MUCC

Criptología. Logaritmo discreto

Criptología y Seguridad de los Datos (CSD)

©Damián López

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria **Informàtica**

December 20, 2022

Índice

Bibliografía

- Handbook of applied crytography. *A. J. Menezes, P. C. van Oorshot and S. A. Vanstone*. CRC Press. 1996.
- Understanding Cryptography. *C. Paar and J. Pelzl*. Springer. 2010.
- A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms *T. ElGamal*. IEEE Transactions on Information Theory, Vol. IT-31, No. 4, July 1985.
- Digital Signature Standard (DSS). FIPS 186-4 (2013) y FIPS 186-5 (draft).
- Schnorr Non-interactive Zero-Knowledge Proof. RFC 8235
- The Secure Shell (SSH) Transport Layer Protocol. RFC 4253
- Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer. RFC 5656

Criptografía basada en el logaritmo discreto

Consideremos un entero n, el grupo multiplicativo $\mathbb{Z}_n^*$, y un elemento α de $\mathbb{Z}_n^*$.

Dado $\beta \in \mathbb{Z}_p^*$, se busca el valor k tal que:

$$\beta = \alpha^k \bmod n,$$

a este valor k se le conoce como LOGARITMO DISCRETO DE β (EN BASE α).

Intercambio de clave

Diffie-Hellman

Require: Un valor primo p y un generador α de $\mathbb{Z}_p^*$. // públicos
Require: Dos interlocutores *Alice* y *Bob*
Ensure: Comunicación segura de un valor de $\mathbb{Z}_p^*$

Método
$\langle Alice \rangle$ Genera aleatoriamente k_A // entero secreto
$\langle Alice \rightarrow Bob \rangle$ Envia $S_A = \alpha^{k_A} \bmod p$
$\langle Bob \rangle$ Genera aleatoriamente k_B // entero secreto
$\langle Alice \leftarrow Bob \rangle$ Envia $S_B = \alpha^{k_B} \bmod p$
$\langle Alice \rangle$ Calcula $H = S_B^{k_A} \bmod p$
$\langle Bob \rangle$ Calcula $H = S_A^{k_B} \bmod p$
FinMétodo.

Intercambio de clave

ECDH

Require: Una curva elíptica $y^2 \equiv x^3 + ax + b \pmod{n}$
Require: Un punto G de la curva
Require: Dos interlocutores *Alice* y *Bob*
Ensure: Un punto P de la curva comunicado de forma segura

Método
$\langle Alice \rangle$ Genera aleatoriamente el secreto k_A // entero
$\langle Bob \rangle$ Genera aleatoriamente el secreto k_B // entero
$\langle Alice \rightarrow Bob \rangle$ Enviar $S_A = k_A G \bmod n$
$\langle Alice \leftarrow Bob \rangle$ Enviar $S_B = k_B G \bmod n$
$\langle Alice \rangle$ Calcular $H = k_A S_B \bmod n$
$\langle Bob \rangle$ Calcular $H = k_B S_A \bmod n$
FinMétodo.

Cifrado

ElGamal: Generación de la clave

Ensure: Clave ElGamal $\langle K_{pb},\ K_{pr} \rangle$

1: **Método**
2: Generar un valor primo (grande) p y obtener un generador α de $\mathbb{Z}_p^*$
3: Generar aleatoriamente un entero $1 \leq a \leq p-2$
4: Calcular $\beta = \alpha^a \bmod p$
5: $K_{pb} = (p, \alpha, \beta)$
6: $K_{pr} = a$
7: **return** $\langle K_{pb},\ K_{pr} \rangle$
8: **FinMétodo**

Cifrado

ElGamal: Cifrado

Require: $K^B_{pb} = n$: Componente pública de la clave ElGamal del destinatario
Require: x: Mensaje a cifrar // valor entero módulo p)
Ensure: y: **Cifrado** ElGamal del mensaje x

Método
Generar aleatoriamente un entero $1 \leq k \leq p - 2$ {distinto para cada mensaje}
$\gamma = \alpha^k \bmod p$
$\delta = x \cdot (\beta)^k \bmod p$
return (γ, δ)
FinMétodo

