MA-53G Intr. a la Criptografía **Semestre Primavera, 1996**

Cátedra No. 14 : 9 de Octubre

Transcriptor: Rodrigo González del Barrio

1 Criptosistema de Rabin

Ahora probaremos que el criptosistema de Rabin es computacionalmente seguro. Suponemos entonces que si existe un algoritmo A que conociendo n y B y dada una entrada $y \in \mathcal{C}$ entrega un $x \in \mathcal{P}$ tal que $e_K(x) = y$.

El siguiente algoritmo toma como entrada n y B y utiliza A como subrutina y factoriza el módulo n con probabilidad de a lo menos $\frac{1}{2}$.

1. Eligir $r \in \{1, \ldots, n-1\}$ al azar.
2. Calcular $y = r^2 - \frac{B^2}{4}$.
3. Obtener $x = A(y)$.
4. Calcular $x_1 = x + \frac{B}{2}$.
5. Si $x_1 \equiv \pm r (\mathrm{mod}\ n)$
   parar(fracaso)
   en caso contrario
   $\mathrm{mcd}((,x)_1 + r, n) = p \text{ o } q$

Existen varios puntos a explicar en este algoritmo. Primero observemos que:

$$e_K(r - \frac{B}{2}) = y$$

Luego en el paso 3, A retorna un valor $x \in \mathcal{P}$ tal que:

$$x_1^2 \equiv (x + \frac{B}{2})^2 \equiv x(x+B) + \frac{B^2}{4} \equiv y + \frac{B^2}{4} \equiv r^2 (\mathrm{mod}\ n)$$

De esta manera obtenemos que:

$$x_1 \equiv \pm r (\bmod n) \quad o \quad x_1 \equiv \pm wr (\bmod n).$$

donde w es una raíz cuadrada no trivial de 1 modulo n. Si $x_1 \not\equiv \pm 1 (\bmod n)$ entonces como $n|(x_1 + r)(x_1 - r)$ sigue que, al calcular $\text{mcd}((,x)_1 + r, n)$ se obtiene p o q.

Para determinar la probabilidad de éxito de quebrar el criptosistema de Rabin, definimos en $\mathcal{Z}_n \backslash \{0\}$ la relación de equivalencia:

$$r_1 \sim r_2 \Leftrightarrow r_1^2 \equiv r_2^2 (\bmod n).$$

Se verifica fácilmente que $\sim$ es relación de equivalencia. Notar que la clase de equivalencia de un elemento r tiene cardinal 4, en efecto:

$$[r] = \{\pm r, \pm wr (\bmod n)\}.$$

Pero todo elemento de $[r]$ genera el mismo $y \in \mathcal{C}$ para el paso 2, y luego da lugar al mismo x_1 en el paso 4. Luego con probabilidad de a lo menos $\frac{1}{2}$ en el paso 1 se eligio un término tal que $x_1 \not\equiv \pm r (\bmod n)$, luego el Criptosistema de Rabin es computacionalmente seguro bajo ataques de texto plano elegido, bajo el supuesto que no existe un algoritmo probabilistico eficiente que permita factorizar números de la forma $n = p \cdot q$, p y q primos.

Cabe hacer notar que bajo ataques de texto cifrado conocido el Criptosistema de Rabin es inseguro, dado que el algoritmo A es reemplazado por el algoritmo de decripción del Criptosistema de Rabin conocido por la persona que realiza el ataque.

2 Criptosistema de ElGamal

El Criptosistema de ElGamal se basa en la premisa que el problema de Logaritmos Discretos es computacionalment infactible. A continuación se plantea el problema de Logaritmos Discretos en el cuerpo finito $\mathcal{Z}_p$, donde p es un primo, esto es:

Entrada: $I = (p, \alpha, \beta)$, donde p es primo,
$\alpha \in \mathcal{Z}_p$ es un generador, y $\beta \in \mathcal{Z}_p^*$.
Salida: Encontrar el único i, $0 \leq i \leq p-2$, tal que
$\alpha^i \equiv \beta (\bmod p)$.
Denotaremos $i = log_\alpha \beta$.

En el problema de Logaritmos Discretos en $\mathcal{Z}_P$ su dificultad radica en como se a elegido p. En particular no se conoce un algoritmo determinista de tiempo polinomial para el cálculo de Logaritmo Discreto. Para frustrar ataques conocidos, p debe ser de al menos 150 dígitos, y $p-1$ debe tener a lo menos un factor primo "grande".

