

8. Criptosistemas basados en el problema del logaritmo discreto

Recordatorio:

Definición

Una función es de una dirección si es sencillo calcular $f(x)=y$ pero computacionalmente no viable calcular $f^{-1}(y)=x$.

Nota: El problema de factorización de enteros es una función de una dirección de RSA.

¿Hay otras funciones de una dirección para construir sistemas criptográficos asimétricos?

Sí

El problema del logaritmo discreto

La mayoría de los algoritmos de llave pública distintos a RSA están basados en esto

8.1 Intercambio de llaves Diffie-Hellman (DHKE)

- Propuesto por Whitfield Diffie y Martin Hellman en 1976.
- Primer esquema asimétrico publicado en la "literatura abierta".
- Usando en muchos protocolos criptográficos como Secure Shell (SSH), Transport Layer Security (TLS) e Internet Protocol Security (IPSec).

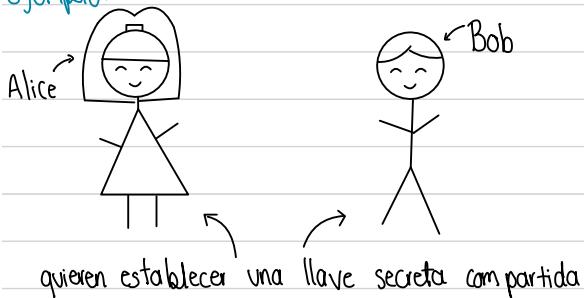
Idea general:

Para un primo p , la exponenciación en \mathbb{Z}_p^* es una función de una dirección y es conmutativa, i.e.

$$k \equiv (\alpha^x)^y \equiv (\alpha^y)^x \pmod{p}$$

la clave del porqué funciona
Este es el secreto del DHKE

Ejemplo:



Probablemente hay una tercera persona involucrada que elige y publica los parámetros públicos que se necesitan para el intercambio de llaves.

También lo pueden hacer únicamente Alice y Bob.

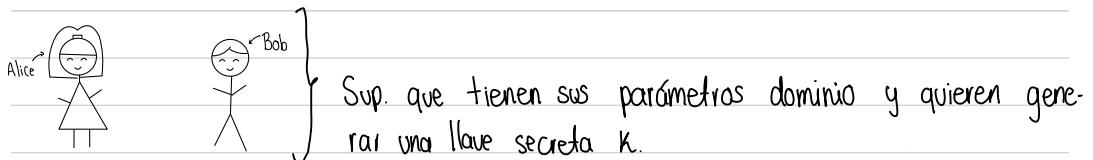
- DHKE consta de dos protocolos:

- el set-up
- el protocolo principal

Diffie-Hellman setup:

- 1: Elige un primo p grande.
- 2: Elige un entero $\alpha \in \{2, 3, \dots, p-2\}$.
- 3: Publicar p y α .

parámetros dominio

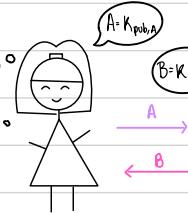


Intercambio de llaves Diffie-Hellman

Alice:

- Elige $a = K_{pr,A} \in \{2, \dots, p-2\}$
- Calcula $A = K_{pub,A} \equiv a^a \pmod{p}$

$$K_{AB} = K_{pub,B}^{K_{pr,A}} \equiv B^a \pmod{p}$$



Bob:

- Elige $b = K_{pr,B} \in \{2, \dots, p-2\}$
- Calcula $B = K_{pub,B} \equiv a^b \pmod{p}$

$$K_{AB} = K_{pub,A}^{K_{pr,B}} \equiv A^b \pmod{p}$$

Afirmación
 $K_{AB} = K_{AB}$

La llave conjunta K_{AB} puede ser utilizada para establecer una comunicación segura entre Bob y Alice con algoritmos como AES, 3DES o un stream cipher.

