicas criptográficas. Denotaremos el conjunto de todos estos textos como "$\mathbf{M}$".

- Llamaremos "*criptograma*" al texto una vez que ha sido transformado mediante alguna técnica criptográfica. Este texto resulta ilegible a no ser que se conozca la clave para volver a recuperar el "*texto plano*" original. Denotaremos el conjunto de todos estos textos como "$\mathbf{C}$".

- Llamaremos "*encriptación*" al proceso que transforma un texto plano en un criptograma.

- Llamaremos "*desencriptación*" al proceso que recupera el texplo plano de un criptograma.

- Denotaremos por **K** a todo el conjunto de claves que se pueden utilizar para encriptar mensajes utilizando un determinado sistema criptográfico.

- Llamaremos "*dispositivo de encriptación*" , y lo denotaremos como "**E**" , a cualquier dispositivo que transforme un elemento de **M** en un elemento de **C**.

- Llamaremos "*dispositivo de desencriptación*" , y lo denotaremos como "**D**" , a cualquier dispositivo que transforme un elemento de **C** en un elemento de **M**.

- Llamaremos "*criptosistema*" , "*sistema criptográfico*" o "*sistema de cifrado*" al conjunto (M,C,K,E,D).

¿Que son los alfabetos?

Tanto los textos planos como los criptogramas estan formados por palabras, y estas estan constituidas por símbolos. Por ejemplo en la escritura estos símbolos son las letras, números y signos de puntuación.

Llamaremos "*alfabetos*" al conjunto de símbolos utilizados en los textos planos o en los criptogramas. Los símbolos utilizados en los textos planos y en los criptogramas no tienen que ser los mismos. Denotaremos como Σ_M al alfabeto utilizado en los textos planos y Σ_C al alfabeto utilizado en los criptogramas.

El proceso de encriptación

Dado un criptosistema cualquiera (M,C,K,E,D) un dispositivo de encriptación será, desde el punto de visto matemático, una función:

$$\begin{array}{rccccccc} E: & M & \times & K & \longrightarrow & C & \times & K \\ & (m & , & k) & \longrightarrow & (c_k & , & k) \end{array}$$

Dado un texto plano y una clave nos genera un criptograma.

El proceso de desencriptación

Dado un criptosistema cualquiera (M,C,K,E,D) un dispositivo de desencriptación será, desde el punto de visto matemático, una función:

$$\begin{array}{rccccccc} D: & C & \times & K & \longrightarrow & M & \times & K \\ & (c_k & , & k) & \longrightarrow & (m & , & k) \end{array}$$

Dado un criptograma y una clave recuperamos el texto plano.

¿Todo esto funciona?

Dado un criptosistema cualquiera (M,C,K,E,D) ¿funciona realmente el proceso de encriptación y desencriptación?.

La respuesta es si, pero hay que exigir algo más a los sistemas de encriptación y desencriptación:

$$D(E(m,k)) = (m,k)$$

Es decir, si encriptamos un mensaje y luego lo desencriptamos obtenemos el texto plano original. Si después de encriptar un mensaje no pudieramos obtener el texto plano de nuevo el sistema no serviría de nada.

Algunas precauciones

La criptografía no es segura al 100 %. Todos los algoritmos tienen puntos debiles, o se supone que los tienen. Hay que tener en cuenta algunos detalles:

1. Las claves han de permanecer seguras. Si se compromete la seguridad de las claves empleadas en nuestro sistema no importa lo seguro o infranqueable que sea, terminará por caer tarde o temprano.

2. Existen ciertas claves, denominadas "*claves débiles*" , que comprometen la seguridad. Estas claves actuan de la siguiente manera:

$$\exists\ k \in\ K \qquad E(m,k) = (m,k)$$

La clave k es una clave débil ya que deja el criptograma igual que el texto plano.

3. Otro tipo de "*clave débil*" sería aquella que deja el criptograma muy parecido al texto plano, con lo cual . . .

4. Y otro tipo, más, de "*clave débil*" sería aquella que si encriptamos un texto plano n veces consecutivas obtenemos el texto plano de nuevo:

$$E^{n)}(m,k) = (m,k)$$

Todo este tipo de claves pueden llegar a comprometer la seguridad de nuestro sistema, luego es importante conocer cuantas claves de este tipo existen en nuestro sistema y evitarlas a toda costa. En un buen criptosistema la cantidad de este tipo de claves es nula o muy pequeña, comparada con la cantidad total de claves, con lo cual la probabilidad de utilizar una de estas claves es nula o practicamente nula.

Todos estos mandamientos se resumen en uno solo:

La potencia sin control no sirve de nada.

¿Que es el criptoanálisis?

Suponiendo conocidos los algoritmos de encriptación el criptoanálisis consiste en comprometer la seguridad de un sistema criptográfico. El criptoanálisis consiste en buscar los puntos débiles de un sistema criptográfico.

Existen diferentes formas de atacar un sistema criptográfico:

1. **Ataque por fuerza bruta**, si se tiene un criptograma mediante este método se probarán todas las claves posibles para obtener el texto plano. Si el conjunto de posibles claves es alto este sistema es inviable. Normalmente a este tipo de ataques no se les suele considerar como una forma de criptoanálisis ya que no busca puntos débiles, únicamente utiliza todas las claves posibles.

2. **Ataque por texto plano escogido**, consiste en elegir varios textos planos y obtener sus criptogramas asociados. Esto implica tener acceso al dispositivo de encriptación, pero no a la clave de encriptación.

3. **Ataque a partir de texto plano**, el atacante tiene acceso a textos planos y a sus correspondientes criptogramas.

4. **Análisis por frecuencias**, este tipo de ataque es utilizado para romper sistemas criptográficos simétricos y se basa en estudiar la frecuencia con la que aparecen los distintos símbolos en un lenguaje determinado y luego estudiar la frecuencia con la que aparecen en los criptogramas, y de esta manera establecer una relación y obtener el texto plano.

Sistemas criptográficos

Los sistemas criptográficos los podemos clasificar según como encripten los diferentes símbolos del alfabeto que estemos utilizando:

- **Monoalfabéticos**, son aquellos en los que cada símbolo del alfabeto se encripta siempre con el mismo símbolo, la encriptación de cada símbolo es independiente del mensaje. Por ejemplo todas las "B" que aparecen en el texto plano siempre apareceran en el criptograma como "L" . . .

  Este tipo de sistemas son inseguros ya que se pueden romper mediante "*análisis estadísticos de frecuencias*" .

- **Polialfabéticos**, son aquellos en los que cada símbolo del alfabeto **NO** se encripta con el mismo símbolo, la encriptación depende del mensaje. Por ejemplo las "B" que aparecen en el mensaje se encriptan unas veces con "L" , otras veces con "T" , otras veces con "Ñ" , . . .

El primer requisito para que un sistema criptográfico sea seguro es que sea polialfabético.

Un poco de matemáticas

¿Que es un grupo?

Desde el punto de vista matemático un "*grupo*" es un conjunto, "G" , dotado de una operación, "$\oplus$" . Llamaremos grupo al par (G,$\oplus$).

Muchas veces la operación $\oplus$ se denotará como "$+$" , diciendo entonces que estamos utilizando notación "*aditiva*" o como "$*$" , diciendo entonces que estamos utilizando notación multiplicativa. Tanto "$+$" como "*" **NO** tienen por que ser la suma y producto que conocemos sobre los números reales, $\mathbb{R}$.

Un grupo ha de verificar las siguientes condiciones:

1. Dados $a, b \in$ G, cualesquiera, se tiene que $a \bigoplus B \in$ G.

2. Existencia de elemento neutro:

   - Utilizando notación aditiva existirá un elemento, al que denotaremos, 0, tal que:
     $0 \in$ G y $0 + a = a + 0 = a \ \forall \ a \in$ G

   - Utilizando notación multiplicativa existirá un elemento, al que denotaremos 1, tal que:
     $1 \in$ G y $0 * 1 = a * 1 = a \ \forall \ a \in$ G

3. Para cada $a \in$ G se tiene que:

   - Utilizando notación aditiva existirá un elemento, al que denotaremos $-a$, tal que:
     $-a \in$ G y $a + (-a) = -a + a = 0$.
     El elemento $-a$ recibe el nombre de elemento opuesto de a.

   - Utilizando notación multiplicativa existirá un elemento, al que denotaremos a^{-1}, tal que:
     $a^{-1} \in$ G y $a * a^{-1} = a^{-1} * a = 1$.
     El elemento a^{-1} recibe el nombre de elemento inverso de a.

Llamaremos orden de un "*grupo*" **G** y lo denotaremos como $|\mathbf{G}|$ al número de elementos que posee el grupo.

El grupo $(\mathbb{Z}/n,+)$

Un grupo muy utilizado en criptografía es $(\mathbb{Z}/n,+)$, donde $n \in \mathbb{N}$.

Fijado un número n si dividimos por él entonces obtenemos los siguientes restos:

$$\{0, 1, 2, \ldots, n-1\}$$

Luego tendremos que:

$$(\mathbb{Z}/n, +) = \{0, 1, 2, \ldots, n-1\}$$

$(\mathbb{Z}/n,+)$ es un grupo con la operación $+$ definida de la siguiente manera:

$a + b =$ resto de $a + b$ al dividir por n.

Con esta suma podemos sumar cualquier número entero en $\mathbb{Z}/n$. Por ejemplo para $n = 5$ tendremos:

0	1	2	3	4	5	6	7	8	En $\mathbb{Z}$
0	1	2	3	4	0	1	2	3	En $\mathbb{Z}/n$

Para expresar cualquier elemento de $\mathbb{Z}$ como un elemento de $\mathbb{Z}/n$ bastará con dividirlo por n y quedarnos con su resto.

Diremos que dos elementos de $\mathbb{Z}$ son equivalentes en $\mathbb{Z}/n$ o simplemente son equivalentes módulo n siempre y cuando tengan el mismo resto al dividir por n.

En el ejemplo anterior tendremos que:

$$5 \equiv 0 \ mod \ 5$$
$$6 \equiv 1 \ mod \ 5$$
$$7 \equiv 2 \ mod \ 5$$
$$8 \equiv 3 \ mod \ 5$$

Sistemas de encriptación con estructura de grupo

Algo muy a tener en cuenta en los sistemas de encriptación, independientemente de si son simétricos o asimétricos, es si tienen o no estructura de grupo.

Diremos que un sistema criptográfico tiene estructura de grupo si $\forall\ k_1,\ k_2 \in \mathbf{K}\ \exists\ k_3 \in \mathbf{K}$ tal que para cualquier $m \in \mathbf{M}$ se verifica que:

$$E(E(m, k_2), k_1) = E(m, k_3)$$

Es importante que un sistema criptográfico **NO** tenga estructura de grupo, ya que si ciframos un mensaje con una clave k_1 y luego ciframos el criptograma obtenido con otra clave k_2 entonces aumentamos la seguridad del sistema. Mientras que si el sistema hubiera tenido estructura de grupo no habriamos hecho nada, ya que existiría una clave k_3 que realizaría la misma operación, con lo cual no habriamos incrementado la seguridad del sistema.

Permutaciones

Sea **P** un conjunto de n elementos. Llamaremos permutaciones de n elementos al conjunto formado por todas las reordenaciones de esos elementos y lo denotaremos por S_n.