Cifrado

ElGamal: Descifrado

Require: $K_{pr}^{B} = a$: Componente privada de la clave ElGamal del destinatario
Require: $y = (\gamma, \delta)$: Criptograma dirigido al destinatario
// enteros módulo p)
Ensure: x: **Descifrado** del mensaje cifrado ElGamal
1: **Método**
2: Calcular $\gamma^{p-1-a} \bmod p = \gamma^{-a} \bmod p$
3: $x = (\gamma^{-a} \cdot \delta) \bmod p$
4: **return** x
5: **FinMétodo**

Firma digital

DSA: Generación de la clave

Require: Tamaños N y L que determinan la clave.
Ensure: Clave firma DSS $\langle K_{pb},\ K_{pr}\rangle$

```
Método
    Seleccionar q primo tal que 2^(N−1) < q < 2^N
    Escoger p primo 2^(L−1) < p < 2^L tal que q divide (p − 1)
    Seleccionar α generador del grupo de orden q en Z*_p
    Escoger aleatoriamente 1 ≤ a ≤ q − 1
    β = α^a mod p
    K_pb = (p, q, α, β)                 //Clave de verificación
    K_pr = (a)                          //Clave de firma
    return ⟨K_pb, K_pr⟩
FinMétodo
```

En notación matemática:

- $2^{N-1} < q < 2^N$
- $2^{L-1} < p < 2^L$, q divide $(p-1)$
- α generador del grupo de orden q en $\mathbb{Z}_p^*$
- $1 \leq a \leq q-1$
- $\beta = \alpha^a \bmod p$
- $K_{pb} = (p, q, \alpha, \beta)$ //Clave de verificación
- $K_{pr} = (a)$ //Clave de firma
- **return** $\langle K_{pb},\ K_{pr}\rangle$

Firma digital

DSA: Proceso de firma

Require: $K_{pr}^{A} = (a)$: Clave de firma DSA.
Require: Función resumen h de M bits.
Ensure: $x_f = (\gamma, \delta)$: **Firma** DSA del mensaje x.

Método
Generar aleatoriamente k tal que $1 \leq h < q$ y $mcd(k, q) = 1$ // k tiene inverso módulo q
Guardar en x' los $min(N, M)$ bits más significativos de $h(x)$.
$\gamma = (\alpha^k \bmod p) \bmod q$
$\delta = k^{-1}(x' + a\gamma) \bmod q$
$x_f = (\gamma, \delta)$
return x_f
FinMétodo

Firma digital

DSA: Verificación de firma

Require: Un número entero positivo compuesto n
Ensure: Booleano que determina la validez de la firma

```
Método
if not 0 < |γ|, |δ| < q then
    return False
end if
Calcular δ^{-1} mod q
u_1 = δ^{-1} x' mod q
u_2 = δ^{-1} γ mod q
if (α^{u_1} β^{u_2} mod p) mod q = γ then Firma Vericada else
    Firma no verificada end if
FinMétodo
```

Identificación

Schnorr: Configuración

1: **Autoridad de confianza** <u>PARÁMETROS DEL SISTEMA</u>

2: Selección de enteros primo p y q tal que $p-1$ es divisible por q $//p \geq 2^{1024},\ q \geq 2^{512}$

3: Selección de β un generador de un grupo de orden q.

4: Distribución segura de $\langle p, q, \beta \rangle$ y $cert_T$ a los usuarios.

5: Elección de un parámetro de confianza t $//\ 2^t < q$, (p.e. $t = 60$)

6: **Usuarios** <u>PARÁMETROS DE USUARIO</u>

7: Asignación a cada usuario A de una identidad única I_A

8: Elección por A de una clave privada $0 \leq a \leq q-1$

9: Cálculo de $v = \beta^{-a} \bmod p$.

10: Registro de I_A y v por la autoridad T, obteniendo $cert_A = \langle I_A, v, F_T(I_A, v) \rangle$.

Identificación

Schnorr: Identificación

Método
A escoge aleatoriamente un entero $1 \leq r \leq q-1$
A calcula $x = \beta^r \bmod p$
$A \;\rightarrow\; B: \quad < cert_A, x >$ // (witness)
B valida el certificado de A
B selecciona un desafío aleatorio $1 \leq e \leq 2^t < q$.
$A \;\leftarrow\; B: \quad < e >$ // (challenge) e no utilizado anteriormente
A comprueba que $1 \leq e \leq 2^t$
$A \;\rightarrow\; B: \quad < y = ae + r \bmod q >$ // (response)

B calcula $z = \beta^y v^e \pmod{p}$
if $x = z$ **then** *Id.correcta* **else** *Id.fallida* **end if**
FinMétodo.

