Public-key encryption

Introduction to cryptology

Bruno Grenet

M1 INFO, MOSIG & AM

Université Grenoble Alpes – IM²AG

https://membres-ljk.imag.fr/Bruno.Grenet/IntroCrypto.html
https://membres-ljk.imag.fr/Pierre.Karpman/tea.html

Introduction

Symmetric (or *private key*) encryption

- Alice and Bob share a common key k
- Alice wants to send m to Bob:
  1. Alice computes $c \leftarrow \mathrm{Enc}_k(m)$
  2. Alice sends c to Bob
  3. Bob computes $m' \leftarrow \mathrm{Dec}_k(c)$

and if all goes well: $m = m'$

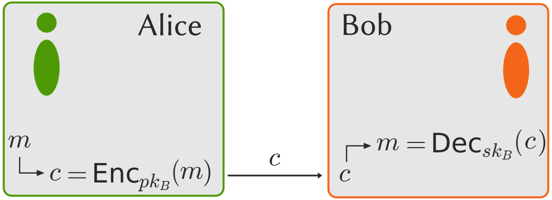
Key exchange

- Alice and Bob must agree on a common key k.
- Diffie-Hellman protocol based on cyclic groups

Public-key (*a.k.a* **a**symmetric) cryptography: no prior key exchange!

Contents

Principle



Encryption	Alice encrypts m with Bob's public key: $c \leftarrow \mathrm{Enc}_{pk_B}(m)$
Decryption	Bob decrypts c with his private key: $m' \leftarrow \mathrm{Dec}_{sk_B}(c)$
Correctness	if $m = m'$
Security	if an adversary cannot compute m, knowing both c **and** pk_B

Loosely based on: J. Katz, Y. Lindell. Introduction to modern cryptography. 3rd ed, CRC Press, 2021.

Formalization of public-key encryption

Definition

A public-key encryption scheme is given by 3 algorithms:

$\mathrm{Gen}_n()$	returns a pair of keys (pk, sk) where n is the *security parameter*
$\mathrm{Enc}_{pk}(m)$	returns a ciphertext c for a message $m \in \mathcal{M}_{pk}$
$\mathrm{Dec}_{sk}(c)$	returns a message m or an error

Correctness: for all $(pk, sk) \leftarrow \mathrm{Gen}_n()$ and all $c \leftarrow \mathrm{Enc}_{pk}(m)$, $\mathrm{Dec}_{sk}(c) = m$

Remarks

- pk is the *public key* and sk the *private (or secret) key*.
- The public key defines the message space $\mathcal{M}_{pk}$
  - require a mapping from $\{0,1\}^*$ to $\mathcal{M}_{pk}$
  - often obvious
- The security parameter n sets the keys lengths — *often implicit*
- Gen is implicit for symetric encryption — *e.g*: return $k \twoheadleftarrow \{0,1\}^n$

CPA-security

CPA indistinguishability game

Challenger: $(pk, sk) \leftarrow \mathsf{Gen}()$
Adversary: given pk, produces m_0, $m_1 \in \mathcal{M}_{pk}$ of the same size
Challenger: $b \twoheadleftarrow \{0,1\}$; $c \leftarrow \mathsf{Enc}_{pk}(m_b)$
Adversary: given c, returns a bit b'; *success* if $b = b'$

Advantages

- $\mathsf{Adv}_{\mathsf{Enc}}^{\mathsf{IND-CPA}}(A) = \left|\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]\right| = |2\Pr[\textsc{success}] - 1|$
- $\mathsf{Adv}_{\mathsf{Enc}}^{\mathsf{IND-CPA}}(t) = \max_{A_t} \mathsf{Adv}_{\mathsf{Enc}}^{\mathsf{IND-CPA}}(A_t)$ where A_t has running time $\leq t$

Remarks

- Extremely similar with IND-CPA for symmetric encryption
  - No *oracle access* to $\mathsf{Enc}_{pk}(\cdot)$ *The public key is… public!*
- $\mathsf{Enc}_{pk}(\cdot)$ must be randomized: Why?
- No *perfectly secret* public-key encryption

CCA-security

CCA indistinguishability game

Challenger: $(pk, sk) \leftarrow \mathsf{Gen}()$

Adversary: *has oracle access to* $\mathsf{Dec}_{sk}(\cdot)$ *during the whole experiment*

given pk, produces $m_0, m_1 \in \mathcal{M}_{pk}$ of same size

Challenger: $b \twoheadleftarrow \{0,1\}$; $c \leftarrow \mathsf{Enc}_{pk}(m_b)$

Adversary: given c, returns a bit b'; *success* if $b' = b$ *not allowed to ask* $\mathsf{Dec}_{sk}(c)$*!*