Supongamos que **P**= $\{1, 2, 3\}$ entonces tendremos que las posibles reordenaciones de **P** serán:

$(1, 2, 3), (1, 3, 2), (3, 2, 1), (2, 1, 3), (3, 1, 2), (2, 3, 1)$

Consideraremos los elementos de S_3 como aplicaciones biyectivas de **P** en **P**:

$$\pi_i : \mathbf{P} \xrightarrow{\sim} \mathbf{P} \; i = 1, 2, \ldots, 6$$

La notación que utilizaremos será la siguiente:

$$\pi_5 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$$

Lo cual nos indica que π_5 es la aplicación que manda el 1 al 3, el 2 al 1 y el 3 al 2.

Tendremos entonces que:

$$S_3 = \{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6\}$$

Podemos definir una operación dentro de S_n, a esta operación la denotaremos como "$\circ$" y tendremos que $(S_n, \circ)$ es un grupo. A esta operación la llamaremos composición.

Sea

$$\pi_6 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$$

Tendremos entonces que:

$$\pi_5 \circ \pi_6 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix} \circ \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$$

El orden del grupo S_n vendrá dado por $n!$. De donde se deduce que el orden de S_3 será $3! = 6$.

Criptografía simétrica

¿Que es la criptografía simétrica?

En este tipo de criptografía tanto el emisor como el receptor del mensaje han de conocer la clave. El conocimiento de la clave implica el poder encriptar y desencriptar mensajes. En este tipo de criptografía normalmente se utilizan dos claves:

1. Una para encriptar.

2. Otra para desencriptar.

Normalmente se dice que se emplea una clave ya que conociendo la clave de encriptación es facil, la mayoría de las veces inmediato, calcular la clave de desencriptación y viceversa.

Ventajas

- Son algoritmos “faciles” de implementar.
- Requieren “poco” tiempo de computo.

Desventajas

- Las claves han de transmitirse por un canal seguro. Algo difícil de realizar.

Simple sustitución

Denotemos por Σ el alfabeto utilizado, supongamos que es el abecedario. Sea $\pi \neq Id$ una permutación de Σ, es decir una aplicación biyectiva:

$$\pi : \Sigma \xrightarrow{\sim} \Sigma$$

Observaciones sobre este sistema criptográfico:

- Este sistema criptográfico es simétrico. Ya que si se conoce la permutación que realiza el encriptado es facil calcular la permutación que realiza el desencriptado, por lo tanto ambas deben permanecer en secreto.

- Este sistema es monoalfabético. Cada símbolo del alfabeto Σ en el texto plano siempre es reemplazado por el mismo símbolo en el criptograma.

- Como consecuencia de ser monoalfabético es sensible a un ataque por análisis estadístico de frecuencias.

- Dado que las permutaciones de n elementos tienen estructura de grupo respecto a la composición este sistema criptográfico posee estructura de grupo. Encriptar con dos claves no nos da más seguridad ya que existira una tercera clave que produce el mismo efecto que el encriptar con las dos claves anteriores.

El cifrador del Cesar

Este sistema de encriptación lo utilizaba *Julio Cesar* y es un caso particular de una familia de sistemas de encriptación conocidos como "*Vigenere*" .

Este sistema es completamente inseguro, pero es útil para explicar los fundamentos de los metodos criptográficos.

Supongamos que tenemos un alfabeto con n elementos.El metodo consiste en asignar un número a cada símbolo del alfabeto, $A = 0$, $\ldots Z = n - 1$. Para encriptar a cada letra le sumaremos un número, menor estrictamente, que n módulo n y le haremos corresponder la letra asociada.

Por ejemplo si nuestro alfabeto tiene 26 símbolos y elegimos como clave para encriptar 3 tendremos:

$A \to 0$ y $A + 3 = 3 \Rightarrow 3 \equiv 3 \mod 26$, $3 \to D$ ya que a D le corresponde el 3.

Si para encriptar sumamos 3 mod n, entonces para desencriptar restamos 3 mod n. Siendo n el número de símbolos de nuestro alfabeto.

Observaciones sobre este sistema criptográfico:

- Este sistema es simétrico. Las claves para encriptar y desencriptar deben mantenerse en secreto, ya que conocer una de ellas implica conocer la otra.

- Este sistema es monoalfabético.

- Es sensible a ataques por análisis estadístico de frecuencias.

- Tiene estructura de grupo, ya que si primero desplazamos m unidades cada letra, y luego

lo hacemos n unidades es como si lo hubiesemos desplazado $m+n$ unidades desde el principio.

El algoritmo D.E.S.

¿Que es D.E.S.?

D.E.S. (Data Encryption Standard o estándar de encriptación de datos) es, probablemente, el algoritmo simétrico más utilizado en el mundo.

Fue desarrollado por IBM en los 70 por encargo del NBS(National Bureau of Standards) hoy conocido como NIST (National Institute of Standards and Technology). En 1,977 fue modificado y adoptado como estándar para encriptación de datos no clasificados por el gobierno de los EE.UU.

Originalmente era conocido como Lucifer, trabajaba sobre bloques de 128 bits y tenia una clave de 128 bits. Unicamente utilizaba operaciones booleanas y era fácil de implementar tanto en hardware como en software.

Las modificaciones que introdujo el NBS fueron basicamente reducir la longitud tanto de la clave como de los bloques a 64 bits.

Descripción del DES

Este algoritmo simétrico encripta bloques de 64 bits de longitud con una clave de 64 bits de longitud. Dentro de la clave el último bit de cada byte es de paridad, con lo cual tenemos que la clave en realidad es de 56 bits, esto hace que haya 2^{56} posibles claves para este algoritmo.

Este algoritmo utiliza un "dispositivo" denominado SBB (Standard Building Block o constructor estándar de bloques), el cual requiere como entrada un bloque de 64 bits y una clave de 48 bits, produciendo una salida de 64 bits. El DES requiere 16 dispositivos SBB.

Tenemos una clave original, K, de 64 bits, 56 en realidad. De ella se extraen 16 subclaves K_i de 48 bits de longitud. El algoritmo es el siguiente:

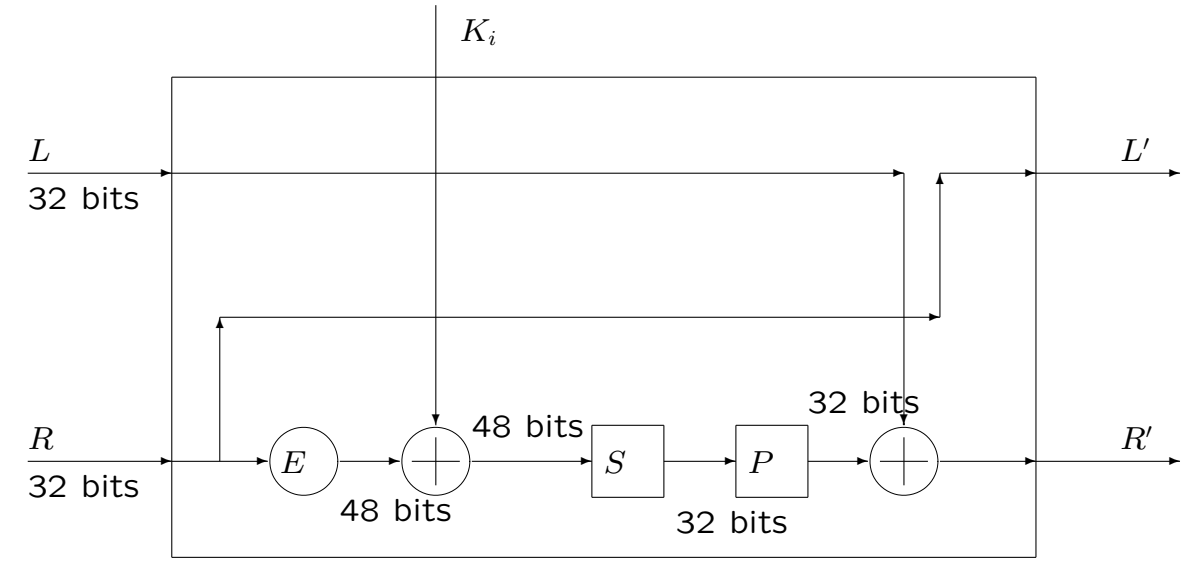
1. Se aplica una permutación original, IP, a cada bloque de 64 bits. Produciendo una salida B_j de 64 bits.

2. Pasamos B_j con la subclave K_1 por el primer SBB, la salida la pasamos por el segundo SBB con la subclave K_2 y asi con los 16 SBB.

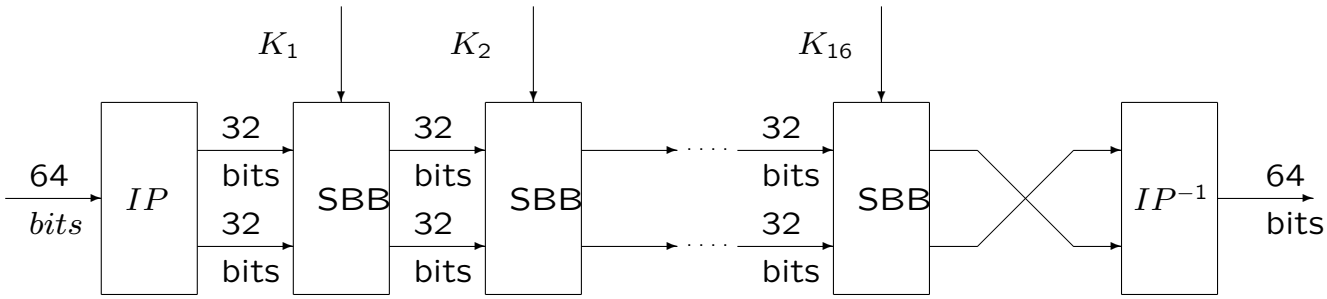
3. A la salida del último SBB le aplicamos la permutacion IP^{-1}. De donde obtenemos el texto encriptado.

Para desencriptar tomamos como entrada el texto encriptado y aplicamos las subclaves K_i en orden inverso, es decir en el primer SBB

utilizamos K_{16}, en el segundo K_{15} . . . y en el último SBB utilizamos K_1.



Dispositivo SBB.



Encriptación con DES.

Seguridad en DES

A mediados de 1,988 se demostró que un ataque por fuerza bruta contra el algoritmo DES ya era posible, gracias al avanze de la informática entre otras cosas. Pero la debilidad no la tiene el algoritmo, sino que la tiene la clave, no posee suficiente longitud la clave. Luego si aumentamos la clave del DES este algoritmo sigue siendo seguro.

También se conocen claves débiles y semidébiles para este algoritmo, pero su número es tan pequeño en comparación con el total de claves posibles que no supone mucha preocupación.

Cuando se implantó DES en 1,977 W.Diffie y E.Hellman analizaron una máquina capaz de encontrar la clave de cifrado en doce horas partiendo de un texto en claro y su correspondiente criptograma. Su coste era de 20 millones de dolares.

En Julio de 1,988 una empresa sin ánimo de lucro construyó una máquina que desencriptaba mensajes DES en menos de tres días, el costo de este engendro era menos de 40 millones de pesetas. Pocas semanas antes un alto cargo de la NSA declaraba que DES era seguro y que descifrar mensajes DES era demasiado costoso, incluso para el gobierno . . .

Variantes del DES

Dado al avance de la informática los ataques al DES por fuerza bruta son cada vez más fáciles y rápidos de llevar a cabo, y dado que la debilidad no reside en el algoritmo sino en la longitud de la clave los usuarios se sienten reticentes a cambiar de algoritmo. Prefieren utilizar variantes del algoritmo y de esta forma aprovechar las implementaciones por software y hardware que existen.

DES Múltiple

Consiste en aplicar el algoritmo DES, con diferentes claves, varias veces. Este método aumenta la seguridad ya que el DES **NO** posee estructura de grupo.