Criptoanálisis: Baby-step Giant-step

Dado p primo, el grupo $\mathbb{Z}_p^*$ y números $\alpha, \beta \in \mathbb{Z}_n$, para obtener k $\beta = \alpha^k \bmod p$, se considera que:

Si $n = \sqrt{p-1}$, entonces $k = qn + r$, por lo que:

$$\alpha^k \equiv \alpha^{qn+r} \equiv \alpha^{qn}\alpha^r \pmod{p},$$

por lo que, si podemos obtener α^{-qn}, entonces:

$$\alpha^k\alpha^{-qn} \equiv \alpha^{qn}\alpha^{-qn}\alpha^r \equiv \alpha^r \pmod{p}$$

Criptoanálisis: Baby-step Giant-step

Algoritmo

Require: Un valor primo p, un generador α de $\mathbb{Z}_p^*$
Require: Un valor β de $\mathbb{Z}_p^*$
Ensure: k tal que $\beta \equiv \alpha^k \pmod{p}$

```
n = ⌈√p⌉
for r = 0 to n − 1 do
    Almacena en T el par ⟨r, α^r mod p⟩ indexado por α^r mod p
end for //El tiempo de acceso a la tabla T debe ser constante
Calcula α^(−n) mod p y asigna γ = β
for q = 0 to n − 1, q + + do
    if Existe un par ⟨j, γ⟩ en la tabla T then k = qn + j  end if
    γ = γα^(−n) mod p
end for
```

Criptoanálisis: Baby-step Giant-step

Ejemplo

Logaritmo discreto de $\beta = 124$ en base 2 módulo 383:
$n = \lceil\sqrt{383}\rceil = 20$. El primer paso es construir la tabla con los baby-steps:

$$
\begin{array}{ccccccc}
\langle 1,2\rangle & \langle 2,4\rangle & \langle 3,8\rangle & \langle 4,16\rangle & \langle 5,32\rangle & \langle 16,43\rangle & \langle 6,64\rangle \\
\langle 17,86\rangle & \langle 7,128\rangle & \langle 9,129\rangle & \langle 11,133\rangle & \langle 13,149\rangle & \langle 18,172\rangle & \langle 15,213\rangle \\
\langle 8,256\rangle & \langle 10,258\rangle & \langle 12,266\rangle & \langle 14,298\rangle & \langle 19,344\rangle & &
\end{array}
$$

Calculamos $2^{-20} \bmod 383 = 54$, e iteramos:

q	γ	$T[\gamma]$
0	$\gamma = \beta = 124$	$\nexists\ \langle \_, 124\rangle$
1	$\gamma = \gamma \cdot 2^{-20} \bmod 383 = 185$	$\nexists\ \langle \_, 185\rangle$
2	$\gamma = \gamma \cdot 2^{-20} \bmod 383 = 32$	$\langle 5, 32\rangle$

Con lo que $k = 2 \cdot 20 + 5 = 45$, en efecto, $2^{45} \bmod 383 = 124$.

Criptoanálisis: Pollard-rho

- Algoritmo Montecarlo que hace probable encontrar la solución con complejidad temporal de $\mathcal{O}(\sqrt{p})$, necesidades muy bajas de memoria.
- Modifica iterativamente valores x_i, a_i y b_i que cumplen siempre que:

$$x_i = \alpha^{a_i}\beta^{b_i} \bmod p$$

Criptoanálisis: Pollard-rho

- Si se encuentran x_i y x_{2i} equivalentes, entonces podemos obtener el logaritmo discreto de β.
- Para realizar el cálculo es necesario contar con o, el tamaño del grupo generado por α (el valor tal que $\alpha^o \bmod p = 1$).

$$
\begin{aligned}
\alpha^{a_i}\beta^{b_i} &\equiv \alpha^{a_{2i}}\beta^{b_{2i}} \pmod{p}\\
\beta^{b_i}\beta^{-b_{2i}} &\equiv \alpha^{a_{2i}}\alpha^{-a_i} \pmod{p}\\
\alpha^{t(b_i-b_{2i})} &\equiv \alpha^{(a_{2i}-a_i)} \pmod{p}\\
t(b_i-b_{2i}) &\equiv (a_{2i}-a_i) \pmod{o}
\end{aligned}
$$

donde t es valor que se busca.