Observación

$$\begin{aligned} B^a &\equiv (a^b)^a \pmod{p} = a^{ba} \pmod{p} = (a^a)^b \pmod{p} \equiv A^b \\ &\Rightarrow B^a \pmod{p} = A^b \pmod{p} \Rightarrow K_{AB} = K_{AB} \end{aligned}$$

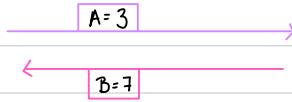
Ejemplo:

Consideremos los parámetros dominio $p=29$ y $a=2$

Alice

$$p-2 = 29-2$$

- Escoge $a = K_{pr,A} = 5 \in \{2, \dots, 27\}$
- $A = K_{pub,A} = 2^5 \equiv 3 \pmod{29}$



$$K_{AB} = B^a = 7^5 \equiv 16 \pmod{29}$$

Bob

- Escoge $b = K_{pr,B} = 12 \in \{2, \dots, 27\}$
- $B = K_{pub,B} = 2^{12} \equiv 7 \pmod{29}$

$$K_{AB} = A^b = 3^{12} \equiv 16 \pmod{29}$$

- Los aspectos computacionales de DHKE son similares a los de RSA.
 \Rightarrow DHKE debe tener una longitud similar a la de RSA mod n para ser seguro (≥ 2048 bits)
- El entero α debe ser primitivo (Definición: Sea G un grupo cíclico, un elemento $a \in G$ es **primitivo** si tiene orden máximo)
 \Leftrightarrow (i.e. $\text{ord}(a) = |G|$)
- K_{AB} tiene la misma longitud de bits que p .
- a y b deben provenir de un generador de números aleatorios (para que no se puedan adivinar fácilmente)
- El protocolo se puede generalizar (ej. para grupos de curvas elípticas \rightsquigarrow criptografía de curvas elípticas)

8.2 Álgebra Abstracta

- Grupos. (Nota: se busca trabajar con grupos finitos)
- Grupos cíclicos
- Teoremas de grupos cíclicos que son relevantes en la criptografía.

Teorema

Para todo primo p , (\mathbb{Z}_p^*, \cdot) es un grupo cíclico abeliano finito.



Relevante puesto que casi todo buscador web usa un criptosistema sobre \mathbb{Z}_p^* .

Teorema

Sea G un grupo cíclico finito. Entonces para todo $a \in G$ se tiene que

1: $a^{|G|} = 1$

2: $\text{ord}(a)$ divide a $|G|$

Teorema

Sea G un grupo cíclico finito, entonces

1- El número de elementos primativos de G es $\phi(|G|)$.

2- Si $|G|$ es primo entonces todos los elementos $a \neq 1 \in G$ son primativos.

• Subgrupos

Teorema

Sea G un grupo cíclico finito de orden n y sea α un generador de G . Entonces para todo entero k que divide n existe exactamente un subgrupo cíclico H de G de orden k . Este subgrupo está generado por $\alpha^{\frac{n}{k}}$. H consiste exactamente de los elementos $a \in G$ tales que $a^k = 1$. No hay otros subgrupos.

8.3 El problema del logaritmo discreto en campos primos

Sea p primo

Definición (El problema del logaritmo discreto (DLP) en \mathbb{Z}_p^*)

Dados el grupo cíclico \mathbb{Z}_p^* , un elemento primitivo $\alpha \in \mathbb{Z}_p^*$ y algún otro elemento $\beta \in \mathbb{Z}_p^*$. El DLP es el problema de encontrar el entero $1 \leq x \leq p-1$ tal que

$$\alpha^x \equiv \beta \pmod{p}$$

Observación

Como α es primitivo \Rightarrow dicho x debe de existir

• A x se le llama el **logaritmo discreto** de β base α y escribimos

$$x = \log_{\alpha} \beta \pmod{p}$$

Nota: Para parámetros suficientemente grandes, calcular logaritmos discretos módulo un primo es un problema muy difícil.

Observación

Como $\alpha^x \equiv \beta \pmod{p}$ es computacionalmente fácil (exponenciación)
⇒ tenemos una función de una dirección.

Nota: Como $|\mathbb{Z}_p^*| = p-1$ no es primo, a menudo se usa DLPs en subgrupos de \mathbb{Z}_p^* de orden primo

Ejemplo:

Consideremos el grupo \mathbb{Z}_{47}^* de orden 46
⇒ los subgrupos de \mathbb{Z}_{47}^* tienen orden 23, 2 o 1

Como 23 es primo ⇒ todos los elementos en el grupo de cardinalidad 23 son generadores.

Escogemos el 2 de dicho subgrupo

Un DLP posible está dado por $p=36$:

$$2^x \equiv 36 \pmod{47}$$

Usando fuerza bruta se obtiene $x=17$.

