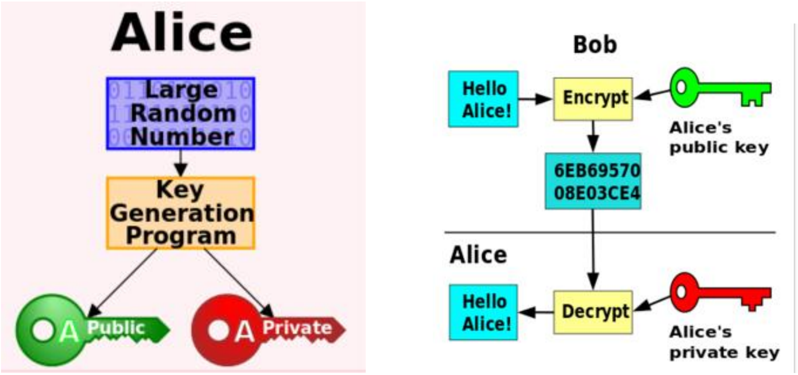
Public-Key Encryption (公钥加密)

Sheng Zhong Yuan Zhang

Computer Science and Technology Department
Nanjing University

A public-key encryption scenario

When Alice sends a secret message to Bob using a public-key encryption scheme, three major operations are involved: *key generation*, *encryption*, and *decryption*.



(a) Key generation

(b) Encryption and decryption

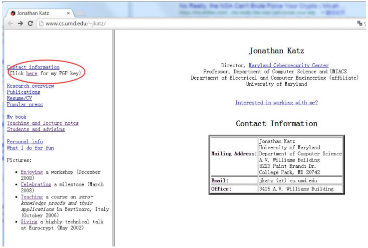
图 1: A public-key encryption usage scenario (Figs. from Wikipeida)

One detail remains unclear

Q: How does Bob know Alice's public key?

A: Alice sends it to Bob via an authenticated channel between Alice and Bob, or Alice publicizes its public key to everyone.

(a) Alipay sends its public key via an authenticated channel



(b) Prof. J. Katz publicizes his PGP key on his mainpage

图 2: Two public-key distribution manners

Formal definition of a public-key encryption scheme

Formally, we can define a public-key encryption scheme as follows.

DEFINITION 11.1

A public-key encryption scheme is a *triple of PPT algorithms* (Gen, Enc, Dec) such that:

1. The **key-generation algorithm** Gen takes as input the security parameter 1^n and outputs a pair of keys (pk, sk).
2. The **encryption algorithm** Enc takes input a public key pk and a message m from some message space (that may depend on pk). It outputs a ciphertext c, and we write this as $c \leftarrow Enc_{pk}(m)$.
3. The **decryption algorithm** Dec takes input a private key sk and a ciphertext c, outputs a message m or a special symbol $\perp$ denoting failure. We write this as $m := Dec_{sk}(c)$.

It is required that, except possibly with negligible probability over (pk, sk) output by $Gen(1^n)$, we have $Dec_{sk}(Enc_{pk}(m)) = m$ for any legal message m.

Security against an eavesdropper

We begin examine the security of a public-key encryption by considering an eavesdropping adversary first.

The eavesdropping indistinguishability experiment $\mathrm{PubK}^{eav}_{\mathcal{A},\Pi}(n)$:

1. $Gen(1^n)$ is run to obtain keys (pk, sk).
2. Adversary $\mathcal{A}$ is given pk, and then outputs a pair of equal-length messages m_0, m_1 in the message space.
3. A uniform bit $b \in \{0, 1\}$ is chosen, and then a ciphertext $c \leftarrow Enc_{pk}(m_b)$ is computed and given to $\mathcal{A}$. We call c the **challenge ciphertext**.
4. $\mathcal{A}$ outputs a bit b'. The output of the experiment is 1 if $b' = b$, and 0 otherwise. If $b' = b$ we say that $\mathcal{A}$ **succeeds**.

