Criptografía: Seguridad en la información

Guillermo Morales-Luna
Sección de Computación
CINVESTAV-IPN
gmorales@cs.cinvestav.mx

México, D.F., a 30 de octubre de 2000.

Resumen

En la actualidad, la gran influencia que las telecomunicaciones tienen en todos los aspectos de la cotidianidad ha hecho que las comunicaciones seguras cobren también particular importancia.

Hay diferentes enfoques para tratar el problema de seguridad: Tanto de aislamiento de los sistemas y de entrenamiento a usuarios y administradores de los sistemas de cómputo y de comunicaciones, como de uso de técnicas criptográficas.

En esta conferencia mostramos los fundamentos matemáticos de los principales procedimientos de criptografía.

Nos referiremos por Criptografía a la disciplina referente a la construcción de sistemas de encriptamiento; por Criptoanálisis a la disciplina referente al rompimiento de sistemas de encriptamiento, y por Criptología la conjunción de la criptografía y el criptoanálisis.

A grandes rasgos, la criptografía puede dividirse en dos grandes tipos: La de llave secreta y la de llave pública. Presentaremos como representante t
i pico de llave secreta al sistema DES, asumido por el gobierno de los EUA como un estándar desde hace ya tres décadas. Como representante de llave pública presentamos el método de RSA.

La conferencia es de divulgación con la intención primordial de atraer la atención de los escuchas hacia la Criptografía.

Palabras claves: Criptografía, llave secreta, llave pública.

Contenido

Tema 1

Introducción a criptografía y bases matemáticas

1.1 Seguridad en la información

La seguridad de la información es vulnerable y susceptible de ataques varios: Errores humanos, empleados incumplidos, empleados desleales, intromisiones externas y factores imponderables tales como incendios o desastres naturales. Para reforzarla se puede asumir diversos enfoques:

- *Personal*: Accesos mediante tarjetas de control o reconocimiento de rasgos físicos
- *Física*: Infraestructura de control integral de acceso
- *Administrativo*: Educación del personal en medidas de seguridad
- *Redes de datos*: Mediante encriptamiento y configuraciones de control
- *Software y sistemas operativos*: Sistemas confiables
- *Hardware*: Resistencia a intromisiones, encriptamiento

En todo sistema de seguridad hay que considerar las *comunicaciones* y los *sistemas de cómputo*.

Un enfoque muy importante es el *encriptamiento* (*cifrado*, *codificación*) de la información.

1.2 Introducción a la criptografía

La **criptografía** es el estudio de técnicas matemáticas relativas a la seguridad de la información. Su objetivo consiste en ofrecer los servicios siguientes:

- *Confidencialidad*: Mantener la información secreta respecto a quienes no están autorizados a conocerla
- *Control de acceso*: Restricciones de acceso a ciertos recursos de información permitiendo sólo usuarios distinguidos
- *Privacidad*: Evitar intromisiones indeseadas a la información propia
- *Integridad*: Mantener la exactitud de la información, es decir, que no sufra alteraciones por usuarios o medios no-autorizados

- *Autentificación de identidades*: Reconocimiento de remitentes, corroboración de sus identidades
- *Autentificación de mensajes*: Reconocimiento de orígenes de la información,
- *Anonimato*: Reserva de la identidad de una entidad involucrada en un proceso,
- *Firmado*: Ligamento de partículas de información con entidades
- *Sostenimiento de compromisos (Non-repudation)*: Protección contra actos de desdecirse

- *Autorización*: Delegación de privilegios a otra entidad para realizar actos propios
- *Aval*: Otorgamiento temporal de privilegios a otra entidad para realizar actos propios
- *Certificación*: Respaldo de información por una entidad confiable
- *Revocación*: Suspensión de certificados o de autorizaciones
- *Propiedad*: Maneras de otorgar permisos a una entidad para uso o transferencia de recursos
- *Testimonio*: Verificación de creación o de vigencia de partículas de información por otras partes que los creadores
- *Confirmación*: Reconocimiento de que ciertos servicios han sido provistos
- *Recibo*: Acuses de que partículas de información han sido recibidas

- *Disponibilidad*: Asegurar la entrega oportuna de información
- *Fechado o marca de tiempo*: Ligamento de partículas de información con momentos de creación o de vigencia

Criptografía (*kryptos*, griego: Oculto) Disciplina referente a la construcción de sistemas de encriptamiento.

Criptoanálisis Disciplina referente al *rompimiento* de sistemas de encriptamiento.

Criptología Conjunción de la criptografía y el criptoanálisis.

1.3 Algo que no es criptografía

En la **esteganografía** (*steganos*, griego: Cubierto), los mensajes no se transforman mas se ocultan en otro discurso, o bien, se presenta el mensaje con sinónimos:

Lingüística: El *caló* (*jerga de arrabal* y *albureo*). El mensaje simplemente se plantea en vocablos sinónimos o deformados: *Ay nos vidrios en tu cantón, y te caes con la luz.* Otros ejemplos los dan los códigos utilizados en comunicaciones por radio (taxistas, policías, radioaficionados).

Escrita: Los *acrósticos* o la referencia a símbolos dentro de un texto:

> "Un nuevo régimen fiscal para Pemex – Fox anunció en Chile –. El fisco ha de tener ingresos alternativos para obtener si no todas las utilidades actuales, sí lo que compense lo no logrado al hacer más eficiente el gasto público."

si se refiere a los símbolos subrayados

> "Un nuevo régimen **f**iscal pa**ra** Pemex – Fox an**u**nció en Chile – . El fisco ha **de** t**en**er ingresos al**t**ernativos par**a** o**b**tener si no todas l**as** utilidades actuales, sí lo que **co**mpense lo no logrado al hacer más eficiente el gasto público."

se lee **Fraude en Tabasco**

En 1508, el abad Johannes Trithemius, inició la publicación de su *Polygraphiæ*, obra en seis volúmenes que presentaba una colección de palabras en latín que codificaba a las letras del alfabeto, lo que se conoció posteriormente como el código *Ave Maria*.

A	Deus
B	Creator
C	Conditor
D	Opisex
E	Dominus
F	Dominator
G	Consolator
H	Arbiter

A	clemens
B	clementissimus
C	pius
D	pijssimus
E	magnus
F	excelsus
G	maximus
H	optimus

Así, por ejemplo *¡acecha!* podría ponerse

Deus pius Dominus magnus Conditor optimus clemens

1.4 Características criptográficas

Un *buen* sistema criptográfico ha de tener varias características:

- pequeñas variaciones de textos llanos: grandes variaciones de textos cifrados,
- los tamaños de los textos planos deben ser comparables con los cifrados,
- los textos cifrados deben calcularse eficientemente a partir de los planos, y
- la relación entre textos planos y cifrados debe ser impredecible.

Un *mal* sistema criptográfico se caracteriza porque:

- aparenta una relación aleatoria entre planos y cifrados, pero en realidad no lo es,
- es susceptible a criptoanálisis elementales,
- el cálculo de cifrados es ineficiente en tiempo y en espacio, y
- es vulnerable a sus propios fabricantes.

Un sistema criptográfico típico de *llave secreta* consta de los objetos siguientes

- $\mathcal{M}$: un *espacio de mensajes*,
- $\mathcal{L}$: un *espacio de llaves*,
- $\mathcal{C}$: un *espacio de cifrados de textos*,
- $E : \mathcal{L} \times \mathcal{M} \to \mathcal{C}$: función de *encriptamiento*,
- $D : \mathcal{L} \times \mathcal{C} \to \mathcal{M}$: función de *desencriptamiento*, tales que
  - $\forall k \in \mathcal{L}, m \in \mathcal{M} : \ D(k, E(k, m)) = m$ &
  - $\forall k \in \mathcal{L}, c \in \mathcal{C} : \ E(k, D(k, c)) = c.$

El sistema es *seguro* si a pesar de conocer $E(k, m)$, desconociendo k no se puede calcular m.

Ejemplo. *Rellenado de una sola vez (one-time pad)*

En $\mathbb{Z}_2$ la operación suma $\oplus$, u *"O"-excluyente*, es de orden 2: $x \oplus x = 0$, y $\forall n \in \mathbb{N}$, la misma operación componente a componente cumple lo mismo: $\forall \mathbf{x} \in \mathbb{Z}_2^n : \ \mathbf{x} \oplus \mathbf{x} = \mathbf{0}$. Así pues, se puede tomar $\mathcal{M} = \mathcal{L} = \mathcal{C} = \mathbb{Z}_2^n$ y

$$\begin{array}{rcl} E : \mathcal{L} \times \mathcal{M} & \to & \mathcal{C} \\ (\mathbf{k}, \mathbf{m}) & \mapsto & \mathbf{c} = \mathbf{k} \oplus \mathbf{m} \end{array} \qquad \begin{array}{rcl} D : \mathcal{L} \times \mathcal{C} & \to & \mathcal{M} \\ (\mathbf{k}, \mathbf{c}) & \mapsto & \mathbf{m} = \mathbf{k} \oplus \mathbf{c} \end{array}$$

el sistema es seguro pues conocido $\mathbf{c}$ si no se conoce $\mathbf{k}$ no es recuperable $\mathbf{m}$ (de hecho, es posible generar cualquier $\mathbf{m}$ pues siempre existe $\mathbf{k} = \mathbf{m} \oplus \mathbf{c}$ tal que $\mathbf{m} = D(\mathbf{k}, \mathbf{c})$.

