Cryptography

Public Key Cryptography & RSA

M S Vilkhu

02 Nov 2024(C1/C3/C5)

Learning Objectives

- Present an overview of the basic principles of public-key cryptosystems.
- Explain the two distinct uses of public-key cryptosystems.
- List and explain the **requirements** for a public-key cryptosystem.
- Present an **overview** of the **RSA** algorithm.
- Understand the **timing attack**.
- Summarize the relevant issues related to the complexity of algorithms.

Topics

Principles of Public-Key Cryptosystems

Public-Key Cryptosystems

Applications for Public-Key Cryptosystems

Requirements for Public-Key Cryptography

Public-Key **Cryptanalysis**

The RSA Algorithm

Description of the Algorithm

Computational Aspects

The Security of RSA

Introduction

- The development of public-key, or asymmetric, cryptography is the **greatest and perhaps the only true revolution in the entire history of cryptography**.
- From its earliest beginnings to modern times, virtually all cryptographic systems have been based on the **elementary tools of substitution and permutation**.
- After millennia of working with algorithms that could be calculated by hand, a major advance in symmetric cryptography occurred with the development of the rotor encryption/decryption machine. The **electromechanical** rotor enabled the development of fiendishly complex cipher systems.
- With the availability of **computers**, even more complex systems were devised, the most prominent of which was the Lucifer effort at IBM that culminated in the Data Encryption Standard (DES).
- But both rotor machines and DES, although representing significant advances, still relied on the **bread-and-butter tools of substitution and permutation**.

Introduction

- Public-key cryptography provides a **radical departure** from all that has gone before.
- For one thing, **public-key algorithms** are **based on mathematical functions** rather than on **substitution and permutation**.
- More important, public-key cryptography is **asymmetric**, involving the **use of two separate keys**, in contrast to symmetric encryption, which uses only one key.
- The use of two keys has **profound consequences** in the areas of
  - **confidentiality**,
  - **key distribution,** and
  - **authentication**,

Common Misconceptions

Common misconceptions concerning public-key encryption.

1. One such misconception is that **public-key encryption is more secure** from **cryptanalysis than is symmetric encryption**.

- **Reality** In fact, the security of **any encryption scheme** depends on the **length of the key** and the **computational work** involved in **breaking a cipher**.
- There is **nothing in principle about either symmetric or public-key** encryption that makes **one superior to another** from the **point of view** of **resisting cryptanalysis**.

2. A **second** misconception is that **public-key encryption is a general-purpose** technique that has **made symmetric encryption obsolete**.

- **Reality** On the contrary, **because of the computational overhead** of current public-key encryption schemes, there **seems no foreseeable likelihood that symmetric encryption will be abandoned**

Approach

Overview of public-key cryptography –

- **First**, we look at its **conceptual framework**.
  - Interestingly, the concept for this technique was developed and published before it was shown to be practical to adopt it.
- **Next**, we examine **the RSA algorithm**, which is the most important encryption/decryption algorithm that has been shown to be feasible for public-key encryption.

Approach

- Much of the theory of **public-key cryptosystems** is based on **number theory.**
- If one is **prepared to accept the results** given now, then an understanding of number theory is **not strictly necessary**.
- However, to gain a full appreciation of public-key algorithms, some understanding of number theory is required.

Principles of Public-key cryptosystems

- The **concept of public-key cryptography evolved** from an attempt to att**ack two of the most difficult problems** associated with **symmetric encryption**.
- **The first problem** is that of **key distribution**,
- The **second problem** that Diffie pondered, and one that was apparently unrelated to the first, was that of **digital signatures**.

Principles of Public-key cryptosystems

- **The first problem** is that of **key distribution**,
- key distribution under symmetric encryption requires either

  (1) that two communicants **already share a key**, which somehow has been distributed to them; or

  (2) the use of a **key distribution center**.

- **Whitfield Diffie**, one of the discoverers of public-key encryption (along with **Martin Hellman**, both at Stanford University at the time), reasoned that this **second requirement negated the very essence of cryptography**: the **ability to maintain total secrecy over your own communication**.

