

Introdução à Chave Pública

- ▶ Troca de chaves Diffie-Hellman
- ▶ Problemas de logaritmo discreto
- ▶ Ataques a problemas de logaritmo discreto
- ▶ Encriptação Elgamal

Troca de Chaves de Diffie-Hellman

- ▶ Parâmetros públicos p, α
- ▶ Alice:
 - 1 Sorteia $a = K_{prA} \in \{2, 3, \dots, p - 2\}$
 - 3 Envia para Bob $A = \alpha^a \bmod p$
 - 5 Calcular $K_{AB} = B^a \bmod p$
- ▶ Bob:
 - 2 Sorteia $b = K_{prB} \in \{2, 3, \dots, p - 2\}$
 - 4 Envia para Alice $B = \alpha^b \bmod p$
 - 5 Calcular $K_{BA} = K_{AB} = A^b \bmod p$
- ▶ Depois da troca de chaves, usar K_{AB} como chave secreta (simétrica) no AES!

Explicando Diffie-Hellman

- ▶ Como escolher primo p ?
- ▶ Como escolher inteiro α ?
- ▶ Como o uso desses parâmetros garante a segurança?
 - ▶ Teoria de Números

O problema de Diffie-Hellman em \mathbb{Z}_p^*

- ▶ Dados um grupo cíclico finito \mathbb{Z}_p^* de ordem $p - 1$ e um elemento primitivo $\alpha \in \mathbb{Z}_p^*$ e outro elemento $\beta \in \mathbb{Z}_p^*$.
- ▶ O problema de Diffie-Hellman é encontrar o inteiro x , com $1 \leq x \leq p - 1$ tal que

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

- ▶ Também conhecido como o problema de computar o logaritmo discreto.

Problemas de Diffie-Hellman (premissas de segurança)

- ▶ Seja um grupo G com gerador α e elementos $A, B \in G$
- ▶ Definir $DH_\alpha(A, B) = DH_\alpha(\alpha^a, \alpha^b) = \alpha^{ab}$
- ▶ Problema **Computacional** de Diffie-Hellman (CDH):
 - ▶ Dados gerador α e elementos α^a e α^b , quanto é α^{ab} ?
- ▶ Problema de **Decisão** de Diffie-Hellman (DDH):
 - ▶ Dados gerador α e elementos α^a e α^b , verificar se $x \in G$ é uma solução correta para o problema computacional de Diffie-Hellman, ou seja, $x = \alpha^{ab}$
 - ▶ Premissa mais exigente porque exige o uso de um grupo em que a verificação (decisão) é difícil (nem todos grupos são difíceis de fazer a verificação sem resolver a computação)

Problemas de logaritmo discreto em criptografia

- ▶ $(G, \circ) = (Z_p, \times \text{ mod } p)$ para p primo ou em um subgrupo em Z_p
 - ▶ Fatoração: módulo 2048 bits (NIST, segurança de 112 bits)
 - ▶ log discreto, subgrupo de ordem q de Z_p^* : $\|q\| = 224$,
 $\|p\| = 2048$
- ▶ Curvas elípticas
 - ▶ log discreto, grupo de curva elíptica de ordem q : $\|q\| = 224$
- ▶ $(GF(2^m), \times)$
- ▶ Curvas hiperelípticas

Ataques contra o Problema do Logaritmo Discreto

- ▶ Algoritmos genéricos
 - ▶ Força bruta
 - ▶ Algoritmo “passo de bebê, passo de gigante”
 - ▶ Método de ρ (“rô”) de Pollard
- ▶ Algoritmos dependentes dos tamanho dos fatores primos da ordem do grupo
 - ▶ Algoritmo de Pohlig-Hellman

Força bruta

- ▶ Tentar potências até achar x correto

$$\alpha^1 = \beta?$$

$$\alpha^2 = \beta?$$

...

$$\alpha^x = \beta$$

- ▶ $O(|G|)$ passos para β aleatório em G
- ▶ Defesa: garantir que $|G|$ é grande
- ▶ Exemplo:
 - ▶ Em \mathbb{Z}_p^* , escolher p primo grande tal que $\log_2(p-1) > 2^{80}$

