Public-key encryption

\[
\begin{align*}
&Alice & \xrightarrow{Enc(PK_b, m)} & Bob \\
&PK_A, SK_A & & PK_B, SK_B \text{ (Eve)} \\
1. & \text{KeyGen}(\cdot) \rightarrow (PK, SK) \\
2. & Enc(PK, m) \rightarrow C \\
3. & Dec(SK, C) \rightarrow m
\end{align*}
\]

Correctness: \( \forall PK, SK \in \text{KeyGen}, \forall m, C = Enc(PK, m) \Rightarrow Dec(SK, C) = m \)

Security: similar in spirit to IND-CPA
Semantic security

\[ \text{Ch} \]
\[
\text{KeyGen()} \rightarrow \text{PK, SK}
\]

chooses a message at random
\[ b \in \{0, 1\} \]
\[ \text{mb} \]

\[ \text{if Adv,} \]
\[ \Pr [\text{Adv wins } (b' = b)] \leq \frac{1}{2} + \text{negl} \]
ElGamal cryptosystem (1985)

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

$PK = g^K \mod p$ (gip public)

Publish $PK$, keep $SK$ secret

Due to the DLP assumption, cannot guess $K$

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

- pick $\$ r \in [1, p-1]$
$C = (g^r \mod p, m \cdot PK^r \mod p)$

Dec($SK, C_1, C_2$):

\[
C_2 \cdot C_1^K \mod p = m
\]

\[
C_2 \cdot (g^K \mod p)^r \mod p \mod p = m
\]

Correctness

Discrete log problem must hold

(not sufficient)
Ed-Gamal Encryption Scheme

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

\[
m - \text{Alice} \quad r \in [1, p-1] \quad (g^r, (g^b)^r \cdot m) \quad \text{Bob} - b
\]

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

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

... Embed message in exponent? i.e. \( g^m \)

\[ \rightarrow \text{This hides the message but isn't decryptable} \]

- We want something like \( m \cdot k \) where \( k \)
  is only known to Alice & Bob

\[ \rightarrow \text{This is just a OTP!} \]
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 & gb
  → This is DH Key exch. where gb is static

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

→ Ed-Gamad Encryption can be thought as a OTP where the key is randomly generated on each encryption via DH Key Exch.
Padding

Varying message sizes

plaintext bits

\[ 1000000 \]

\[ m = 101000 \]

\[ \text{pad} \]

\[ \text{remove padding} \]

Enc: add padding

Dec: remove padding

\[ \text{padding scheme works only for messages of size < plaintext bits} \]

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 $\text{sym key } K$ (AES-CTR)

    $\text{Enc}_{\text{sym}}(K, M) \cdot \text{Enc}_{\text{pub}}(PK, K)$

Decrypt(SK, (E_1; E_2)):
    Dec_{\text{pub}}(SK, E_2) \rightarrow K
    Dec_{\text{sym}}(K, E_1) \rightarrow M
Digital signatures

Alice \[ M, \text{sign}(SK_A, M) = \text{sig} \] \rightarrow Bob

SK_A, PK_A

integrity & authenticity in the asymmetric setting

Syntax:

\[
\begin{align*}
\text{Keygen}() & \rightarrow SK, PK \\
\text{Sign}(SK, M) & \rightarrow \text{sig} \\
\text{Verify}(PK, m, \text{sig}) & \rightarrow 0 / 1 
\end{align*}
\]

Correctness: \( \forall m, SK, PK \)

\[
\text{Verify}(PK, M, \text{Sign}(SK, M)) = 1 \checkmark
\]
Security: EU-CPA

existential unforgeable under CPA...

\[
\begin{align*}
\text{Ch} & \quad \text{PK} \quad \text{everyone knows it} \quad \text{Adv} \\
\text{SK, PK} & \quad \leftarrow \quad M_i \quad \text{Sign(SK, Mi)} \quad \rightarrow \quad Q \\
& \quad \leftarrow \quad M'_i \quad \text{Sign} \\
& \quad \# \quad \leftarrow \quad \{M_i\} \\
& \text{(Adv wins if } M'_i \neq \{M_i\} \text{ and Verify(PK, } M', \text{sig)} = \text{yes)} \\
& \text{Adv} \quad \neg \text{Adv} \\
\text{Pr[Adv wins]} & \leq \text{negl}
\end{align*}
\]
RSA Signature

**Keygen()**: pick two random primes
   \( p \) and \( q \) of 2048 bits \((\text{both } 2 \text{ mod } 3)\)
   \( n = p \cdot q = \text{PK} = n \)

\( \phi(n) = \text{Euler's totient function} \)
   \( = \# \text{ of integers } \geq 0 \text{ that are } \gcd(\cdot, n) = 1 \)

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

\( \forall a, a^{\phi(n)} \equiv 1 \mod n \)

Compute \( d \) s.t. \( 3d \equiv 1 \mod \phi(n) \)

**SK = d**

\( \exists r \) s.t.
   \( 3d = r \cdot \phi(n) + 1 \)
\[ \text{Sign}(SK, m) = \frac{\text{hash}(m)^d}{H} \mod n \]

\[ \text{Verify}(PK, m, \text{Sig}) : \quad \text{Sig}^3 \mod n = H(m) \mod n \]

**Correctness:**

\[ (\text{hash}(m)^d)^3 \mod n = \text{hash}(m)^{3d} \mod n \]

\[ = \text{hash}(m)^{r \cdot \phi(n) + 1} \mod n \]

\[ = (\text{hash}(m)^{\phi(n)})^r \cdot \text{hash}(m) \mod n \]

\[ = \text{hash}(m) \mod n \]
\[ \text{sign(}sk, m) = \frac{md}{\text{sig}} \mod n \]

How can you forge?

Signature for 1 is 1

\[ \text{sign(}sk, 1) = 1^d \mod n = 1 \]

\[ \text{sign(}sk, 0) = 0^d \mod n = 0 \]
Necessary assumption for security:

No Adv can factor large numbers.

Difficulty of factoring problem

If Adv could factor \( n \)

\[ n \rightarrow p, q \quad \Rightarrow \quad \phi(n) \quad \Rightarrow \quad d = SK \]
Alice gives Bob her public key (PKb). Bob encrypts message (m) with PKb and sends it to Alice. However, an attacker (MitM) intercepts the message and replaces it with their own Enc(PKA_{MitM}, m). Alice encrypts this with her private key (PK_A) and sends it to MitM. MitM substitutes the original message with their own Enc(PKA_{MitM}, m). Alice decrypts with PK_A and receives a corrupted message M.

Trusted directory:
- Random domain chosen by Alice
- Send me PK Bob's ID: {name: PK} - sign(KTD, PK) (Bob's key, nonce, MitM)
- PK_{KTD} hardcoded in her device
- MitM

Updating a key:
- Assume update happens securely

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