> As Diffie put it [DIFF88],
>
> "what good would it do after all to **develop impenetrable cryptosystems**, if their **users were forced to share their keys with a KDC** that could be **compromised by either burglary or subpoena**?"

Principles of Public-key cryptosystems

- The **second problem** that Diffie pondered, and one that was apparently unrelated to the first, was that of **digital signatures**.
- If the **use of cryptography was to become widespread**, not just in military situations but for commercial and private purposes, then **electronic messages and documents** would **need the equivalent of signatures** used in paper documents.
- That is, could a **method be devised** that would stipulate, to the satisfaction of all parties, that a **digital message had been sent by a particular person**?
- This is a somewhat **broader requirement** than that of **authentication**

Principles of Public-key cryptosystems

- **Diffie and Hellman** achieved an **astounding breakthrough in 19**76 a, by coming up with a method that **addressed both problems** and was radically different from all previous approaches to cryptography

Asymmetric algorithms rely on **one key for encryption** and a **different but related key** for **decryption**.

Cryptography

Public Key Cryptography & RSA

M S Vilkhu

02 Nov 2024(C1/C3/C5)

04 Nov 2024(C1/C3)

-08 Nov 2024(C5)

Public-Key Cryptosystems

- **Asymmetric algorithms** rely on **one key for encryption** and a **different but related key** for **decryption**.
- These algorithms have the following important characteristic.
  - It is **computationally infeasible** to determine the **decryption key** given only knowledge of the cryptographic algorithm and the encryption key.

Computational infeasibility means that, with current technology and known algorithms, it would take an **unrealistic amount of time** (often measured in thousands or millions of years) to calculate the decryption key from the encryption key or encrypted data. This is due to the sheer number of possible combinations that would need to be tested (a concept known as a **brute-force attack**).

Public-Key Cryptosystems

In addition, some algorithms, such as RSA, also exhibit the following characteristic.

- **Either of the two related keys** can be used for **encryption**, with the other used for **decryption**.

Public-Key Cryptosystems

characteristic.

- **computationally infeasible** to determine the **decryption key**
- **Either of the two related keys** can be used for **encryption**, with the other used for **decryption**.