El Criptosistema de Elgamal es el siguiente:

Sea p un primo tal que el problema de Logaritmos Discretos en $\mathcal{Z}_p$ es computacionalmente infactible. Sea $\mathcal{P} = \mathcal{Z}_p^*$, $\mathcal{C} = \mathcal{Z}_p^* \times \mathcal{Z}_p^*$, y
$\mathcal{K} = \{(p, \alpha, i, \beta) : \alpha \in \mathcal{Z}_p^* \text{ un generador}, \beta \equiv \alpha^i (\bmod p)\}$.
Los valores p, α y β son públicos, e i es secreto.
Para $K = (p, \alpha, i, \beta)$, y para un número secreto aleatorio
$j \in \{0, \ldots, p-2\}$, sea:
$$e_K(x, j) = (y_1, y_2)$$
donde
$$y_1 = \alpha^j (\bmod p)$$
y
$$y_2 = x\beta^j (\bmod p).$$
Para $y_1, y_2 \in \mathcal{Z}_p^*$, sea:
$$d_K(y_1, y_2) = y_2 (y_1^i)^{-1} (\bmod p).$$

Observemos que:

$$d_K(e_K(x, j)) = d_K(\alpha^j(\bmod p), x\beta^j(\bmod p)) = x\beta^j[(\alpha^j)^i]^{-1}(\bmod p) = x\beta^j[\beta^j]^{-1}(\bmod p)$$

El Criptosistema de ElGamal es no determinista, puesto que el texto cifrado depende a la vez de el texto plano x y del valor aleatorio j elegido. Así, muchos textos cifrados distintos, pueden obtenerse de la encripción de un mismo texto plano.

Informalmente, se puede decir que el Criptosistema de ElGamal, trabaja de la siguiente manera. Un texto plano x es enmascarado multiplicando por β^j, produciendo y_2. El valor de α^j es también transmitido como parte del texto cifrado. Bob, quien conoce el exponente i, puede calcular β^j de α^j. Entonces el puede remover la máscara dividiendo y_2 por β^j obtiendo x.

2.1 Algoritmos para el problema de Logaritmo Discreto

En lo que sigue se asume que p es primo y α es un generador de $\mathcal{Z}_p$, ambos fijos.

Claramente, el problema de Logaritmo Discreto puede resolverse mediante búsqueda exhaustiva en O(p) de tiempo y O(1) de espacio (se omitiran los factores logaritmicos en p de los ordenes), i.e., calcular uno por uno hasta encontrarlo. Precalculando todos los posibles α^i, y ordenando los pares $(i, \alpha^i (\bmod p))$ con respecto a la segunda coordenada, se puede resolver el problema de Logaritmo Discreto en O(1) de tiempo con O(p) de precomputación y O(p) de memoria.

Algoritmo de Shank

El algoritmo de Shank logra un compromiso entre los requerimientos de espacio y tiempo mencionado en el punto anterior. Sea $m \in \{1, \ldots, p\}$ y $n = \lceil \frac{\sqrt{p-1}}{m} \rceil$, luego el algoritmo de Shank toma como entrada p, α y β y ejecuta los siguientes pasos:

1. Calcular $\alpha^{nj} (\bmod p)$, $0 \leq j \leq m-1$
2. Ordenar los m pares $(j, \alpha^{nj} (\bmod p))$ con respecto a su segunda coordenada, obteniendo una lista L_1
3. Calcular $\beta\alpha^{-i} (\bmod p)$, $0 \leq i \leq n-1$
4. Ordenar los n pares $(i, \beta\alpha^{-i} (\bmod p))$ con respecto su segunda coordenada, obteniendo una lista L_2
5. Encontrar un par $(j, y) \in L_1$ y un par $(i, y) \in L_2$ (i.e., un par que tenga su segunda coordenada idéntica)
6. Definir $log_\alpha \beta = nj + i (\bmod\ (p-1))$.

Los pasos 1 y 2 pueden ser precomputados. Además, se observa que si $(j, y) \in L_1$ y $(i, y) \in L_2$, entonces

$$\alpha^{jn} = y = \beta\alpha^{-i}$$

así

$$\alpha^{jn+i} = \beta$$

como se quería. Por otro lado, cualquier β, se puede escribir como:

$$log_\alpha \beta = nj + i.$$

donde $0 \leq j, i \leq n-1$, por lo tanto la búsqueda en el paso 5 será exitosa (puede ser hecha con una pasada simultanea a través de cada una de las dos listas L_1 y L_2).

Con este algoritmo se resuelve el problema de Logaritmo Discreto en $O(m)$ de precomputación, $O(n)$ de tiempo y de $O(\max\{n, m\})$ de memoria (omitiendo factores logaritmicos de los ordenes).

Si el tiempo y el espacio nos "cuesta" lo mismo, se obtiene que, eligiendo $m = \lceil\sqrt{p-1}\rceil$, el algoritmo de Shank corre en $O(m)$ tanto en tiempo como en memoria. Con esto se logra que el orden del algoritmo de Shank sea mínimo.

Algoritmo de Pohlig-Hellman

Supongamos que:

$$p - 1 = \prod_{i=1}^{k} p_i^{c_i},$$

donde los p_i son primos distintos. El valor de $a = log_\alpha \beta$ es único módulo $p-1$. Si calculamos $a (\bmod\, p_i^{c_i})$ para cada i con $i \in \{1, \ldots, k\}$, luego podemos calcular $a (\bmod\, (p-1))$ mediante el teorema Chino del resto, sea q primo,$p - 1 \equiv 0 (\bmod\, q^c)$ y $p - 1 \not\equiv 0 (\bmod\, q^{c+1})$.

