Public-key encryption

Alice $\xrightarrow{Enc(PK_B, m)}$ Bob

PK_A, SK_A $\qquad\qquad$ PK_B, SK_B

Eve $\quad$ $Enc(PK_B, m)$

~~Dec(S~~

$Dec(SK_B, C) = m$

1. $KeyGen() \to (PK, SK)$
2. $Enc(PK, m) \to C$
3. $Dec(SK, C) \to m$

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

Ch	PK	Adv
$\text{KeyGen}() \to PK, SK$	$\xrightarrow{\hspace{3em}}$	PK
	$\xleftarrow{m_0, m_1}$ s.t. $\lvert m_0\rvert = \lvert m_1\rvert$	
chooses a message at random $b \xleftarrow{\$} \{0,1\}$	$\xrightarrow{Enc(PK, m_b)}$	
m_b	$\xleftarrow{b'}$	

$\forall$ Adv,

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

1

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): $m \in [1, \dots, p-1]$

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

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

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, $(C_1; C_2)$):

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

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

✓ Correctness

El-Gamal Encryption Scheme

p - large prime
$g \in [2, p-2]$ - generator
g^b - Bob's public key

m - **Alice** | **Bob** - b

$r \xleftarrow{\$} [1, p-1]$

$$\left(\underbrace{g^r}_{=c_1}, \underbrace{(g^b)^r \cdot m}_{=c_2}\right) \longrightarrow$$

$$\frac{c_2}{(c_1)^b} = \frac{g^{br} \cdot m}{g^{rb}} \equiv m$$

- We know discrete log is hard... how to build encryption from it?

. . . . Embed message in exponent? ie. g^m

→ This hides the message but isn't decryptable

- We want something like $m \cdot K$ where K is only known to Alice & Bob

→ This is just a OTP!

Idea: Use DH Key exch. to create a new K for every ciphertext

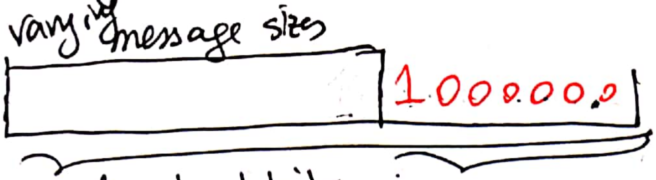
For each encryption:

- $K = g^{br}$ → Alice can compute since she knows r & g^b
  → This is DH Key exch. where g^b is static

- $c = (g^r, K \cdot m)$ → Bob can compute K & decrypt since he knows g^r & b

⟹ El-Gamal Encryption can be thought as a OTP where the key is randomly generated on each encryption via DH Key Exch.

Padding



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

Enc: add padding

Dec: remove padding

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

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)):

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

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

Digital signatures

$$\text{Alice} \xrightarrow{M,\ \text{sign}(SK_A, M) = Sig} \text{Bob}$$

SK_A, PK_A

SK_B, PK_B

$\text{Verify}(PK_A, M, Sig) = \checkmark$

integrity & authenticity in the asymmetric setting

Syntax:

$\text{Keygen}() \to SK, PK$

$\text{Sign}(SK, m) \to Sig$

$\text{Verify}(PK, m, Sig) \to 0/1$

Correctness: $\forall m, SK, PK$

$$\text{Verify}(PK, m, \text{Sign}(SK, m)) = 1 \checkmark$$

Security: EU-CPA

existential unforgeable under CPA...

Ch	PK	everyone knows it	Adv

SK, PK

$$\xleftarrow{\quad M_i \quad}$$

$$\xrightarrow{\quad Sign(SK, M_i) \quad}$$

$$\xleftarrow{\quad M', \ sig \quad}$$

$$M' \neq \{M_i\}$$

Adv wins if $M' \neq \{M_i\}$ and $Verify(PK, M', sig) = yes$

$\forall$ Adv

$Pr[\text{Adv wins}] \le negl$

RSA Signature

$\boxed{\text{Keygen()}}$: pick two random primes p and q of 2048 bits (both 2 mod 3)

$n = p \cdot q$ $\Rightarrow$ $\boxed{PK = n}$

$\phi(n)$ = Euler's totient function
= # of integers ≥ 0 that are $\gcd(\cdot, n) = 1$

$\phi(n) = (p-1)(q-1)$ order of group modulo n

$\forall a, \; a^{\phi(n)} \equiv 1 \bmod n$

Compute d s.t. $\underbrace{3d \equiv 1 \bmod \phi(n)}$

$\Downarrow$

$\boxed{SK = d}$

$\exists r$ s.t.

$3d = r \cdot \phi(n) + 1$

$$\mathrm{sign}(\underset{d}{SK}, m) = \underset{H}{\underline{\mathrm{hash}}}(m)^d \bmod n$$

$$\mathrm{verify}(\underset{n}{PK}, m, \mathrm{sig}): \quad \mathrm{sig}^3 \bmod n \stackrel{?}{=} H(m) \bmod n$$

Correctness:

$$\left(\mathrm{hash}(m)^d\right)^3 \bmod n = \mathrm{hash}(m)^{3d} \bmod n$$

$$= \mathrm{hash}(m)^{r\cdot\phi(n)+1} \bmod n$$

$$= \left(\cancel{\mathrm{hash}(m)^{\phi(n)}}^{\,1}\right)^r \cdot \mathrm{hash}(m) \bmod n$$

$$= \mathrm{hash}(m) \bmod n \quad \checkmark$$

$$\text{Sign}(sk, m) = \underbrace{m^d \bmod n}_{sig}$$

Insecure scheme.

How can you forge?

Signature for 1 is 1

for 0 is 0

$$\text{Sign}(sk, 1) = 1^d \bmod n = 1$$

$$\text{Sign}(sk, 0) = 0^d \bmod n = 0$$

Necessary assumption for security:

No Adv can factor large numbers.

Difficulty of factoring problem

If Adv could factor n

$n \Rightarrow p, q \Rightarrow \phi(n) \Rightarrow d = SK$

Alice give me your PK Bob

PK_B

M $Enc(PK_B, m)$ m

attacker MiTM

PK_{Adv} PK_B

$Enc(PK_{Adv}, M)$ $Enc(PK_B, M)$

M

Trusted directory

random chosen by Alice

nonce
send me PK Bob

PK_B, sign(SK_{TD}, PK_B is Bob's nonce)

Alice
PK_{TD}
hardcoded in her device

MiTM

TD:

name	PK
$\boxed{\text{Bob}}$	~~0x52....~~ 0x28..
MiTM	0xae...

sign(SK_{TD}, PK_{MiTM} is MiTM)

Updating a Key

Assume update happens securely

Replay attack:
Attacker replays old information
(old sig with old PK)