Ataques a Supuestos de Teoría de Números

Logaritmo discreto y Diffie-Hellman

Emanuel Nicolás Herrador

Facultad de Matemática, Astronomía, Física y Computación
Universidad Nacional de Córdoba

09 de Diciembre 2025

Índice

DL (1) - Repaso

Suponemos $\mathbb{G}$ un grupo cíclico de orden q generado por $g \in \mathbb{G}$.
Dado $u \in \mathbb{G}$, encontrar $\alpha \in \mathbb{Z}_q$ tal que $g^\alpha = u$.

Nota

Consideramos q general, i.e., puede o no ser primo.
α es llamado como el logaritmo discreto de u base g.

DL (2) - Fuerza bruta

Intentar $\beta = 1, \ldots, q \Rightarrow \mathcal{O}(q)$ operaciones en $\mathbb{G}$.

Algoritmo 1: Fuerza bruta

Input: $u, g \in \mathbb{G}$
Output: $\beta \in \mathbb{Z}_q : g^\beta = u$
$v \leftarrow 1 \in \mathbb{G}$
$\beta \leftarrow 0 \in \mathbb{Z}_q$
while $v \neq u$ **do**
    $v \leftarrow v \cdot g$
    $\beta \leftarrow \beta + 1$
return β

DL (3) - MitM

Meet-in-the-Middle approach $\Rightarrow \mathcal{O}(2\sqrt{q})$ operaciones en $\mathbb{G}$.
Sea $m = \lfloor \sqrt{q} \rfloor$, suponemos $0 \leq \beta, \gamma < m$:

$$\alpha = \gamma \cdot m + \beta$$

$$u = (g^m)^\gamma \cdot g^\beta \Rightarrow u \cdot (g^{-m})^\gamma = g^\beta$$

Algoritmo 2: Construcción

Input: $g \in \mathbb{G}$
$v \leftarrow 1 \in \mathbb{G}$
for $\beta \in [0, m)$ **do**
    $L_r[v] \leftarrow \beta$
    $v \leftarrow v \cdot g$

Algoritmo 3: Query

Input: $u, g \in \mathbb{G}$
Output: $\alpha \in \mathbb{Z} : g^\alpha = u$
$g' \leftarrow g^{-m}$, $v \leftarrow u$, $\gamma \leftarrow 0$
while $v \notin Dom(L_r)$ **do**
    $v \leftarrow v \cdot g'$
    $\gamma \leftarrow \gamma + 1$
$\beta \leftarrow L_r[v]$
$\alpha \leftarrow \gamma \cdot m + \beta$ **return** α

DL (3) - MitM

Meet-in-the-Middle approach $\Rightarrow \mathcal{O}(2\sqrt{q})$ operaciones en $\mathbb{G}$.

Nota

Contra $\rightarrow$ $\mathcal{O}(m) = \mathcal{O}(\sqrt{q})$ valores en memoria.
Trade-off $\rightarrow$ Otro m: $\mathcal{O}\left(\frac{q}{m}\right)$, $\mathcal{O}(m)$ en runtime, memoria.
Adaptable $\rightarrow$ Rango $[A, B]$ con $\mathcal{O}(\sqrt{\ell})$ donde $\ell = B - A + 1$.

DL (4) - Pollard's Rho

Pollard's Rho DL con Floyd cycle detection $\Rightarrow$ $\mathcal{O}(\sqrt{q}), \mathcal{O}(1)$ en ops, memoria con probabilidad de falla $\mathcal{O}(e^{-cq})$ con $c > 0$.

Algoritmo de la tortuga y liebre (Floyd)

Encontrar un ciclo en una linked list.

- *Presencia*: dos punteros (slow, fast) comenzando al principio. Slow mueve 1, fast 2 en cada iteración. Si colisionan, hay ciclo.
- *Inicio del ciclo*: reseteá slow comenzando al principio. Ambos mueven 1. Se encontrarán en el inicio del ciclo.