Ejemplo. *Transformaciones lineales no-singulares*

Sea $\mathbf{A} \in (\mathbb{Z}_2)^{n\times n}$ una matriz tal que Det $\mathbf{A} \neq 0$ y sea $\mathbf{b} \in \mathbb{Z}_2^n$. Entonces la pareja $(\mathbf{A}, \mathbf{b})$, que contiene $n^2 + n$ "entradas" en $\mathbb{Z}_2$, determina una transformación afín

$$T_{(\mathbf{A},\mathbf{b})} : \mathbf{m} \mapsto \mathbf{A}\mathbf{m} + \mathbf{b}$$

con inversa $T^{-1}_{(\mathbf{A},\mathbf{b})} = T_{(\mathbf{A}^{-1},\mathbf{0})} \circ t_{-\mathbf{b}}$, donde $t_{-\mathbf{b}}$ es la translación $\mathbf{x} \mapsto \mathbf{x} - \mathbf{b}$.

Ejemplo. *Permutaciones-sustituciones de Shannon*

Sustituciones. Para $n \in \mathbb{N}$, sea $\mathbb{Z}_2^n$ el espacio de cadenas binarias de longitud n. Sea $\mathbf{k} : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ una permutación. A cada mensaje $\mathbf{m} = [m_i]_{i\leq l} \in (\mathbb{Z}_2^n)^*$ se le asocia el código $\mathbf{c} = [\mathbf{k}(m_i)]_{i\leq l} \in (\mathbb{Z}_2^n)^*$. En este caso se tiene un espacio de llaves de tamaño $(2^n)!$.

Permutaciones. Sea $\mathbf{k} : [\![1, n]\!] \to [\![1, n]\!]$ una permutación. A cada mensaje, dividido en bloques, $\mathbf{m} = \left[m_i = [a_{ij}]_{j\leq n}\right]_{i\leq l} \in (\mathbb{Z}_2^n)^*$ se le asocia el código $\mathbf{c} = \left[\left[a_{i\mathbf{k}(j)}\right]_{j\leq n}\right]_{i\leq l} \in (\mathbb{Z}_2^n)^*$. En este caso se tiene un espacio de llaves de tamaño $n!$.

1.5 Encriptamiento por bloques

Dado un mensaje $\mathbf{m} = (a_i)_{i\leq m}$, se parte en *bloques* $\mathbf{m}_i = \left(a_{(i-1)k+j}\right)_{j\leq k}$, donde $k|n$. Se encripta cada bloque para obtener el cifrado $\mathbf{c}_i = E_{Llave}(\mathbf{m}_i)$ y el cifrado de todo el mensaje es: $\mathbf{c} = (\mathbf{c}_i)_{i\leq \frac{m}{k}}$. Hay esencialmente dos tipos de encriptamiento de bloques:

de ráfaga: En cada bloque i se usa una llave distinta y los cifrados anteriores: $\mathbf{c}_i = E_{i,Llave_i}(\mathbf{m}_i, \mathbf{c}_{i-1})$. Permite bloques pequeños. Hay varios *modos de operación*:

1. En el *modo de encadenamiento de bloques cifrados (cipher block chaining mode (CBC))* se toma como E_{Llave} una permutación (la llave se mantiene constante) y

$$\mathbf{c}_i = E_{Llave}(\mathbf{m}_i \oplus \mathbf{c}_{i-1})$$

donde $\mathbf{c}_0$ es un texto *inicial.* El mensaje se recupera haciendo

$$\mathbf{m}_i = D_{Llave}(\mathbf{c}_i \oplus \mathbf{c}_{i-1}) \qquad \text{con} \qquad D_{Llave} = E^{-1}_{Llave}.$$

2. En el *modo de retroalimentación con salida (output feedback mode (OFB))* se toma como E_{Llave} una permutación (la llave se mantiene constante) y se hace

$$\mathbf{v}_i = E_{Llave}(\mathbf{v}_{i-1}) \;\;;\;\; \mathbf{c}_i = E_{Llave}(\mathbf{m}_i \oplus \mathbf{v}_i)$$

donde $\mathbf{v}_0$ es una *semilla inicial.* El mensaje se recupera haciendo

$$\mathbf{m}_i = D_{Llave}(\mathbf{c}_i \oplus \mathbf{v}_i).$$

3. En el *modo de retroalimentación con cifrado (cipher feedback mode (OFB))*,

$$\mathbf{c}_i = \mathbf{m}_i \oplus E_{Llave}(\mathbf{c}_{i-1})$$

donde $\mathbf{c}_0$ es un texto *inicial*. El mensaje se recupera haciendo

$$\mathbf{m}_i = \mathbf{c}_i \oplus E_{Llave}(\mathbf{c}_{i-1}).$$

uniforme: En cada bloque i se utiliza una misma función y una misma llave: $\mathbf{c}_i = E_{Llave}(\mathbf{m}_i)$. Aquí se tiene el *modo de código de libro electrónico (electronic code book mode (ECB))*. La característica principal es que:

$$\forall i, j : \quad \mathbf{c}_i = \mathbf{c}_j \;\Leftrightarrow\; \mathbf{m}_i = \mathbf{m}_j.$$

Tema 2

Data Encryption Standard (DES)

2.1 Presentación del método

El *estándar de encriptamiento de datos*, *Data Encryption Standard* (DES), ha sido de los más utilizados en la historia de la criptografía:

- Se desarrolló en los años 70, por IBM, principalmente.
- En 1977 se adoptó como un estándar por la Oficina Nacional de Estándares (NBS: National Bureau of Standards, en la actualidad el National Institute of Standards and Technology) del Gobierno de los EUA.
- Es de encriptamiento por bloques, con bloques de 64 bits.
- Utiliza llaves de 56 bits (dadas en 8 bytes, en cada uno de los cuales 7 bits son de la llave y el octavo es de *paridad*).
- Consecuentemente, el espacio de búsqueda para la llave es de tamaño 2^{56} (en promedio se requiere de 2^{55} pasos de búsqueda).
- El encriptamiento se hace con 16 ciclos de reiteración. En cada ciclo, los paraámetros de la función de encriptamiento dependen de los bloques de datos y de llaves, actuales y previos.
- En cada ciclo, la (sub-)llave $Llave_i$, $i = 1, \ldots, 16$ se construye mediante un algoritmo de preparación de llaves (*key scheduling algorithm*).

Dado un mensaje $\mathbf{m} = \mathbf{m}_{Izq}\mathbf{m}_{Der}$ de 64 bits, es decir, 8 bytes, $\mathbf{m}_{Izq}, \mathbf{m}_{Der}$ de 4 bytes cada uno, se procede como sigue:

1. Sea $\mathbf{m}^{(1)}_{Izq}\mathbf{m}^{(1)}_{Der} = \mathbf{m}^{(1)} = T(\mathbf{m})$, donde T es una permutación.

2. Para $i = 2, \ldots, 16$ hágase

$$\begin{aligned} \mathbf{m}_{Izq}^{(i)} &= \mathbf{m}_{Der}^{(i-1)} \\ \mathbf{m}_{Der}^{(i)} &= \mathbf{m}_{Izq}^{(i-1)} \oplus f\left(\mathbf{m}_{Der}^{(i-1)}, Llave_i\right) \end{aligned}$$

   donde $Llave_i$ tiene una longitud de 48 bits.

3. $\mathbf{c} = T^{-1}(\mathbf{m}_{Izq}^{(16)}\mathbf{m}_{Der}^{(16)})$.

El cálculo de $f : \mathbb{Z}_2^{32} \times \mathbb{Z}_2^{48} \to \mathbb{Z}_2^{32}$, $(\mathbf{m}_{Der}, Llave_i) \mapsto f(\mathbf{m}_{Der}, Llave_i)$ utiliza 8 transformaciones $S_j : \mathbb{Z}_2^6 \to \mathbb{Z}_2^4$, $j = 1, \ldots, 8$, llamadas *S-cajas*, **fijadas de antemano por los estándares**. f se calcula como sigue:

1. Duplicando bits en 16 posiciones, la cadena $\mathbf{m}_{Der}$ de 32 bits se expande a una cadena $\mathbf{m}'_{Der}$ de 48 bits.
2. Sea $\mathbf{r} = \mathbf{m}'_{Der} \oplus Llave_i \in \mathbb{Z}_2^{48}$.
3. Escríbase $\mathbf{r} = [\mathbf{r}_j]_{j=1}^8$, donde cada $\mathbf{r}_j \in \mathbb{Z}_2^6$ es una cadena de 6 bits.
4. Hágase $f\left(\mathbf{m}_{Der}^{(i-1)}, Llave_i\right) = P\left[S_j(\mathbf{r}_j)\right]_{j=1}^4 P\left[S_j(\mathbf{r}_j)\right]_{j=5}^8$, donde P es una permutación $\mathbb{Z}_2^{32} \to \mathbb{Z}_2^{32}$.