Dentro de esta familia el más utilizado es el Triple-DES (3DES). Elegimos dos claves K_1 y K_2, el procedimiento es el siguiente:

$$C = E_{K_1}(D_{K_2}(E_{K_1}(M)))$$

La clave en este caso tendría 112 bits.

Otros criptosistemas simétricos

- **LOKI**, Australiano. Roto mediante criptoanálisis diferencial.

- **LOKI91**, Australiano. Versión modificada del anterior. De momento ha resistido los ataques por criptoanálisis diferencial.

- **RC2**, diseñado por *R. Rivest* para RSA Data Security Inc. No es público ni esta patentado.

- **RC4**, versión simplificada del anterior. El gobierno de los EEUU autorizó su exportación. Utiliza una clave de 40 bits.

- **RC5**, diseñado por *R. Rivest*.

- **GOST**, Rusia. Similar al DES.

- **CAST**, Canada. Inmune a ataques basados tanto en criptoanálisis diferencial como en criptoanálisis lineal.

- **BLOWFISH**, diseñado por *Bruce Schneier*.

- **IDEA**, desarrollado por *X. Lai* y *J.L. Massey* en el politécnico de Zurich.

- **AKELARRE**, inspirado en IDEA y RC5. Se diseño a partir de estos para evitar algunas de los inconvenientes que tenian ambos métodos.

- **SKIPJACK**, EEUU. Desarrollado por la NSA (National Security Agency). Este algoritmo es el que incorporan los chips CLIPPER y CAPSTONE. Tanto el algoritmo como los chips estan clasificados por el gobierno de los EEUU como secretos.

- **RIJNDAEL**, Belga. Desarrollado por *Vincent Rijmen* y *Joan Daemen*. Totalmente público.

Advanced Encription Standard (AES)

Que es AES

AES es el nuevo estándar de encriptación de datos propuesto por el NIST. Es el sustituto de DES.

Este estándar ha sido desarrollado para sustituir a DES, cuya longitud de clave (56 bits), hoy en día resulta ineficiente.

Este estándar goza de más confianza que su predecesor, DES, ya que ha sido desarrollado y examinado de forma pública desde el primer momento.

En 1,997 se presentó la convocatoria para la elección de un sustituto a DES. Dicho sustituto debía cumplir los siguientes requisitos:

- Ser un algoritmo simétrico.
- Ser público.
- Que su diseño permita aumentar la longitud de la clave según las necesidades de los usuarios.
- Que sea facilmente implementable tanto en software como en hardware.
- No esté patentado.

En la primera conferencia de candidatos (AES1 en 1,998) se presentaron 15 candidatos:

CAST-256, CRYPTON, DEAL, DFC, E2, FROG, HPC, LOKI97, MAGENTA, MARS, RC6, RIJNDAEL, SAFER+, SERPENT y TWOFISH

En la segunda conferencia de candidatos (AES2 en 1,999) se seleccionaron cinco finalistas:

MARS, IBM.

RC6**,** RSA Laboratories.

RIJNDAEL, Joan Daemen y Vincent Rijmen.

SERPENT,

TWOFISH, Bruce Schneier

El 2 de Octubre del 2,000 el NIST anunció a Rijndael como sustituto de DES.

Caracteristicas de Rijndael

Rijndael es un algoritmo simétrico de encriptación desarrollado para encriptar bloques de longitud 128, 192 o 256 bits utilizando claves de longitud 128, 192 o 256 bits.

Puede ser utilizado en cualquiera de las nueve combinaciones posibles de bloque/longitud.

Es facilmente extensible a multiplos de 32 bits tanto como para la clave como para la longitud de bloque.

Complejidad computacional

Complejidad computacional

Podemos definir la "complejidad computacional" como el tiempo que tarda un ordenador en realizar una tarea (tiempo que tarda en procesar un algoritmo).

En criptografía es importante desde dos puntos de vista:

1. Interesa que los algoritmos de encriptación, desencriptación y elección de claves requieran poco tiempo de calculo.

2. Cualquier algoritmo de desencriptación, suponiendo no conocida la clave, requiera un alto coste computacional.

El coste computacional de un algoritmo depende de:

1. El propio algoritmo, número de operaciones a realizar y "que" operaciones tiene que realizar.

2. El ordenador sobre el que se esta ejecutando el algoritmo.

Entenderemos por coste computacional:

- A una función que depende del tamaño de la entrada.

- Para un tamaño n entenderemos por coste computacional el peor de todos los casos posibles.

Los ordenadores trabajan con "*bits*" , "*bytes*" , . . . Es por esta razón que para calcular el coste computacional de una operación calcularemos el número de operaciones, a nivel de "bits" , que son necesarias para realizarla.

Sea $n \in \mathbb{Z}$ y sea $a_r a_{r-1} \dots a_1 a_0$ su expresión en binario:

$$a_r a_{r-1} \dots a_1 a_0 = \sum_{i=0}^{r} a_i \cdot 2^i = n$$

donde los $a_i \in \{0, 1\}$.

$$r = [\log_2 n] + 1 = [\frac{\log n}{\log 2}] + 1$$

Con los algoritmos clásicos de suma y multiplicación tendremos que:

- Para sumar dos números de k bits necesitaremos realizar $2 \cdot k$ operaciones, diremos que el coste computacional de ese algoritmo es $O(2 \cdot k)$.

- Para multiplicar dos enteros de m y n bits se necesitan realizar $m \cdot n$ operaciones, diremos que el coste computacional de ese algoritmo es $O(m \cdot n)$.

El número de operaciones para calcular $n!$ será:

$$(n-2) \cdot n \cdot ([\log_2 n] + 1)^2$$

Si desarrollamos esta expresión veremos que el término que más aporta a la suma es:

$$n^2 \cdot [\log 2 \cdot n]^2$$

a medida que crece n se puede despreciar el resto de terminos que dependen de n, con lo cual se dice que el algoritmo tiene un coste computacional $O(n^2 \cdot [\log 2 \cdot n]^2)$.

Tiempo polinomial

Diremos que un algoritmo tiene una complejidad de tipo polinomial, o simplemente que se puede resolver en tiempo polinomial si su complejidad computacional esta acotada por un polinomio que depende del tamaño de la entrada:

$$O(algoritmo) \leq p(n)$$

Interesa que los siguientes algoritmos sean de tipo polinomial:

- Elección de claves.
- Encriptación.
- Desencriptación, conocida la clave.

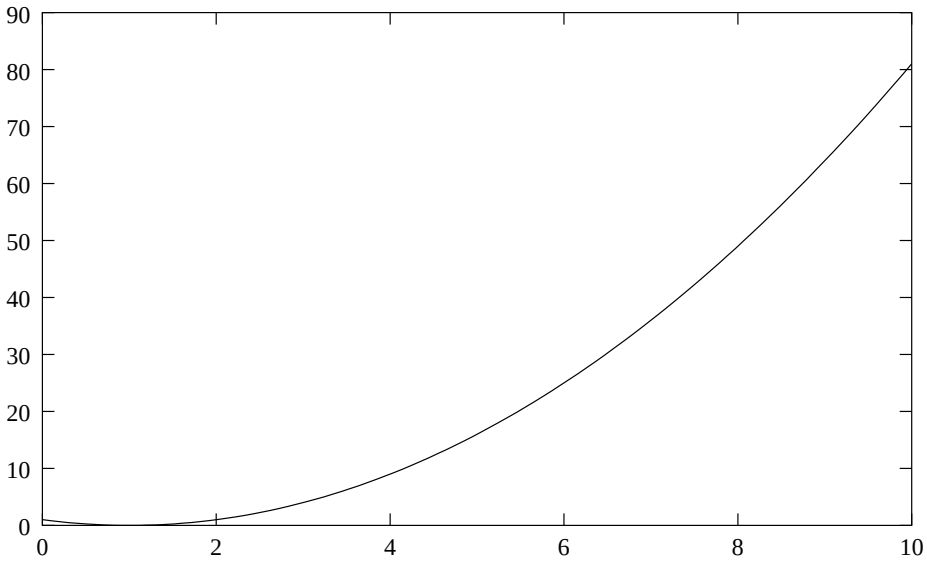
Dado un criptosistema cualquiera, si existe un algoritmo de tipo polinomial que puede encontrar la clave de dicho criptosistema diremos que ese criptosistema **NO** es seguro.

Tiempo exponencial

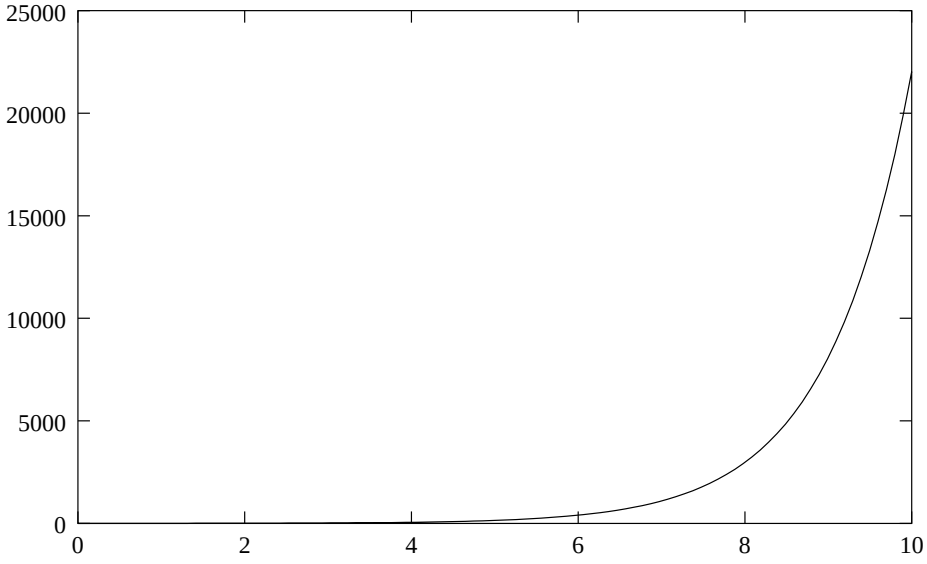
Diremos que un algoritmo es de tipo exponencial o simplemente que se puede resolver en tiempo exponencial si el tiempo necesario para llevarlo a cabo aumenta de forma exponencial según varia el tamaño de la entrada. En este caso su complejidad computacional será $O(e^{f(n)})$.

Normalmente denominaremos como algoritmo de tipo exponencial a todo aquel algoritmo que no pueda ser incluido en los algoritmos de tipo polinómico.

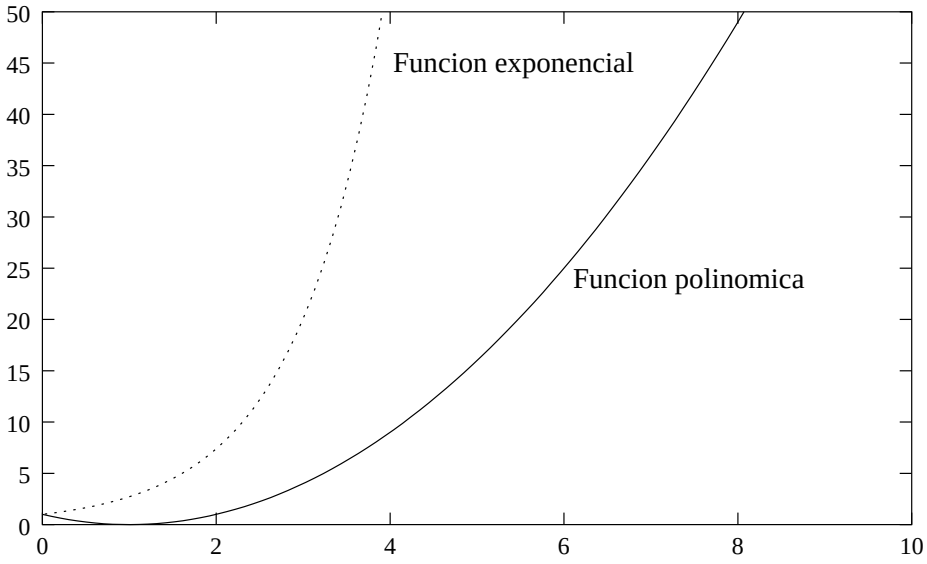
Gráfica de una función polinomial:



Gráfica de la función exponencial:



Funciones exponencial y polinómica en la misma escala:



Criptografía asimétrica

Introducción

La criptografía asimétrica surge para solucionar el problema que tiene la criptografía simétrica de distribución de la clave. Esta distribución se tenía que hacer mediante un canal seguro, algo difícil de realizar.