Public-Key Cryptosystems

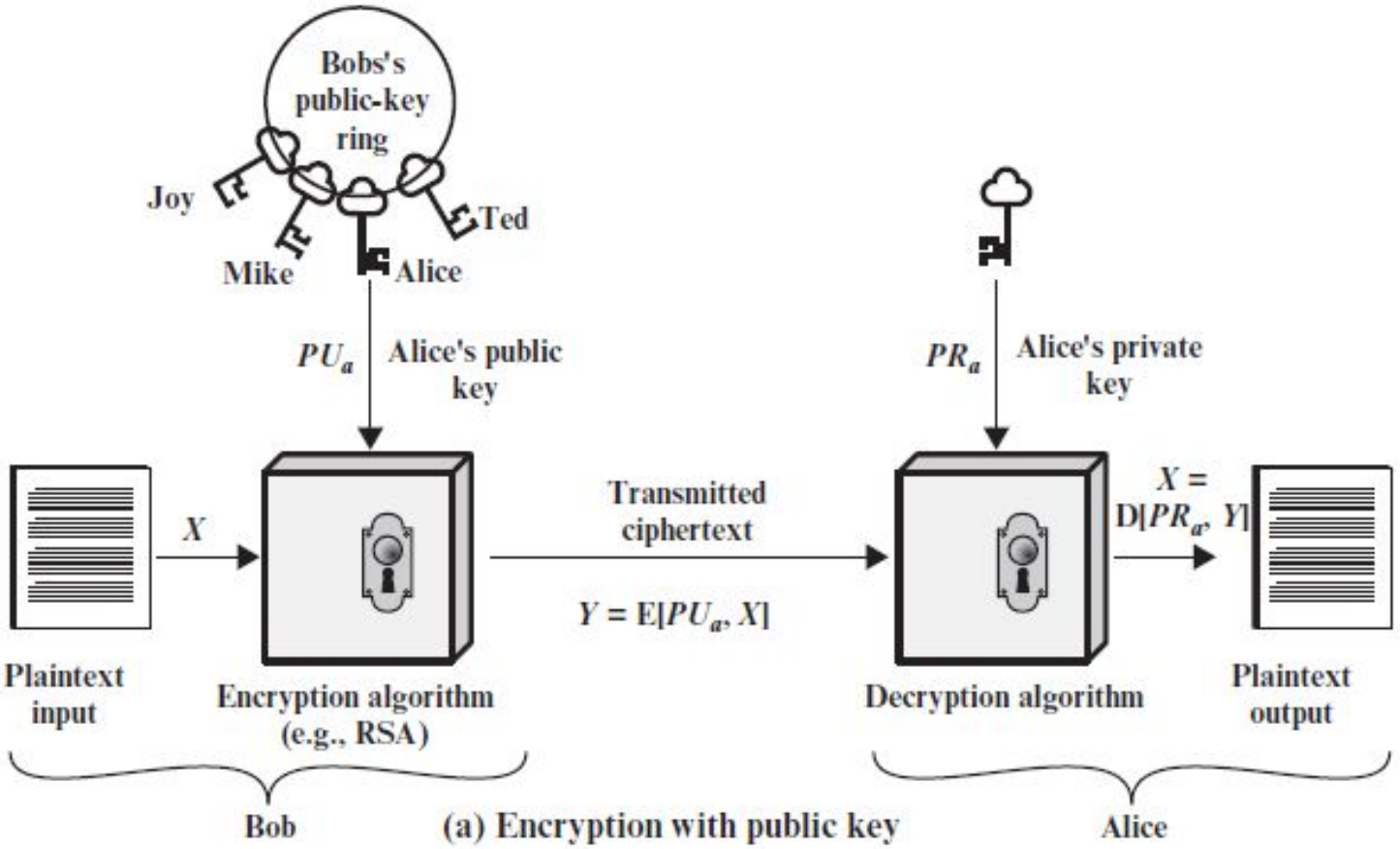
In addition, some algorithms, such as RSA, also exhibit the following characteristic.

- A **public-key encryption** scheme **has six ingredients**
  - **Plaintext**: This is the readable message or data that is fed into the algorithm as input.

    Encryption algorithm:
  - **Encryption algorithm** performs various transformations on the plaintext.
  - **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as Input.
  - **Ciphertext:** This is the encrypted message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
  - **Decryption algorithm**: This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