Security against an eavesdropper

Given $\text{PubK}^{eav}_{\mathcal{A},\Pi}(n)$, we can define a public-key encryption scheme's security against an eavesdropping adversary:

DEFINITION 11.2

A public-key encryption scheme $\Pi = (Gen, Enc, Dec)$ has indistinguishable encryptions in the presence of an eavesdropper if for all PPT adversaries $\mathcal{A}$ there is a negligible function *negl* such that

$$Pr[\text{PubK}^{eav}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + negl(n).$$

Indistinguishable encryption against the CPA adversary

Different from private-key encryption, for public-key encryption we have

PROPOSITION 11.3

If a public-key encryption scheme has indistinguishable encryptions in the presence of an eavesdropper, it is CPA-secure, i.e. has indistinguishable encryptions against the CPA adversary.

Q: Can you see why this is true?

A: Because 1) $\mathcal{A}$ is given *pk* (used for encryption), 2)*Enc* is publicly known. Thus, $\mathcal{A}$ can encrypt any message by itself, which is equivalent to have an encrypting oracle.

CPA-security is equivalent to indistinguishable multiple encryption

Recall that we use the indistinguishable multiple encryptions to model the security of using the same key or key pair for encrypting multiple messages. Similar to private-key encryption, we have the following result.

THEOREM 11.6

If public-key encryption scheme Π is CPA-secure, then it also has indistinguishable **multiple** encryptions.

CPA-secure fixed-length encryption implies CPA-secure arbitrary-length encryption

Let $\Pi = (Gen, Enc, Dec)$ be a CPA-secure public-key encryption for messages with a fixed length l, then we can construct a new public-key encryption $\Pi' = (Gen, Enc', Dec')$ that has message space $\{0,1\}^*$ as follows:

$$Enc'_{pk}(m) = Enc_{pk}(m_1), \ldots, Enc_{pk}(m_k),$$

where $k = \lceil \frac{|m|}{l} \rceil$, and $m_1 \ldots m_k$ equals m processed by some padding operation that extends its length to be a multiple of l.

CPA-secure fixed-length encryption implies CPA-secure arbitrary-length encryption

Let $\Pi = (Gen, Enc, Dec)$ be a CPA-secure public-key encryption for messages with a fixed length l, then we can construct a new public-key encryption $\Pi' = (Gen, Enc', Dec')$ that has message space $\{0,1\}^*$ as follows:

$$Enc'_{pk}(m) = Enc_{pk}(m_1), \ldots, Enc_{pk}(m_k),$$

where $k = \lceil \frac{|m|}{l} \rceil$, and $m_1 \ldots m_k$ equals m processed by some padding operation that extends its length to be a multiple of l.

CLAIM 11.7

If Π is CPA-secure, then so is Π'.

- This is true since we can view an encryption of message m with Π' as an encryption of k messages with Π, and Thm 11.6 tells us Π has indistinguishable multiple encryptions.

Security against chosen-ciphertext attacks

We use the following experiment to define CCA-security:

The CCA indistinguishability experiment $\text{PubK}^{cca}_{\mathcal{A},\Pi}(n)$:

1. $Gen(1^n)$ is run to obtain keys (pk,sk).
2. The adversary $\mathcal{A}$ is given pk and access to a decryption oracle $Dec_{sk}(\cdot)$. It outputs a pair of legal messages m_0, m_1 of the same length.
3. A uniform bit b is chosen, then a challenge ciphertext $c \leftarrow Enc_{pk}(m_b)$ is computed and given to $\mathcal{A}$.
4. $\mathcal{A}$ continues to interact with $Dec_{sk}(\cdot)$, but can not request a decryption of c itself.
5. $\mathcal{A}$ outputs a bit b'.