Una solución a esto la dieron "*Diffie*" y "*Hellman*" en 1,976. Su solución proponía utilizar "*funciones de un sentido*" en canales abiertos.

Funciones de un sentido

Supongamos que $f(x)$ es una función de un sentido, entonces:

1. Es fácil el cálculo $y = f(x)$, conocido x.

2. Conocido y es computacionalmente imposible el cálculo de $x = f^{-1}(y)$.

Un ejemplo típico de una función de este tipo es:

$$y \equiv g^x \pmod{p}$$

donde $g, x \in \mathbb{Z}$ y p es un número primo con más de 200 digitos. Esta función es conocida como "*exponenciación modular*". La complejidad computacional de esta función es $O(\log p)$.

Su función inversa será:

$$x \equiv \log_g y \pmod{p}$$

su complejidad computacional es de tipo exponencial $O(e^{\sqrt{\log p \log\log p}})$. Cuando p tiene un tamaño como el que se ha dicho antes es practicamente imposible el cálculo de esta función.

Esta función se conoce como "*logaritmo discreto*" y es una gran importancia en la criptografía asimétrica.

Intercambio de Diffie-Hellman

Este intercambio de claves propuesto por Diffie y Hellman utiliza la función de exponenciación modular en canales abiertos.

En este esquema se tienen datos públicos, que conoce todo el mundo, y datos privados, que unicamente conocen aquellos que los han generado. Ambos datos, tanto públicos como privados, se utilizan para generar una clave secreta común que puede ser utilizada en cualquier criptosistema simétrico.

Sean "A" y "B" dos personas que desean comunicarse:

1. "A" y "B" eligen un primo p con más de 200 digitos.

2. Se elige $g \in \mathbb{Z}/p$ tal que $< g >= \mathbb{Z}/p$.

3. Los valores p y g son públicos.

4. Tanto "A" como "B" eligen sendos valores aleatorios, privados, x_A y x_B tales que:

$$x_A, x_B \in \mathbb{Z}/p$$

5. "A" manda a "B", $y_A \equiv g^{x_A} \pmod{p}$.

6. "B" manda a "A", $y_B \equiv g^{x_B} \pmod{p}$.

7. Se calcula la clave secreta de encriptación $z_{AB} = z_{BA}$:

   - "A" la calcula:

$$z_{BA} \equiv y_B^{x_A} \equiv g^{x_B \cdot x_A} \pmod{p}$$

   - "B" la calcula:

$$z_{AB} \equiv y_A^{x_B} \equiv g^{x_A \cdot x_B} \pmod{p}$$

En este esquema los datos públicos son (p, g, y_A, y_B). En el caso de que alguien quisiera conocer la clave secreta a partir de los datos públicos, tendría que conocer x_A o x_B para generar la clave secreta z_{AB}, lo que equivale a resover una de estas dos ecuaciones:

$$\begin{aligned} x_A &\equiv \log_g y_A \pmod{p} \\ x_B &\equiv \log_g y_B \pmod{p} \end{aligned}$$

con lo cual se tendría que resolver un problema con complejidad computacional exponencial, y dado que p posee 200 o más digitos tenemos que no es computacionalmente posible.

Funciones con trampa

Este tipo de funciones son funciones de un sentido, pero tienen una pecularidad:

- El cálculo de su función inversa tiene “truco” .

Si se conoce ese “truco” se puede calcular su función inversa en tiempo polinomial.

El logaritmo discreto

Recordemos la exponenciación modular:

$$y = a^x \pmod{n}$$

diremos que x es el "*logaritmo discreto*" de y en la base a módulo n:

$$x = \log_a y \pmod{n}$$

El cálculo del logaritmo discreto es un problema de tipo exponencial cuando n es suficientemente grande, 200 digitos o más.

Computación cuántica

Aunque el logaritmo discreto sea un problema de tipo exponencial y esto induzca a creer que los criptosistemas basados en esta función gozarán de seguridad por los siglos de los siglos . . . esto no es cierto.

Ahora se esta investigando sobre ordenadores y computación cuántica y muchos de los problemas que corren en tiempo exponencial en ordenadores clásicos corren en tiempo polinomial sobre ordenadores cuánticos. Existen algoritmos que se ejecutan en tiempo polinomial en ordenadores cuánticos para resolver problemas de tipo exponencial (sobre ordenadores clásicos).

El algoritmo de Shor es un ejemplo de esto.

El algoritmo RSA

Este algoritmo es de clave pública y debe su nombre a sus tres inventores:

Rivest, Ron.
Shamir, Adi.
Adleman, Leonard.

Es el algoritmo público más utilizado y seguramente el más sencillo tanto a su comprensión, como a su implementación.

Este algoritmo basa su seguridad en la factorización de números primos.

Descripción del RSA

Un usuario elige dos números primos p y q de 200 digitos aproximadamente. Sea $n = p \cdot q$.

Buscamos e tal que sea primo con:

$$\Phi(n) = (p-1) \cdot (q-1)$$

Como e y $\Phi(n)$ son primos entre sí, entonces existe d tal que:

$$e \cdot d \equiv 1 \text{ mod } \Phi(n) = n + 1 - p - q$$

d se puede calcular mediante el algoritmo de Euclides.

- Clave Pública: (e,n).

- Clave Privada: (d,n).

Los datos que han de mantenerse privados son:

- p y q.
- $\Phi(n)$.
- d.

Cualquiera que conozca estos datos podrá descifrar los mensajes del propietario de la clave.

Además de estos datos hemos de fijar la longitud de bloque:

- Longitud del bloque que vamos a cifrar.
- Longitud del bloque cifrado.

Ejemplo

Elegimos dos primos $p = 281$ y $q = 167$.

$$\begin{aligned} n &= 281 \cdot 167 = 46{,}927 \\ \Phi(n) &= (280 - 1) \cdot (167 - 1) = 46{,}480 \end{aligned}$$

Buscamos e y d tales que:

$$e \cdot d \equiv 1 \text{ mod } \Phi(46{,}927)$$

por ejemplo:

$$\begin{aligned} e &= 39{,}423 \\ d &= 26{,}767 \end{aligned}$$

Las claves serán:

- Clave Pública: $(39{,}423, 46{,}927)$.
- Clave Privada: $(26{,}767, 46{,}927)$.

Supongamos que vamos a encriptar bloques de dos letras en bloques de tres letras, y que queremos encriptar "HOLA" utilizando un alfabeto de 36 símbolos.

El procedimiento es el siguiente:

1. Asignamos a cada letra un número según un alfabeto:

$$\text{HOLA} = (17, 24, 21, 10)$$

2. Bloques a cifrar: $(17, 24)$ y $(21, 10)$.

3. Expresamos ambos bloques como un número en base 36:

$$\begin{aligned} (17, 24) &= 17 \cdot 36^0 + 24 \cdot 36 = 881 \\ (21, 10) &= 21 \cdot 36^0 + 10 \cdot 36 = 381 \end{aligned}$$

4. Elevamos estos números a e módulo 46,927:

$$\begin{aligned} 881^{39,423} &\equiv 45{,}840 \bmod 46{,}927 \\ 381^{39,423} &\equiv 26{,}074 \bmod 46{,}927 \end{aligned}$$

5. Expresamos estos números en base 36, teniendo en cuenta que vamos a tener tres componentes:

$$\begin{aligned} 45{,}840 &= 12 \cdot 36^0 + 13 \cdot 36 + 35 \cdot 36^2 = \\ &= (12, 13, 35) \\ 26{,}074 &= 10 \cdot 36^0 + 4 \cdot 36 + 20 \cdot 36^2 = \\ &= (10, 4, 20) \end{aligned}$$

6. Según el alfabeto considerado a cada número le asignamos una letra:

$$\begin{aligned} (12, 13, 35) &\Longrightarrow CDZ \\ (10, 4, 20) &\Longrightarrow A4K \end{aligned}$$

Luego el mensaje encriptado es "CDZA4K".

Para desencriptar habría que hacer el mismo proceso, pero partiendo de bloques de tres letras y terminando en bloques de dos letras y elevando a e en lugar de d.

Seguridad RSA

Como las claves públicas son públicas, cualquiera puede encriptar un texto a partir de un texto plano e intentar averiguar la clave privada. Supongamos que encriptamos el texto "HOLA", durante el proceso de desencriptación tendremos:

$$45{,}840^d \equiv 881 \mod 46{,}927$$
$$26{,}074^d \equiv 381 \mod 46{,}927$$

o lo que es lo mismo:

$$d = \log_{45{,}840} 881 \mod 46{,}927$$
$$d = \log_{26{,}074} 381 \mod 46{,}927$$

d no es conocido ya que forma parte de la clave privada, para romper este criptosistema lo podemos intentar de varias formas:

1. A fuerza bruta . . .

2. Mediante un ataque de intermediario.

3. Intentando resolver cualquiera de los dos logaritmos discretos anteriores . . .

4. Resolviendo:

$$e \cdot d \equiv 1 \text{ mod } \Phi(46{,}927)$$

Lo cual equivale a conocer $\Phi(46{,}927)$, que a su vez equivale a conocer la factorización en números primos de 46,927.

Lo cual es un problema con el mismo grado de complejidad que el logaritmo discreto(problema de tipo exponencial) para números lo suficientemente grandes . . .

Otros criptosistemas asimétricos

- **ElGamal**.

- **Rabin**, su seguridad se basa en la dificultad de calcular raíces cuadradas módulo un primo compuesto. Problema equivalente al de la factorización de dicho número.

Algoritmo de exponenciación rápida

Un algoritmo para realizar exponenciaciones de una forma rápida es:

Sea $(b_0, b_1, \ldots, b_n)_2$ la expresión en binario de b_{10}:

$$b_{10} = \sum_{i=0}^{n} b_i \cdot 2^i$$

donde $b_i \in \{0, 1\}$.

$$a^b = a^{b_0 \cdot 2^0 + \ldots + b_n \cdot 2^n} = \prod_{i=0}^{n} a^{2^i \cdot b_i}$$

Unicamente tendremos que calcular los a^{2^i} cuyo b_i sea 1. Observar que tenemos una sucesión:

$$\{a^{2^0}, a^{2^1}, a^{2^2}, \ldots, a^{2^n}\}$$

y como $a^{2^i} = (a^{2^{i-1}})^2$ podemos calcular todos los elementos de la sucesión a partir del siguiente elevando a cuadrado.

Cálculo de inversos modulares

Sea $m \in \mathbb{Z}/n$ m es invertible sí y solo sí el máximo común divisor de ambos es 1.