Advantages

- $\mathsf{Adv}_{\mathsf{Enc}}^{\mathsf{IND-CCA}}(A) = \left|\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]\right| = |2\Pr[\text{success}] - 1|$
- $\mathsf{Adv}_{\mathsf{Enc}}^{\mathsf{IND-CCA}}(q, t) = \max_{A_t} \mathsf{Adv}_{\mathsf{Enc}}^{\mathsf{IND-CCA}}(A_{q,t})$ where $A_{q,t}$ has running time $\leq t$ and makes $\leq q$ queries to $\mathsf{Dec}_{sk}(\cdot)$

Remarks

- The security notion needed in practice
- Implies *non-malleability*: Knowing $c \leftarrow \mathsf{Enc}_{pk}(m)$ but not m, it is *hard* to compute c' such that $\mathsf{Dec}_{sk}(c') = f(m)$ for some chosen $f(\cdot)$

What about *multiple* encryptions?

Two (equivalent) questions

- What happens if we re-use the same public key several times?
- Can we encrypt arbitrary long messages?

Reminder in the symmetric case

- Block ciphers $\rightarrow$ fixed-length deterministic encryption
- Modes of operations $\rightarrow$ variable-length randomized encryption

Security for multiple encryption

- The building block is already randomized
- No modes of operations $\rightarrow$ only ECB $\quad \mathsf{Enc}_{pk}(m_1) \| \cdots \| \mathsf{Enc}_{pk}(m_B)$
- Formally: IND-CPA $\Rightarrow$ IND-CPA for multiple encryptions

Encryption: public-key or symmetric + key exchange?

Advantages of symmetric encryption + key exchange

- Symmetric encryption usually lighter than public-key encryption
  - Reduced communications
  - Reduced computations

Advantages of public-key encryption

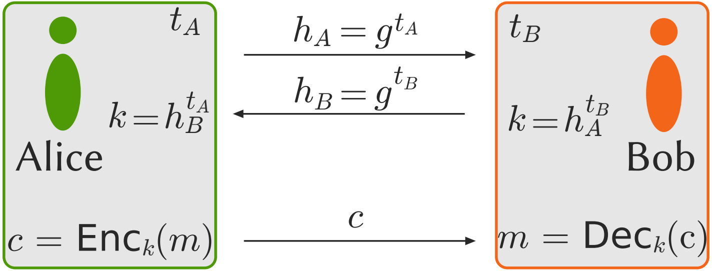
- Only one protocol to manage $\rightarrow$ fewer points of weakness
- Each user has only one private key to keep in the long run
- Works in asynchronous situations

Hybrid encryption

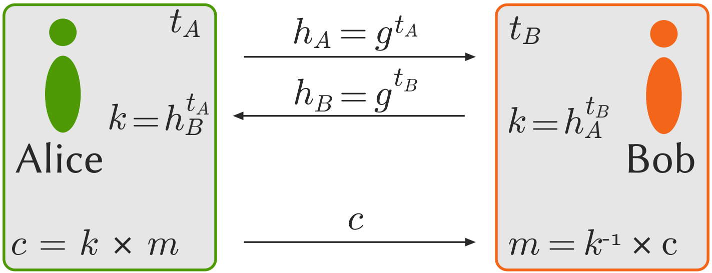
- General idea
  - Encrypt the message m with a symmetric key $k \rightarrow c$
  - Encrypt the key k with a public key $pk \rightarrow c'$
  - Send c and $c' \rightarrow$ decryption in the obvious manner
- More general framework: we can do *better* than encrypting the key k
  - KEM/DEM Paradigm

Contents

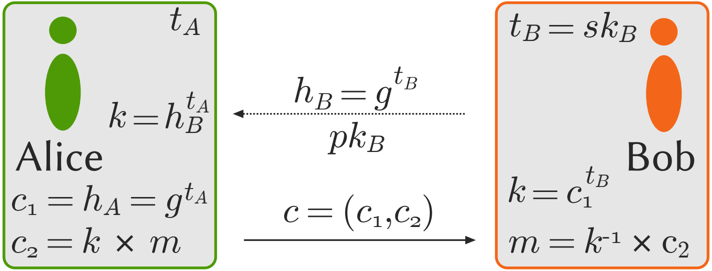
From Diffie-Hellman to ElGamal



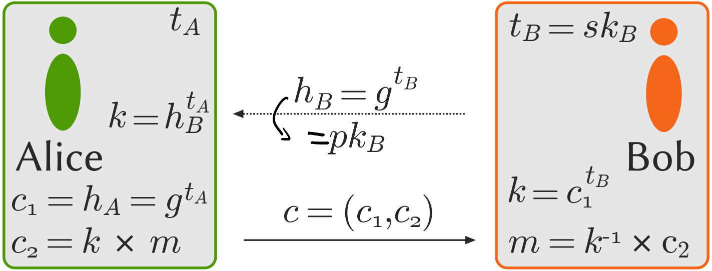
From Diffie-Hellman to ElGamal



From Diffie-Hellman to ElGamal



From Diffie-Hellman to ElGamal



Question

Prove that $\mathrm{Enc}_k(m) = k \times m$ provides a secure encryption scheme

Remark

Several senders can all use Bob's public key:
security for a single encryption $\Rightarrow$ security for multiple encryptions