En cuanto a la generación de sub-llaves, dada la llave del usuario de 56 bits, primero se transpone a sus bits, y se fracciona en dos partes de 28 bits cada una. Se va rotando a estas partes uno o dos bits a la izquierda, dependiendo del índice del ciclo, y de acuerdo con una regla fija, se compone a las partes para producir la llave $Llave_i$ de 48 bits:

1. Sea $Llave = [k_i]_{i=1}^{64}$ una llave dada por 64 bits. Los bits k_{8i}, $i = 1, \ldots, 8$ son de paridad. Se les descarta y los restantes forman la llave de 56 bits propia del usuario.
2. Sea $L = \mathbf{P}_1(Llave)$ la cadena de 56 bits formada mediante la aplicación de una permutación $\mathbf{P}_1$, a $Llave$. Escríbasela como un par de cadenas de 28 bits, $[C_0, D_0] = L$.
3. Para cada ciclo $i \leq 16$, hágase $v_i = \begin{cases} 1 & \text{si } i \in \{1,2,9,16\}, \\ 2 & \text{en otro caso.} \end{cases}$

   En otras palabras, de acuerdo con el índice de cada ciclo se ha de rotar como se indica a continuación:

Ciclo i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Rotaciones v_i	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

   Sea $\gamma : \mathbb{Z}_2^{28} \to \mathbb{Z}_2^{28}$ la rotación consistente de "correr un lugar hacia la izquierda".

4. Para cada $i = 2, \ldots, 16$ hágase $C_i = \gamma^{v_i}(C_{i-1})$, $D_i = \gamma^{v_i}(D_{i-1})$ y sea $Llave_i = \mathbf{P}_2([C_i, D_i])$ la llave de 48 bits formada mediante una permutación $\mathbf{P}_2$.

Para desencriptar se sigue el mismo procedimiento, pero en sentido inverso de aplicación.

Observaciones:

- Las S-cajas definen transformaciones que no son lineales ni afines. En tanto sean más complejas, la seguridad será mayor.
- Una llave es débil si $E_{Llave}(E_{Llave}(\mathbf{m})) = \mathbf{m}$.
- **Propiedad de complemento.** Si $\mathbf{c} = E_{Llave}(\mathbf{m})$ entonces $\overline{\mathbf{c}} = E_{\overline{Llave}}(\overline{\mathbf{m}})$.
- En la actualidad, la llave resulta pequeña.
- La implementación en hardware de las S-cajas es propiedad de IBM y no se ha publicado su diseño, el cual puede ser un elemento de vulnerabilidad.

Un posible ataque:

1. Escójase $\mathbf{m}$.
2. Obténgase los cifrados $\mathbf{c}_0$ de $\mathbf{m}$ y $\mathbf{c}_1$ de $\overline{\mathbf{m}}$, así $\mathbf{c}_0 = E_{Llave}(\mathbf{m})$ y $\mathbf{c}_1 = E_{Llave}(\overline{\mathbf{m}})$.
3. Para las 2^{55} llaves $Llave'$ cuyo bit más significativo es 0, revísese si $E_{Llave'}(\mathbf{m}) \in \{\mathbf{c}_0, \mathbf{c}_1\}$. En tal caso, se tendrá evidencia de que $Llave = Llave'$ si $E_{Llave'}(\mathbf{m}) = \mathbf{c}_0$ o bien de que $Llave = \overline{Llave'}$ si $E_{Llave'}(\mathbf{m}) = \mathbf{c}_1$.

2.2 Triple DES

Como un reforzamiento de DES, *Triple DES* aplica tres veces el método convencional DES para obtener un cifrado final de cada mensaje. Sea $DES : (Llave, \mathbf{m}) \mapsto \mathbf{c}$ la función de encriptamiento descrita anteriormente. Escribamos

$$DES_{Llave} : \mathbf{m} \mapsto DES(Llave, \mathbf{m}).$$

Llaves de 168 bits Dadas tres llaves $Llave_1, Llave_2, Llave_3$ de 56 bits cada una (juntas por tanto dan 168 bits) se hace, para cada mensaje $\mathbf{m}$:

$$\mathbf{c} = DES_{Llave_3} \circ DES^{-1}_{Llave_2} \circ DES_{Llave_1}(\mathbf{m})$$

Llaves de 112 bits Dadas dos llaves $Llave_1, Llave_2$ de 56 bits cada una (juntas por tanto dan 112 bits) se hace, para cada mensaje $\mathbf{m}$:

$$\mathbf{c} = DES_{Llave_1} \circ DES^{-1}_{Llave_2} \circ DES_{Llave_1}(\mathbf{m})$$

En cualquier caso, se puede utilizar cualesquiera de los modos ECB, CBC, OFB y CFB.

2.3 Otros esquemas de encriptamiento por bloques

IDEA (*International Data Encryption Algorithm*) Bloques de 64 bits se encriptan con llaves de 128 bits. Cada bloque se divide en 4 sub-bloques de 16 bits cada uno.

Lucifer

RC5 de RSADSI.

Safer K-64 y **K-128**

... y muchos más.

Tema 3

Métodos de llave pública

El concepto de *llave pública* apareció en el 76 en un artículo debido a Diffie-Hellman [3].

En este esquema cada usuario tiene un "buzón, con una ranura de entrada: cualquier otro usuario sabe cómo meter un mensaje en ese buzón, mas sólo el dueño del buzón puede extraer mensajes". El término inglés *trapdoor one-way* denota precisamente esta idea: Un ratón puede entrar con facilidad a una trampa mas de ella no puede salir. Utilizaremos, con todas las reservas del caso, el término *unidireccional* para referirnos a *one-way*.

En todo esquema de llave pública cualquier usuario U posee dos llaves (e_U, d_U). Cuando un usuario V ha de enviar un mensaje m a U, calcula un código $c = f(m, e_U)$ utilizando la llave pública de U. U desencripta el mensaje calculando $m = g(c, d_U)$ utilizando su propia llave secreta d_U. Así pues la llave secreta de U está en función de su llave pública e_U. Naturalmente, d_U no ha de ser calculable sólo a partir de e_U. Por otro lado, las funciones f y g, que guardan cierta relación de inversibilidad, una respecto a la otra, han de ser funciones de iguales complejidades.

En la actualidad, RSA es el prototípico esquema de llave pública. Veremos en esta lección algunos otros esquemas de llave pública.

3.1 El-Gamal

3.1.1 En anillos de residuos

Recordamos que si p es un número primo, entonces $\mathbb{Z}_p$ es un campo, y su grupo multiplicativo $\mathbb{Z}_p^* = \mathbb{Z}_p - \{0\}$ es un grupo cíclico. Cada generador de $\mathbb{Z}_p^*$ es un *elemento primitivo* en $\mathbb{Z}_p^*$. Si x es un elemento primitivo, entonces

$$\forall y \in \mathbb{Z}_p^* \exists! z \in [\![0, p-2]\!] : \ x^z = y \bmod p$$

tal z es el *logaritmo discreto* de y en base x, en $\mathbb{Z}_p^*$. Escribimos, $z = \log_x(y)$.

En la tabla 3.1 vemos que 14 es primitivo en $\mathbb{Z}_{17}^*$ así como todas sus potencias impares (esto porque $17 - 1 = 16$ es una potencia de 2), a saber: 14, 7, 12, 6, 3, 10, 5, 11.

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14^k	1	14	9	7	13	12	15	6	16	3	8	10	4	5	2	11

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\log_{14} x$	0	14	9	12	13	7	3	10	2	11	15	5	4	1	6	8

Tabla 3.1: Potencias de 14 módulo 17 y logaritmos discretos en base 14 en $\mathbb{Z}_{17}^*$.

En esa misma tabla, observamos que la inversa de la función "potencia de 14", a saber el logaritmo discreto en base 14, tiene un comportamiento que parece aleatorio.

La función exponenciación es relativamente sencilla: Si $k = \sum_i a_i 2^i$ entonces

$$x^k = \prod_i (x^{a_i})^{2^i} = \prod_{\{i|a_i=1\}} x^{2^i}.$$

Ahora bien, si se define la sucesión $(x_i)_i$ como

$$x_0 = x \qquad , \qquad x_i = x_{i-1}x_{i-1}$$

entonces $x^{2^i} = x_i$.

Sin embargo, el cálculo de logaritmos discretos es un problema completo-NP. En este sentido es que la exponenciación es una función unidireccional.

El *Método de ElGamal* se resume como sigue:

Espacio de llaves. $\mathcal{L} = \{(p,x,z,y)|x^z = y \bmod p\}$. En cada cuarteta $(p,x,z,y) \in \mathcal{L}$ la terceta $e = (p,x,y)$ es la *llave pública* y $d = z$ es la llave secreta.