Si $m, n \in \mathbb{Z}$ con n positivo y no nulo para calcular el inverso de m en $\mathbb{Z}/n$ lo podemos calcular de la siguiente forma:

1. Comprobamos que m es invertible en $\mathbb{Z}/n$.

2. Si fuera invertible tendremos que:

   $$m \cdot m^{-1} \equiv 1 \mod n$$

   o lo que es lo mismo:

   $$m \cdot m^{-1} = \lambda \cdot n + 1$$

   donde $\lambda \in \mathbb{Z}$. Es decir:

   $$m \cdot m^{-1} + \lambda \cdot n = 1$$

   Esta ecuación recibe el nombre de diofántica y se puede resolver utilizando el Algoritmo de Euclides.

Números aleatorios

La generación de números aleatorios es muy importante en criptografía ya que la generación de claves se realiza mediante la generación aleatoria de números.

Un mal(predecible) generador aleatorio puede ocasionar que nuestras claves sean predecibles.

Esto fue lo que paso con Netscape y su generador aleatorio en 1,995 cuando dos estudiantes de Berkeley, Ian Goldberg y David Wagner rompieron el SSL de Netscape.

Estos estudiantes descubrieron que se estaban utilizando tres valores para generar la semilla aleatoria del generador de números aleatorios:

1. Un `PID`.

2. Un `PPID`.

3. La hora en segundos y microsegundos.

Para obtener la hora se puede hacer utilizando la herramienta `tcpdump`, con esto se obtenía la hora en segundos con una precisión de un segundo y como en un segundo hay un millón de microsegundos, bastaría con comprobar todos los microsegundos. En algunos casos la encriptación con SSL se podía romper en menos de un minuto.

Generación de claves

Para la generación de claves de n bits se procederá de la siguiente manera:

1. Se pone el bit más significativo a uno.

2. Se generan $n-1$ bits aleatoriamente.

3. Se comprueba que la clave así obtenida verifica las condiciones del algoritmo.

En el caso de la criptografía asimétrica las claves son números primos:

1. Se ponen el bit más y menos significativo a uno.

2. Se generan $n-2$ bits aleatoriamente.

3. Se comprueba la primalidad del número así obtenido. Normalmente dividiendo por una tabla de números primos precalculados y si después de esto no se ha deducido que el número en cuestión es compuesto se le aplicará un algoritmo probabilístico.

Una vez generado un número de n bits se debe comprobar su primalidad, esto se hace empleando algoritmos probabilísticos:

- `Algoritmo de Lehmann`.

  Si un número pasa el test n veces la probabilidad de que no sea primo es de 1 contra 2^n.

- `Algoritmo de Rabin-Miller`.

- `Test de Lucas-Lehmer`.

Se utiliza para comprobar la fiabilidad de los superordenadores CRAY. No es útil en la práctica ya que sólo es aplicable a números de Mersenne.

- `Test de primalidad de Solovay-Strassen`.

- `Test de APRCL`.

  Permite demostrar en pocos minutos que un número de 200 cifras es primo.

- `Test de Atkin-Morain`.

  Permite demostrar en pocos minutos que un número de 200 cifras es primo.

Para asegurar la fiabilidad de las claves no basta con utilizar primos grandes, sino que ademas los números primos deben poseer otras características.

En este tipo de criptografía se suele utilizar un número $n = p \cdot q$, donde p y q son dos primos grandes. Para que sea más dificil de factorizar n se impone que p y q sean "*primos fuertes*" :

- El máximo común divisor de $p-1$ y $q-1$ deber ser pequeño.

- $p-1$ y $q-1$ deben tener algún factor primo grande p' y q'.

- Tanto $p'-1$ como $q'-1$ deben tener factores primos grandes.

- Tanto $p'+1$ como $q'+1$ deben tener factores primos grandes.

Criptografía con curvas elípticas

¿Que es?

Este tipo de criptografía es de clave pública y también es conocida como ECC.

Elliptic
Curve
Cryptosystem

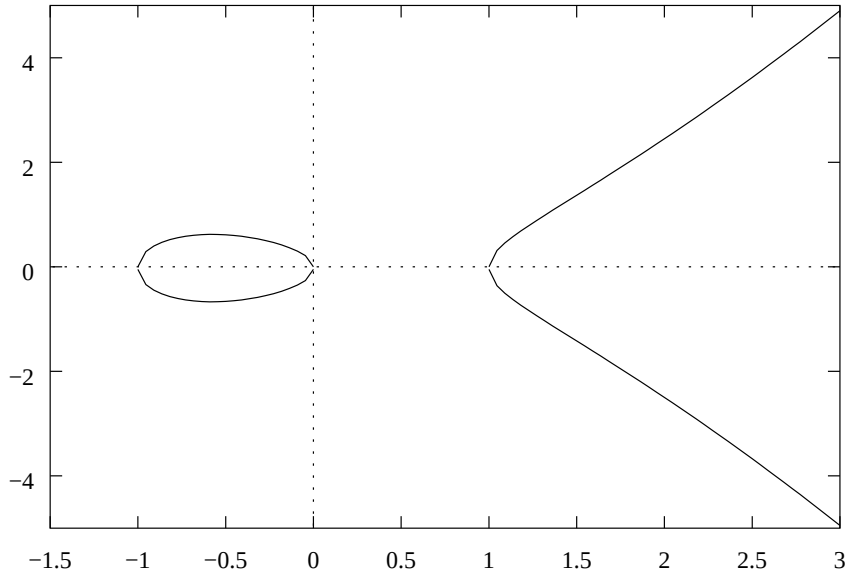
Debido al aumento de las prestaciones de los ordenadores en los sistemas clásicos de clave pública es necesario aumentar el tamaño de las claves según pasa el tiempo para asegurar la seguridad de los criptosistemas.

En 1,985 *Neil Koblitz* y *Victor Miller* propusieron el ECC, cuya seguridad descansa en logaritmo discreto. La diferencia es que en estos sistemas no se utilizan números enteros como símbolos del alfabeto, se utilizan los puntos de las curvas elípticas, pero sobre cuerpos finitos.

Estos metodos son más seguros ya que necesitan menos longitud de clave para obtener la misma seguridad que los metodos clásicos públicos. Por lo tanto son más rápidos y consumen menos recursos.

¿Que son las curvas elípticas?

Lo siguiente es un ejemplo de una curva elíptica sobre $\mathbb{R}$:



Una curva elíptica es una ecuación del tipo:

$$y^2 + a_1 \cdot x \cdot y + a_3 \cdot y = x^3 + a_2 \cdot x^2 + a_4 \cdot x + a_5$$

donde los a_i son elementos de un cuerpo.

Estructura de grupo

Las curvas elípticas que nos interesan para nuestros propositos serán:

$$y^2 = x^3 + \cdot x + b$$

donde $x^3 + \cdot x + b$ no tiene raices multiples o lo que es lo mismo:

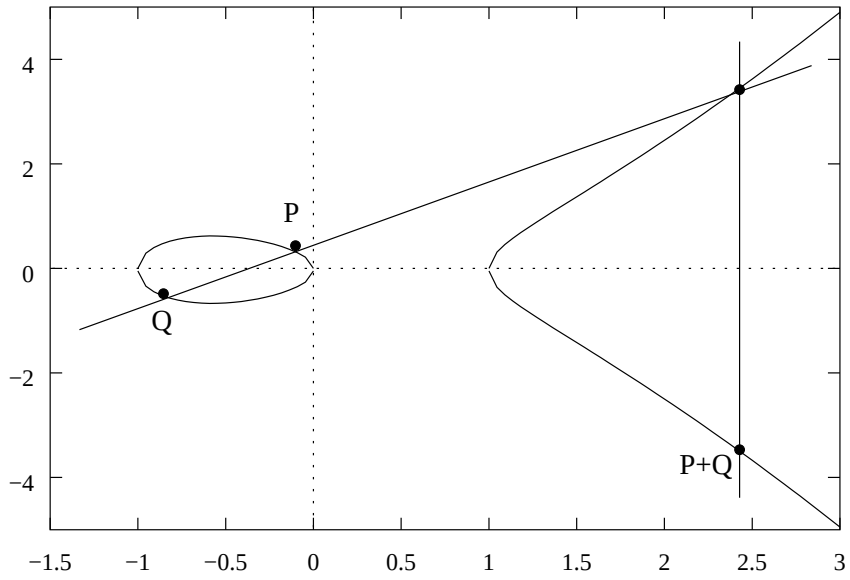
$$\Delta = 4 \cdot a^3 + 27 \cdot b^2 \neq 0$$

En estas condiciones:

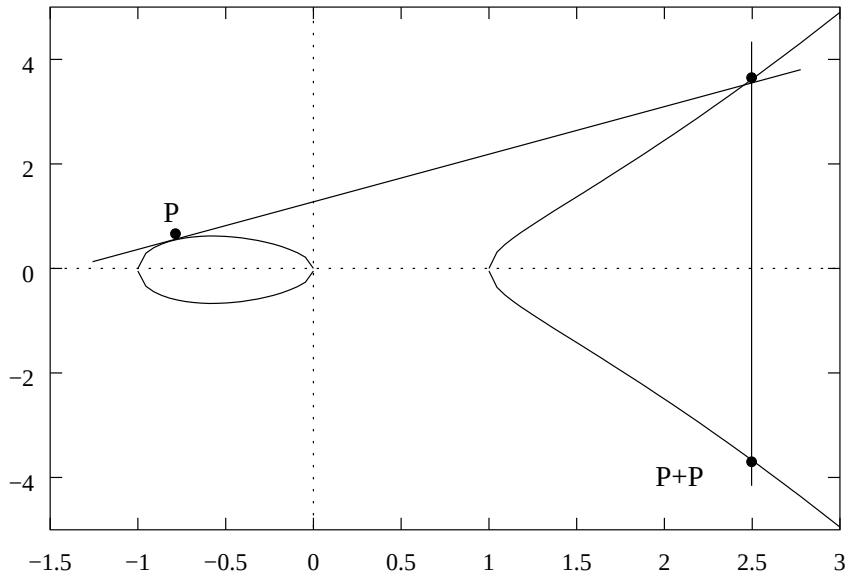
$$E = \{(x, y) \in \mathbb{K} \times \mathbb{K}\} \bigcup \{\mathcal{O}\}$$

tales que (x, y) verifican la curva elíptica anterior y $\mathcal{O}$ es un punto que no esta en el plano y se llama "*punto del infinito*" forman un grupo abeliano respecto a la suma.

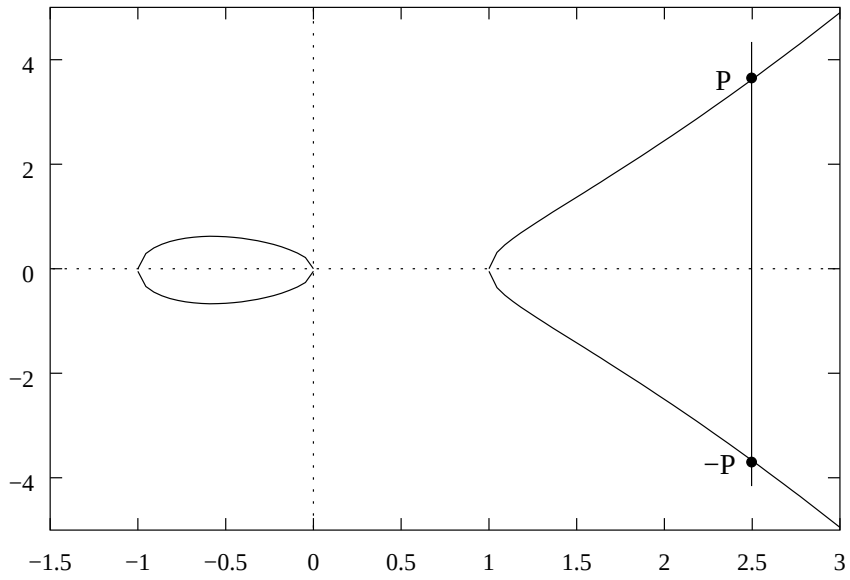
Suma de dos elementos de una curva elíptica:



Suma de un elemento con él mismo:



Suma de un elemento con su opuesto:



Podemos definir en E un producto por escalares de la forma:

$$n \cdot P = P + \overset{n)}{\ldots} + P$$

donde $P \in$ E y $n \in \mathbb{Z}$.