ElGamal encryption scheme

Construction

Public: a cyclic group G of order $q \simeq 2^n$ with generator g

Gen(): 1. $x \twoheadleftarrow \{0, \ldots, q-1\}$
2. $h \leftarrow g^x$
3. Return $pk = h$ and $sk = x$ $\quad (\mathcal{M}_{pk} = G)$

$\mathrm{Enc}_{pk}(m)$: 1. $y \twoheadleftarrow \{0, \ldots, q-1\}$
2. $c_1 \leftarrow g^y$; $c_2 \leftarrow h^y \cdot m$
3. Return $c = (c_1, c_2)$

h

$\mathrm{Dec}_{sk}(c_1, c_2)$: 1. Return $\hat{m} = c_2 \cdot c_1^{-x}$

Correction

$$\hat{m} = c_2 \cdot c_1^{-x} = h^y \cdot m \cdot (g^y)^{-x} = g^{xy} \cdot m \cdot g^{-xy} = m$$

Group multiplication for encryption

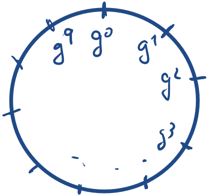
Lemma

Let G be a cyclic group of order q and generator g and $z \twoheadleftarrow \{0, \dots, q-1\}$ (uniformly):

(1) ▶ g^z is a uniform element of G

(2) ▶ for any $m \in G$, $g^z \cdot m$ is uniform in G

(1) $\Leftrightarrow$ $\Pr_z[g^z = h] = 1/q$ for all $h \in G$: this is true since for all h, there is a unique $z \in \{0, \dots, q-1\}$ such that $g^z = h$



(2) If h is uniform in G, then $h \cdot m$ is uniform in G.

Let $\ell \in G$. $\Pr[h \cdot m = \ell] = \Pr[h = m^{-1} \cdot \ell] = 1/q$

Security proof

Theorem

If DDH holds for G, ElGamal encryption scheme is IND-CPA secure. More precisely, $\text{Adv}^{\text{IND-CPA}}_{\text{ElGamal(G)}}(t) \leq 2 \cdot \text{Adv}^{\text{DDH}}_{G}(t)$ for all t.

$\text{Exp}^{\text{DDH}}_{G}(A)$:

C: simulates the DH protocol

$b \twoheadleftarrow \{0,1\}$ $x_1, x_2, x_3 \twoheadleftarrow \{0, \dots, q-1\}$

Sends $h_1 = g^{x_1}$, $h_2 = g^{x_2}$, $h_3 = \begin{cases} g^{x_1 x_2} & \text{if } b=1 \\ g^{x_3} & \text{if } b=0 \end{cases}$

A: Outputs $\hat{b}$

$\text{Exp}^{\text{IND-CPA}}_{EG(G)}(A')$:

A': Sends m_0, m_1

C: $b' \twoheadleftarrow \{0,1\}$ and $c \leftarrow \text{Enc}_{pk}(m_{b'})$

A': Outputs $\hat{b}'$

↳ Assume A' has advantage α'

We build A for $\text{Exp}^{\text{DDH}}_{G}$.

A receives $h_1 = g^{x_1}$, $h_2 = g^{x_2}$, $h_3 = \begin{cases} g^{x_1 x_2} & \text{if } b=1 \\ g^{x_3} & \text{if } b=0 \end{cases}$

1. A calls A' to get m_0, m_1
2. A chooses $b' \twoheadleftarrow \{0,1\}$ and $c \leftarrow \text{Enc}_{pk}(m_{b'})$
3. A asks A' for a bit $\hat{b}'$
4. A outputs $\begin{cases} \hat{b} = 1 & \text{if } \hat{b}' = b' \\ \hat{b} = 0 & \text{otherwise.} \end{cases}$

$\text{Adv}^{\text{DDH}}_{G}(A) = \left| \underbrace{\Pr[\hat{b} = 1 \mid b = 1]}_{\frac{1}{2}(\alpha' + 1)} - \underbrace{\Pr[\hat{b} = 1 \mid b = 0]}_{1/2} \right|$

Additional remarks

Choice of the group G

- The order q must be prime, for DDH
- Several choices (subgroup of $(\mathbb{Z}/p\mathbb{Z})^\times, \ldots$)
  - different security levels
  - standardization by NIST and other agencies

$\log p$	$\log q$	security
2048	224	112
3072	256	128
7680	384	192
15360	512	256

Message space G?