Public-Key Cryptosystems

- Encryption with **public key**



(a) Encryption with public key

Public-Key Cryptosystems

- Encryption with **private key**

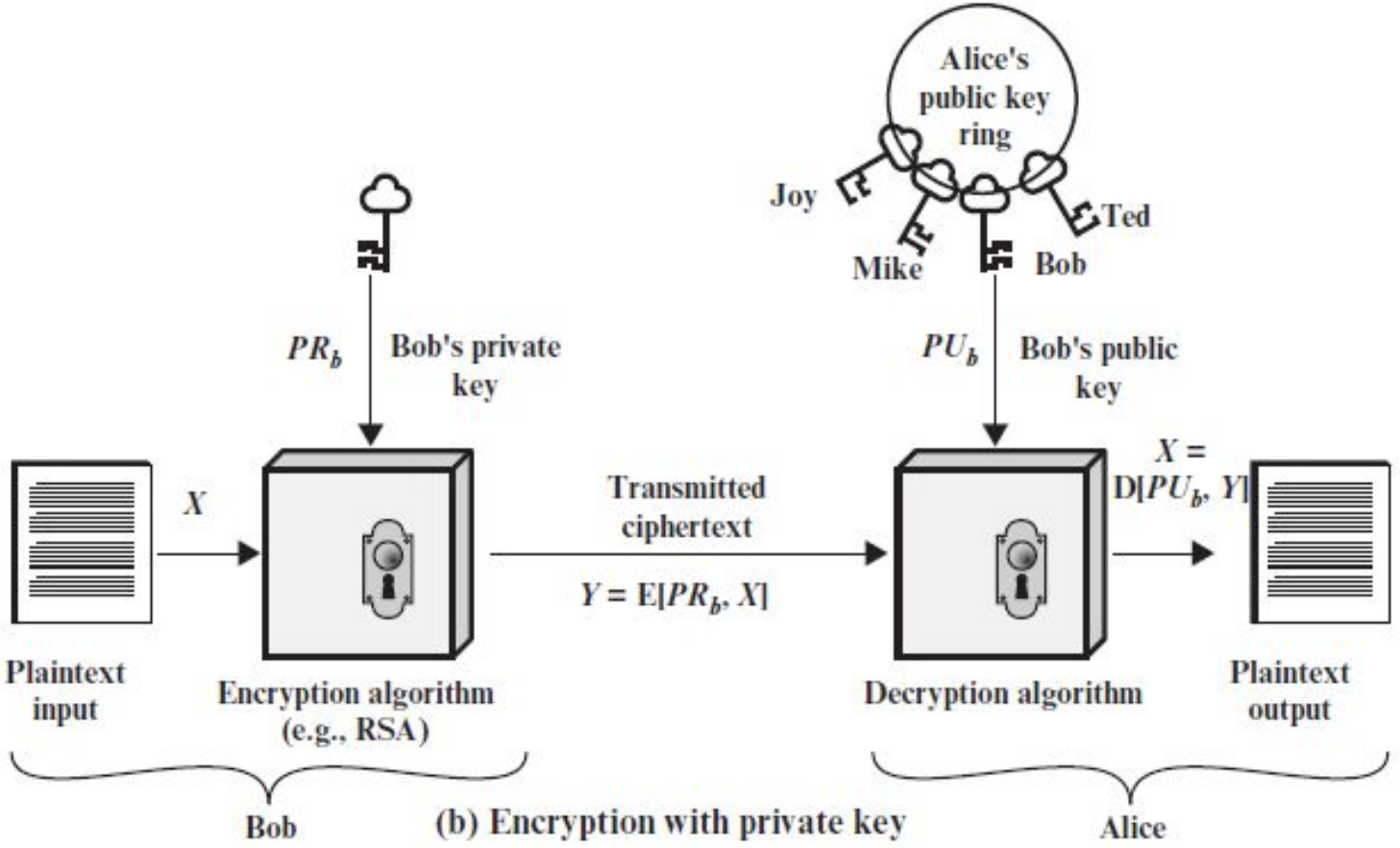


Figure 9.1 Public-Key Cryptography

Essential steps

The **essential steps** are the following.

1. **Each user generates a pair of keys** to be used for the encryption and decryption of messages.

2. Each user places **one of the two keys in a public register** or other accessible file. This is the **public key**. The **companion key is kept private**. Each user maintains a collection of public keys obtained from others.

3. If **Bob** wishes to send a **confidential** message to **Alice**, Bob encrypts the message using **Alice's public key**.

4. When **Alice** receives the message, she **decrypts it using her private key**. No other **recipient can decrypt** the message because only Alice knows Alice's private key.

As long as a user's private key remains protected and secret, incoming communication is secure

At any time, a system can **change its private key** and **publish the companion public key** to replace its old public key

Comparison

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if the key is kept secret. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and a related algorithm for decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if one of the keys is kept secret. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Process