Firmas digitales

Que son las firmas digitales

Las firmas digitales son una solución que ofrece la criptografía para verificar:

- La integridad de documentos.
- La procedencia de documentos.

Las firmas digitales se pueden realizar tanto con criptografía simétrica como asimétrica.

Funciones resumen (Hash)

Estas funciones producen resumenes de textos, es decir, dado un mensaje una función "hash" produce un resumen del mismo.

Para que este tipo de funciones sean utiles, criptográficamente, deben cumplir:

- El resumen es de longitud fija.
- Calcular el resumen de un mensaje es fácil. (tiempo polinomial)
- Dado un resumen es computacionalmente imposible calcular el mensaje que lo generó. (tiempo exponencial)
- Es practicamente imposible obtener dos mensajes que generen el mismo resumen.

Se recomienda utilizar signaturas de al menos 128 bits. El tamaño más usado para signaturas es de 160 bits.

Como se firma digitalmente

Primero se produce un resumen del mensaje, luego se encripta este resumen. Si estamos utilizando criptografía asimétrica lo encriptamos con nuestra clave privada.

De esta forma la única persona que conozca la clave privada será capaz de firmar digitalmente en nuestro nombre.

Normalmente para hacer hash de un mensaje se divide en bloques y se suele incluir la longitud del mensaje original para evitar que dos mensajes de diferente longitud produzcan el mismo resumen.

Con algoritmos asimétricos nunca se debe firmar un mensaje después de encriptarlo ya que existen ataques para romper el criptosistema.

Como comprobar una firma digital

Procederemos de la siguiente forma:

- Desencriptamos la firma digital, usando la clave pública si han utilizado un método asimétrico. Obtenemos el resumen del mensaje original.

- Hacemos un hash sobre el mensaje original.

- Comprobamos nuestro resumen con el obtenido al desencriptar y si coinciden la firma digital es válida.

Algoritmos de firmas digitales

Algoritmos asimétricos empleados en firmas digitales:

- RSA.
- ElGammal.
- MD5.

  Desarrollado por Ron Rivest, fue una modificación del MD4 y produce firmas de 128 bits sobre bloques de 512 bits.

- SHA.

  Desarrollado por la NSA. Se considera seguro y produce firmas de 160 bits a partir de bloques de 512 bits.

Certificados

Que son y para que valen

Mediante la criptografía podemos establecer comunicaciones seguras con otras personas. ¿Pero esas personas son quienes dicen que son?. ¿Quien nos lo asegura?.

Para resolver este problema surgen unas entidades, que tienen que ser de confianza para todo el mundo. Estas entidades "certifican" las claves públicas de los usuarios que se las remiten emitiendo un certificado.

Existen cuatro tipos de certificados, cada uno de ellos ofrece un nivel de seguridad diferente.

Quien da certificados

Los certificados los tiene que conceder "alguien". Tiene que haber una tercera parte, de confianza para emisor y receptor, que certifique a ambos.

Estas terceras partes reciben el nombre de "Trusted Third Parties (TTP)", terceras partes de confianza. Su labor, basicamente, consiste en identificar a usuarios, empresas o servicios con una determinada clave pública.

Las entidades que emiten certificados reciben el nombre de "Certification Authority (CA)", las cuales han de ser de confianza y pueden estar certificadas por ellas mismas o por otra CA superior.

En España tenemos la "Agencia de Certificación Española (ACE)", `www.ace.es`, que ofrece el servicio de certificacion SET.

En que consiste un certificado

Básicamente un certificado esta compuesto por varios campos:

- Identidad del propietario.
- Clave pública.
- Periodo de validez.
- Identidad y clave pública de la CA que lo expidió.
- Firma digital del certificado. Esta firma esta realizada por la CA.

Revocaciones de certificados

La seguridad de un certificado reside en la confidencialidad de la clave privada. Al ser los certificados de dominio público cualquiera puede comprobarlos y lo unico que le certifican es quien es el propietario de una determinada clave pública. Ese propietario es la "única" persona que conoce la clave privada correspondiente a la pública certificada.

La persona que conozca una determinada clave privada puede certificarse utilizando el certificado de la correspondiente clave pública y de esta forma ofrecer una "falsa" seguridad a la persona que ha verificado el certificado (suplantación de personalidad).

Cuando un certificado ha sido revocado la CA que lo expidió lo coloca en su "Certificacion Revocation Lists (CRL)", lista de revocación de certificados.

Comprobación de certificados

Cuando queremos establecer una comunicación con "alguien" debemos poseer su clave pública. Una vez conseguida debemos asegurarnos que es quien dice ser. Para ello solicitamos su certificado y comprobamos su validez de la siguiente forma:

1. Comprobamos su caducidad.

2. Comprobamos que CA lo expidió.

3. Comprobamos el certificado de la CA, asi como todos los certificados de CA's superiores que haya, si se da el caso, certificado a la CA anterior.

4. Comprobamos que el certificado no fue manipulado, comprobando la firma digital de la CA.

5. Comprobamos CRL de la CA para verificar que no ha sido revocado.

Después de todo esto podemos establecer una conexión “segura” y “autenticada”.

Criptografía en LiNUX

Autenticación en LiNUX (I)

La autenticación en LiNUX se lleva a cabo mediante el proceso `login`. Para identificarse ante el sistema cada usuario tiene un "login" y un "password".

Estos datos, asi como otros, los almacena el sistema en `/etc/passwd`, por cada usuario hay una linea como la siguiente:

```
root:59AFDADF7D:UID:GUID:root:/root:/bin/bash
```

En el segundo campo esta la contraseña del usuario encriptada con el programa `crypt`, utilizando `MD5` ...

Este fichero ha de ser público y accesible a todo el mundo, razón por la cual se podía acceder al fichero, no solo por usuarios del sistema, sino por ftp anónimo (cuando era posible) y esto daba la posibilidad de ataques a base de diccionario.

El programa Crypt

Es una variacion del DES. Sus modificaciones van encaminadas a impedir que se puedan desarrollar herramientas para la busqueda de claves. Es irreversible, si se conoce un texto encriptado con `crypt` es imposible desencriptarlo.

`Crypt` encripta la contraseña de cada usuario utilizando la propia contraseña y "algo" llamado `salt`.

El "salt" no es mas que una secuencia de 12 bits que se utiliza para permutar la contraseña elegida por el usuario. Se elige de forma aleatoria para cada usuario. Esto se utiliza para evitar que dos contraseñas iguales den un mismo texto encriptado.

Según la “documentación” la forma en la que es almacenada la password encriptada es:

`login:frargtYuJ09IKMh:UID:GID:Nombre:Home:Shell`

donde “`fr`” es la salt con la que fue encriptada el password y “`argtYuJ09IKMh`” es el password encriptado con la salt anterior.

Hoy en día todas las distribuciones dan la opción de utilizar `MD5` como función de hashing en lugar de `Crypt`. Es recomendable utilizar `MD5` en lugar de `Crypt`.

Autenticación en LiNUX (II)

Supongamos que en `/etc/passwd` tenemos la siguiente linea:

`tux:frargtYuJ09IKMh:...`

cuando el usuario **tux** hace `login` para verificar al usuario se hace lo siguiente:

1. Solicitar el `password`.

2. Obtener la `salt` de `/etc/passwd`, en este caso la salt sería "`fr`".

3. Encriptar el password suministrado utilizando la `salt` anterior. Mediante el programa `crypt`.

4. Comprobar la encriptación obtenida con la almacenada en "`/etc/passwd`":

   `argtYuJ09IKMh`

5. Si ambas coinciden se permite la entrada al sistema del usuario, en caso contrario no se permite.

Shadowing de contraseñas

El sistema anterior tiene un problema y es que el fichero "`/etc/passwd`" ha de tener permiso de lectura para todo el mundo, con lo cual la salt con la que cada contraseña fue encriptada es conocida, así como la encriptación. Entonces cualquier usuario puede atacar el sistema utilizando un diccionario junto con herramientas desarrolladas para ello.

La solución a esto es utilizar "*shadow password*" . Basicamente lo que hace esto es reemplazar en el fichero "`/etc/passwd`" el password del usuario por un caracter, normalmente "`x`", y almacenar el password en un fichero que unicamente pueda leer el superusuario, "`/etc/shadow`" .

Instalar *shadow passwords* implica el modificar ciertas utilidades del sistema, ya que aquellas utilidades que requieran comprobar el password de un usuario no podrán ya que ahora el password se encuentra en otro fichero. La suite *Shadow* trae una serie de utlidades que reemplazarán a las más comunes en el sistema:

```
chfn, chsh, id, login, newgrp, passwd,
su
```

También añade otra serie de utilidades.

Elección de buenas contraseñas

Para que una contraseña sea buena tiene que cumplir varios requisitos:

- Que no sea una palabra que exista en algún idioma o que tenga algún significado (siglas, fechas, matriculas,. . .)

- Ha de ser una combinación (grande) de mayúsculas, minúsculas, números . . .

- El único sitio donde puede estar almacenada será en nuestra cabeza o en su defecto en algún sitio con garantías (una caja de seguridad).

- Ha de verificar el protocolo **KISS**:

Keep **I**t **S**ECRET, Stupid!

Una clave por muy segura que sea cuantas más personas la conozcan más fácil será que alguien ajeno al sistema la consiga. B. Franklin dijó una vez:

> *Tres personas pueden compartir un secreto si dos estan muertas.*

P.A.M.

P.A.M. (**P**luggable **A**uthentication **M**odules) es una librería que permite al administrador de un sistema elegir que métodos de autenticación va a utilizar el sistema. Esto permite adaptar la seguridad de nuestro sistema según nuestras necesidades.

Utilizando esta librería podemos personalizar los métodos que utilizaran las aplicaciones para autenticar a los usuarios, siempre y cuando dispongamos del código fuente de las aplicaciones. Con esta librería no es necesario realizar muchos cambios en el código fuente de las aplicaciones.

C.F.S

C.F.S. (**C**ryptographic **F**ile **S**ystem)

Esta utilidad permite la creacion de ficheros y directorios encriptados dentro del sistema de ficheros que estamos utilizando. Para poder acceder a estos ficheros se necesitará una contraseña que habrá sido establecida cuando se creó el directorio.

Este software esta basado en D.E.S. y esta sometido a las leyes de exportación de los Estados Unidos.

T.C.F.S

C.F.S. (**T**ransparent **C**ryptographic **F**ile **S**ystem)

Desarrollado por la Universidad de Salerno específicamente para Linux. Su propósito es proporcionar seguridad en sistemas de ficheros NFS. La desencriptación se realiza en el cliente.

Este software funciona a nivel de nucleo, no a nivel de usuario, con lo cual se gana en seguridad.

Esta basado en D.E.S. y para funcionar las máquinas clientes necesitan una versión compilada del nucleo para soporte de T.C.F.S.

Kerberos

`Kerberos` es un sistema de autenticación. Este sistema trata de solucionar el problema de la autenticación tradicional, la cual manda las passwords como texto en claro.