Procedimiento de encriptamiento. Es no-determinista. Dado $m \in \mathbb{Z}_p^*$, elíjase aleatoriamente $k \in \mathbb{Z}_p^*$ y hágase

$$c = f(m,k) = \left(x^k \bmod p, m \cdot y^k \bmod p\right).$$

Procedimiento de desencriptamiento. Dado $c = (v_1, v_2)$, y conocida la llave secreta $d = z$ hágase

$$g(v_1,v_2) = v_2 \left(v_1^z\right)^{-1} \bmod p.$$

El procedimiento es correcto pues si se tuviera

$$c = (v_1,v_2) = \left(x^k \bmod p, m \cdot y^k \bmod p\right)$$

entonces

$$v_1^z = \left(x^k\right)^z \bmod p = y^k \bmod p$$

y, en consecuencia,

$$v_2 \left(v_1^z\right)^{-1} \bmod p = m \cdot y^k \cdot y^{-k} \bmod p = m.$$

Ejemplo: Supongamos elegido el primo $p = 1438033$, el cual es el 109789-ésimo primo.

Entonces $p - 1 = 1438032 = 2^4 \cdot 3 \cdot 29959$. Consideremos el elemento primitivo $x = 278783 = 17 \cdot 23^2 \cdot 31$. Y como llave secreta el exponente $z = 45276$. Entonces $y = x^z \bmod p = 1153553$.

La llave pública es pues $(p, x, y) = (1438033, 278783, 1153553)$.

Ahora, si alguien quiere enviar un mensaje, digamos $m = 89279$, entonces genera un número aleatorio $k \in [\![0, p-2]\!]$, digamos $k = 1376276$. Calcula

$$\begin{array}{rcr} v_1 & = & x^k \bmod p = 1435954 \\ v_2 & = & m \cdot y^k \bmod p = 502371 \end{array}$$

El cifrado es pues $c = (v_1, v_2) = (1435954, 502371)$.

Ahora, partiendo de c, con la llave secreta z se calcula $w_1 = v_1^z \bmod p = 748458$. Su inverso multiplicativo en $\mathbb{Z}_p$ es $w_1^{-1} = 1015266$. Así pues el producto

$$m' = v_2 w_1^{-1} \bmod p = 502371 \cdot 1015266 \bmod 1438033 = 89279 = m$$

efectivamente coincide con m.

La seguridad del método de ElGamal se basa en la dificultad del

Problema del logaritmo discreto:

Instancia: $x, y \in \mathbb{Z}_p^*$

Solución: z tal que $y = x^z \bmod p$

3.1.2 En campos finitos

Sea α un elemento de $GF(p^n)^*$, no necesariamente un generador.

Mensajes: Elementos $x \in GF(p^n)^*$.

Procedimiento: El Receptor elige un entero $t \in [1, p^n - 2]$ que ha de mantener sólo para sí, y hace público el valor $\beta = \alpha^t$.

El Transmisor elige un entero k y, para un mensaje x, transmite la pareja $(u, v) = (\alpha^k, \beta^k \cdot x)$.

Tenemos pues que

$$\begin{array}{rcl} v \cdot u^{p^n-1-t} & = & (\alpha^k)^{p^n-1-t} \cdot \beta^k \cdot x \\ & = & \alpha^{k(p^n-1-t)} \alpha^{t \cdot k} x \\ & = & (\alpha^{p^n-1-t+t})^k x \\ & = & x \end{array}$$

Cifrados: Parejas $(u, v) = (\alpha^k, \beta^k \cdot x)$. Los cifrados duplican la *longitud* de los mensajes.

Llave pública: (β, α, p^n).

Desencriptamiento: Conocida la llave t se recobra el mensaje calculando $x = v \cdot u^{tinv}$.

3.2 Método de Maussey-Omura

Los elementos de este procedimiento son:

Mensajes: Elementos de $GF(p^n)^*$.

Procedimiento: El Transmisor elige un $t \in \mathbb{Z}$ y calcula $tinv$ tal que $t \cdot tinv = 1 \mod (p^n - 1)$.

El Receptor elige un $r \in \mathbb{Z}$ y calcula $rinv$ tal que $r \cdot rinv = 1 \mod (p^n - 1)$.

Dado el mensaje $m \in GF(p^n)^*$:

1. El Transmisor calcula $m_1 = m^t$ y envía m_1.
2. El Receptor calcula $m_2 = m_1^r$ y devuelve m_2.
3. El Transmisor calcula $m_3 = m_2^{tinv}$ y envía m_3.
4. El Receptor calcula $m_4 = m_3^{rinv}$ y ha recibido m pues $m = m_4$.

Efectivamente, tenemos que se cumplen las igualdades siguientes:

$$\begin{aligned} m_4 &= m_3^{rinv} \\ &= (m_2^{tinv})^{rinv} \\ &= ((m_1^r)^{tinv})^{rinv} \\ &= (((m^t)^r)^{tinv})^{rinv} \\ &= (m^{t \cdot tinv})^{r \cdot rinv} \\ &= m \end{aligned}$$

Un Intruso puede conocer $m_1 = m^t$, $m_2 = m^{t \cdot r}$ y $m_3 = m^r$.

Su tarea, para romper el procedimiento, consiste en calcular bien t o bien r.

3.3 Algoritmos para el logaritmo discreto

3.3.1 En anillos de residuos

Algoritmo de Shank

El Algoritmo de Shank resuelve el problema del logaritmo discreto en tiempo $O(\sqrt{p})$.

En lo que sigue, todas las operaciones aritméticas son hechas en $\mathbb{Z}_p$.

Formemos la lista de $k = \left\lceil\sqrt{p-1}\right\rceil$ entradas $L = \left(x^{kj}\right)_{j=0}^{k-1}$, y sea L' la lista de parejas de la forma (j, x^{kj}), $j = 0, \ldots, k$, ordenada respecto a su segunda componente.

Formemos ahora la lista $M = (yx^{-i})_{i=0}^{k-1}$, y sea M' la lista de parejas de la forma (i, yx^{-i}), $i = 0, \ldots, k$, ordenada respecto a su segunda componente.

Sea y_0 cualquier elemento, sean j_0 la posición en la que aparece y_0 en L e i_0 la posición en la que aparece y_0 en M. La localización de estos índices se hace con los arreglos L' y M'.

El logaritmo discreto de y, respecto a x, es: $\log_x(y) = (kj_0 + i_0) \mod (p-1)$.

En efecto, se tiene $x^{kj_0} = y_0 = yx^{-i_0}$ y por tanto $y = x^{kj_0+i_0}$.

Algoritmo de Pohlig-Hellman

Con las notaciones anteriores, supongamos factorizado como un producto de primos $p-1 = \prod_{i=1}^{t} p_i^{e_i}$.

Como se quiere conocer el exponente $z \mod (p-1)$, por el Teorema Chino del Residuo, basta calcular $z \mod (p_i)$, $i = 1, \ldots, t$.

Consideremos pues el siguiente problema:

Instancia: Primos p, q y un exponente e tales que $p-1 = 0 \mod q^e$ pero $p-1 \neq 0 \mod q^{e+1}$.

Solución: Un entero z_q tal que $z_q = z \mod q^e$.

Escribamos a la solución del problema en base q: $z_q = \sum_{i=0}^{e-1} a_i q^i$. Ya que $z = z_q \mod q^e$ se tiene $z = z_q + sq^e$, para algún $s \in \mathbb{N}$.

Observamos que

$$z_q + sq^e - a_0 = \sum_{i=1}^{e-1} a_i q^i + sq^e = q\left[\sum_{i=0}^{e-2} a_{i+1} q^i + sq^{e-1}\right]$$

es decir $\frac{1}{q}\left[z_q + sq^e - a_0\right]$ es un entero. Consecuentemente

$$\eta := \frac{p-1}{q}\left[z_q + sq^e\right] = \frac{p-1}{q} a_0 \mod (p-1) =: \theta \mod (p-1).$$

Por tanto,

$$y^{\frac{p-1}{q}} = x^\eta = x^\theta.$$

Calculemos pues $v = y^{\frac{p-1}{q}}$. Si $v = 1$ entonces $a_0 = 0$. En otro caso, el mínimo i tal que $v = x^{i\frac{p-1}{q}}$ dará el valor de a_0.

```
Pohlig-Hellman(y, x, p, q, e)
{   para i = 0 hasta q − 1 hágase α_i = x^(i (p−1)/q) ;
    j := 0 ; y_j := y ;
    mientras que j ≤ e − 1 hágase
    {   v := y_j^((p−1)/q^(j+1)) ;
        encuéntrese i tal que v = α_i ;
        a_j := i ;
        y_(j+1) := y_j · x^(−a_j q^j) mod p ;
        j + +
    } ;
    dése como resultado z_q = Σ_{i=0}^{e−1} a_i q^i
}
```

Recuadro 3.1: Algoritmo de Pohlig-Hellman.