DL (4) - Pollard's Rho

Pollard's Rho DL con Floyd cycle detection $\Rightarrow$ $\mathcal{O}(\sqrt{q}), \mathcal{O}(1)$ en ops, memoria con probabilidad de falla $\mathcal{O}(e^{-cq})$ con $c > 0$.

Presencia.

Sean L, a longitud del ciclo e iteraciones requeridas para slow para alcanzar el ciclo. Sea $k > 0 : kL \geq a$, cuando slow mueva kL, fast movió $2kL$ y ambos entran al ciclo. Como fast hizo kL pasos más y el largo del ciclo es L, se encuentran en el mismo punto. □

Inicio.

Ambos punteros mueven $a + xL + b(x \geq 0)$ pasos (a para alcanzar el ciclo, x veces el ciclo completo y b al final). Sea x para slow e y para fast: $a + yL + b = 2(a + xL + b) \Rightarrow a = (y - 2x)L - b$. Luego, al estar b pasos del inicio del ciclo, si mueven a más lo alcanzan. I.e., resetear slow y mover ambos de a 1 paso a veces funciona. □

DL (4) - Pollard's Rho

Pollard's Rho DL con Floyd cycle detection $\Rightarrow$ $\mathcal{O}(\sqrt{q}), \mathcal{O}(1)$ en ops, memoria con probabilidad de falla $\mathcal{O}(e^{-cq})$ con $c > 0$.
Suponemos $F : \mathbb{G} \to \{0, \ldots, q-1\}$ pseudo-random y $H : \mathbb{G} \to \mathbb{G}$ tal que $H(v) = vug^{F(v)}$.

Algoritmo 4: Pollard's Rho

Input: $u, g \in \mathbb{G}, q \in \mathbb{Z}$
Output: $\alpha \in \mathbb{Z} : g^{\alpha} = u$ o "Failed"
$i \leftarrow 1,\ x \leftarrow 0,\ v \leftarrow u$
$x' \leftarrow F(v),\ v' \leftarrow H(v)$
while $v \neq v'$ **do**
  $x \leftarrow (x + F(v)) \mod q,\ v \leftarrow H(v)$
  $x' \leftarrow (x' + F(v')) \mod q,\ v' \leftarrow H(v')$
  $x' \leftarrow (x' + F(v')) \mod q,\ v' \leftarrow H(v')$
  $i \leftarrow i + 1$
if $i < q$ **then**
  return $\alpha = (x - x')i^{-1} \mod q$
return *"Failed"*

DL (4) - Pollard's Rho

Correctitud.

Si suponemos que termina, $u^i g^x = v = v' = u^{2i} g^{x'}$. Luego, $u = (g^{x-x'})^{i^{-1}} = g^{(x-x')i^{-1}}$ □

Cantidad de iteraciones.

Si consideramos $q+1$ elementos, por principio del palomar hay un ciclo trivial que Floyd encuentra. Luego, esta es la cota con q iteraciones hechas. Se considera $i < q$ porque $u^q g^x = u^{2q} g^{x'} \Rightarrow g^x = g^{x'}$ y no se puede obtener información. □

Complejidad y probabilidad.

Por *Birthday paradox*, bastan $\sqrt{q}$ iteraciones para encontrar la colisión, suponiendo F random (y por consecuencia H también). Probabilidad exponencialmente chica sale del mismo resultado. □

DL (5) - Orden q^e

Suponemos $\mathbb{G}$ con orden q^e ($q > 1, e \geq 1$). Luego:

$$g_f := g^{(q^f)} \in \mathbb{G} \quad \text{genera subgrupo de orden } q^{e-f}$$