8.3.2 El problema del logaritmo discreto generalizado

Nota: Algo que hace a DLP muy útil es que puede definirse sobre cualquier grupo cíclico.

Definición (Problema del logaritmo discreto generalizado)

Dado un grupo cíclico finito G con operación de grupo \circ y cardinalidad n . Consideraremos un elemento primitivo $\alpha \in G$ y algún otro elemento $\beta \in G$. El problema del logaritmo discreto es encontrar el entero x , con $1 \leq x \leq n$, tal que

$$\beta = \underbrace{\alpha \circ \alpha \circ \dots \circ \alpha}_{x\text{-veces}} = \alpha^x$$

Observación

Como α es primitivo $\Rightarrow x$ debe existir

iNOTA! Existen grupos cíclicos donde el problema del logaritmo discreto no es difícil \Rightarrow No pueden usarse en criptosistemas de llave pública.

Ejemplo: Consideremos el grupo aditivo $G = (\mathbb{Z}_{11}, +)$.

G es un grupo cíclico finito y $\alpha = 2 \in G$ es un elemento primitivo del mismo

Tratemos de resolver el DLP para $\beta = 3$; esto es, busquemos x tal que $1 \leq x \leq 11$ y

$$x \cdot 2 = \underbrace{2 + 2 + \dots + 2}_{x\text{-veces}} \equiv 3 \pmod{11}$$

$$2^x \equiv 6 \pmod{11} \Rightarrow x \equiv 2^{-1} \cdot 3 \pmod{11} = 6 \cdot 3 \pmod{11} = 18 \pmod{11} \equiv 7 \pmod{11}$$

Listado de DLPs propuestos para ser usados en criptografía:

- 1: Grupo multiplicativo (o un subgrupo) el campo primo \mathbb{Z}_p .
- 2: Grupo cíclico formado por una curva elíptica (Capítulo 9).
- 3: El grupo multiplicativo de un campo de Galois $GF(2^m)$ o un subgrupo de este. **Nota:** No son tan populares porque los ataques hacia estos son más poderosos que los de DLP en $\mathbb{Z}_p \Rightarrow$ se requiere una llave de mayor longitud

4. Curvas hiperelípticas o variedades algebraicas (raramente usadas)

8.3.3 Ataques en contra del DLP

Los algoritmos para calcular logaritmos discretos que veremos se pueden clasificar en **algoritmos genéricos** y **algoritmos no genéricos**.

Algoritmos genéricos

↑ usan únicamente la operación del grupo
no se analizan las propiedades específicas de los grupos
⇒ funcionan el cualquier grupo cíclico

Se pueden subdividir en dos clases:

- los que su tiempo de ejecución depende del tamaño del grupo cíclico.
Por ejemplo: fuerza bruta, el algoritmo baby-step giant-step y el método de Pollard's rho.
- los que su tiempo de ejecución depende del tamaño de los factores primos del orden del grupo.
Ejemplo: algoritmo Pohlig-Hellman.

Fuerza bruta

Se calculan las potencias de α hasta que coincide con β .

$$\alpha^1 \stackrel{?}{=} \beta$$

$$\alpha^2 \stackrel{?}{=} \beta$$

⋮

Usando este método se espera, para un exponente aleatorio x , se espera encontrar una solución después de verificar con la mitad de todos los posibles valores de x .

$$\alpha^x \stackrel{?}{=} \beta$$

⇒ Esto nos da una complejidad de $\mathcal{O}(|G|)$ pasos, donde $|G|$ es la cardinalidad de G .

⇒ La cardinalidad $|G|$ debe ser suficientemente grande para evitar ataques de fuerza bruta en criptosistemas DLP-basados.

Ejemplo: Para el grupo \mathbb{Z}_p^* con p primo se requieren $\frac{(p-1)}{2}$ intentos (en promedio) para calcular un logaritmo discreto.

⇒ Para evitar ataques de fuerza bruta $|G|=p-1$ debe estar en el rango de 2^{128} .

El método Baby-step Giant-step de Shank

La idea se basa en reescribir el logaritmo discreto $x = \log_a \beta$ en una representación de dos dígitos:

$$x = x_g m + x_b \quad \text{para } 0 \leq x_g, x_b < m$$

El entero m se elige como la raíz cuadrada del orden del grupo
 $m = \sqrt{|G|}$.

$$\Rightarrow \beta = a^x = a^{x_g m + x_b}$$

La idea del algoritmo es encontrar (x_g, x_b) (\Rightarrow tenemos el valor de x)

¿Cómo?