Security against chosen-ciphertext attacks

We use the following experiment to define CCA-security:

The CCA indistinguishability experiment $\mathsf{PubK}^{cca}_{\mathcal{A},\Pi}(n)$:

1. $Gen(1^n)$ is run to obtain keys (pk, sk).
2. The adversary $\mathcal{A}$ is given pk and access to a decryption oracle $Dec_{sk}(\cdot)$. It outputs a pair of legal messages m_0, m_1 of the same length.
3. A uniform bit b is chosen, then a challenge ciphertext $c \leftarrow Enc_{pk}(m_b)$ is computed and given to $\mathcal{A}$.
4. $\mathcal{A}$ continues to interact with $Dec_{sk}(\cdot)$, but can not request a decryption of c itself.
5. $\mathcal{A}$ outputs a bit b'.
6. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

CCA-secure public-key encryption

With the CCA indistinguishability experiment $\mathsf{PubK}^{cca}_{\mathcal{A},\Pi}(n)$, we can define the CCA-secure public-key encryption as follows:

DEFINITION 11.8

A public-key encryption scheme $\Pi = (Gen, Enc, Dec)$ has indistinguishable encryptions under a chosen-ciphertext attack (or is CCA-secure) if for all PPT adversaries $\mathcal{A}$ there exists a negligible function *negl* such that

$$Pr[PubK^{cca}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + negl(n).$$

CCA-security is equivalent to indistinguishable encryptions

- Similar to CPA-secure public-key encryption, we have:
  If Π is CCA-secure, Π has also indistinguishable (multiple) encryptions under chosen ciphertext attacks.
- However, different from CPA-secure public-key encryption, extending a CCA-secure fixed-length encryption scheme to a arbitrary-length encryption as in Claim 11.7 DOSE NOT yield a CCA-secure encryption for arbitrary-length messages.
  - Why not? Cause $\mathcal{A}$ can query the decryption oracle with a "new" ciphertext which reorders the blocks of the challenging ciphertext.

Comparisons between Private-key encryption and Public-key encryption

Compared with private-key encryption,

- The advantages of public-key encryption include:
  - Key distribution is much easier.
  - More convenient for "open systems" where the numbers and identities of potential senders need not be known in advance.

- The disadvantages of public-key encryption include:
  - Public-key encryption is slower than private key encryption (e.g. roughly 2 to 3 orders of magnitude slower.).
  - Public-key encryption generally has greater ciphertext expansion (thus you may need store/transmit more bits.).

What is a hybrid encryption?

To utilize the merits and avoid the disadvantages of private-key encryptions and public-key encryptions, a mixed scheme called "**hybrid encryption**" can be adopted.

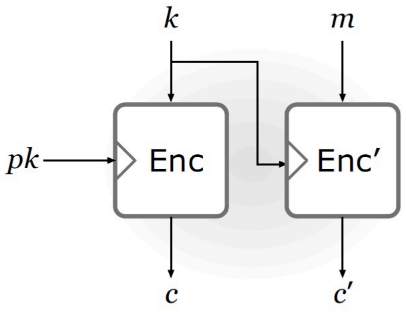


图 3: A example of a hybrid encryption, where *Enc* and *Enc′* denote the encryption of a public-key encryption scheme and a private-key encryption scheme resp.

The advantages of hybrid encryptions

The advantages of hybrid encryptions include:

- easier key distribution (inherited from the public-key encryption)
- good efficiency and small ciphertext expansion (inherited from private-key encryption)
- offers good security guarantees when selecting Enc and Enc' properly.

The key-encapsulation mechanism (KEM)

A **key-encapsulation mechanism** (KEM) is a public-key primitive that efficiently generates an encryption key for the private-key encryption k and its ciphertext c in a hybrid encryption scheme. Accordingly, the private-key encryption scheme is called a **data-encapsulation mechanism** (DEM) here.