Sea $u = g^\alpha$ para $0 \leq \alpha < q^e$. Si $e = 1$, resolvemos normalmente (u en subgrupo de orden q). Si $e > 1$, sea $f \in [1, e)$:

$$\alpha = q^f \gamma + \beta \text{ con } 0 \leq \beta < q^f,\ 0 \leq \gamma < q^{e-f}$$

$$u = g^a = g^{q^f \gamma + \beta} = g_f^\gamma g^\beta \Rightarrow u^{(q^{e-f})} = g_{e-f}^\beta$$

Luego, computamos $\text{DLOG}(u^{(q^{e-f})},\ g_{e-f}) = \beta$ y obtenemos $\frac{u}{g^\beta} = g_f^\gamma$. Computando $\text{DLOG}(u/g^B, g_f) = \gamma$, encontramos α.
Idea del algoritmo: recursivamente en log pasos tomando $f \approx \frac{e}{2}$.

DL (5) - Orden q^e

Suponemos $\mathbb{G}$ con orden q^e $(q > 1, e \geq 1)$. Luego:

$$g_f := g^{(q^f)} \in \mathbb{G} \quad \text{genera subgrupo de orden } q^{e-f}$$

Algoritmo 5: RDL: Atacando orden q^e

Input: $g, u \in \mathbb{G}, q > 1, e \geq 1 \in \mathbb{Z}$
Output: $\alpha \in \mathbb{Z} : g^\alpha = u$
if $e = 1$ **then**
  return $\log_g u$ (Pollard's rho por ejemplo)
$f \leftarrow \left\lfloor \frac{e}{2} \right\rfloor$
$\beta \leftarrow \text{RDL}(q, f, g_{e-f}, u^{(q^{e-f})})$
$\gamma \leftarrow \text{RDL}(q, e - f, g_f, u/g^B)$
return $q^f \gamma + \beta$

DL (5) - Orden q^e

La complejidad es $\mathcal{O}(e \cdot T_{\text{base}} + e \log e \log q)$.
En el cuerpo de RDL (sin llamadas recursivas) se hacen $\mathcal{O}(e \log q)$ operaciones de grupo. Si T_{base} es la cantidad de operaciones de grupo para el caso base, entonces la complejidad se aproxima por:

$$T(e) = 2T(e/2) + O(e \log q)$$

Finalmente, entonces, si se resuelve la recurrencia tenemos $T(e) = O(e \cdot T_{\text{base}} + e \log e \log q)$.

Nota

T_{base} es el término dominante y, por ende, DLOG en $\mathbb{G}$ es difícil como computarlo para un subgrupo de orden q.

DL (6) - Pohlig-Hellman

Un poco de repaso:

Teorema (Teorema Chino del Resto (CRT))

Sean $\{n_i\}_{i=1}^k$ *una familia de enteros positivos coprimos y* $a_1, \ldots, a_k$ *enteros arbitrarios. Entonces* $\exists a \in \mathbb{Z}$ *solución del sistema de congruencias*

$$a \equiv a_i \pmod{n_i} \quad i = 1, \ldots, k$$

Más aún, cualquier $a' \in \mathbb{Z}$ *es solución sii* $a \equiv a' \pmod{n}$ *donde* $n = \prod_{i=1}^k n_i$.

Nota

Dado n, se puede computar el único entero $0 \leq a < n$ que satisface en tiempo $\mathcal{O}(len(n)^2)$.

DL (6) - Pohlig-Hellman

Suponemos $\mathbb{G}$ con orden $n = q_1^{e_1} \dots q_r^{e_r}$ (factorización en primos).
Asumimos que sabemos la factorización.
Sea $i = 1, \dots, r$, se define $q_i^* := n/q_i^{e_i} \in \mathbb{Z}$. Luego, $u^{q_i^*} = \left(g^{q_i^*}\right)^{\alpha}$ donde $g^{q_i^*}$ tiene orden $q_i^{e_i}$ en $\mathbb{G}$. Sea $0 \leq \alpha_i < q_i^{e_i}$ el DL de $u^{q_i^*}$ base $g^{q_i^*}$, $\alpha \equiv \alpha_i \pmod{q_i^{e_i}}$.
Por ello, computando $\alpha_1, \dots, \alpha_r$ usando RDL, obtenemos α usando CRT.

