Alejandro Corletti Estrada (acorletti@hotmail.com)

Algunos Conceptos de Criptografía

Por: Alejandro Corleti

Algoritmos de autenticación y encriptado.

HMAc con MD5 [RFC-2403]
HMAC con SHA-1 [RFC-2404].
HMAC [RFC-2104].
Diffie-Hellman.
DES (Data Encriptation Standard) [ANSI X3.106] en modo CBC.
MD5 (Message Digest Algorithm Versión 5) [RFC-1321] y SHA (Secure Hash Standard) [FIPS- 180-1, de NIST].
3DES (triple DES) para encriptado.
Tiger para Hash.
DSS (Digital Standard Signature).
RSA (Rivest, Shamir and Aldeman).

a. Empleo y conceptos de clave simétrica y asimétrica.

La criptografía es la ciencia que permite convertir un determinado conjunto de códigos generalmente comprensibles en otro cuyas características lo hacen de difícil interpretación.

Este proceso se lleva a cabo a través de algoritmos matemáticos combinados con claves. El mayor o menor grado de complejidad del algoritmo, combinado con una determinada longitud de clave, harán que el nivel de dificultad para poder descifrar o interpretar un mensaje sin ser parte de los miembros que realizan estos pasos (es decir intruso) sea una tarea simple, dificultosa o prácticamente imposible.

Este arte, es milenario, y se empleó generalmente en el transporte de información de carácter militar o política. En sus comienzos se lo utilizaba para mensajes en tránsito, pero casi asociado a la aparición de la informática, se comienza a hacer necesario su empleo también en el almacenamiento de la información, pues la misma se presenta ahora de una manera más accesible a individuos que no necesariamente deberían tener acceso. Es evidente que con la aparición de las redes y la posibilidad de interconexión que existe hoy, la importancia de mantener la confidencialidad de la información por todos los métodos existentes, hace necesario el uso de esta herramienta en muchas actividades cotidianas.

Uno de los conceptos a tener en cuenta es que la información en tránsito en todos los casos es escuchada por muchas personas, ya sea dentro de una red LAN como cuando se desplaza a través de vínculos WAN tanto dedicados como por medio de redes X.25 o Frame Relay. Cualquiera que posea las herramientas adecuadas puede decodificar los distintos encabezados de estas tramas o paquetes y acceder a la información que está viajando.

Al criptografiar información, si se intenta interpretarla, será necesario poder revertir el proceso, para volver al mensaje original. Como se mencionó con anterioridad, el empleo de la clave es lo que hace posible esta actividad, la cual es análoga a la llave de un candado o cerradura y el algoritmo es

el conjunto de engranajes o mecanismos. La cantidad de trabas o bolillas que posea esta cerradura, determinará la cantidad de posibilidades que existen para abrir o cerrar ese acceso. Si se mantiene este ejemplo para el caso de un candado de clave numérica, este podría tener cuatro ruedas de diez dígitos de las cuales una sola combinación liberaría el mecanismo, generando una posibilidad entre diez mil. Criptográficamente hablando, tendría un **ESPACIO DE CLAVES** de 10.000 posibilidades.

Si alguien tuviera acceso a este candado sin conocer la combinación (Clave), podría probar desde el 0000, luego el 0001, el 0002..... y así sucesivamente hasta lograr abrir el candado. Nuevamente en terminología criptográfica, esto se llamaría **FUERZA BRUTA**. Se debe tener en cuenta que si este intruso pudiera tener algún indicio respecto a esta combinación, como por ejemplo que la misma empieza con el dígito cuatro , el espacio de claves queda reducido a cien, con lo cual el tiempo necesario para abrir el candado sería diez veces menor. Esta última, es de especial interés pues en informática existe un sinnúmero de métodos de reducir el ESPACIO DE CLAVES, minimizando el tiempo necesario para vulnerar textos cifrados. Por ejemplo, un método muy conocido de cifrado (que se mencionará posteriormente) es el **DES** que tiene una clave de 64 bit. A primera vista se pensaría que su espacio de claves es 2^{64}, pero como primer detalle, esta clave emplea 8 bit de paridad, con lo cual ya se tiene un espacio de claves de 2^{56} claves independientes. Como segunda medida se puede emplear por ejemplo un método llamado criptoanálisis diferencial que reduce las posibilidades a 2^{34}, con lo cual se reducen los tiempos casi a la mitad. Esto se menciona sólo a título de ejemplo, pues si se tiene en cuenta que cada intento de estos trae aparejado la comparación de un texto para verificar si es entendible, la realidad hace que si se trata de un mensaje extenso esta actividad sea poco rentable, existiendo métodos de análisis mucho más eficientes.