• Baby-step:

Calcular y almacenar todos los valores a^{x_b} , con $0 \leq x_b < m$.

Aquí estamos haciendo $m \approx \sqrt{|G|}$ pasos y almacenando $m \approx \sqrt{|G|}$ elementos

• Giant-step:

Verificar para todos los x_g con $0 \leq x_g < m$ si se satisface
 $\beta \cdot (\alpha^{-m})^{x_g} = \alpha^{x_b}$
 para algún α^{x_b} calculado en la fase Baby-step.

- Si se satisface

$$\Rightarrow \beta \cdot (\alpha^{-m})^{x_{g,0}} = \alpha^{x_{b,0}} \text{ para algún par } (x_{g,0}, x_{b,0})$$

$$\Rightarrow x = x_{g,0}m + x_{b,0}$$

Este método requiere $O(\sqrt{|G|})$ pasos y la misma cantidad de memoria.

Ejemplo: En un grupo G con $|G| = 2^{128}$ el atacante necesita aprox. $2^{64} = \sqrt{2^{128}}$ cálculos y espacios de memoria (lo cual actualmente es posible de realizar)

\Rightarrow Para tener un ataque de complejidad 2^{128} el grupo debe tener cardinalidad $|G| \geq 2^{256}$

Si $G = \mathbb{Z}_p^*$ $\Rightarrow p$ debe tener longitud de al menos 256 bits.

El método Rho de Pollard \leftarrow método probabilístico

A igual que el método previo, tiene un tiempo de ejecución de $O(\sqrt{|G|})$ pero utiliza menos memoria

Idea general: generar un grupo de elementos de la forma $\alpha^i \beta^j$ (de manera pseudoaleatoria) teniendo en cuenta los valores i y j (rastreándolos)

Se generan estos elementos hasta obtener

$$\alpha^{i_1} \beta^{j_1} = \alpha^{i_2} \beta^{j_2}$$

Si sustituimos $\beta = \alpha^x$ y comparamos los exponentes a cada lado de la ecuación previa tenemos

$$i_1 + x j_1 \equiv i_2 + x j_2 \pmod{16}$$

$$\Rightarrow x \equiv \frac{i_2 - i_1}{j_1 - j_2} \pmod{16}$$

- Actualmente es el mejor algoritmo conocido para calcular logaritmos discretos en grupos de curvas elípticas.

Algoritmo Pohlig-Hellman

Basado en el Teorema Chino del Residuo

Normalmente se utiliza en conjunto con otro ataque DLP.

Sea $|G| = p_1^{e_1} \cdot p_2^{e_2} \cdots p_r^{e_r}$ la factorización en primos de $|G|$.

Se calculan los logaritmos discretos

$$x_i \equiv x \pmod{p_i^{e_i}}$$

en cada subgrupo de orden $p_i^{e_i}$

} con el algoritmo que quieras
(método Rho ó Baby-step
Giant step)

Se obtiene x de todos los x_i usando el Teorema Chino del Residuo.

El tiempo de ejecución del algoritmo depende de los factores primos de $|G|$.

Para prevenir el ataque, el grupo debe tener un orden tal que el factor primo más grande esté en el rango de 2^{256}

Este algoritmo necesita conocer una factorización de $|G|$. En el caso de criptosistemas de curvas elípticas, calcular el orden del grupo cíclico no siempre es sencillo.

Algoritmos no genéricos

El método de cálculo de índice

Explota propiedades especiales del grupo en específico

Es un algoritmo muy eficiente para calcular logaritmos discretos en \mathbb{Z}_p^* y $GF(2^n)^*$.

Tiempo de ejecución subexponencial

Propiedad: Una porción significativa de elementos en G puede ser expresada como un producto de elementos de un subconjunto pequeño de G .

Ejemplo: En \mathbb{Z}_p^* muchos elementos pueden expresarse como un producto de primos pequeños

Nota: No se sabe si esta propiedad se satisface para grupos de curvas elípticas.

Para garantizar seguridad ante este ataque de 128 bits (i.e. el atacante debe hacer 128 pasos), el primo p en un DLP en \mathbb{Z}_p^* debe tener al menos 3072 bits de longitud.