DL (6) - Pohlig-Hellman

Nota

Sea $T(w)$, monótona creciente, el nro de operaciones necesarias para calcular DL en un subgrupo de $\mathbb{G}$ de orden w. Luego:

$$\sum_{i=1}^{r} \mathcal{O}(e_i T(q_i) + e_i \log e_i \log q_i) = \mathcal{O}(T(q_{\max}) \log n + \log n \log\log n)$$

Y, por ende, la dificultad está determinada por la complejidad de calcular DLOG en un grupo cíclico con orden $q_{\max}$.

Nota

DL es fácil en grupos con orden producto de primos chicos. Luego, el orden de $\mathbb{G}$ *debe tener a menos un factor primo "largo"*.

Índice

DDH - Leakage en orden compuesto

Caso $\mathbb{G}$ de orden $n = 2m$ (Generalizable a primos chicos)

Sea $u = g^{\alpha} \in \mathbb{G}$,

$$\alpha \text{ es par} \iff u^{n/2} = 1. \tag{1}$$

Sea $u = g^{\alpha}, v = g^{\beta} \in \mathbb{G}$,

$$\alpha\beta \in \mathbb{Z}_n \text{ es par} \iff (u^{n/2} = 1 \vee v^{n/2} = 1). \tag{2}$$

Podemos aprender un bit de información sobre el secreto de DH ($g^{\alpha\beta}$). Sea $(g^{\alpha}, g^{\beta}, w = g^{\gamma}) \in \mathbb{G}^3$ una tupla DDH, obtenemos paridad de γ por (1) y de $\alpha\beta$ por (2). Si coinciden, aceptamos y sino rechazamos.
Como siempre acepta cuando $w = g^{\alpha\beta}$, pero cuando $w \xleftarrow{\$} G$ lo hace con probabilidad $1/2$, sacamos ventaja $1/2$ de romper DDH.

Índice

SDH - The Brown-Gallant-Cheon algorithm

Algo de repaso:

Definición (Juego SDH)

Sea $\mathbb{G}$ grupo cíclico de orden primo q, para un adversario $\mathcal{A}$ el juego es:

- Challenger elige $a \xleftarrow{\$} \mathbb{Z}_q$
- $\mathcal{A}$ hace SDH queries: envía $v \in \mathbb{G}$ y recibe $w := v^{\alpha} \in \mathbb{G}$.
- $\mathcal{A}$ responde $\tilde{\alpha} \in \mathbb{Z}_q$

Donde $\mathcal{A}$ gana si $\alpha = \tilde{\alpha}$.

SDH - The Brown-Gallant-Cheon algorithm

En algunos grupos, SDH es más fácil que DLOG. Consideremos $\mathcal{A}$ dado por:

1. $g_0 := g$
2. Para $i = 1, \ldots, d$, SDH query para g_{i-1} obteniendo $g_i := g_{i-1}^{\alpha}$
3. $\mathcal{A}$ tiene:

$$v_\alpha := \left(g, g^\alpha, g^{(\alpha^2)}, \ldots, g^{(\alpha^d)}\right)$$

Y se puede obtener α con mejor complejidad si $d|q-1$ o $d|q+1$.

En particular, si $d|q-1$, se usan $\mathcal{O}(\sqrt{q/d} + \sqrt{d})$ operaciones, mientras que si $d|q+1$ se usan $\mathcal{O}(\sqrt{q/d} + d)$.

Nota

Un adversario SDH en un grupo genérico con d queries y probabilidad $1/2$ de éxito toma tiempo al menos $\Omega(\sqrt{q/d})$. Luego, $\sqrt{d}$ es el mejor speed-up en un grupo genérico.