Criptoanálisis: Pollard-rho

El algoritmo original clasifica los valores x_i en tres bloques de tamaño similar, aplicando una operación distinta para obtener el valor x_{i+1}:

$$x_{i+1} = \begin{cases} \beta x_i \bmod p & \text{si } x_i \in S_1 \\ x_i^2 \bmod p & \text{si } x_i \in S_2 \\ \alpha x_i \bmod p & \text{si } x_i \in S_3 \end{cases}$$

La modificación de los x_i implica modificar los valores asociados a_i y b_i para mantener la igualdad $x_i = \alpha^{a_i}\beta^{b_i} \bmod p$:

Criptoanálisis: Pollard-rho

Habitualmente:

$$x_{i+1} = \begin{cases} \beta x_i \bmod p & \text{si } x_i \bmod 3 = 1 \\ x_i^2 \bmod p & \text{si } x_i \bmod 3 = 0 \\ \alpha x_i \bmod p & \text{si } x_i \bmod 3 = 2 \end{cases}$$

$$a_{i+1} \ // \ b_{i+1} = \begin{cases} a_i \quad // \quad b_i + 1 \bmod p-1 & \text{si } x_i \bmod 3 = 1 \\ 2a_i \bmod p-1 \quad // \quad 2b_i \bmod p-1 & \text{si } x_i \bmod 3 = 0 \\ a_i + 1 \bmod p-1 \quad // \quad b_i & \text{si } x_i \bmod 3 = 2 \end{cases}$$

Denotando con $f(x_i, a_i, b_i)$ la función que obtiene, de acuerdo con la clasificación particular de x_i, los valores x_{i+1}, a_{i+1} y b_{i+1} (o bien $f(x_i, a_i, b_i, p, o)$ para un valor modular p y orden de α igual a o).

Criptoanálisis: Pollard-rho

Algoritmo

Require: Un entero p, un generador α de un subgrupo de $\mathbb{Z}_p^*$ de orden o y un valor β de $<\alpha>$

Ensure: k tal que $\beta \equiv \alpha^k \pmod{p}$

$a = b = aa = bb = 0$; $i = x = xx = 1$
while $i < p$ **do**
    $x = f(x, a, b, p, o)$
    $xx = f(xx, aa, bb, p, o)$; $xx = f(xx, aa, bb, p, o)$
    if $x == xx$ **then**
        if $mcd(b - bb, o) \neq 1$ **then return** *False* **end if**
        return $(aa - a)(b - bb)^{-1} \bmod o$
    end if
end while
return *False*

Criptoanálisis: Pollard-rho

Ejemplo

Buscamos el logaritmo discreto de $\beta = 804$ en base $\alpha = 9$ módulo $p = 853$.
Tenemos en cuenta que $| < 9 > | = 71$.
Clasificamos los valores de acuerdo con la regla:

$$\begin{cases} x_i \in S_1 & \text{si } x_i \bmod 3 = 1, \\ x_i \in S_2 & \text{si } x_i \bmod 3 = 0, \\ x_i \in S_3 & \text{si } x_i \bmod 3 = 2, \end{cases}$$

Criptoanálisis: Pollard-rho

Ejemplo

Las iteraciones se resumen en la tabla:

i	x_i	a_i	b_i	x_{2i}	a_{2i}	b_{2i}
0	1	0	0	1	0	0
1	804	0	1	695	0	2
2	695	0	2	850	2	2
3	284	1	2	284	4	6

$$x_i = \alpha^{a_i}\beta^{b_i} \bmod p$$

Para obtener t a partir de $t(2-6) \equiv (4-1) \pmod{71}$ necesitamos:

$$(2-6)^{-1} \bmod 71 = (2-6+71)^{-1} \bmod 71 = (67)^{-1} \bmod 71 = 53.$$

Así, $t = (4-1)67^{-1} = (4-1) \cdot 53 \bmod 71 = 17$.
En efecto, $9^{17} \bmod 853 = 804$.

