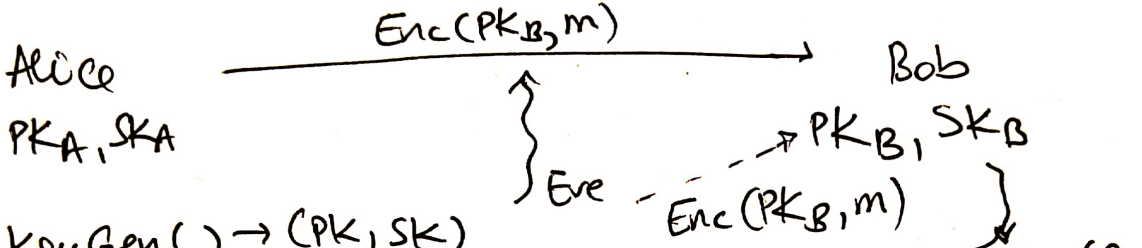
Public-key encryption



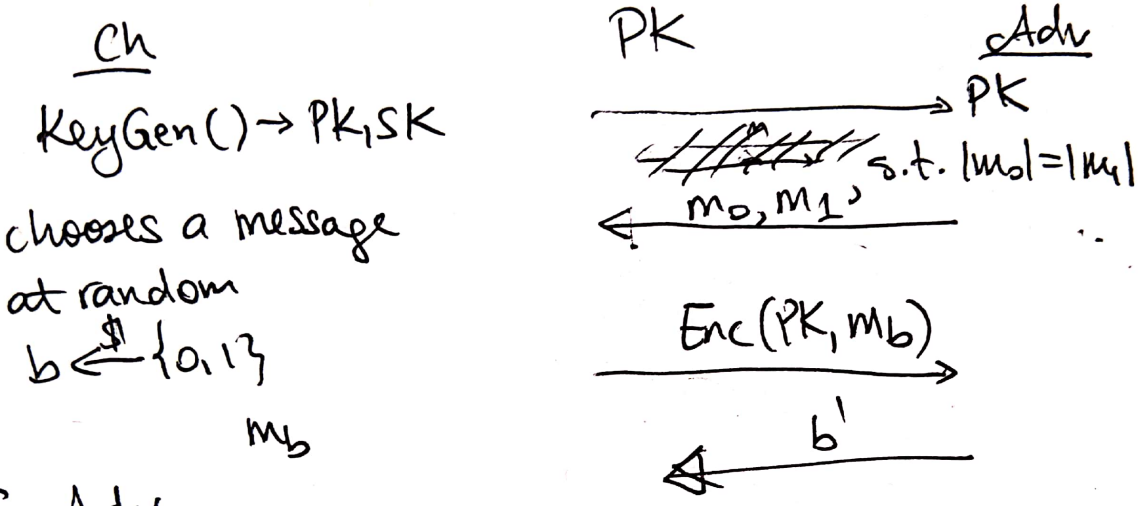
1. KeyGen() → (PK, SK)
2. Enc(PK, m) → C
3. Dec(SK, C) → m

Correctness: $\forall PK, SK \leftarrow KeyGen, \forall m,\ C = Enc(PK, m)$

$Dec(SK, C) = m$

Security: similar in spirit to IND-CPA

Semantic security



$\forall$ Adv,

$Pr[\text{Adv wins } (b' = b)] \leq 1/2 + negl$

ElGamal cryptosystem (1985)

Keygen()

- generate $ a large prime p (2048-bit) $\sim 2^{2048}$
- $g \in [2, p-1]$
- generate $ a secret key k $\in [2, p-2]$

k = SK

- $PK = g^k \bmod p$; $(g, p$ public$)$

Publish PK, keep SK secret

Due to the DLP assumption, cannot guess k

Enc(PK, m): $m \in [1, \ldots, p-1]$

- pick $ $r \in [1, \ldots, p-1]$

$$C = \left(\underbrace{g^r \bmod p}_{C_1} ; \underbrace{m \cdot PK^r \bmod p}_{C_2} \right)$$

Discrete Log Problem must hold (not sufficient)

$(g, p, g^k, C_1, C_2 \approx g, p, g^k, C_1, R)$

Dec(SK, (C_1, C_2)):

$$\frac{C_2}{C_1^k} \bmod p = m$$

k = SK

$$\frac{m \cdot (g^k \bmod p)^r \bmod p \bmod p}{(g^r \bmod p)^k} = m$$

✓ correctness

Padding

varying message sizes

Enc: add padding

Dec: remove padding

↓ padding scheme works only for messages of size < plaintext bits

$m = 1010\underbrace{00}_{\text{pad}}$

remove padding

Using this, you can encrypt 0 with ElGamal

What if I want to encrypt a very long message? GB

Encrypt(PK, very long M):

generate $ sym key K (AES-CTR)

$\underbrace{Enc_{sym}(K, M)}_{C_1}$; $\underbrace{Enc_{pub}(PK, K)}_{C_2}$

Decrypt(SK, $(C_1; C_2)$):

$Dec_{pub}(SK, C_2) \rightarrow K$

$Dec_{sym}(K, C_1) \rightarrow M$