Otro detalle significativo es que la **ENTROPIA** de los códigos reales no es máxima en ningún caso, es decir que la frecuencia de aparición de los distintos símbolos no es equiprobable. Si se analiza cualquier texto escrito en Castellano, se podría apreciar que determinadas letras tienen mayores ocurrencias de aparición que otras, por ejemplo la E, N, S, A, etc. y por el contrario es poco frecuente encontrar X, W. Esta característica es de sumo interés pues un mensaje cifrado si se puede determinar su alfabeto fuente (por ejemplo Castellano) allanará las posibilidades de criptoanálisis. Es evidente que este detalle causa más impacto cuanto mayor sea la longitud del texto, pues en unas pocas palabras esta frecuencia de aparición puede no ser cierta, y absolutamente lo será si la longitud tiende a infinito. También se tiene en cuenta el **CODIGO EN SU EXTENSION**, pues volviendo al mismo ejemplo, la secuencia de extensión dos TH en Castellano tiene una probabilidad excesivamente baja pero en Inglés es cotidiana; más aún si se extiende en grado tres THR en Castellano no existe y en Inglés sí. Estos breves conceptos sirven para ilustrar escuetamente cómo se puede reducir de distintas maneras la tarea de **criptoanalizar** un mensaje, desde ya que estos conceptos son básicos, y que en la actualidad, esta actividad se encuentra altamente avanzada y tremendamente favorecida por las altas velocidades de procesamiento actuales.

El principio de funcionamiento de los distintos algoritmos criptográficos y sus claves correspondientes, se puede clasificar clásicamente de tres formas: **Simétrico, asimétrico e híbrido**. Se podría considerar una cuarta forma que se debería llamar **irreversible** (como se puede considerar el sistema de claves de Unix)**.** A continuación se detallan cada una de ellas.

¿Cuánto se tarda en romper una clave hoy?

Password Length	All Characters	Only Lowercase
3 characters	0.86 seconds	0.02 seconds
4 characters	1.36 minutes	.046 seconds
5 characters	2.15 hours	11.9 seconds
6 characters	8.51 days	5.15 minutes
7 characters	2.21 years	2.23 hours
8 characters	2.10 centuries	2.42 days
9 characters	20 millennia	2.07 months
10 characters	1,899 millennia	4.48 years
11 characters	180,365 millennia	1.16 centuries
12 characters	17,184,705 millennia	3.03 millennia
13 characters	1,627,797,068 millennia	78.7 millennia
14 characters	154,640,721,434 millennia	2,046 millennia

Cifrado simétrico:

Este tipo de algoritmo, emplea la misma clave para cifrar que para descifrar. Este algoritmo es el primero de todos, y se empleo casi con exclusividad hasta principios de los ochenta. Los ejemplos más conocidos son el DES, Triple DES, CAST, RC 4 y RC 5, Blowfish, IDEA, y CAST.

Como ejemplo simple, se desea encriptar el siguiente:

código ASCII: Mje_{fuente} = {77, 69, 83, 65} = {MESA}
El algoritmo consiste en sumarle siete a cada símbolo (Clave = 7).
$Mensaje_{Cripto}$ = {84, 76, 90, 72} = {T, L, Z, H}
Para desencriptarlo, el algoritmo será la resta, y la Clave será la misma = 7 .

Este ejemplo que a simple vista parece trivial, de hecho no lo es tanto pues fue utilizado durante muchos siglos, y se llamaba el algoritmo del César. Se puede hacer la prueba de redactar un texto cualquiera y emplear distintas claves, comprobando que no es tan simple decodificarlo, también se puede incrementar la complejidad del algoritmo con distintas operaciones simples e inclusive con combinaciones de ellas, logrando paso a paso un grado de dificultad cada vez mayor.
Como conclusión, se puede apreciar que se emplea la misma clave para encriptar que para desencriptar, y el algoritmo es la inversa. Si se desea incrementar el grado de dificultad, se realiza fácilmente, incrementando la cantidad de operaciones y la longitud de la clave.
El gran problema radica en la forma en la cual se hace llegar la clave pues debería ser por otro medio de comunicaciones debido a que justamente este no es seguro. Este método posee la gran debilidad que esta clave no puede ser difundida pues a medida que más de dos personas conocen un secreto, este poco a poco va dejando de serlo. El gran inconveniente radica en que se está creyendo que la información es confidencial y sobre este concepto se fundamenta la toma de decisiones, siendo esto más peligroso que si se es consciente que la información puede ser escuchada y se opera al respecto.