Si $c = 1$ entonces $z_q = a_0$. En otro caso, hacemos $y_1 = y \cdot x^{-a_0}$. Denotemos por z_1 al exponente tal que $z_1 = (\log_x(y_1) \bmod q^e$. De hecho, $z_1 = \sum_{i=1}^{e-1} a_i q^i$. Como antes,

$$y_1^{\frac{p-1}{q^2}} = x^{\frac{p-1}{q} a_1} \bmod p.$$

Calculemos pues $v_1 = y_1^{\frac{p-1}{q^2}}$. Si $v_1 = 1$ entonces $a_1 = 0$. En otro caso, el mínimo i tal que $v_1 = x^{i\frac{p-1}{q}}$ dará el valor de a_1.

Si $c = 2$ entonces $z_q = a_0 + a_1 q$. En otro caso, se repite el procedimiento.

3.3.2 En campos finitos

Algoritmo de Silver (1978)

Los algoritmos siguientes son en campos finitos.

Dado un elemento $\alpha \in GF(p^n)^*$ y otro elemento β que sea una potencia de α habrá que revisar una a una todas las potencias para encontrar el logaritmo discreto de β en base α.

El algoritmo de Silver, que presentamos a continuación reduce el número de posibilidades a revisar como potencias candidatas a ser el logaritmo, de hecho es idéntico al de Shank. El algoritmo aún sigue siendo bastante costoso e irrealizable cuando p y n son grandes.

Este tiene una complejidad de $O(\sqrt{p^n - 1})$ revisiones de igualdad de elementos en $GF(p^n)^*$.

Informalmente, el algoritmo se desempeña como sigue:

Se forman dos listas, *EltosConsec* y *EltosBloque*, de longitud $\sqrt{p^n - 1}$ aproximadamente. La primera contiene $\sqrt{p^n - 1}$ potencias consecutivas de la base α, corridas por el elemento cuyo logaritmo se quiere calcular, y la otra contiene $\sqrt{p^n - 1}$ potencias de la base distribuídas

Entrada: Dos elementos $\alpha, \beta \in GF(p^n)^*$.
Salida: El logaritmo discreto $LogD$ de α en base β.

$\{$ $Raíz := \sqrt{p^n - 1} + 1$;
  para $i = 1$ **hasta** $Raíz$ **hágase**
  $\{$ $EltosConsec[i] := \alpha \cdot \beta^i$;
    $EltosBloque[i] := \beta^{i \cdot Raíz}$
  $\}$;
  escójase $i, j \in [\![0, Raíz - 1]\!]$ tales que
    $EltosConsec[i] = EltosBloque[j]$;
  si Búsqueda Exitosa **entonces**
    $LogD := (j \cdot Raíz - i) \bmod (p^n - 1)$
  en otro caso 'No existe el logaritmo buscado'
$\}$

Recuadro 3.2: Algoritmo de Silver.

uniformemente entre las $p^n - 1$ posibles. Un elemento en la intersección de las listas da el logaritmo discreto buscado.

En el recuadro 3.2 presentamos el algoritmo en un seudocódigo que se explica a sí mismo.

Algoritmo de Adleman [1979]

Sea $\alpha \in GF(p)$ un generador. Sea F un conjunto de primos en $GF(p)$.

1. Supongamos que $\forall\, x \in F$ se ha calculado $\log_\alpha x$.

   Dado $y \in GF(p)$, elegimos $Z \in [\![0, p-2]\!]$ aleatoriamente y calculamos $y_1 = y \cdot \alpha^z$.

   Si y_1 se factoriza completamente con factores en F, es decir, si expresamos

   $$y_1 = \prod_{x \in F} x^{\gamma(x)} \bmod p$$

   entonces

   $$\log_\alpha y = \left(\sum_{x \in F} \gamma(x) \cdot \log_\alpha x - z \right) \bmod (p-1)$$

   Así pues, expresamos a $\log_\alpha y$ en términos de los números $\log_\alpha x, x \in F$. Veamos pues cómo calcular a estos últimos.

2. Cálculo de $\log_\alpha x \in GF(p)$.

Generemos $\mathrm{card}(F)$ exponentes $z \in [\![0, p-2]\!]$ tales que α^z se factoriza completamente en F:

$$\alpha^z = \prod_{x \in F} x^{\gamma(z,x)} \bmod p.$$

Entonces

$$z = \left(\sum_{x \in F} \gamma(z,x) \cdot \log_\alpha x - z \right) \bmod (p-1)$$

lo cual dará un sistema (no singular) de $\mathrm{card}(F)$ ecuaciones lineales en $GF(p)$, con $\mathrm{card}(F)$ incógnitas $\{\log_\alpha x\}_{x \in F}$.

La dificultad de esta etapa consiste en elegir a los números z.

Otro problema es la elección del mismo conjunto F. Un criterio es el siguiente:

3. Sea $\beta > 0$ y sea $B = e^{\beta\sqrt{\log p \cdot \log\log p}}$. Elijamos $F = \{x \in GF(p) | x$ es un primo y $x < B\}$.

El algoritmo descrito tendrá complejidad en tiempo $O(e^{\gamma\sqrt{\log p \cdot \log\log p}})$ donde γ es una constante.

Tema 4

Método de llave pública RSA

4.1 Algoritmos básicos

Un resultado clásico de Teoría de Números es el:

Teorema Pequeño de Fermat. Si p es primo y $a \in I\!N$ no es divisible entre p, entonces $a^{p-1} \bmod p = 1$.

Sean p, q dos números primos, y sea $n = p \cdot q$ su producto. Por el teorema de Fermat, se tiene que para cualquier $x \neq 0$,

$$x^{p-1} \bmod p = 1 \qquad \text{y} \qquad x^{q-1} \bmod q = 1,$$

consecuentemente

$$x^{(p-1)(q-1)} \bmod p = 1 \qquad \text{y} \qquad x^{(p-1)(q-1)} \bmod q = 1.$$

Es decir

$$p \Big| (x^{(p-1)(q-1)} - 1) \qquad \text{y} \qquad q \Big| (x^{(p-1)(q-1)} - 1)\ .$$

Así pues, $n \Big| (x^{(p-1)(q-1)} - 1)$. Es decir, $\boxed{x^{(p-1)(q-1)} \bmod n = 1}$.

El teorema de Fermat puede plantearse de manera más general. Se dice que dos números enteros n y m son *primos relativos* si no tienen factores comunes, es decir, si m.c.d.$(n, m) = 1$.

La *función de Euler* es

$$\phi : n \mapsto \text{card}\{m \leq n | m \text{ es primo relativo con } n\}.$$

Entre las propiedades más importantes de esta función están:

- Si p es primo, entonces $\phi(p) = p - 1$.
- ϕ es *multiplicativa*, es decir, si m y n son primos relativos entonces $\phi(mn) = \phi(m)\phi(n)$.

- Si p es primo y $r \in I\!N$, entonces $\phi(p^r) = p^{r-1}(p-1)$.
- $\limsup_{n\to+\infty} \frac{\phi(n)}{n} = 1$.
- $\liminf_{n\to+\infty} \frac{\phi(n)\ln(\ln(n))}{n} = e^{-\gamma}$, donde $\gamma = 0.577215\ldots$ es la, así llamada, *constante de Euler.*

Teorema Pequeño de Fermat (bis). Si n es un entero no-nulo y $a \in I\!N$ es primo relativo con n, entonces $a^{\phi(n)} \bmod n = 1$.

Además, dado que la función "módulo" es un homomorfismo, se tiene

$$r \bmod \phi(n) = s \;\Rightarrow\; a^r \bmod n = a^s.$$

Consecuentemente, si e, d son enteros tales que $e \cdot d \bmod (p-1)(q-1) = 1$ se tiene que: $a^{ed} \bmod n = a$. Esta es la base del algoritmo de encriptamiento: Si a es el mensaje, se hace $c = a^c$ su cifrado. Entonces, $c^d = m$.

La pareja (n, e) es la llave, que puede hacerse *pública*. La llave secreta es (n, d). Para calcular d sabiendo e, es necesario conocer la factorización de n como producto de los dos primos p y q.

4.1.1 Encriptamiento

De manera aún más restringida, si p y q son primos, $n = pq$ y $\lambda(n) = \text{m.c.m.}(p-1, q-1)$ entonces también vale

$$e \cdot d \bmod \lambda(n) = 1 \;\Rightarrow\; a^{ed} \bmod n = a.$$

Observamos que si se conoce p, q, entonces dado e se calcula, mediante el Algoritmo de Euclides para el Cálculo del Máximo Común Divisor, a su inverso multiplicativo d en el anillo $Z\!\!Z^*_{\text{m.c.m.}(p-1,q-1)}$. Cualquier ataque contra el sistema de encriptamiento tiene como objetivo calcular d.

Los procedimientos de encriptamiento y desencriptamiento mostrados en el algoritmo (4.1) son correctos.

Ejemplo. Consideremos dos primos *grandes*: $p = P_{23007} = 262231$ y $q = P_{23008} = 262237$. Su producto es entonces $n = pq = 68766670747$ el cual número se escribe con un número de bits igual a $\lfloor \log_2 n \rfloor + 1 = 37$.

