Encryption from Diffie-Hellman Assumption

IND-CPA secure Constructions: El-Gamal Encryption

- Setup Same as Diffie-Hellman Key Exchange with prime p and element g.

- Keygen
  $sk := x \xleftarrow{\$} \mathbb{Z}_p$
  $pk :=$
  g^x mod p
  return (pk, sk)

- Encrypt(pk, m)
  $y \xleftarrow{\$} \mathbb{Z}_p$
  $C_1 = g^y$ mod p
  $C_2 =$
  $m \cdot pk^y$ mod p
  return (C_1, C_2)

- Decrypt(sk, C)
  $x = sk$
  $(C_1, C_2) = C$
  $t = C_1^x$ mod p
  $m =$
  $C_2 \cdot t^{-1}$ mod p
  return m

El-Gamal Encryption:Correctness

- Keygen
  $sk := x \xleftarrow{\$} \mathbb{Z}_p$
  $pk :=$
  g^x mod p
  return (pk, sk)

- Encrypt(pk, m)
  $y \xleftarrow{\$} \mathbb{Z}_p$
  $C_1 = g^y$ mod p
  $C_2 =$
  $m \cdot pk^y$ mod p
  return (C_1, C_2)

- Decrypt(sk, C)
  $x = sk$
  $(C_1, C_2) = C$
  $t = C_1^x$ mod p
  $m =$
  $C_2 \cdot t^{-1}$ mod p
  return m

Correctness

For all $m \in G, y \in \mathbb{Z}_p$ it holds that

$$\begin{aligned}
\text{Decrypt}\left(sk, \text{Encrypt}(pk, m)\right) &= \text{Decrypt}\left(x, \text{Encrypt}(g^x \text{ mod } p, m)\right) \\
&= \text{Decrypt}\left(x, (g^y \text{ mod } p, m \cdot g^{xy} \text{ mod } p)\right) \\
&= m \cdot g^{xy} \cdot (g^{-xy}) \text{ mod } p \\
&= m
\end{aligned}$$

The idea of El-Gamal

El-Gamal ciphertext

Encrypt$(pk = g^x \bmod p, m) \rightarrow (g^y \bmod p, m \cdot g^{xy} \bmod p)$

The idea of El-Gamal

El-Gamal ciphertext

$\mathsf{Encrypt}(pk = g^x \bmod p, m) \to (g^y \bmod p, m \cdot g^{xy} \bmod p)$

- Encrypt using a "session-key": One-time pad using $g^{xy} \bmod p$

The idea of El-Gamal

El-Gamal ciphertext

$\mathsf{Encrypt}(pk = g^x \text{ mod } p, m) \rightarrow (g^y \text{ mod } p, m \cdot g^{xy} \text{ mod } p)$

- Encrypt using a “session-key”: One-time pad using g^{xy} mod p
- Decryption requires key-derivation: Compute g^{xy} mod p from

The idea of El-Gamal

El-Gamal ciphertext

$\mathsf{Encrypt}(pk = g^x \text{ mod } p, m) \rightarrow (g^y \text{ mod } p, m \cdot g^{xy} \text{ mod } p)$

- Encrypt using a "session-key": One-time pad using g^{xy} mod p
- Decryption requires key-derivation: Compute g^{xy} mod p from
  - Secret key x.
  - First part of ciphertext g^y mod p

The idea of El-Gamal

El-Gamal ciphertext

$\mathsf{Encrypt}(pk = g^x \bmod p, m) \to (g^y \bmod p, m \cdot g^{xy} \bmod p)$

- Encrypt using a "session-key": One-time pad using $g^{xy} \bmod p$
- Decryption requires key-derivation: Compute $g^{xy} \bmod p$ from
  - Secret key x.
  - First part of ciphertext $g^y \bmod p$
- Encrypt again? Use another session-key by choosing new y' and computing $g^{xy'} \bmod p$.

Key Exchange Revisited

- Key Exchange allows to agree on a secret key, derived from public-keys.
- Communication happens in multiple sessions.
- What to do when each session requires a new secret key?

Key Exchange Revisited

- Key Exchange allows to agree on a secret key, derived from public-keys.
- Communication happens in multiple sessions.
- What to do when each session requires a new secret key?

Key Encapsulation Mechanism

Algorithms

A Key Encapsulation Mechanism consists of three algorithms $KEM = (\mathsf{Keygen}, \mathsf{Encap}, \mathsf{Decap})$

- $(pk, sk) \leftarrow \mathsf{Keygen}(1^n)$: Keygen generates the keypair (pk, sk). pk is the public-key. sk is the secret-key.
- $c \leftarrow \mathsf{Encap}(pk)$: The randomized encapsulation algorithm Encap takes the public-key pk as input, and outputs a "key-ciphertext" pair (c, k).
- $k \leftarrow \mathsf{Decap}(sk, c)$: The decapsulation algorithm Decap takes the secret key sk and a ciphertext c. The output is the corresponding session key k.

Properties

The session key is uniquely determined from the “ciphertext”

Correctness

For all $(pk, sk) \leftarrow \mathsf{Keygen}(1^n)$, it should hold that

$$Prob[\mathsf{Decap}(sk, c) = k \mid (c, k) \leftarrow \mathsf{Encap}(pk)] = 1$$

Properties

The session key is uniquely determined from the “ciphertext”

Correctness

For all $(pk, sk) \leftarrow \mathsf{Keygen}(1^n)$, it should hold that

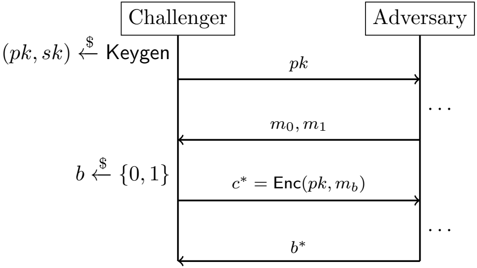
$$Prob[\mathsf{Decap}(sk, c) = k \mid (c, k) \leftarrow \mathsf{Encap}(pk)] = 1$$

Note: Modern systems sometimes allow little correctness errors.