Cifrado Asimétrico:

A mediados de los años 70` se descubre esta nueva técnica que permite el cifrado de una manera diferente, haciendo especial hincapié en la preservación del secreto, el cual como se mencionó anteriormente, sólo es seguro si lo conoce una sola persona (es decir el propietario) pues si ya se difundió a alguien más se pierde la certeza de su no distribución, los algoritmos más empleados en la actualidad son **RSA** y **Diffie-Hellman**. Se basan todos en un par de claves denominados "Pública y privada". Hoy básicamente, existen tres tipos de algoritmos matemáticos:

- **Logaritmos entero discretos.**
- **Factorización de Números primos.**
- **Curvas elípticas.**

Esta técnica se basa en el empleo de dos claves, una **PUBLICA** la cual se difunde sin ninguna limitación, como se hace con un número telefónico en la guía correspondiente, esta clave es la que se emplea para emitir un mensaje cifrado, es decir la emplea cualquier persona que desee enviar un mensaje seguro hacia un determinado remitente (por supuesto que cada remitente tendrá su propia clave pública, la cual dará a conocer a todas las personas que necesiten emplearla sin limitación). La segunda es la clave **PRIVADA**, la cual es conocida únicamente por su propietario y nadie más, y es la que se emplea para descifrar el mensaje cifrado con su correspondiente clave pública. Como se puede suponer, ambas claves están asociadas de alguna manera, conformando un PAR DE CLAVES, causa por la cual no es imposible partiendo desde la clave pública obtener la privada, pero en la actualidad aún es excesivamente cara la inversión de tiempo, recursos y algoritmos necesarios para hacerlo, y más aún a medida que se emplean claves extensas; esta cierta complejidad es la que hace a esta técnica altamente confiable y la convierte en la más segura que se emplea en la actualidad.
La lógica es la siguiente:

$Mje_{fuente} = \{M\}$
ClAVE PÚBLICA $<Mje_{fuente} = \{M\}> = Mje_{cripto}$ ⟶ Emite
Recibe ⟶ ClAVE PRIVADA $<Mje_{cripto}> = Mje_{fuente}$

En este método se soluciona el problema de la distribución de claves, pues no se necesita otro canal o procedimiento para esta tarea, pues no es necesario.

El primer problema que se plantea es cómo se puede estar seguro que la clave pública que dice ser, realmente es; es decir una persona puede confiar en la guía telefónica que publica anualmente la o las Empresas de telefonía local porque sabe que las mismas fueron impresas por esta Empresa y es su responsabilidad ser veraces, también se puede confiar en un teléfono o dirección que me suministra una persona conocida siempre y cuando la misma esté considerada como "confiable"; pero qué sucedería si la clave pública que se obtiene no es en realidad la que se corresponde con el remitente al cual se le desea enviar un mensaje cifrado, como ejemplo se presenta el siguiente:

PROCEDER CORRECTO

A desea enviar un mensaje a B.
A busca en la guía G la clave pública de B, la cual será $K_{B(Pub)}$
Redacta el mensaje M = Mje M.
Lo criptografía con la clave pública de B, ⟶ $K_{B(Pub)}\{Mje\ M\} = Mje_{cripto}$
Lo envía a B.
B lo recibe.
B lo decriptografía con su clave privada $K_{B(Priv)}$ ⟶ $K_{B(Priv)}\{Mje_{cripto}\} = Mje\ M$
Obteniendo el mensaje original, Mje M.

PROCEDER INCORRECTO

J publica su lista falsa de claves públicas.
A desea enviar un mensaje a B.
A busca en la guía J la clave pública de B, la cual será $K_{B(Pub\ falsa)}$
Redacta el mensaje M = Mje M.