Ahora calculamos $x = a (\bmod\, q^c)$, donde $x \in \{0, \ldots, q^c - 1\}$. se puede expresar x

como sigue:

$$x = \sum_{i=0}^{c-1} a_i q^i$$

donde $a_i \in \{0, \ldots, q-1\}$ para $i \in \{0, \ldots, c-1\}$. Además a se puede expresar como $a = x + q^c s$ para algún entero s.

El primer paso del algoritmo consiste en calcular a_0. Para esto se observa que:

$$\beta^{(p-1)/q} \equiv \alpha^{(p-1)a_0/q} (\bmod\, p).$$[1]

Por lo tanto, primero calculamos $\beta^{(p-1)/q} (\bmod\, p)$. Si $\beta^{(p-1)/q} \equiv 1 (\bmod\, p)$, entonces $a_0 = 0$. De otro modo calculamos en forma sucesiva:

$$\gamma = \alpha^{(p-1)/q} (\bmod\, p), \gamma^2 (\bmod\, p), \ldots,$$

hasta que $\gamma^i \equiv \beta^{(p-1)/q} (\bmod\, p)$ para algún i, luego $a_0 = i$

Ahora, si $c = 1$, listo. Si $c > 1$, debemos calcular a_1, se define $\beta_1 = \beta\alpha^{-a_0}$ y denotamos $x_1 = log_\alpha \beta_1 (\bmod\, q^c)$. Luego

$$x_1 = \sum_{i=1}^{c-1} a_i q^i$$

Por lo tanto, se sigue:

$$\beta_1^{(p-1)/q^2} \equiv \alpha^{(p-1)a_1/q} (\bmod\, p).$$

Así, calculamos $\beta_1^{(p-1)/q^2} (\bmod\, p)$, y encontramos i tal que $\gamma^i \equiv \beta_1^{(p-1)/q^2} (\bmod\, p)$, luego $a_1 = i$.

Si c=2, terminamos. Si $c > 2$ repetimos el proceso para obtener $a_2, \ldots, a_{c-1}$.

Luego con los $a_0, \ldots, a_{c-1}$. calculamos:

$$log_\alpha \beta (\bmod\, q^c) = \sum_{i=0}^{c-1} a_i q^i$$

El orden del algoritmo Pohlig-Hellman es de O(q) tiempo, donde q es el mayor factor primo de $p-1$, luego el factor q debe ser "grande", al seleccionar el primo p.

[1] Douglas R. Stinson, *"Cryptography, Theory and Practice"*, 1995, pp. 167

Algoritmo de Cálculo de Indices

Este algoritmo usa una base de factores $\mathcal{B}$, sea $\mathcal{B} = \{p_1, p_2, \ldots, p_B\}$. Primero se precomputan los logaritmos de los B primos en la base de factores. Segundo calcular el Logaritmos Discreto de un elemento de $\mathcal{B}$, usando los logaritmos discretos calculados, en el punto anterior.

En la precomputación, se construyen $C = B + 10$ congruencias módulo p como sigue:

$$\alpha^{x_j} \equiv p_1^{a_{1j}} p_2^{a_{2j}} \cdots p_B^{a_{Bj}} (\mathrm{mod}\ p),$$

$j \in \{1, \ldots, C\}$. Que es equivalente a escribir:

$$x_j \equiv a_{1j} log_\alpha p_1 + \cdots + a_{Bj} log_\alpha p_B (\mathrm{mod}\ (p-1)),$$

$j \in \{1, \ldots, C\}$. Dadas C congruencias en los B $log_\alpha p_i$ $i \in \{1, \ldots, B\}$, esperamos que exista una única solución módulo $p-1$. En este caso, calculamos el logaritmo de los elementos de la base de factores.

Ahora bien, se calcula el logaritmo $log_\alpha \beta$ mediante un algoritmo probabilistico tipo Las Vegas (se verá más adelante, este tipo de algoritmo).

Se elige un entero s ($s \in \{1, \ldots, p-2\}$) y se calcula $\gamma = \beta\alpha^s(\ \mathrm{mod}\ p)$. Si γ se factoriza en factores de la base $\mathcal{B}$ se obtiene la congruencia:

$$\beta\alpha^s \equiv p_1^{c_1} p_2^{c_2} \cdots p_B^{c_B} (\mathrm{mod}\ p).$$

Que es equivalente a:

$$log_\alpha \beta + s \equiv c_1 log_\alpha p_1 + \cdots + c_B log_\alpha p_B (\mathrm{mod}\ (p-1)).$$

Esta última congruencia es fácil de resolver para $log_\alpha \beta$