Para funcionar necesita haber un servidor `kerberos`, lo cual implica unos problemas:

- Si el servidor no funcionara toda la red estaría inactiva.

- Si se ve comprometida la seguridad del servidor `kerberos` quedaría comprometida la seguridad de toda la red.

También tenemos otros problemas:

- Necesitamos `kerbetizar` todas las aplicaciones y servicios que requieran autenticación con kerberos.

- Necesitamos disponer del código fuente de las aplicaciones y servicios. Con software propietario esto es imposible, a menos que exista la versión `kerbetizada` y estemos dispuestos a pagar por ella.

- La `kerbetización` no es siempre fácil.

El proceso de autenticación en kerberos es:

1. El cliente se autentica una vez en el centro de distribución de claves (KDC).

2. El KDC da un `Ticket Granting Ticket`, TGT. Ticket especial para obtener otros Tickets. Tiene un periodo de caducidad, normalmente de unas 8 horas.

3. Cuando necesitamos utilizar un recurso solicitamos un ticket utilizando el TGT para obtener un ticket `Ticket Granting Server`, TGS. El cual nos da acceso al recurso solicitado.

IPSec

Como es sabido el protocolo http es un protocolo que manda información como texto plano, sin encriptar.

Este protocolo se encarga de encriptar los "datos" de los paquetes a nivel IP, esto significa que no hace falta el modificar las aplicaciones que utilizan estos protocolos para utilizarlo. De esta forma podemos establecer comunicaciones "seguras" con unos cambios mínimos en el sistema.

Este protocolo esta disponible para otros S.O. por lo cual podremos establecer comunicaciones "seguras" con otros S.O.

Para poder instalar esto necesitamos tener compilado el nucleo con soporte para IPSec :-(.

SSH

Secure Shell o SSH es un paquete orientado a establecer comunicaciones seguras entre equipos.

Programas como `telnet` mandan toda la información en claro, incluido el login y password.

Este paquete contiene aplicaciones que nos permiten interactuar con otros equipos de forma segura y transparente para el usuario.

GnuPG

GnuPg también conocido com Gpg es la versión libre del conocido PGP de Phil Zimmermann. Es libre para uso no comercial y para uso comercial. Además se dispone del código fuente con lo cual es posible auditar el código y comprobar si tiene fallos de seguridad y/o funcionalidades no documentadas.

Permite encriptar y desencriptar ficheros y realizar firmas digitales. No utiliza algoritmos patentados.

Generación de claves

Para generar nuestras claves tendremos que hacer:

```
gpg --gen-key
```

A continuación tendremos que elegir el tipo de clave:

- DSA y ElGamal.

  Creamos un par de claves. El primer par, DSA, para firmar y el segundo para encriptar/desencriptar.

- DSA. Sólo firma.

  Creamos una clave, sólo para firmar.

- ElGamal. Firma y cifrado.

Creamos una clave que utilizaremos para firmar y encriptar/desencriptar.

A continuación tendremos que elegir el tamaño de la clave:

- 768 bits. Insegura.
- 1024 bits. Seguridad buena aunque esta en la zona de riesgo.
- 2048 bits. “Segura”.

Elegiremos la fecha de caducidad e introduciremos varios datos:

- Nombre y apellidos.
- Dirección de correo electrónico.
- Comentario.

A continuación necesitamos introducir una contraseña para proteger nuestras claves.

Caracteristicas de GnuPg

1. Ver nuestras claves públicas.
   `gpg --list-public-keys`.

2. Ver nuestras claves secretas.
   `gpg --list-secret-keys`.

3. Generar certificados de revocación.
   `gpg --gen-revoke`.

4. Crear firmas digitales.
   `gpg --sign`.

5. Crear firmas digitales en formato ASCII.
   `gpg --clearsign`.

6. Verificar firmas digitales.
   `gpg --verify`.

7. Encriptar datos.
   `gpg --encrypt`.

8. Desencriptar datos.
   `gpg --decrypt`.

9. Generar una salida con armadura ASCII.
   `gpg --armor`.

10. Ver las huellas digitales.
    `gpg --fingerprint`.

Anillos de confianza

Loa anillos de confianza son el sustituto a las autoridades de certificación en GnuPg y tiene como fin la autenticación de las claves públicas de los usuarios.

La forma de hacer esto es firmar las claves de aquellos en los que confiemos plenamente y tengamos identificados. De esta forma todos aquellos que confien en nosotros confiarán también en aquellos a los que hayamos firmado su clave pública.

A pesar de tener firmada una clave podemos asignarle un grado de confianza. Existen cuatro niveles de confianza:

Unknown, desconocido.

None, no nos fiamos. El usuario que firmó la clave se puede dedicar a firmar todas las claves sin comprobaciones.

Marginal, confiamos pero con reservas.

Full, confianza plena.

Comercio electrónico

+

Que es SSL

SSL es un acrónimo de:

Secure
Sockets
Layer

Fue diseñado por Netscape en 1,994 con el proposito de mejorar la seguridad de las comunicaciones a traves de Internet.

Hoy en día es un estándar.

Ofrece los siguientes servicios:

- Encriptación de datos. (DES, 3-DES, RC2, RC4, IDEA, RSA)

- Integridad de mensajes. (MD5, SHA)

- Autenticación tanto del servidor de destino, como del cliente (opcional). (RSA)

SSL es un protocolo para comunicaciones “seguras”.

Como funciona SSL

Una comunicación mediante SSL tiene varias etapas:

1. Ambas partes se ponen de acuerdo en los algoritmos que van a utilizar y la longitud de claves.

2. Autenticación. El servidor se identifica con un certificado X.509v3(suministrando su clave pública) y si hay autenticación de cliente el navegador solicita el certificado X.509v3 al cliente.

3. Se establece la clave de sesión que será utilizada posteriormente con un algoritmo simétrico para establecer la comunicación. El cliente manda una clave inicial encriptada con las clave pública del servidor y a partir de esa clave se genera la clave de sesión.

4. Se verifica la autenticidad de los datos, que el canal es seguro y se puede empezar la transmisión.

SSL es un protocolo que se instala entre los niveles de transporte y de aplicación, con lo cual puede ser utilizado con pequeñas modificaciones en los programas que utilizan protocolos de red.

Ventajas y desventajas de SSL

Ventajas:

- Proporciona seguridad durante la transmisión.
- Es transparente para el usuario.
- No son necesarias muchas modificaciones en los programas que lo utilizan.

Desventajas:

- Debido a las fuertes restricciones que tenian los EEUU hasta hace poco, las versiones de exportación tanto de Netscape como del otro navegador utilizaban claves de sesión de 40 bits.

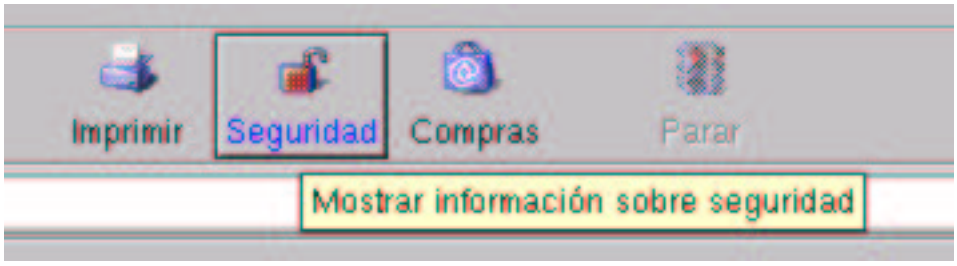
  Estas longitudes de clave garantizan la seguridad durante la transmisión, pero no después de ella, ya que estos datos pueden ser almacenados y criptoanalizados con éxito al tener una clave de 40 bits.

  Comprueba tú navegador preferido y desactiva todas las opciones para utilizar algoritmos simétricos con claves menores de 128 bits y claves menores de 1,024 bits con algoritmos asimétricos.

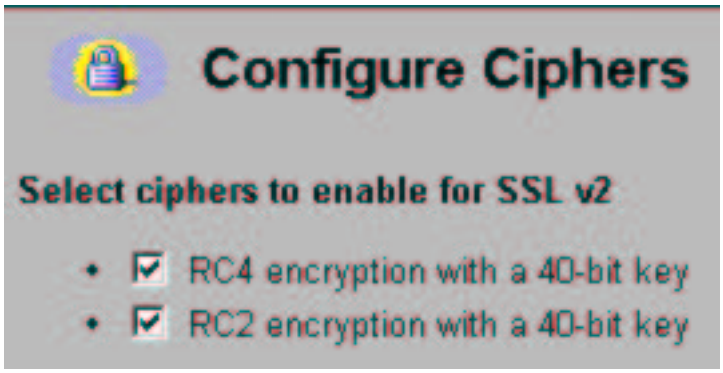
Fortify

`Fortify` es un programa para Netscape que existe en todos los S.O. en los que funciona Netscape. Este programa activa las opciones de criptografía fuerte presentes en Netscape y que no son accesibles en las versiones de exportación. Hasta hace poco todos los productos Americanos de criptografía tenían unas fuertes restricciones para su exportación.

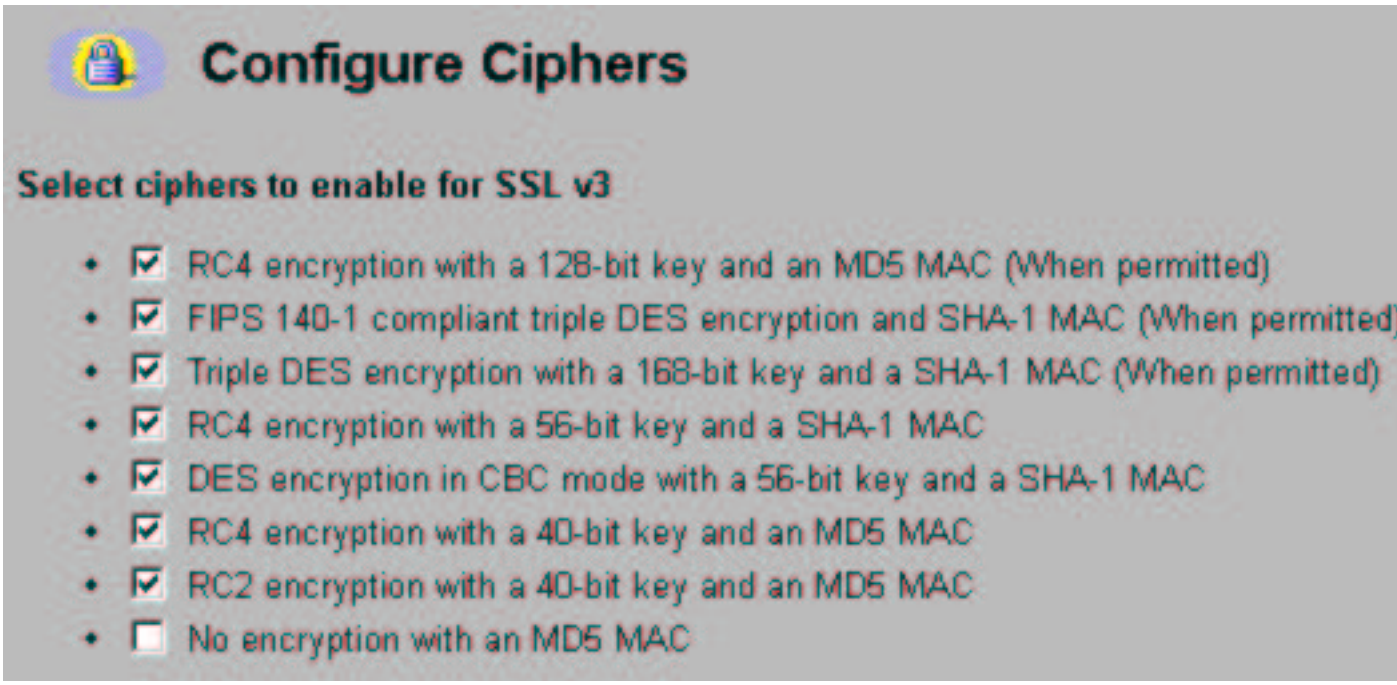
Para ver las opciones de seguridad:



Las versiones de exportación usan claves de 40 bits para SSL versión 2:



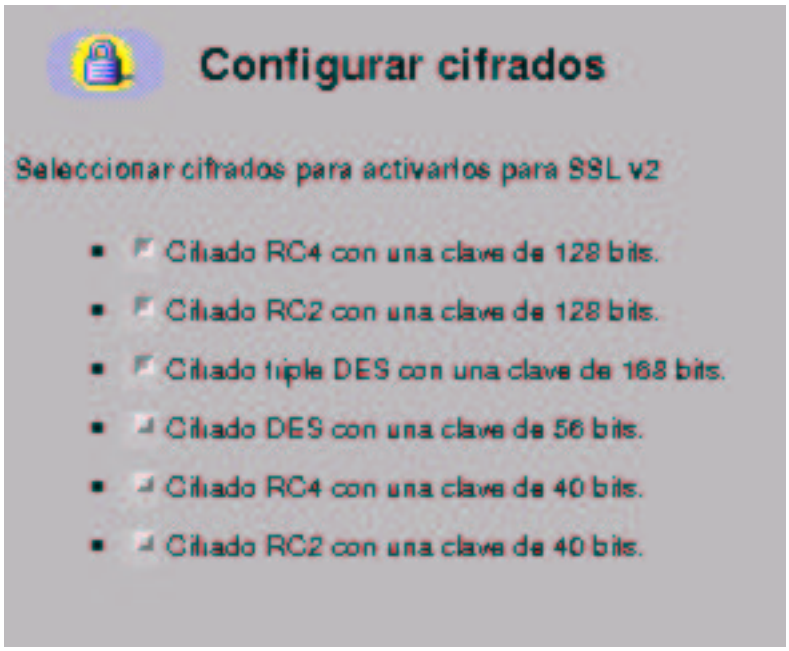
Y para la versión 3 de SSL las cosas no mejoran mucho:



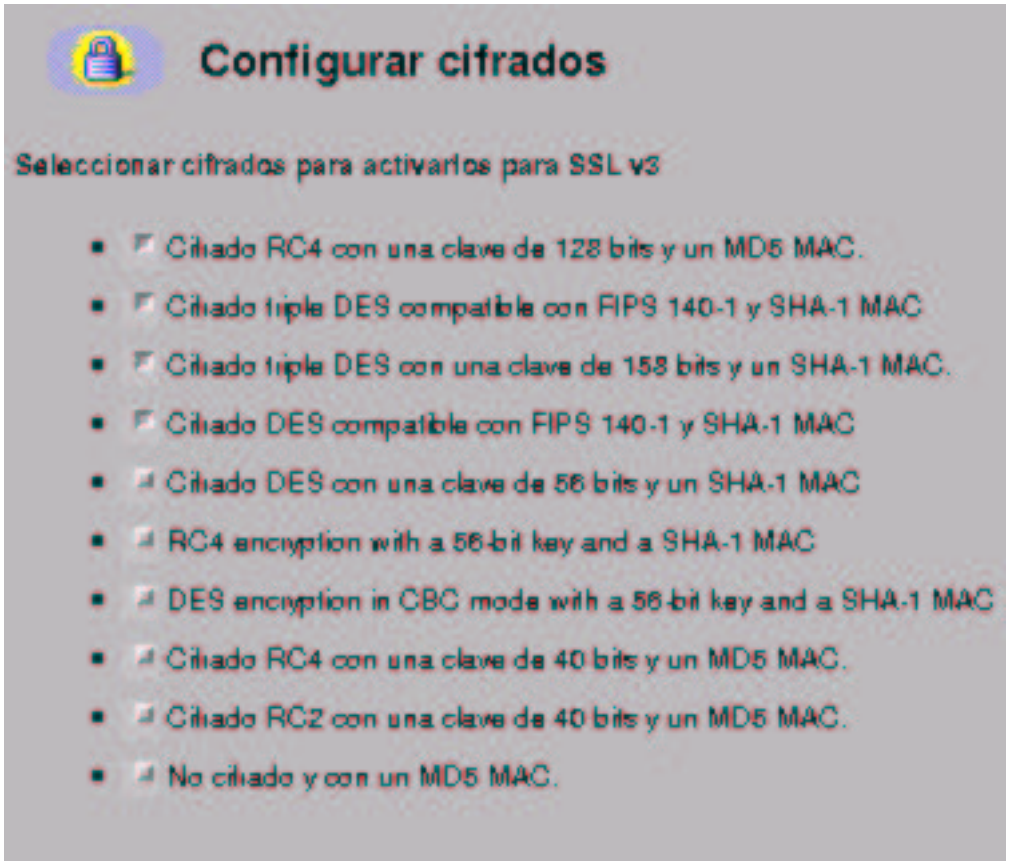
Estas claves unicamente protegen los datos durante el tránsito, pero no después ya que pueden romperse en menos de un día con los ordenadores de hoy en día.

Utilizando `Fortify` obtendremos más seguridad en SSL.

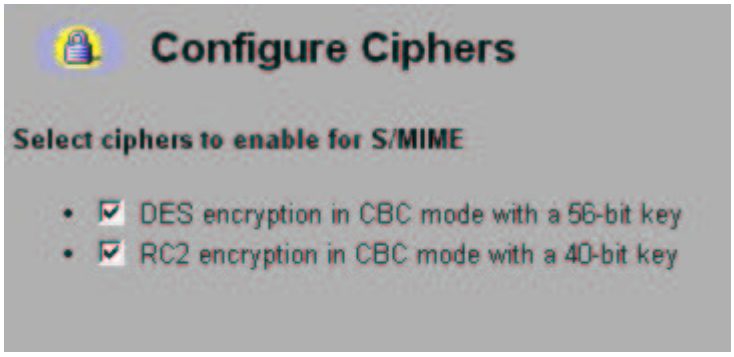
Para SSL versión 2:



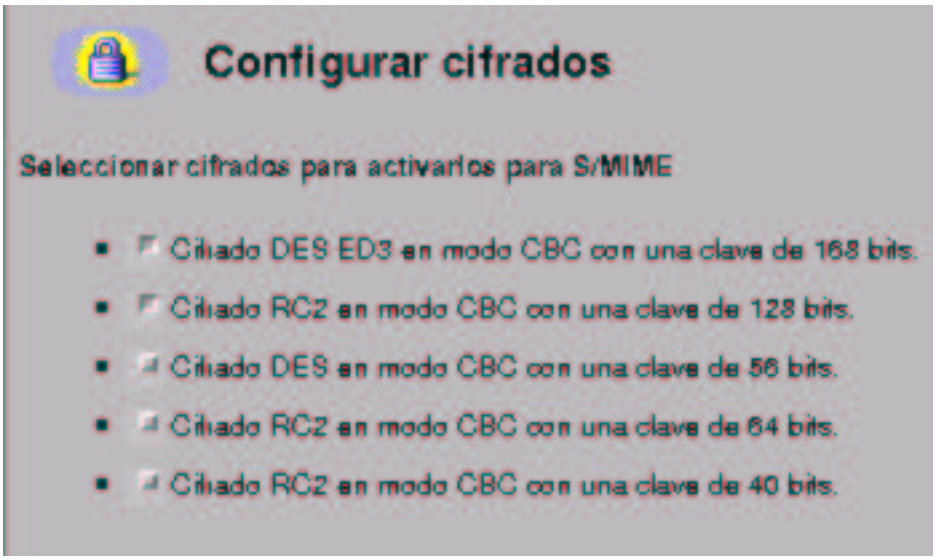
y para SSL versión 3:



Haciendo lo mismo para messenger tendremos que antes de utilizar `Fortify`:



y depués de utilizar `Fortify`:



Que es SET

En 1,985 las compañias Visa y MasterCard junto con otras empresas punteras en el sector tecnológico como RSA, IBM, Netscape, VeriSign y Microsoft (si, ya se que he dicho punteras ;-)) decidieron desarrollar un protocolo respaldado por la industria y estandarizado desde el primer momento. El proposito de este protocolo es favorecer las transacciones electrónicas con tarjetas de credito a través de redes electrónicas.

Dicho protocolo recibe el nombre de "Secure Electronic Transaction(SET)".

SSL fue diseñado para proteger comunicaciones entre dos usuarios, entidades, . . . Además SSL no protege contra usos fraudulentos de las tarjetas de credito.

Una transacción económica mediante una tarjeta de credito involucra a más de dos entidades(cliente, comerciante, bancos, CA's).

Como funciona SET

SET funciona de manera que permite confiar en todos los participantes en la transacción electrónica:

Autenticación, todas las partes pueden autenticarse de forma fiable mediante la utilización de certificados digitales. Evitando de esta forma el uso fraudulento de las tarjetas, suplantación de sitios web (web spoofing).

Confidencialidad, SET encripta el número de la tarjeta de credito, de tal forma que el vendedor no tiene acceso al número de la tarjeta, evitando un posible uso fraudulento por parte del comerciante.

Integridad, para verificar que la información no ha sido alterada se utilizan las firmas digitales.

Intimidad, el banco no tiene acceso a las compras del cliente con lo que no puede elaborar estudios de mercado, hábitos de consumo, . . .

Verificación inmediata, antes de completar la compra el vendedor puede verificar si el cliente dispone del saldo necesario para pagar.

No repudio, mediante el uso de certificados digitales X.509v3. Lo que impide hacer una compra y luego negarse a pagarla con el pretexto de no haberla realizado.

Software

Software Criptográfico

Sistemas criptográficos de ficheros

cfs, *Cryptographic File System*

sfs, *Self-certifying File System*
`http://www.fs.net`

tcfs, *Transparent Cryptographic File System*
`http://www.tcfs.it`

Seguridad en comunicaciones

Fortify, activa opciones de criptografía fuerte en versiones de exportación de Netscape.
`http://www.fortify.net`

Gnupg y PGP

PGP, programa para la encriptación de documentos, correo electrónico, . . .
`http://www.pgpi.com` (libre para uso no comercial)

Gnupg, programa para la encriptación de documentos, correo electrónico, . . .
`http://www.gnupg.org` (libre)

Pgp4pine, añade funcionalidad para el uso de `gnupg` con `pine` (libre)

TkPgp, interface gráfica para `gnupg` basada en Tcl/tk.

Librerias

Miracl, libreria de C/C++ para criptografía de clave pública.
`ftp://ftp.compapp.dcu.ie/pub/crypto/miracl.zip`
(libre para usos académicos)

Protocolos criptográficos

SSH, Conjunto de aplicaciones para establecer comunicaciones encriptadas.
`http://www.ssh.com` (producto comercial)

OpenSSH, Conjunto de aplicaciones para establecer comunicaciones encriptadas, basado en SSH y desarrollado con código libre.
`http://www.openssh.org`

SSL, Protocolo para establecer comunicaciones encriptadas a traves de la web.
`http://www.ssl.com` (producto comercial)

OpenSSL, Protocolo para establecer comunicaciones encriptadas a traves de la web, basado en SSL y desarrollado con código libre.
`http://www.openssl.org`