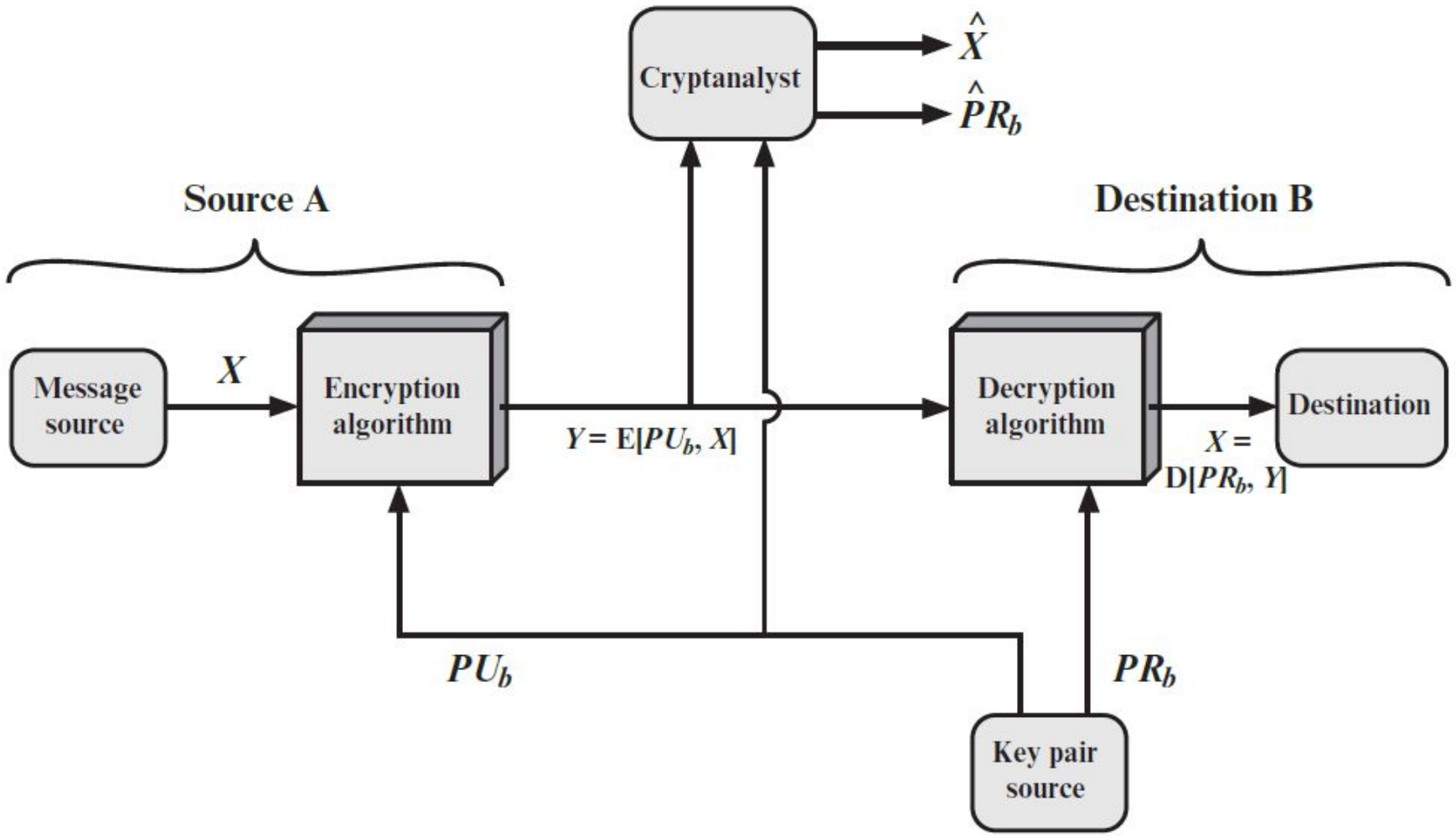
- There is some source **A** that produces a message in plaintext, **X = [X1, X2, …, X_M]**. The M elements of X are letters in some finite alphabet. The message is intended for destination **B**.
- B generates a related pair of keys: a **public key, PU_b,** and a **private key, PR_b** known only to B, whereas PU_b is publicly available and therefore accessible by A.
- With the message X and the encryption key PU_b as input, A forms the ciphertext Y = [Y1, Y2, …, Y_N]:

$$Y = E(PU_b, X)$$

- The intended receiver, in possession of the **matching private key**, is able to invert the transformation:

$$X = D(PR_b, Y)$$

End-to-end system

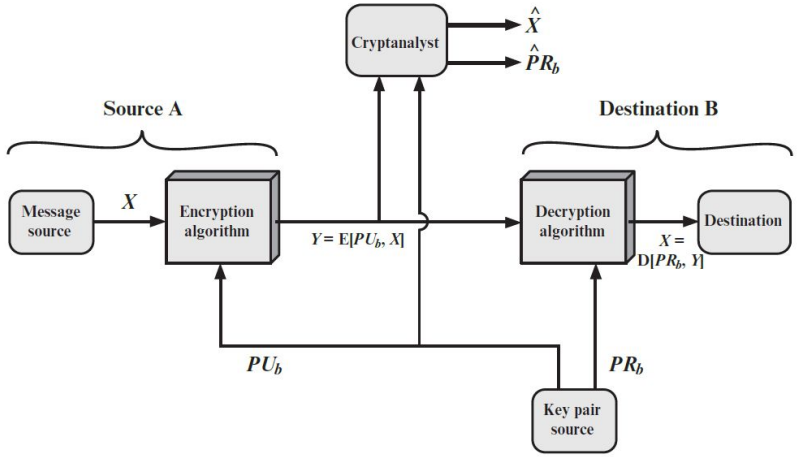


Public-Key Cryptosystems

- the **two related keys** can be **used for encryption**, with the **other** being used for **decryption**.
- This enables a **rather different cryptographic scheme** to be implemented.
- Whereas the scheme illustrated in Figure provides **confidentiality**,
- Figures show the use of public-key encryption to provide **authentication**:

$$Y = \mathrm{E}(PRa, X)$$
$$X = \mathrm{D}(PUa, Y)$$

Confidentiality



Authentication

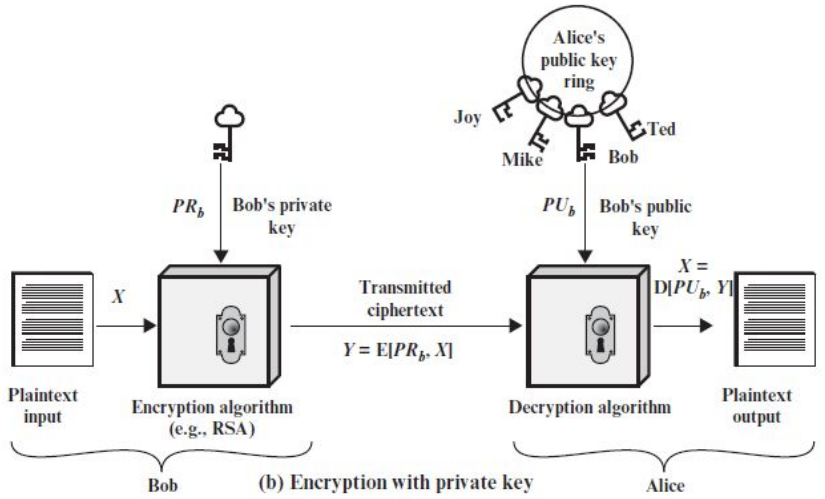
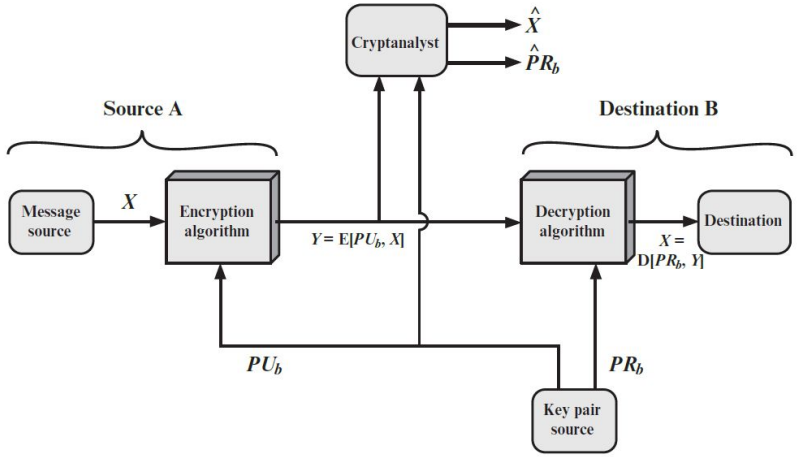