Lo criptografía con la clave pública falsa de B, ⟶ $K_{B(Pub\ falsa)}$ {Mje M} = Mje $_{cripto\ falso}$
Lo envía a B.
J lo escucha y lo decriptografía con la clave privada falsa de B,
⟶ $K_{B(Priv\ falsa)}$ {Mje $_{cripto\ falso}$} = Mje M
J criptografía el mensaje con la verdadera clave pública de B,
⟶ $K_{B(Pub)}$ {Mje M} = Mje $_{cripto}$
J lo envía a B.
B lo recibe.
B lo decriptografía con su clave privada $K_{B(Priv)}$ ⟶ $K_{B(Priv)}$ {Mje $_{cripto}$} = Mje M
Obteniendo el mensaje original, Mje M.

Nótese como un intruso obtuvo el mensaje original, el cual desde ya que si se realiza el proceder inverso, también se tomará conocimiento de la respuesta de B hacia A.

Este como se mencionó es el primer problema que plantea este algoritmo, y es por esta razón que es de suma importancia **garantizar la veraz distribución de claves públicas**, la cual se realizará a través de listas conocidas de distribución que puedan garantizar la consistencia de sus datos o lo que es más eficiente a través de la seguridad individual de cada emisor, el cual deberá estar plenamente convencido de la confiabilidad de las fuentes de obtención, lo cual puede ser a través del propio receptor telefónicamente, vía terceros que son de confianza, dependencias dentro de la organización que garanticen, etc.

El segundo gran problema que presenta este método es la demora que introduce (llegando a ser en algunos casos hasta mil veces más lentos que las técnicas simétricas), y el mayor volumen de información que genera. Ambos problemas son naturales en todo proceso que se desee optimizar la seguridad, como regla general:

SIEMPRE QUE SE INCREMENTA LA SEGURIDAD, SE INTRODUCEN DEMORAS.

Esto da origen al cifrado híbrido.

Cifrado Híbrido:

Esta técnica es la que se emplea en la mayoría de las aplicaciones de Software comercial, y se basa en el empleo de las dos algoritmos (Simétrico y asimétrico), para mejorar la velocidad y el volumen de datos, se procede a encriptar todo el mensaje con clave simétrica, luego se toma esta clave y se la criptografía con clave pública enviando todo el conjunto; el receptor entonces recibe un mensaje que en realidad está compuesto de dos partes, la primera de ellas es la clave simétrica criptografiada a través de una clave pública; sobre esta parte se aplica la clave privada obteniendo como resultado la clave simétrica original, una vez obtenida esta, se aplica sobre la segunda parte (mensaje criptografiado), obteniendo el texto original.

NOTA: Esta técnica en general para optimizar la seguridad se suele emplear generando en cada sesión una clave simétrica en forma aleatoria, es decir para cada mensaje se implementará una clave simétrica diferente. Esta mejora permite incrementar el algoritmo pues aún en el caso de lograr descifrar la clave simétrica, esta sola sería de utilidad para ese mensaje y nada más.

Cifrado Irreversible:

Este procedimiento, consiste en poder criptografiar código, pero si bien puede existir un método de desencriptado, este no se difunde o no se emplea. Su aplicación más común se puede observar en la forma en que los sistemas operativos de red suelen tratar las cuentas de usuario y contraseñas, como por ejemplo Unix o Windows NT. Estos sistemas operativos, al crear una cuenta de usuario de red, la guardan en archivos dentro de alguna estructura de directorios en forma criptografiada. Al hacerse presente este usuario en la red, solicita validarse ante un servidor, colocando su nombre de usuario y contraseña. El o los servidores que reciben esta petición, aplican el mismo algoritmo de encriptado y comparan los resultados, si son los mismos códigos, lo reconocen como usuario de red, caso contrario le niegan el acceso. Lo importante a tener en cuenta es que en ningún momento se compara texto plano, sino código criptografiado, por lo tanto no se necesita el procedimiento inverso para desencriptar.

Lamentablemente este tipo de cifrado tiene poca difusión, pues tiene la enorme potencia de no poder volver atrás. Si se tiene en cuenta este detalle, se podría implementar en discursos políticos, programas de TV, etc, en los cuales sería importantísimo que ni siquiera el que generó el discurso o diálogo pueda volver a repetirlo.

Métodos de autenticación y no repudio

Diffie_Hellman.