Si se toma el exponente $e = 12521$ se tiene que e es primo relativo con el mínimo común múltiplo de $p-1$ y $q-1$. Sea $\lambda = \text{m.c.m.}(p-1, q-1) = 11461024380$. De hecho, se tiene $1 = d \cdot e + k \cdot \lambda$, donde $d = 1132280741$ y $k = -1237$.

Entrada: Un mensaje $m < n$ La llave pública (n, e). **Salida:** El cifrado c. `Encriptar` $\{\quad c := m^e \bmod n$ $\}$	**Entrada:** Un cifrado $c < n$ La llave privada (n, d). **Salida:** El mensaje m. `Desencriptar` $\{\quad m := c^d \bmod n$ $\}$

Algoritmo 4.1: Encriptado y desencriptado RSA.

Para un valor como $m = 1562435$ el código correspondiente es

$$c = m^e \bmod n = 56009798215$$

y, en efecto, si calculamos la potencia correspondiente al exponente d,

$$m' = c^d \bmod n = 1562435,$$

vemos que ésta coincide con el *mensaje* original m.

Ejemplo. Como un segundo ejemplo, consideremos la cadena de caracteres

Mexicanos al grito de guerra

al tomar uno a uno los caracteres

$$M\ ,\ e\ ,\ x\ ,\ i\ ,\ c\ ,\ a\ ,\ n\ ,\ o\ ,\ s\ ,\ \ \ ,\ a\ ,\ l\ ,\ \ \ ,\ g\ ,\ r\ ,\ i\ ,\ t\ ,\ o\ ,\ \ \ ,\ d\ ,\ e\ ,\ \ \ ,\ g\ ,\ u\ ,\ e\ ,\ r\ ,\ r\ ,\ a$$

y al ponerlos en código ASCII, obtenemos la sucesión de números

77, 101, 120, 105, 99, 97, 110, 111, 115, 32, 97, 108, 32, 103, 114, 105, 116, 111, 32, 100,
101, 32, 103, 117, 101, 114, 114, 97

Al tomarlos de cuatro en cuatro, cada tira de cuatro números se interpreta como un entero, entre 0 y $256^4 - 1$, escrito en base 256. Aplicamos pues una conversión de base 256 a base 10. Los 28 caracteres dan 7 números:

1298495593 1667329647 1931501932 543650409 1953439844 1696622453 1701999201

Aplicamos a cada uno la función de encriptamiento:

63933001817 25675825739 5190155225 15978589175 28389357636 14741227757 32565199854

y al aplicar la función de reconversión a base 256

226, 180, 48, 89, 250, 102, 2, 75, 53, 91, 123, 217, 184, 101, 235, 247, 156, 35, 56, 68, 110, 165, 72, 237, 149, 9, 131, 238

que son códigos correspondientes a los caracteres

â , ′ , 0 , Y , ú , f , ⟨no-imprimible⟩ , K , 5 , [, { , Ù , ∠ , e , ë , ÷ , ⟨no-imprimible⟩ , # , 8 , D , n , ¥ , H , í , ⟨no-imprimible⟩ ,   , ⟨no-imprimible⟩ , î

algunos de los cuales no son imprimibles. La cadena yuxtapuesta es

â′0Yúf⟨no-imprimible⟩K5[{Ù∠eë÷⟨no-imprimible⟩#8Dn¥Hí⟨no-imprimible⟩
⟨no-imprimible⟩î

Ahora procediendo en sentido inverso, partiendo de

63933001817 25675825739 5190155225 15978589175 28389357636 14741227757 32565199854

calculamos la función de desencriptamiento, para obtener

1298495593 1667329647 1931501932 543650409 1953439844 1696622453 1701999201

la cual sucesión efectivamente coincide con la “original”

1298495593 1667329647 1931501932 543650409 1953439844 1696622453 1701999201

4.1.2 Generación de llaves

En el algoritmo (4.2) presentamos un esquema para generar las llaves pública y privada en RSA. Los primos p, q pueden ser desechados tras la generación de llaves, o bien pueden mantenerse encriptados, con fines de facilitar operaciones de encriptamiento y de firmas digitales.

Un punto muy fino en el algoritmo anterior es la elección de dos primos grandes. Para esto, son necesarios,

- un generador de números aleatorios (véase el algoritmo (4.3),
- que la diferencia $|p-q|$ entre los dos primos elegidos, sea grande (de otra forma empezando por $\lceil\sqrt{n}\rceil$ se podría buscar la factorización de n), y
- con fines de evitar ataques por factorización, se pide que tanto p como q sean *primos fuertes*. Un número p es *primo fuerte* si tanto $p-1$ como $p+1$ poseen divisores primos que sean muy grandes, además un divisor grande r de $p-1$ ha de ser tal que $r-1$ posee un divisor grande.

Entrada: Nada
Salida: Llave pública : (n, e).
Llave privada : (n, d).

`Generación de llaves`
{ escójase dos primos grandes p, q ;
$n := p \cdot q$ /* Módulo público */ ;
sea $e < n$ tal que m.c.d.$(e, (p-1)(q-1)) = 1$;
sea $d = e^{-1}$ mod m.c.m.$(p-1, q-1)$;
Llave pública $:=(n, e)$;
Llave privada $:=(n, d)$
}

Algoritmo 4.2: Generación de llaves.

Entrada: k: Número de bits aleatorios a generar
Salida: $[z_i]_{i=1}^k$: Sucesión de bits aleatorios.

`Generación de bits aleatorios`
{ escójase dos primos p, q tales que $p \bmod 4 = 3 = q \bmod 4$;
$n := p \cdot q$;
sea $s < n$ tal que m.c.d.$(s, n) = 1$ una semilla ;
$x_0 := s^2 \bmod n$;
para $i \leq k$ **hágase**
{ $x_i = x_{i-1}^2 \bmod n$; $z_i = x_i \bmod 2$ } ;
dése como resultado $[z_i]_{i=1}^k$ }

Algoritmo 4.3: Algoritmo de Blum para la generación de bits aleatorios.

1	2	4	8	15	16	23	27	29	30
32	35	39	46	47	54	58	60	61	63
64	70	78	85	89	91	92	94	95	97
101	108	109	116	120	122	123	125	126	128
139	140	147	151	153	156	157	159	163	170
171	178	182	184	185	188	190	194	201	202
213	215	216	218	219	221	225	232	233	240
244	246	247	249	250	252	256	263	271	277
278	280	281	283	287	294	295	302	306	309
311	312	314	318	325	326	333	337	339	340

Tabla 4.1: Conjunto c_{341}^{-1}: Imagen inversa de 1 bajo $a \mapsto a^{340} \bmod 341$.

4.2 Pruebas de primacía

Recordamos el famoso

Teorema de los números primos. Consideremos la función que a cada n le asocia el número de primos entre 2 y n inclusive:

$$\pi : n \mapsto \mathrm{card}\{m \in [\![2, n]\!] | m \text{ es primo }\}.$$

Entonces $\lim_{n\to+\infty} \frac{\pi(n)\ln(n)}{n} = 1$.

Es decir, a la larga, el número de primos entre 2 y n es aproximadamente $\frac{n}{\ln(n)}$.

De hecho, si p_n es el n-ésimo primo, entonces para $n > 6$: $n \ln n < p_n < n(\ln n + \ln\ln n)$.

Ahora bien, atendiendo al Teorema Pequeño de Fermat, dado n, se dice que $a \in [\![1, n]\!]$ es un *testigo según Fermat* de que n es compuesto si $a^{n-1} \neq 1 \bmod n$. n es *seudoprimo según Fermat respecto a* $a \leq n-1$ si $a^{n-1} \neq 1 \bmod n$.

Naturalmente, si p es primo entonces p es seudoprimo según Fermat respecto a cualquiera de los enteros que lo preceden.

Ejemplo. El número $341 = 11 \cdot 31$ es seudoprimo según Fermat respecto a 100 enteros que lo preceden.

En efecto, la función $c : \mathbb{Z}_{341} \to \mathbb{Z}_{341}, a \mapsto a^{340} \bmod 341$ tiene como imagen al conjunto $\{1, 56, 67, 253\}$. De hecho, la imagen inversa de 1, llamémosla c_{341}^{-1} consta de los números enlistados en la tabla 4.1, y este conjunto puede expresarse como la unión de 19 órbitas de funciones de la forma $k \mapsto a^k \bmod 341$, mostradas en la tabla 4.2,

1	1	1	1	1	1	1	1	1	1
2	4	8	16	32	64	128	256	171	1
15	225	306	157	309	202	302	97	91	1
23	188	232	221	309	287	122	78	89	1
27	47	246	163	309	159	201	312	240	1
29	159	178	47	340	312	182	163	294	1
30	218	61	125	340	311	123	280	216	1
35	202	250	225	32	97	326	157	39	1
46	70	151	126	340	295	271	190	215	1
54	188	263	221	340	287	153	78	120	1
58	295	60	70	309	190	108	126	147	1
63	218	94	125	32	311	156	280	249	1
85	64	325	4	340	256	277	16	337	1
92	280	185	311	309	125	247	218	278	1
95	159	101	47	32	312	314	163	140	1
109	287	252	188	32	78	318	221	219	1
116	157	139	97	340	225	184	202	244	1
170	256	213	64	309	16	333	4	339	1
194	126	233	190	32	70	281	295	283	1

19 órbitas de funciones de la forma $k \mapsto a^k \bmod 341$. En cada renglón se muestra una tal órbita en la que el generador a, de orden 10, es el primer elemento que aparece en el renglón. El último en cada renglón, naturalmente, es el 1. La unión de estas órbitas es c_{341}^{-1}, de lo cual se ve inmediatamente que las órbitas no son ajenas a pares.