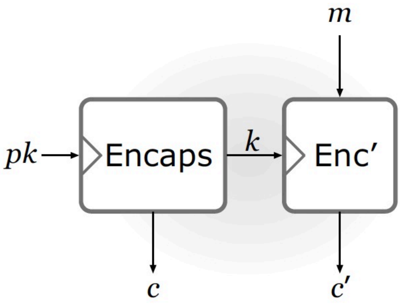


图 4: Hybrid encryption using the KEM/DEM approach

An overview of El Gamal Encryption

El Gamal encryption scheme is a famous, and widely used public encryption scheme:

- based on the Diffie-Hellman key-exchange protocol.
- proposed by Taher El Gamal in 1985.
- used a lot in secure multiparty computations due to its homomorphic encryption

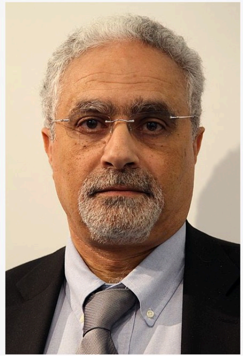


图 5: Taher A. Elgamal, Egyptian cryptographer (pic from wikipedia)

El Gamal Encryption

The El Gamal encryption scheme is described in the following:

CONSTRUCTION 11.16: The El Gamal encryption scheme

Let $\mathcal{G}$ be a PPT algorithm that takes as input 1^n outputs a description of a cyclic group $\mathbb{G}$, its order q with $||q|| = n$, and a generator g of $\mathbb{G}$:

- **Gen**: on input 1^n, run $\mathcal{G}$ to obtain $(\mathbb{G}, q, g)$. Then choose a uniform $x \in \mathbb{Z}_q$ and compute $h := g^x$. The public key is $\langle \mathbb{G}, q, g, h \rangle$ and the private key is $\langle \mathbb{G}, q, g, x \rangle$. The message space is $\mathbb{G}$.
- **Enc**: on input a public key $pk = \langle \mathbb{G}, q, g, h \rangle$ and a message $m \in \mathbb{G}$, choose a uniform $y \in \mathbb{Z}_q$ and output the ciphertext
$$\langle g^y, h^y \cdot m \rangle.$$
- **Dec**: on input a private key $sk = \langle \mathbb{G}, q, g, x \rangle$, and a ciphertext $\langle c_1, c_2 \rangle$, output
$$\hat{m} := c_2 / c_1^x.$$

An example of El Gamal encryption

Example: Let $\mathbb{G} = \mathbb{Z}_5^*$ and $g = 2$, it easy to verify this is a cyclic group with order $q = 4$.
Say $x = 3$ is generated by uniformly chosen from $\mathbb{Z}_4$, then the public key is

$$pk = \langle p, q, g, h \rangle = \langle 5, 4, 2, [2^3 \mod 5] \rangle = \langle 5, 4, 2, 3 \rangle.$$

When encrypting a message $m \in \mathbb{Z}_5^*$, say $y = 2$ is chosen, then the ciphertext equals

$$\langle c_1, c_2 \rangle = \langle [2^2 \mod 5], [3^2 m \mod 5] \rangle = \langle 4, [4m \mod 5] \rangle.$$

When decrypting the ciphertext, we have

$$\hat{m} = [4m \mod 5]/[4^3 \mod 5] = m.$$

The security of El Gamal encryption

Regarding the security of El Gamal encryption, we have the following theorem:

THEOREM 11.18

If the DDH problem is hard relative to $\mathcal{G}$, then the El Gamal encryption scheme is CPA-secure.

Proof: see page 402-403 in the textbook.

The security of El Gamal encryption

Q: Is El Gamal encryption CCA-secure?

A: NO. Because El Gamal is malleable, i.e. given an encryption c of some unknown message m, it is possible to come up with a ciphertext c' that is an encryption of a message m' such that m' is related to m in some known way.

For example: say $\langle c_1, c_2 \rangle$ is the ciphertext of m using the El Gamal encryption. It is easy to verify that $\langle c_1^2, c_2^2 \rangle$ is an encryption of m^2.