El problema del intercambio de claves es poder ponerse de acuerdo sobre un secreto compartido a través de un canal de comunicaciones no seguro (o público). El esquema propuesto por Diffie-Hellman es el siguiente:

a. El **Equipo A** elige dos valores de clave pública un **número primo m (grande)** y otro primo **g (más pequeño)** el cual es un generador de módulo m, es decir que el resto de g^a/m generará un número n tal que $0 \leq n < m$ para cualquier valor de a.
   A elige también una **clave secreta x** < m, luego calcula X = resto (g^x/m) y le envía al Equipo B los siguientes tres valores: m, g, X.

b. B recibe estos valores y elige una **clave secreta y** < m, calcula Y = resto (g^y/m), y le transmite el valor de Y a A.

   NOTA: tener en cuenta que en ningún momento se han transmitido las claves secretas **x** e **y.**

c. A calcula un nuevo valor S = resto (Y^x/m).
d. B calcula un nuevo valor S = resto (X^y/m).

   NOTA: Tener en cuenta que cada uno lo hace con su clave secreta y empleando un cálculo que lo relaciona con la clave secreta del otro (X e Y).

e. La característica (que se demuestra matemáticamente) que presenta S es que:

   S = resto(X^y/m) = resto(Y^x/m) = resto[$(g^x)^y/m$] = resto[$g^{(xy)}/m$] = resto[$(g^y)^x/m$] = resto[$g^{(yx)}/m$]

f. Por lo tanto el valor **de S es el secreto compartido** entre A y B.

g. En el caso que alguien deseara poder descubrir este secreto, contaría con la información que fue enviada por el canal de comunicaciones es decir: m. g, X e Y. Para participar de este secreto debería resolver una ecuación del tipo Z = resto (g^z/m) para un z desconocido. Esto es conocido como un problema de logaritmos discretos, lo cual en la actualidad es **computacionalmente imposible de resolver para algoritmos que empleen números primos suficientemente extensos**.

h. Los parámetros soportados son hasta 512 bit.

NOTA: Con 75 bit existen $5{,}2*10^{72}$ números primos.
Con 512 bit existen $3{,}1*10^{151}$ números primos.
Se estima que existen **$8{,}3*10^{77}$ electrones en el Universo.**

i. EJEMPLO: Se desarrolla a continuación un ejemplo empleando esta función a través de un valor muy conocido y didáctico de **g =3 y m =17**, por lo tanto la teoría tratada anteriormente se puede explicar a través de pequeños números primos de la siguiente forma:

f(x) = resto (3^x/17)

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
f(x)	3	9	10	13	5	15	11	16	14	8	7	4	12	2	6	1

Esta función con su primeros valores presenta la didáctica característica de permitir asociar todo el conjunto de 16 valores a 16 restos con propiedad biyectiva.

Como primer detalle es claro que dado un valor de x es muy fácil descubrir f(x), pero dado un valor de f(x), para poder realizar la operación inversa, se debe realizar el cálculo de las 16 posibilidades. Si se tiene en cuenta lo expresado en la teoría sobre una clave de 512 bit empleando números primos, se deberían realizar $3{,}1*10^{151}$ operaciones matemáticas, aunque existen algunas técnicas para reducir este espacio, en el mejor de los casos puede llegar a ser la mitad, con lo cual seguirá **siendo computacionalmente imposible**.

Volviendo al ejemplo:

a) A elige dos valores de clave pública **m = 17** y otro primo **g = 3**.
A elige también una **clave secreta x = 4** < m, luego calcula **X** = resto (3^4/17) = **13** y le envía a B los siguientes tres valores:

m = 17, g = 3, X = 13.

b) B recibe estos valores y elige una **clave secreta y = 7** < m, calcula **Y** = resto (3^7/17) = **11**, y le transmite el valor de **Y = 11** a A.

c) A calcula un nuevo valor **S**= resto (Y^x/m) = resto (11^4/17) = **4**.

14.641 | 17
..... 861
R = 4

d) B calcula un nuevo valor **S** = resto (X^y/m) = resto (13^7/17) = **4**.

62.748.517 | 17
..... 3.691.089

R = 4

e) Si existiera un tercero que escucha este diálogo, obtendría los valores que fueron transmitidos por el canal de comunicaciones, es decir: m = 17, g = 3, X = 13 e Y = 11. Si tratara de averiguar el secreto compartido, los cálculos que podría realizar son:

Elegir un número secreto z < m, por ejemplo z = 2.
Obtener Z = resto (g^z/m), por lo tanto Z = resto (3^2/17) = 9.
Con este valor intentaría resolver S. Sólo puede hacer dos operaciones:

Sz = resto (Y^z/m) = resto (11^2/17) = **2**.