Tabla 4.2: Orbitas cuya unión forma c_{341}^{-1}.

561	$3 \cdot 11 \cdot 17$	41041	$7 \cdot 11 \cdot 13 \cdot 41$	172081	$7 \cdot 13 \cdot 31 \cdot 61$
1105	$5 \cdot 13 \cdot 17$	46657	$13 \cdot 37 \cdot 97$	188461	$7 \cdot 13 \cdot 19 \cdot 109$
1729	$7 \cdot 13 \cdot 19$	52633	$7 \cdot 73 \cdot 103$	252601	$41 \cdot 61 \cdot 101$
2465	$5 \cdot 17 \cdot 29$	62745	$3 \cdot 5 \cdot 47 \cdot 89$	278545	$5 \cdot 17 \cdot 29 \cdot 113$
2821	$7 \cdot 13 \cdot 31$	63973	$7 \cdot 13 \cdot 19 \cdot 37$	294409	$37 \cdot 73 \cdot 109$
6601	$7 \cdot 23 \cdot 41$	75361	$11 \cdot 13 \cdot 17 \cdot 31$	314821	$13 \cdot 61 \cdot 397$
8911	$7 \cdot 19 \cdot 67$	101101	$7 \cdot 11 \cdot 13 \cdot 101$	334153	$19 \cdot 43 \cdot 409$
10585	$5 \cdot 29 \cdot 73$	115921	$13 \cdot 37 \cdot 241$	340561	$13 \cdot 17 \cdot 23 \cdot 67$
15841	$7 \cdot 31 \cdot 73$	126217	$7 \cdot 13 \cdot 19 \cdot 73$	399001	$31 \cdot 61 \cdot 211$
29341	$13 \cdot 37 \cdot 61$	162401	$17 \cdot 41 \cdot 233$	410041	$41 \cdot 73 \cdot 137$

Tabla 4.3: Primeros 30 números de Carmichael. Al lado de cada uno se presenta su factorización como producto de primos.

Si n es compuesto y no posee testigos de que sea compuesto, es decir, si

$$\forall a \in [\![2, n-1]\!] : \ \text{m.c.d}(a,n) = 1 \ \Rightarrow \ a^{n-1} = 1 \bmod n. \tag{4.1}$$

se dice que n es *de Carmichael*.

Hecho. n es de Carmichael si y sólo si se cumplen las siguientes dos condiciones siguientes:

- n está *libre de cuadrados*, es decir: $\forall p : [p|n \ \Rightarrow \ p^2 \not| n]$.
- $\forall p : [p|n \ \Rightarrow \ (p-1)|(n-1)]$.

En la tabla 4.3 presentamos una lista de los primeros 30 números de Carmichael.

Se tiene pues que un número de Carmichael supera la prueba de primacía de Fermat, mas no es primo.

Otra prueba de primacía, ésta más fuerte, se debe al siguiente

Hecho. Si n es un primo impar, escribamos $n-1 = 2^s r$, donde r es impar. Entonces para todo $a < n$ tal que m.c.d.$(a,n) = 1$ se tiene

$$a^r \bmod n = 1 \qquad \text{o bien} \qquad \exists t < s : a^{2^t r} \bmod n = -1 \tag{4.2}$$

Un número n se dice ser un *seudoprimo fuerte* si satisface la condición (4.2) para todo $a < n$ tal que m.c.d.$(a,n) = 1$.

Más aún, para cada n, sea

$$W(n) = \{a < n | \text{m.c.d.}(a,n) = 1 \ \& \ \neg(4.2)\}$$

Así pues, n es seudoprimo fuerte si y sólo si $W(n) = \emptyset$.

Entrada: B: Una lista de primos pequeños
k: Número de bits
t: Parámetro de seguridad
Salida: n: Seudoprimo de k bits con probabilidad de error $\frac{1}{4^t}$.

`Búsqueda aleatoria para generar primos`
{ *AúnPorBuscar* := *Verdadero* ;
  mientras que *AúnPorBuscar* **hágase**
  { genérese aleatoriamente un entero impar n de k dígitos ;
    si $(\forall p \in B : \ p \not| n)$ **entonces** *AúnPorBuscar* $= \neg$`Miller-Rabin`(n, t)
  } ;
  dése como resultado n
}

Algoritmo 4.4: Algoritmo de búsqueda aleatoria para generar primos grandes.

Si $W(n) \neq \emptyset$ y $a \in W(n)$, se dice que a es un *testigo* de que n es compuesto, es decir, de que no es primo.

En tanto se prueben más a's y ninguno atestigüe que n es compuesto, mayor será la certeza de que n es primo. Este es el *algoritmo de Miller-Rabin* para revisar si acaso el número dado n es primo. Si t es el número de a's probadas, la probabilidad de que declare a un número n (fuerte seudo-)primo, siendo que n sea compuesto, es decir, la probabilidad de que el algoritmo se equivoque, al declarar a un número como primo, es menor que $\left[\frac{1}{4}\right]^t$. Los parámetros de entrada del algoritmo de Miller-Rabin son n y t y da como respuesta una decisión de si acaso n es seudoprimo o no.

Un procedimiento *de búsqueda aleatoria* para generar primos grandes está dado por el algoritmo 4.4.

Tema 5

Autentificación

La noción de *autentificación* (“*authentication*” en inglés) pudo realizarse con la aparición de la criptografía de llave pública. Entre las aplicaciones de la autentificación están las siguientes:

Firmas. Un mensaje puede ser “firmado” con lo cual queda determinado su origen de manera indudable.

Aceptación de compromisos. Un mensaje con la firma de un remitente, crea una responsabilidad del remitente ante el mensaje, pues solamente él pudo haber generado el mensaje firmado.

5.1 Firmas digitales

Un *esquema de firmado* es una estructura $(\mathcal{M}, \mathcal{F}, \mathcal{L}, \mathcal{R}, \mathcal{V})$ donde

- $\mathcal{M}$: conjunto de *mensajes*,
- $\mathcal{F}$: conjunto de *firmas*,
- $\mathcal{L}$: conjunto de *llaves*,
- $\mathcal{R} = (rub_L)_{L\in\mathcal{L}}$: conjunto de *procedimientos de firma.* $\forall L \in \mathcal{L}:\ rub_L : \mathcal{M} \to \mathcal{F}$ es una función que para cada mensaje genera una firma, y
- $\mathcal{V} = (ver_L)_{L\in\mathcal{L}}$: conjunto de *procedimientos de verificación.*

  $$\forall L \in \mathcal{L}:\ ver_L : \mathcal{M} \times \mathcal{F} \to \{No, Sí\}, (m, f) \mapsto ver_L(m, f) = \begin{cases} Sí & \text{si } f = rub_L(m), \\ No & \text{si } f \neq rub_L(m). \end{cases}$$

  es una función que para cada mensaje m y cada firma f, en presencia de la llave L se verifica si, en efecto $f = rub_L(m)$.

Procedimiento de firmado

En un esquema de firmado $(\mathcal{M}, \mathcal{F}, \mathcal{L}, \mathcal{R}, \mathcal{V})$, utilizando una llave $L \in \mathcal{L}$,

1. un *remitente* R con un *mensaje* m, calcula $f = rub_L(m)$ y envía la pareja (m, f),
2. el *destinatario* D calcula $ver_L(m, f)$. Si da *Sí* acepta al mensaje m como proveniente efectivamente de R.
3. Un intruso ha de hacer las veces de *suplantador* S. Dado el mensaje m, entonces
   (a) S ha de generar una f' tal que $ver_L(m, f') =$ *Sí*, o bien
   (b) ha de propiciar que el destinatario utilice una llave L' que acepte firmas con L'.

   para suplantar ante D la personalidad de R.

5.1.1 Firmado con llave pública

Supongamos un esquema de llave pública. En grupo de usuarios $\mathcal{U}$ cada uno U posee una pareja (e_U, d_U) formada por sus llaves pública y secreta, respectivamente. Cualquier mensaje m hacia U se encripta como $c = F(m, e_U)$ y U lo desencripta como $m = G(c, d_U)$.

