lec 12

ElGamal Encryption Scheme

It's security is based on DL problem

Algorithm for the key generation of ElGamal

summary: each entity creates a public-key & a corresponding private-key

1. Generate a large random prime "p" & a generator "α" of the multiplicative group $\mathbb{Z}_p^*$ of the integers mod "p".

$$\mathbb{Z}_p = \{0, 1, \dots, p-1\}$$

$$\mathbb{Z}_p^* = \{1, 2, \dots, p-1\}$$

Note: you can select a random number (large) & the test it with "primality test alg" to see if it's a prime number.

2. Select a random integer "a", $1 \leqslant a \leqslant p-2$ & then you compute (α^a mod p) using square-&-multiply algorithm.

generator

3. public-key (p, α, α^a), private-key "a"

perform the reduction (mod p)

Algorithm for ElGamal Encryption / Decryption

summary: B encrypts a message "m" for A, A decrypts

1. Encryption

(a) obtain A's public key (p, α, α^a)

(b) represent the message "m" as an integer in the range $\{0, 1, \ldots, p-1\}$

(c) select a random integer k, $1 \leq k \leq p-2$

(d) compute $\gamma = \alpha^k \pmod p$ and $\delta = m \cdot (\underbrace{\alpha^a}_{\text{single value}})^k \bmod p$

(e) send ciphertext $c = (\gamma, \delta)$

2. Decryption

(a) use the private-key "a" to compute

$$\gamma^{\overbrace{p-1-a}} \pmod p$$

private-key

(b) Recover "m"

$$(\gamma^{-a}) \cdot \underbrace{\delta}_{\text{another part of the cipher text}} \pmod p$$

---

proof of correctness

$$\gamma^{p-1-a} \pmod p \equiv \underbrace{\gamma^{p-1}}_{\substack{\text{Fermat's little Theorem} \\ 1}} \cdot \gamma^{-a} \pmod p \equiv \gamma^{-a} \pmod p$$

we know : $\delta = m \cdot (\alpha^a)^k = m \cdot (\underbrace{\alpha^k}_{\gamma})^a = m \cdot \gamma^a$

part (b)-dec : $\gamma^{-a} \cdot \delta = m \cdot \gamma^a \cdot \gamma^{-a} = \boxed{m}$

inverse

single value $\rightarrow$ $\log_\alpha x = a$ (mod p) — a: private key; α: base

$\boxed{\text{short proof}}$

$$\log_\alpha x = a \pmod{p} \longrightarrow \alpha^a = x \pmod{p}$$

In ElGamal: $(\alpha, p, \alpha^a = x)$ pubic

there is no efficient algorithm to find a # which is the private-key of the sender. that's why ElGamal relies on DL problem.

$\boxed{\text{Example}}$ ElGamal

key gen part

"A" selects $p = 2357$ a generater $\alpha = 2$ of $\mathbb{Z}^*_{2357}$

$a = 1751$

public key $(p = 2357,\ \alpha = 2,\ \alpha^a = \underbrace{1185}_{x})$

Encryption

$m = 2035$

"B" random integer $k = 1520$

$$\gamma = \alpha^k = 2^{1520} \pmod{2357} = 1430$$

$$\delta = \underbrace{2035}_{m} \cdot 1185^{1520} \pmod{2357} = 697$$

cipher tent $(1430, 697)$

Deception

"A"

$$\gamma^{p-1-a} = 1430^{605} \pmod{2357} = 872$$

$\underbrace{2356}_{p-1} - \underbrace{1751}_{a} = 605$

$$m = 872 \cdot \underbrace{697}_{\delta} \pmod{2357} = \underbrace{2035}_{\text{message}}$$