121	**17**
- 119	**7**
R=2	

Sz = resto (X^z/m) = resto (13^2/17) = **16**.

169	**17**
- 153	**9**
R=16	

Como puede apreciarse, mediante estos cálculos no puede obtener el secreto compartido.

f) La característica que se demuestra matemáticamente que presenta S es que:

S = resto(X^y/m) = resto(Y^x/m) = resto[$(g^x)^y$/m] = resto[$g^{(xy)}$/m]
= resto[$(g^y)^x$/m] = resto[$g^{(yx)}$/m]

S = resto(13^7/17) = resto(11^4/17) = resto[$(3^4)^7$/17] = resto[$3^{(4*7)}$/17]
= resto[$(3^7)^4$/17] = resto[$3^{(7*4)}$/17] **= 4.**

g) Para cerrar este ejemplo, téngase en cuenta la siguiente hipótesis:
Para calcular x con los datos que circularon por el canal de comunicaciones, es decir g, m, X e Y; se debería resolver la siguiente ecuación:

X = resto (g^x/m), que en el ejemplo es 13 = resto (3^4/17)

81	**17**
13	**4**

g^x	**m**
X	**Result**

Por lo tanto: **Result * m + X = g^x** , es decir 4 * 17 + 13 = 81.
Para obtener x desde esta función se despejaría: $x = \log_g (Result * m + X)$

Si se tiene en cuenta que con 75 bit existen $5{,}2*10^{72}$ números primos, y se eligiera este tamaño de claves, teniendo en cuenta valores extremos como podrían ser:

- Procesadores de 100 GHz.
- Capacidad de poder realizar procesamiento en paralelo de manera tal que a través de n ordenadores, se pudiera realizar el cálculo de x para cada variable en un solo ciclo de reloj, es decir:

En un segundo se obtendrían 10^{11} valores de x.
Si basado en cálculo de probabilidades se tiene en cuenta que la probabilidad de acierto sería a la mitad del espacio de valores, entonces:

$$(5{,}2*10^{72})/\ 2 = 2{,}6*10^{72}$$

por lo tanto si: 1 seg → 10^{11} valores de x
para $2{,}6*10^{72}$ = **$2{,}6*10^{61}$ segundos**

$2{,}6*10^{61}$ segundos es el tiempo que demoraría calcular la cantidad de valores para acertar el valor que corresponda a **x secreto**.

Como última reflexión se puede calcular lo siguiente:

$2{,}6*10^{61}$ segundos
$= 4{,}333*10^{59}$ minutos
$= 7{,}222*10^{57}$ horas
$= 3{,}009*10^{56}$ días
$=$ **$8{,}244*10^{53}$ años**.

Quedando claro que es computacionalmente imposible de realizar.

RSA

Este criptosistema lleva el nombre de sus inventores R. Rivest, A. Shamir y L. Adleman. Permite ser empleado para compartir un secreto y para firma digital. Su seguridad está basada en el problema de factorización de enteros.

Cada entidad crea un par de claves (Pública y Privada) de la siguiente forma:

a. Genera dos números primos largos y distintos **p** y **q**.
b. Se calcula **n** = p*q, y **ø** = (p –1) * (q – 1).
c. Elige un entero (random) **e**, tal que 1<e< ø, y que el máximo común divisor (mcd) de e y ø sea 1, es decir: mcd (e, ø) = 1.
d. Se calcula a través del algoritmo Euclídeo extendido un valor entero **d**, 1<d< ø, y tal que el Resto (e*d / ø) = 1.
e. Se obtiene de esta forma la **clave pública** (n,e) y la **clave privada** (d).

En la terminología RSA, los enteros **e** y **d** son llamados exponentes de encriptado y desencriptado, y **n** es llamado módulo.