Security

Indistinguishability under Chosen Ciphertext Attack

The adversary has no information about the current key even if they know any other Ciphertext-key pair.



Output 1 if $b = b^*$

Output 0 if $b \neq b^*$

IND-CCA secure Constructions: Hashed El-Gamal

- Setup A cyclic group G of order p; a generator of the group g.

- Keygen
  $sk := x \xleftarrow{\$} \mathbb{Z}_p$
  $pk := g^x \bmod p$
  return (pk, sk)

- Encap(pk)
  $y \xleftarrow{\$} \mathbb{Z}_p$
  $c = g^y \bmod p$
  $k = H(pk^y \bmod p)$
  return (c, k)

- Decap(sk, c)
  $x = sk$
  $k = H(c^x \bmod p)$
  return k

IND-CCA secure Constructions: Hashed El-Gamal

- Setup A cyclic group G of order p; a generator of the group g.

- Keygen

  $sk := x \xleftarrow{\$} \mathbb{Z}_p$
  $pk := g^x \bmod p$
  return (pk, sk)

- $\mathsf{Encap}(pk)$

  $y \xleftarrow{\$} \mathbb{Z}_p$
  $c = g^y \bmod p$
  $k = H(pk^y \bmod p)$
  return (c, k)

- $\mathsf{Decap}(sk, c)$

  $x = sk$
  $k = H(c^x \bmod p)$
  return k

correctness

$$
\begin{aligned}
k = H(pk^y \bmod p) &= H\left((g^x)^y \bmod p\right) \\
&= H(g^{xy} \bmod p)
\end{aligned}
$$

IND-CCA secure Constructions: Hashed El-Gamal

- Setup A cyclic group G of order p; a generator of the group g.

- Keygen
  $sk := x \xleftarrow{\$} \mathbb{Z}_p$
  $pk :=$
  g^x mod p
  return (pk, sk)

- Encap(pk)
  $y \xleftarrow{\$} \mathbb{Z}_p$
  $c = g^y$ mod p
  $k =$
  $H(pk^y$ mod $p)$
  return (c, k)

- Decap(sk, c)
  $x = sk$
  $k =$
  $H(c^x$ mod $p)$
  return k

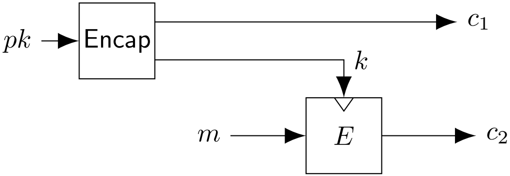
correctness

$$
\begin{aligned}
k = H(pk^y \bmod p) &= H\left((g^x)^y \bmod p\right) \\
&= H(g^{xy} \bmod p)
\end{aligned}
$$

$$k = H(c^x \bmod p) = H\left((g^y)^x \bmod p\right) = H(g^{xy} \bmod p)$$

Hybrid Encryption: KEM+SE $\implies$ PKE

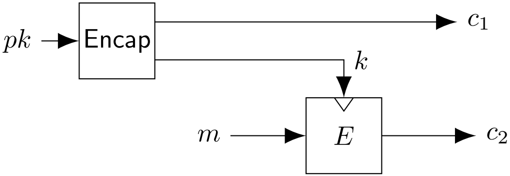
Derive secret key using $\underline{\mathsf{Encap}}$ and apply <u>symmetric encryption</u>.



- $\underline{\mathsf{Encrypt}(pk, m)}$
  $(c_1, k) = \mathsf{Encap}(pk)$
  $c_2 = E_k(m)$ return $c = (c_1, c_2)$

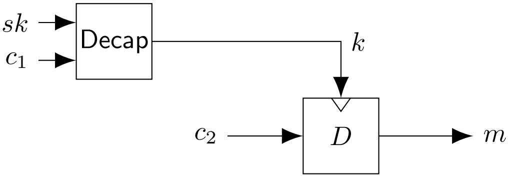
Hybrid Encryption: KEM+SE $\implies$ PKE

Derive secret key using Encap and apply symmetric encryption.



- Encrypt(pk, m)
  $(c_1, k) =$ Encap(pk)
  $c_2 = E_k(m)$ return $c = (c_1, c_2)$

Hybrid Encryption: KEM+SE $\implies$ PKE



- $\underline{\text{Decrypt}(sk, c)}$
  Parse c as (c_1, c_2)
  $k = \text{Decap}(sk, c_1)$
  $m = D_k(c_2)$

Hybrid Encryption: KEM+SE $\Longrightarrow$ PKE

Security

IND-CCA KEM $+$ IND-CCA SE $\Longrightarrow$ IND-CCA PKE

Take Home

- ElGamal Encryption: Diffie-Hellman with one time pad, IND-CPA secure.
- Key Encapsulation Mechanism: Derive session key from fixed public key.
- Hybrid Encryption: KEM+SE $\implies$ PKE.
- IND-CCA KEM $+$ IND-CCA SE $\implies$ IND-CCA PKE.