Algoritmo “passo de bebê, passo de gigante” de Shanks

- ▶ Troca o custo de tempo pelo custo de memória
- ▶ Problema de computar $x = \log_{\alpha} \beta$ é reorganizado em duas partes de ordens de magnitude distintas

$$x = x_g m + x_b$$

onde $0 \leq x_g, x_b < m = \lceil \sqrt{|G|} \rceil$

- ▶ Assim, $\beta = \alpha^x = \alpha^{x_g m + x_b}$ e então $\beta \cdot (\alpha^{-m})^{x_g} = \alpha^{x_b}$
 - ▶ Ideia: **achar par** (x_g, x_b) por “dividir e conquistar”

Passo 1 (bebê) - computar e guardar α^{x_b} com $\sqrt{|G|}$ “contas” e armazenamentos

Passo 2 (gigante) - procurar por x_g tal que vale

$$\beta \cdot (\alpha^{-m})^{x_g} = \alpha^{x_b}$$

- ▶ Possível calcular log discreto de grupos de 2^{80} elementos com $\approx 2 \times 2^{40}$ operações, então usar ao menos 2^{160} elementos (Curvas elípticas)

Método de ρ (“rô”) de Pollard

- ▶ Custo de tempo $O(\sqrt{|G|})$ e custo de espaço negligível
- ▶ Método
 - ▶ Gerar aleatoriamente elementos da forma $\alpha^i \beta^j$
 - ▶ Parar quando encontrar “colisão” da forma

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

- ▶ Se $\beta = \alpha^x$ então $i_1 + xj_1 \equiv i_2 + xj_2 \pmod{|G|}$
- ▶ Computar

$$x \equiv \frac{i_2 - i_1}{j_1 - j_2} \pmod{|G|}$$

Algoritmo de Pohlig-Hellman

- ▶ Exploração de possível fatoração da ordem do grupo
- ▶ Usado em conjunto com outros ataques

$$|G| = p_1^{e_1} \cdot p_2^{e_2} \cdot \dots \cdot p_l^{e_l}$$

- ▶ Dividir e conquistar
- ▶ Computar cada solução para logaritmos discretos dos fatores $e_1 \dots e_j \dots e_l$

$$x_j \equiv x \pmod{p_j^{e_j}}$$

- ▶ obter $x = \log_{\alpha} \beta$ no grupo G é obtido a partir do Teorema do Resto Chinês usando os x_j .

Teorema do Resto Chinês

- ▶ Sendo n_1, n_2, \dots, n_k inteiros coprimos entre si, existe um x para qualquer sequência de inteiros a_1, \dots, a_k que resolve o sistema de equivalências:

$$x \equiv a_1 \pmod{n_1}$$

...

$$x \equiv a_k \pmod{n_k}$$

- ▶ E temos que $x = \sum_{i=1}^k a_i n_i$

Para ataque de Pohlig-Hellman,

$$n_1 = \frac{|G|}{p_1^{e_1}}, n_2 = \frac{|G|}{p_2^{e_2}} \dots$$

sendo $x_1 = a_1, x_2 = a_2 \dots$

Protocolo de Encriptação Elgamal

Bob

escolher primo grande p

escolher elemento primitivo $\alpha \in \mathbb{Z}_p^*$ (ou em subgrupo)

escolher $k_{pr} = d \in \{2, \dots, p-1\}$

computar $k_{pub} = \beta = \alpha^d \pmod p$

$$k_{pub} = (p, \alpha, \beta)$$



Alice

escolher $i \in \{2, \dots, p-1\}$

computar chave efêmera $k_E \equiv \alpha^i \pmod p$

computar chave de máscara $k_m \equiv \beta^i \pmod p$

encriptar mensagem $x \in \mathbb{Z}_p^*$, $y \equiv x \cdot k_m \pmod p$

$$(k_E, y)$$



Bob

computar chave de máscara

$$k_M \equiv k_E^d \pmod p$$

decriptar $x \equiv y \cdot k_M^{-1} \pmod p$

Conclusões

- ▶ Com p e α bem escolhidos garantimos que todos elementos no grupo exceto 1 serão geradores
 - ▶ Ou subgrupos são de tamanho grande
- ▶ Dessa forma, será difícil de achar o valor a $A \equiv \alpha^a \pmod{p}$ usando o log discreto
$$a = \log_{\alpha} A \pmod{p}$$
- ▶ Se alguns ciclos fossem mais curtos, alguns valores de α seriam mais fácil de calcular o log discreto