El mecanismo completo funcionaría de la siguiente manera:

Si A le enviara un mensaje criptografiado a B:

a. Debería obtener la cave pública de B, es decir (**n_b, e_b**).
b. Representar el mensaje **m** como un entero en el intervalo {0, n-1}.
c. Procesar **c = Resto (m^{eb} /n_b).**

d. Enviar el mensaje cifrado **c** a B.
e. Al llegar a B, este debería usar su clave privada $\mathbf{d_b}$, a través de la siguiente fórmula: **m = Resto (c^d / n)**

EJEMPLO:

Generación de claves por parte del equipo A:

a. Se propone dos números primos pequeños: **p** = 5 y **q** = 11 y se calcula **n = p*q** = 55. De estos valores se puede calcular también **ø = (p – 1) * (q – 1)** = 40.
b. A también genera el valor **e = 3**, tal que **1<e< ø** y que el **mcd (e, ø) = 1**. Como puede verificarse 1<3<40 y mcd (3,24) = 1.
c. Se emplea el algoritmo Euclídeo extendido para calcular **d** = 27, teniendo en cuenta que **1<d< ø**, (1<27<49); y que el **Resto (e*d / ø) = 1**; Resto (3*27 / 40) = 1.
d. Con estos parámetros entonces la clave pública de A es (**n** = 55 y **e** = 3) y la clave privada de A es **d** = 27.

Si un equipo B deseara enviarle un mensaje al equipo A, debería primero obtener la clave pública de A, es decir (**n** = 55 y **e** = 3), para luego poder criptografiar el mensaje de la siguiente forma:

a. Se toma como ejemplo un valor m de mensaje que se representa por un número entero en el intervalo {0, n-1}, **m** = 12.
b. Calcula **c** = Resto (m^e/n) = Resto (12^3 / 55) = 23.
c. Envía el valor 23 al equipo A.

El equipo A para desencriptarlo emplearía su clave privada (**d**) procesándolo de la siguiente forma:
m = Resto (c^d / n) = 23^{27} / 55) = 12.

Métodos de verificación de Integridad (HMAC – SHA y MD5)

HMAC (Hashing for Message Authentication Codes) [RFC-2104]

Para proveer un modo de chequear integridad de la información transmitida o almacenada en un medio no confiable es necesario un mecanismo que permita compararla contra algo que se considere válido. Para esto se estandarizó un procedimiento basado en una clave secreta usualmente llamado **Código de Autenticación de Mensajes** (**MAC**). El empleo típico de estos mecanismos es a través de dos partes que comparten una clave secreta para validar la información transmitida entre ellas. HMAC propone el empleo de criptografía aplicada a funciones Hash (resúmenes). En la RFC, estandariza el empleo de HMAC con las funciones Hash definidas como **MD5** (Message Digest versión 5) [RFC-1321] y **SHA-1** (Standard Hash Algorithm Versión 1) [FIPS 180-1]. También hace mención al algoritmo propuesto por RIPE denominado RIPEMD-128/160. Hoy ya ha salido la nueva versión SHA-256, de 256 bits.

HMAC requiere una función Hash (H) que se encargará de comprimir un texto de longitud finita por medio de iteraciones de una función de compresión básica sobre los bloques de datos (B = 64 Byte), y una clave secreta (K); y por medio de ambas se obtendrá un resumen de longitud fija (L), que será de 16 Byte para MD5 y 20 Byte para SHA-1.

Esta función Hash es llamada "**One Way**" pues no es posible a través del resumen de salida obtener el texto de entrada, también resultará computacionalmente imposible obtener un valor de salida igual a través de otro valor de entrada, como así tampoco desde un valor de salida ya calculado, obtener otro valor de entrada diferente al verdadero.

Se definen dos cadenas de longitud fija diferentes una de la otra llamadas:

Ipad = repetición del Byte 36 B veces.
Opad = repetición del Byte 5C B veces.

Luego se ejecuta: H { K_B xor Opad, H (K_B xor Ipad, texto)}

Para esta tarea se debe tener en cuenta:

a. Rellenar con ceros la clave K hasta obtener una longitud de 64 Byte (llamada K_B).
b. Realizar: K_B *xor* Ipad , (Result1).
c. Anexar el texto completo al resultado de b, (Result2).
d. Aplicar la función Hash (H) al resultado de c, (Result3).
e. Realizar: K_B *xor* Opad, (Result4).
f. Anexar el resultado de d, (Result3). Al resultado de e, (Result4).
g. Aplicar la función Hash (H) al resultado de f. Obteniendo el resultado, que acorde a la función Hash empleada será de 16 o 20 Byte.