Some background knowledge about RSA encryption

RSA is probably the best-known public-key cryptosystem:

- It is one of the first practical public-key cryptosystems.
- It is invented by Ronald Rivest, Adi Shamir, and Leonard Adleman in 1977.

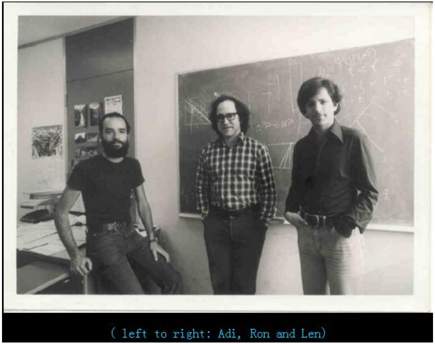


图 6: The inventors of RSA in 1978

Some background knowledge about RSA encryption

RSA is probably the best-known public-key cryptosystem:

- It is one of the first practical public-key cryptosystems.
- It is invented by Ronald Rivest, Adi Shamir, and Leonard Adleman in 1977.

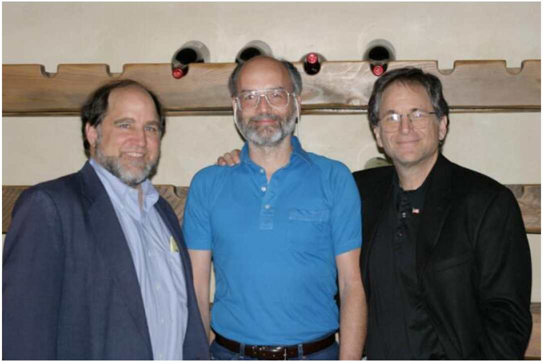


图 6: The inventors of RSA in 1978, and in 2003.

An overview of RSA encryption

The RSA encryption is based on the following trapdoor one-way function:

- "one-way": given public known integers e and N, it is easy to compute

$$c = [m^e \mod N],$$

  for any $m \in \mathbb{Z}_N^*$. , but it is generally hard to compute m from c.
- "trapdoor information": If you know the factorization of N, computing m from c becomes easy.

The RSA key generation

Specifically, the RSA key generation algorithm runs as follows:

ALGORITHM 11.25 RSA key generation GenRSA

- **Input**: Security parameter 1^n.
- **Output**: N, e, d
  1. $(N, p, q) \leftarrow GenModulus(1^n)$.
  2. $\phi(N) = (p-1)(q-1)$.
  3. choose $e > 1$ such that $gcd(e, \phi(N)) = 1$.
  4. compute $d := [e^{-1} \mod \phi(N)]$.
  5. **Return** N, e, d.

The plain RSA encryption scheme

Given *GenRSA*, the (plain) RSA encryption scheme is as follows:

CONSTRUCTION 11.26 The plain RSA encryption scheme:

- *Gen*: on input 1^n, run $GenRSA(1^n)$ to obtain N, e and d. The public key is $\langle N, e \rangle$, and the private key is $\langle N, d \rangle$.
- *Enc*: on input a public key $pk = \langle N, e \rangle$ and a message $m \in \mathbb{Z}_N^*$, compute the ciphertext

$$c := [m^e \mod N].$$

- *Dec*: on input a private key $sk = \langle N, d \rangle$ and a ciphertext $c \in \mathbb{Z}_N^*$, compute the message

$$m := [c^d \mod N].$$

Two examples

Example 1: Say $N = 3 \times 5$, and $e = 3$, what are the public key and private key? What is the message space? What's the encryption of 2?

Example 2 (11.27): Say GenRSA outputs $(N, e, d) = (391, 3, 235)$, and we have a message $m = 158 \in \mathbb{Z}^*_{391}$.
Its ciphertext $c = [158^3 \mod 391] = 295$.
To decrypt c, we compute $[295^{235} \mod 391] = 158$.