Criptoanálisis: Index-calculus

- Algoritmo que considera una base de primos S que permita la representación de la *mayoría* de elementos del grupo como producto de elementos en S
- Obtención de una serie de ecuaciones lineales que permitan obtener eficientemente los logaritmos discretos de los valores en la base.
- Búsqueda de un entero p tal que $\beta \cdot \alpha^p \bmod n$ sea factorizable en S.

Criptoanálisis: Index-calculus

Algoritmo

Require: Un grupo finito cíclico $(G, \cdot \bmod n)$; un generador α de un subgrupo de orden m;

Require: β

Ensure: t tal que $\alpha^t \bmod n = \beta$

1: Selecionar una base $S = \{p_1, p_2, \ldots, p_k\}$

Criptoanálisis: Index-calculus

Algoritmo

while no se disponga de suficientes relaciones lineales **do**
    Generar aleatoriamente r y calcular $\alpha^r \bmod n$
    Descomponer $\alpha^r \bmod n$ como:
    $\alpha^r = \prod_{i=1}^{k} p_i^{e_i}, \quad e_i \geq 0$
    if es posible (α^r es S-smooth) **then**
        Aplicar logaritmos para obtener una relación lineal:
        $r = \sum_{i=1}^{k} e_i \log_\alpha p_i \bmod m, \quad e_i \geq 0$
    end if
end while
Resolver el sistema para obtener los logaritmos de los elementos en S

Criptoanálisis: Index-calculus

Algoritmo

while no se haya encontrado solución **do**
  Generar aleatoriamente r
  if $\beta \cdot \alpha^r \bmod n$ es S-smooth **then**
    Obtener: $\beta\alpha^r = \prod_{i=1}^{k} p_i^{d_i} \bmod n, \quad d_i \geq 0$
    Obtener: $\log_\alpha \beta + r = \sum_{i=1} d_i \log p_i \bmod m, \quad d_i \geq 0$ y despejar $t = \log_\alpha \beta$
    return t
  end if
end while

Criptoanálisis: Index-calculus

Ejemplo

Sea $n = 1109$, y $\alpha = 19$ donde $m = | < \alpha > | = 277$.
Buscamos el logaritmo discreto de $\beta = 274$. Consideraremos la base:

$$S = \{2, 3, 5, 7, 11, 13\}.$$

Primero generamos enteros r aleatorios que conduzcan a valores $\alpha^r \bmod n$ que puedan descomponerse en S (valores S- smooth):

$$\begin{aligned}
\alpha^{83} &\equiv 2^5 \cdot 7 \pmod{1109} \\
\alpha^{235} &\equiv 3^4 \cdot 13 \pmod{1109} \\
\alpha^{324} &\equiv 3^2 \cdot 7^2 \pmod{1109} \\
\alpha^{497} &\equiv 2^2 \cdot 7^2 \pmod{1109} \\
&\ldots
\end{aligned}$$

Criptoanálisis: Index-calculus

Ejemplo

que, aplicando logaritmos, dan lugar a relaciones lineales módulo el orden del grupo:

$$\begin{aligned}
83 &\equiv 5\log_\alpha 2 \cdot \log_\alpha 7 \pmod{277}\\
235 &\equiv 4\log_\alpha 3 \cdot \log_\alpha 13 \pmod{277}\\
324 &\equiv 2\log_\alpha 3 \cdot 2\log_\alpha 7 \pmod{277}\\
497 &\equiv 2\log_\alpha 2 \cdot 2\log_\alpha 7 \pmod{277}\\
&\dots
\end{aligned}$$

Resolviendo el sistema de ecuaciones:

$$\begin{array}{lll}
log_\alpha 2 = 201 & log_\alpha 7 = 186 & log_\alpha 3 = 253\\
log_\alpha 11 = 90 & log_\alpha 5 = 7 & log_\alpha 13 = 54
\end{array}$$

Criptoanálisis: Index-calculus

Ejemplo

Iteramos para buscar un valor t tal que $\beta\alpha^t \bmod n$ sea S-smooth... por ejemplo $\beta\alpha^{17} = 2^8 \bmod 1109$.
A partir de este punto:

$$\begin{aligned} \log_\alpha \beta + 17 &= 8 \log_\alpha 2 \bmod 277 \\ \log_\alpha \beta &= 8 \log_\alpha 2 - 17 \bmod 277 = 206 \end{aligned}$$

En efecto, $\alpha^{206} \bmod 1109 = 274$.