The MD5 Message-Digest Algorithm

Request for Comments: 1321

Este algoritmo toma un mensaje de entrada de longitud arbitraria y entrega una salida de 128 bit de longitud fija. Llamado "Huella digital" o "Recopilación de mensaje (Message Digest). Es computacionalmente imposible producir dos mensajes que posean la misma recopilación, como tampoco regenerar el mensaje a través de la recopilación. Este algoritmo puede ser empleado para aplicaciones de firma digital, donde un texto debe ser comprimido de manera segura antes de ser encriptado con sistemas de clave privada.

El algoritmo MD5:

Se implementa por medio de 5 pasos:

Paso 1: Anexa bit de relleno.

Se anexan bit de relleno para que el mensaje dividido 512 tenga resto 448, es decir 448 (mod 512). El primer bit de relleno será un uno y luego se continuará con una cadena de ceros.

Paso 2: Anexa longitud.

Una representación de 64 bit del mensaje original (antes del paso 1) es anexada al resultado. En este paso el mensaje resultante es un múltiplo exacto de 512 bits, es decir que es múltiplo exacto de 16 palabras de longitud 32 bit.

Paso 3: Inicialización del buffer MD.

Un buffer de cuatro palabras (A, B, C, D) es empleado para procesar el mensaje generado luego del paso 2. Cada uno de estos buffer es un registro de 32 bit. Esos registros son inicializados con los siguientes valores en hexadecimal:

```
A: 01 23 45 67
B: 89 ab cd ef
C: fe dc ba 98
D: 76 54 32 10
```

Paso 4: Procesar el mensaje en bloques de 16 bit.

Se definen cuatro funciones auxiliares donde cada una tiene una entrada de tres palabras de 32 bit y produce una palabra de salida de 32 bit. Las funciones son:

```
F(X,Y,Z) = X.Y OR not(X).Z
G(X,Y,Z) = X.Z OR Y.not(Z)
F(X,Y,Z) = X XOR Y XOR Z
I(X,Y,Z) = Y XOR (X OR not(Z))
```

Como ejemplo de la función F, su tabla es:

X	**Y**	**Z**	**F**
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Se procesa cada bloque de 32 bit bajo una serie de operaciones tabuladas en esta RFC [pág 5] y queda como resultado un valor de 32 bit en cada uno de los buffer A, B, C y D.
Se repiten los pasos a través de asignaciones de variables temporales (AA, BB, CC, DD) hasta el último bloque de texto, repitiendo sucesivamente la asignación que se detalla a continuación.

```
A = A + AA
B = B + BB
C = C + CC
D = D + DD
```

Al finalizar todos los ciclos (recordar que el texto de entrada es múltiplo exacto de 16 palabras de longitud 32 bit) quedan en los buffer A, B, C y D el resultado final del proceso.

Paso 5: Salida.

El mensaje generado es una salida de los Buffer A, B, C y D, los cuales se anexan desde el bit de menor orden de A hasta el último de D, por lo tanto esta salida es un mensaje de 128 bit de longitud.

PGP (Pretty Good Privacy).

Estos temas se desarrollan en forma práctica con el Software de PGP.

1. Clave pública y privada.
2. Verificación y validación de claves.
3. Grado de confianza.
4. Administración de claves.
5. Importación y exportación de claves.
6. Criptografía y firma electrónica.
7. PGP Net.
8. Asociaciones de seguridad.
9. Archivos autodesencriptables.
10. Borrado de archivos.

Sistema de autenticación Kerberos:

Este sistema nace en el Instituto Tecnológico de Massachusetts y su nombre se remonta a la mitología Griega donde así se denominaba el perro guardián de los Dioses. Este sistema de autenticación hoy es soportado por la masa de los sistemas operativos y componentes de red.
El sistema Kerberos está basado en un "Servidor despachador de boletos", al cual se encarga de validar la identidad de los "Principales" los cuales pueden ser:

- Usuarios.
- Servicios.

En cualquiera de los dos casos, un "Principal" queda definido por un "Trío" cuyos componentes son:

- Nombre primario: Nombre de persona o servicio.
- Instancia: Para usuarios es nula o contiene información de ésta. Para un servicio es el nombre de la máquina.
- Reino: Define distintos Dominios de autenticación.

Si un “Principal” obtiene un boleto, este tendrá un tiempo de vida limitado por el Servidor, y a partir de este poseerá una clave privada que sólo conocerán el Principal y el Servidor, por lo tanto será considerada auténtica y a través de esta podrá acceder a los recursos del sistema.