En un tal esquema, se considera el esquema de firmado $(\mathcal{M}, \mathcal{F}, \mathcal{L}, \mathcal{R}, \mathcal{V})$, donde

- $\mathcal{M}$ es el conjunto de mensajes,
- $\mathcal{F}$ coincide con $\mathcal{M}$ (que a su vez coincide con el de cifrados),
- $\mathcal{L}$ coincide con $\mathcal{U}$,
- $\mathcal{R} = (rub_U : m \mapsto G(m, d_U))_{U \in \mathcal{U}}$, y
- $\mathcal{V} = (ver_U)_{U \in \mathcal{U}}$ donde $\forall U \in \mathcal{U}$: $\begin{array}{rcl} ver_U(m, f) = \textit{Sí} & \Leftrightarrow & m = F(f, e_U) \\ ver_U(m, f) = \textit{No} & \Leftrightarrow & m \neq F(f, e_U) \end{array}$

Si además de firmar se ha de encriptar, se puede proceder como sigue:

Firmar y luego encriptar

1. Dado el mensaje m el remitente calcula la firma correspondiente, $f = G(m, d_R)$.
2. Encripta la pareja (m, f) usando la llave pública del destinatario: $c = F(\alpha(m, f), e_D)$, donde $\alpha : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ es una función de apareamiento.
3. El destinatario, recibe c y lo desencripta con su llave secreta: $\alpha(m, f) = G(c, d_D)$.
4. El destinatario verifica la firma: $ver_R(m, f) = (m = F(f, e_R)?)$.

¡El orden es importante! El procedimiento siguiente es inseguro.

Encriptar y luego firmar

1. Dado el mensaje m el remitente lo encripta usando la llave pública del destinatario: $c = F(m, e_D)$.
2. Calcula la firma correspondiente al mensaje cifrado, $f = G(c, d_R)$.
3. El destinatario, recibe la pareja (c, f). Verifica la firma: $ver_R(c, f) = (c = F(f, e_R)?)$.
4. El destinatario queda convencido que el remitente es R.
5. El destinatario desencripta con su llave secreta: $m = G(c, d_D)$.

En efecto, si un intruso S recibe la pareja (c, f), entonces

3. calcula su propia firma correspondiente al mensaje cifrado: $f' = G(c, d_S)$.
4. El destinatario, recibe la pareja (c, f'). Verifica la firma, usando la llave de S: $ver_R(c, f') = (c = F(f, e_S)?)$.
5. El destinatario queda convencido que el remitente es S.
6. El destinatario desencripta con su llave secreta: $m = G(c, d_D)$.

S pues ha suplantado a R.

5.2 Diversos esquemas de autentificación

5.2.1 Esquema RSA

Recordamos que todo usuario posee una llave pública (n_U, e_U) y una llave secreta (n_U, d_U), donde para dos primos p, q, $n = pq$ y $e_U d_U = 1 \mod \phi(n)$. Se tiene pues:

- $rub_U : m \mapsto m^{d_U} \mod n$.
- $ver_U : (m, f) \mapsto [m = f^{e_U} \mod n]?$.

5.2.2 Esquema ElGamal

Recordamos que todo usuario posee una llave pública (p_U, x_U, y_U) y una llave secreta z_U, donde p es primo, x_U es un elemento primitivo de $\mathbb{Z}_p^*$ e $y_U = x_U^{z_U} \mod p$. Se tiene pues:

- Para un entero k aleatorio: $rub_{U,k} : m \mapsto F = (u,e)$ donde $\begin{array}{rcl} u & = & x_U^k \bmod p \\ e & = & (m - z_U u)\, k^{-1} \bmod (p-1) \end{array}$

  ¡Atención! Si se mostrase k entonces podría calcularse la llave secreta z_U a partir de la firma: $z_U = (m - ku)e^{-1} \bmod (p-1)$.

- $ver_U : (m,f) \mapsto [x_U^m = y_U^u u^e \bmod p]?$,
  pues en el caso de que (u,e) resultase del proceso de firmar, necesariamente:
  $$y_U^u u^e = x_U^{z_U u} x_U^{ke} = x_U^m$$
  ya que $z_U u + ke = z_U u + m - z_U u = m \bmod (p-1)$.

Un intruso S que conociese el mensaje m buscaría generar los correspondientes coeficientes (u,e) de la firma.

Si conociese u, el exponente e podría conocerse calculando $\log_u \left(x_U^m y_U^{-u}\right)$.

Si conociese e, el elemento u podría conocerse tratando de reslver la ecuación: $x_U^m = y_U^u u^e \bmod p$.

Algún otro intento consistiría en tratar de calcular simultáneamente u, e. No se conoce de un algoritmo eficiente que lo haga. Tampoco se ha demostrado que no exista tal algoritmo.

Sn embargo, de manera aleatoria un intruso podría generar mensajes firmados: Sean $i, j \in [\![0, p-2]\!]$ tales que j y $p-1$ sean primos relativos. Definamos:

$$\begin{array}{rcll} u & = & x^i y^j \bmod p & \\ e & = & -uj^{-1} \bmod (p-1) & \qquad \left(j \cdot j^{-1} = 1 \bmod (p-1)\right) \\ m & = & -eij^{-1} \bmod (p-1) & \end{array}$$

Entonces (u,e) es una firma de m con la llave pública (p,x,y).

En efecto, módulo p valen las igualadades siguientes:

$$y^u u^e = y^u \left(x^i y^j\right)^{-uj^{-1}} = x^{-uij^{-1}} = x^m.$$

Un segundo procedimiento para generar firmas legítimas sobre ciertos mensajes es el siguiente:

Supongamos conocido unmensaje m con su firma correspondiente (u,e). Sean $h, i, j \in [\![0, p-2]\!]$ tales que $hu - je$ y $p-1$ sean primos relativos. Definamos:

$$\begin{array}{rcll} u' & = & u^h x^i y^j \bmod p & \\ e' & = & eu'\,(hu - je)^{-1} \bmod (p-1) & \qquad \left((hu-je)\cdot(hu-je)^{-1} = 1 \bmod (p-1)\right) \\ m' & = & u'\,(hm - ie)\cdot(hu - je)^{-1} \bmod (p-1) & \end{array}$$

Entonces (u', e') es una firma de m' con la llave pública (p,x,y).

5.2.3 Digital Signature Standard

DSS ("Digital Signature Standard") fue propuesto en 1991 y adoptado como estándar en 1994. ElGamal duplica los tamaños de los mensajes al construir sus firmas. DSS es una alternativa para que al tarabajar en un subgrupo de $\mathbb{Z}_p^*$ se reduzcan los tamaños de las firmas. Aquí consideremos inicialmente

- Para un entero k aleatorio: $rub_{U,k} : m \mapsto F = (u, e)$ donde $\begin{array}{rcl} u & = & x_U^k \bmod p \\ e & = & (m + z_U u)\, k^{-1} \bmod (p-1) \end{array}$
  es decir, en la definición de e se cambia $-$ por $+$.
- $ver_U : (m, f) \mapsto [x_U^m y_U^u = u^e \bmod p]?$.

Pero,

$$x_U^m y_U^u = u^e \bmod p \;\Leftrightarrow\; x_U^{me^{-1}} y_U^{ue^{-1}} = u \bmod p$$

Ahora, supongamos que q es un primo que divide a $(p-1)$ y que x_q es una q-ésima raíz primitiva de 1, por ejemplo $x_q = x^{\frac{p-1}{q}} \bmod p$. Entonces el proceso de verificación puede hacerse módulo q:

- Para un entero k aleatorio: $rub_{U,k} : m \mapsto F = (u, e)$ donde $\begin{array}{rcl} u & = & \left(x_U^k \bmod p\right) \bmod q \\ e & = & (m + z_U u)\, k^{-1} \bmod q \end{array}$
  es decir, los exponentes se reducen módulo q.
- $ver_U : (m, f) \mapsto [x_U^{m_1} y_U^{u_1} = u \bmod q]?$, donde $\begin{array}{rcl} m_1 & = & me^{-1} \bmod q \\ u_1 & = & ue^{-1} \bmod q \end{array}$

Bibliografía

[1] F. Bauer. Decrypted secrets: Methods and maxims of cryptology, Springer, 2000.

[2] G. Brassard. Modern Cryptology. Volume 325 of Lecture Notes in Computer Science,Springer-Verlag, 1988.

[3] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22: 644-654, 1976.

[4] D. Kahn. The Codebreakers. Macmillan Co., New York, 1967.

[5] N. Koblitz. A Course in Number Theory and Cryptography. Springer-Verlag, 1994.

[6] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone. Handbook of Applied Cryptography, CRC Press, 1996. `http://www.cacr.math.uwaterloo.ca/hac/index.html`

[7] R. Rivest. Cryptography , J. van Leeuwen, ed-itor, Handbook of Theoretical Computer Sci-ence, 719 755, Elsevier Science Publishers, 1990.

[8] RSA Laboratories Frequently Asked Questions About Today's Cryptography. `http://www.rsasecurity.com`

[9] B. Schneier. Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley & Sons, New York, 2nd edition, 1996.

[10] G.J. Simmons, editor. Contemporary Cryptology The Science of Information Integrity. IEEE Press, 1992.

[11] D. R. Stinson. Cryptography: Theory and Practice, CRC Press, 1995.