- Solution 1: bijection between G and $\{0,1\}^\ell$ *for some G*
- Solution 2: ElGamal-based KEM + key derivation function

CCA (in)security

- If $(c_1, c_2) \leftarrow \text{Enc}_{pk}(m)$, then $\text{Dec}_{sk}(c_1, m' \cdot c_2) = m' \cdot c_2 \cdot c_1^{-sk} = m' \cdot m$
  $\Rightarrow$ ElGamal encryption scheme is *malleable*, hence not CCA secure
- CCA-secure variants exist, mainly using hybrid encryption

Contents

Introduction

Observation

- Public-key encryption scheme designed for small messages
- Block-by-block encryption possible…
- … but expensive — large *ciphertext expansion*

Use of key exchange

1. Agree on a shared key k
2. Use symmetric encryption with k

The idea of hybrid encryption

Sender: encrypts the message with a key $k \to c$
encrypts the key k with the public key of the receiver — *encapsulated key*

Receiver: decrypts first the encapsulated key with its secret key $\to k$
decrypts c using $k \to m$

The KEM/DEM paradigm

Definition

A Key Encapsulation Mechanism (KEM) is given by three algorithms:

$$\begin{aligned} \mathsf{Gen}_n()&: \text{ produces a pair } (pk, sk) \\ \mathsf{Encaps}_{pk}()&: \text{ produces a pair } (c, k) \\ \mathsf{Decaps}_{sk}(c)&: \text{ returns } k \end{aligned}$$

Usage

To send m using public-key pk:

1. $(c, k) \leftarrow \mathsf{Encaps}_{pk}()$ — *key encapsulation*
2. $c' \leftarrow \mathsf{Enc}_k(m)$ (with symmetric encryption) — *data encapsulation*

Security notions

- Definitions of IND-CPA / IND-CCA security for KEMs
- IND-CPA KEM and symmetric encryption $\Rightarrow$ IND-CPA public-key encryption
- Ditto for IND-CCA

Generic construction from public-key encryption scheme

Definition

Given: Public-key encryption scheme (Enc, Dec)

$\mathrm{Encaps}_{pk}()$:
1. $k \twoheadleftarrow \{0,1\}^n$
2. $c \leftarrow \mathrm{Enc}_{pk}(k)$

$\mathrm{Decaps}_{sk}(c)$:
1. $k \leftarrow \mathrm{Dec}_{sk}(c)$

Security

- If the public-key scheme is IND-CPA secure, the KEM too
- Ditto with IND-CCA security

Comments

- Using ElGamal for instance, must encode k in the group G
- Not the only nor best solution:
  - We need: from pk, produce c and k such that k can be recovered from sk and c
  - We don't need: c to be an actual encryption of k using pk

DDH-based KEM

Construction

Public: a cyclic group G of order q generated by g

Gen():
1. $x \twoheadleftarrow \{0, \ldots, q-1\}$
2. $h \leftarrow g^x$
3. $H \leftarrow$ some hash function from G to $\{0,1\}^\ell$
4. return $pk = (h, H)$ and $sk = (x, H)$

$\mathrm{Encaps}_{pk}()$:
1. $y \twoheadleftarrow \{0, \ldots, q-1\}$
2. return $c \leftarrow g^y$ and $k \leftarrow H(h^y)$

$\mathrm{Decaps}_{sk}(c)$:
1. return $k \leftarrow H(c^x)$

Correction

$$c^x = g^{yx} = (g^x)^y = h^y \Rightarrow H(c^x) = H(h^y)$$

Security (admitted)

- If DDH holds for G and H is *regular*, the KEM is IND-CPA secure
- If CDH holds for G and H is a random oracle, the KEM is IND-CPA secure

Conclusion

Public-key encryption schemes

- Usually heavier than symmetric encryption schemes
- Good solution: use hybrid encryption KEM/DEM paradigm
- Key management can be tricky $\to$ *public key infrastructures*

ElGamal encryption scheme

- Basic idea very close to Diffie-Hellman key exchange protocol
- Requires other tools to make it IND-CCA secure
- Security based on DDH or CDH assumption

Other protocols

- Variant of the DDH based KEM is standardized as DHIES/ECIES
  - IND-CPA or IND-CCA security proofs under suitable assumptions
- Cramer & Shoup protocol: IND-CCA security under DDH assumption
- Other unrelated protocols using completely different assumptions RSA, LWE, …