Logaritmo Discreto

Este documento no pretende ser un curso exhaustivo, en el mejor de los casos, únicamente puede considerarse como un conjunto de notas complementarias en alguna asignatura. Por supuesto, todo documento es susceptible de mejora, cualquier sugerencia o comunicación de error u omisión será bienvenida.

La seguridad de muchos métodos criptográficos está basada en la complejidad de computar el logaritmo discreto, esto es, dado un grupo $\langle G, \oplus \rangle$, un generador α y un valor β obtenido como composición del generador un número determinado de veces, el logaritmo discreto de β en base α es el valor k tal que $\beta = \alpha^k$. Entre los métodos criptográficos que consideran este problema para conseguir seguridad computacional están el esquema Diffie-Hellman de intercambio público de clave, o los esquemas de ElGamal de cifrado o firma. Los esquemas criptográficos basados en curvas elípticas consideran este mismo problema pero considerando un grupo de valores discretos obtenidos a partir de una curva elíptica.

Baby-step giant-step

Un método que, en ocasiones, mejora considerablemente la complejidad de una aproximación por fuerza bruta del cálculo del logaritmo discreto es el algoritmo *bay-step giant-step* de Shank. Este algoritmo establece un compromiso entre memoria y tiempo de computación utilizados (o TMTO, por *time memory trade-off*), permitiendo reducir el tiempo de cómputo.

Consideraremos un primo p y el grupo multiplicador $\mathbb{Z}_p^*$, un elemento α de $\mathbb{Z}_p$ y denotamos $n = \sqrt{p-1}$, buscando obtener para un valor β el valor k tal que $\beta = \alpha^k$ mód p. La idea en la que se basa el método de Shank considera que todo valor k puede descomponerse como $k = qn + r$ para determinados valores q y r menores que n, 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:

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

con lo que podríamos encontrar un valor q para el que $\alpha^k\alpha^{-qn}$ coincide con algún α^r.

Recordamos que podemos obtener α^{-n} mód p de varias formas, bien utilizando el algoritmo extendido de Euclides para calcular el inverso de α:

$$\alpha^{-n} \equiv (\alpha^{-1})^n \pmod{p},$$

o bien reduciendo el exponente módulo el orden del grupo:

$$\alpha^{-n} \equiv \alpha^{p-1-n} \pmod{p},$$

El algoritmo baby-step giant-step considera dos fases: en la primera se calcula y almacenan los valores de α^r para valores de r hasta n (baby-steps), en la segunda fase, se itera buscando una colisión que permita reconstruir k y obtener el logaritmo discreto (giant-steps). El Algoritmo 1 describe este método.

Algoritmo 1 Algoritmo de Shank para el cálculo del logaritmo discreto.

Entrada: Un valor primo p
Entrada: Un generador α de $\mathbb{Z}_p^*$
Entrada: Un valor β de $\mathbb{Z}_p^*$
Salida: k tal que $\beta \equiv \alpha^k \pmod{p}$ //Logaritmo discreto de β en base α módulo p

1: **Método**
2: $n = \lceil \sqrt{p} \rceil$
3: **Para** $i = 0$ to $n - 1$ **hacer**
4: Almacena en la tabla T el par $\langle i,\ \alpha^i \bmod p \rangle$ indexado por la segunda componente
5: **FinPara** //El tiempo de acceso a la tabla T debe ser constante
6: Calcula $\alpha^{-n} \bmod p$
7: $\gamma = \beta$
8: **Para** $i = 0$ to $n - 1$ **hacer**
9: **Si** Existe un par $\langle j, \gamma \rangle$ en la tabla T **entonces**
10: **Devolver** $k = in + j$
11: **FinSi**
12: $\gamma = \gamma\alpha^{-n} \bmod p$
13: **FinPara**
14: **FinMétodo.**

Ejemplo 1. *Utilizaremos el algoritmo baby-step giant-step para calcular el logaritmo discreto de* 124 *en base* 2 *módulo* 383.

Calculamos $n = \lceil \sqrt{383} \rceil = 20$. *El primer paso es construir la tabla con los baby-steps:*

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

La tabla se muestra ordenada en función de la segunda componente, y contiene algunas exponenciaciones modulares, como, por ejemplo, $2^{11} \bmod 383 = 133$ *o* $2^{18} \bmod 383 = 172$.

Para calcular el logaritmo discreto de 124*, calculamos* $2^{-20} \bmod 383 = 54$*, y empezamos a buscar una entrada en la tabla:*

i	γ	$T[\gamma]$
0	124	$\nexists \langle \_, 124 \rangle$
1	185	$\nexists \langle \_, 185 \rangle$
2	32	$\langle 5, 32 \rangle$

Una vez encontrada podemos calcular el logaritmo discreto $k = 2 \cdot 20 + 5 = 45$*, en efecto,* $2^{45} \bmod 383 = 124$.

Ejercicios

Ejercicio 1.

Utilice el algoritmo de Shank para calcular el logaritmo discreto de 30 *en base* 45 *módulo* 463

Ejercicio 2.

Utilice el algoritmo de Shank para calcular el logaritmo discreto de 140 *en base* 33 *módulo* 487

Ejercicio 3.

Utilice el algoritmo de Shank para calcular el logaritmo discreto de 303 *en base* 33 *módulo* 487