Why plain RSA is NOT secure (1) - the need for randomness

Unfortunately, plain RSA is NOT CPA-secure (or DOSE NOT have indistinguishable encryptions against an eavesdropper).

THEOREM 11.4

No deterministic public-key encryption scheme is CPA-secure.

- If the scheme is deterministic, the adversary can win the experiment by simply generating the ciphertexts of m_0 and m_1, and then comparing them with the challenge ciphertext.

Why plain RSA is NOT secure (2) - the prerequisite of the RSA assupmption

Also, we know the security of RSA encryption is based on the RSA assumption, which requires a uniform ciphertext in $\mathbb{Z}_N^*$.

The RSA experiment RSA-inv$_{\mathcal{A},GenRSA}(n)$

1. Run **GenRSA**(1^n) to obtain (N, e, d), where N is a product of two n-bit primes, $gcd(e, \phi(N)) = 1$ and $ed = 1 \mod \phi(N)$.
2. Choose a uniform $y \in \mathbb{Z}_N^*$.
3. $\mathcal{A}$ is given N, e, y, and outputs $x \in \mathbb{Z}_N^*$.
4. The output of the experiment is defined to be 1 if $x^e = y \mod N$, and 0 otherwise.

DEFINITION 8.46

The RSA problem is hard relative to GenRSA if for all PPT algorithms $\mathcal{A}$ there exists a negligible function *negl* such that

$$Pr[\text{RSA-inv}_{\mathcal{A},GenRSA}(n) = 1] \leq negl(n).$$

The RSA assumption is that there exists a **GenRSA** algorithm relative to which the RSA problem is hard.

However, when m is not sampled uniformly at random in $\mathbb{Z}_N^*$, c is not uniformly random neither. Thus the RSA assumption does not apply for plain RSA in this case.

Padded RSA

To overcome the weakness of plain RSA, one method is to use "**Padded RSA**".

- Before encrypting, the message is "randomly padded" (or randomly mapped) to a long message in $\mathbb{Z}_N^*$. The long, random message is fed into the encryption algorithm and generates the ciphertext.
- To decrypt, the ciphertext is fed into the decryption algorithm, and the original message is recovered from the output using a "de-padding" operation (This requires the mapping is reversible.)
- The security depends on critically on the specific mapping that is used.

RSA PKCS #1 v1.5.

The **RSA Laboratories Public-Key Cryptography Standard (PKCS) #1 v1.5.** is a family of standards published by RSA Laboratories.

- RSA PKCS #1 v1.5. utilizes a variant of padded RSA encryption which maps a message m to a k-byte message

$$0x00||0x02||r||0x00||m,$$

  where k is the byte-length of N, and r is a randomly generated string with none of its bytes equal to $0x00$.
- If we force r to be half of the length of N, then it is reasonable to conjecture that the encryption scheme is CPA-secure. (However, we cannot prove it based on the RSA assumption.)

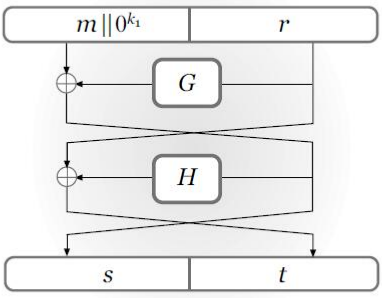
OAEP and RSA-OAEP

The optimal asymmetric encryption padding (OAEP) is a padding scheme that is now often used together with RSA encryption.

- It is proposed by M. Bellare and P. Rogaway in 1994.
- It is standardized in RSA PKCS #1 v2.0.

OAEP and RSA-OAEP

- It uses a two-round Feistel network to construct the mapping.



- Recovered message has to pass the formation verification, otherwise the decryption is rejected.
- It can be proved that RSA encryption with OAEP padding (RSA-OAEP) can achieve CCA-security in the random oracle model.