Figure 9.1 Public-Key Cryptography

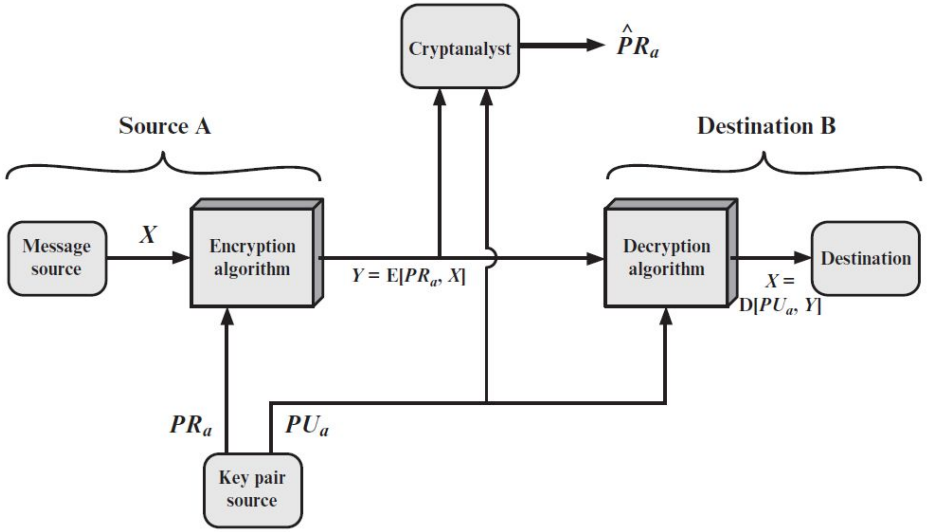
Public-Key Cryptosystems

- use of public-key encryption to provide **authentication**:

Confidentiality



Authentication



Authentication

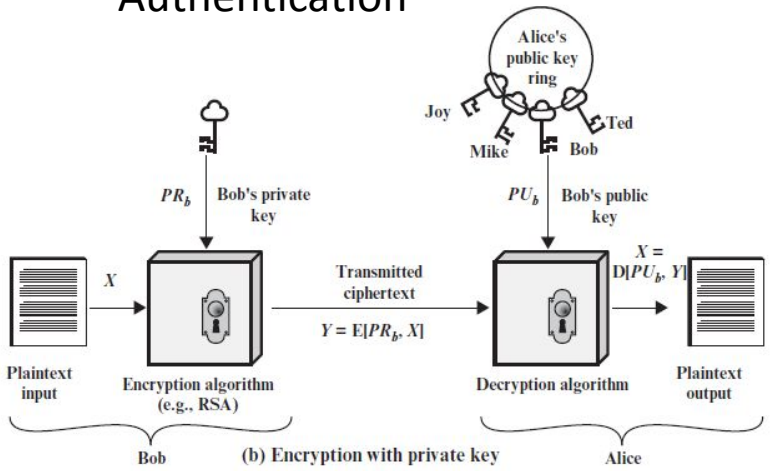
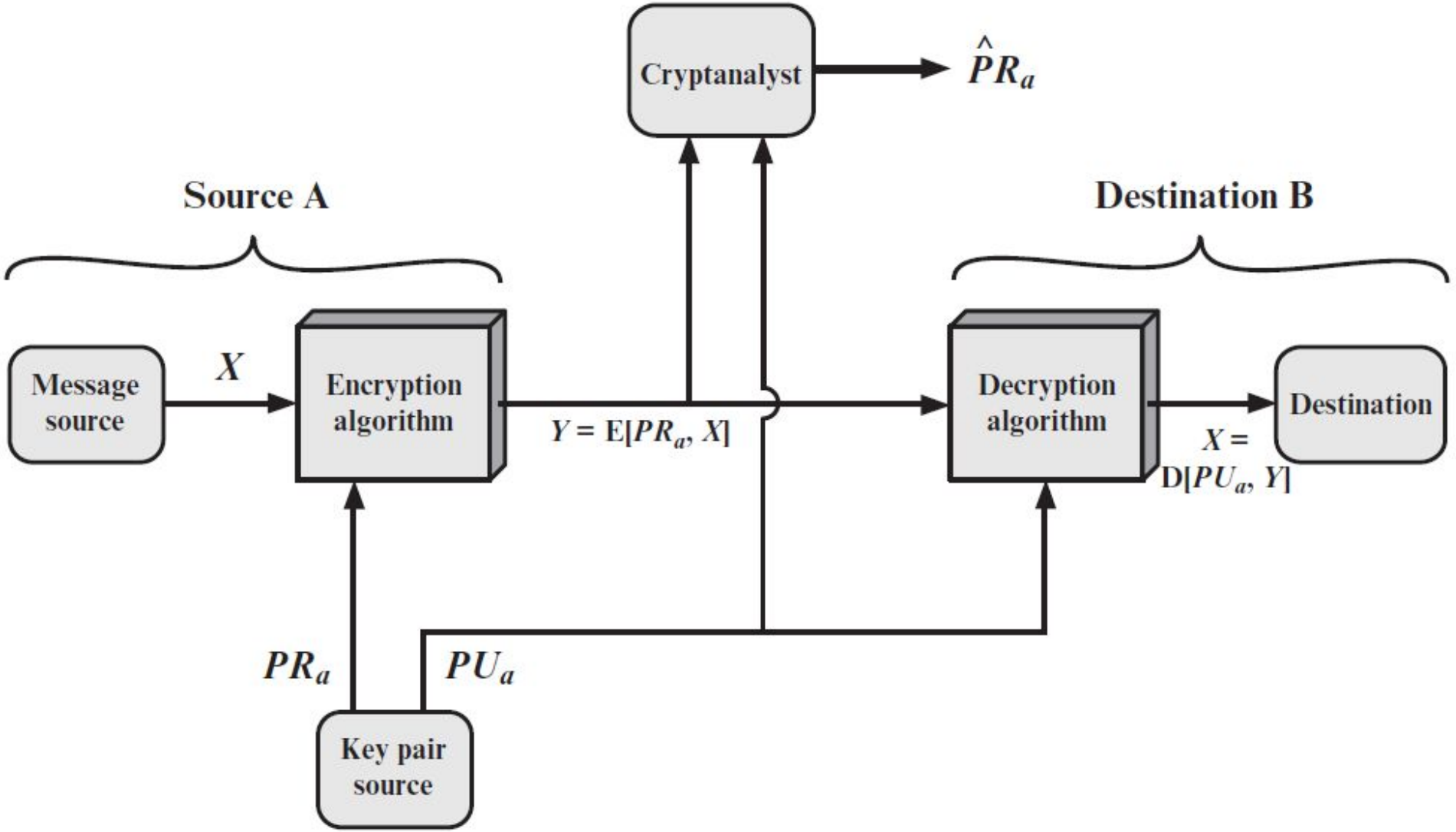


Figure 9.1 Public-Key Cryptography

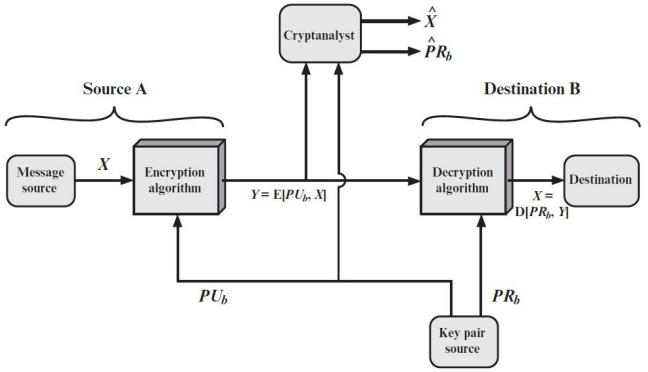
Public-Key Cryptosystems



Public-Key Cryptosystems

- **A** prepares a message to **B**
- **encrypts it using A's private key** before transmitting it.
- **B** can **decrypt the message using A's public** key. Because the message was encrypted using **A's private key**, **only** **A could have prepared the message**.
- Therefore, the **entire encrypted message** serves as a **digital signature.**
- it is **impossible to alter the message** without access to **A's private key**, so the message is **authenticated** both in terms of **source** and in terms of **data integrity**.

Confidentiality



Authentication

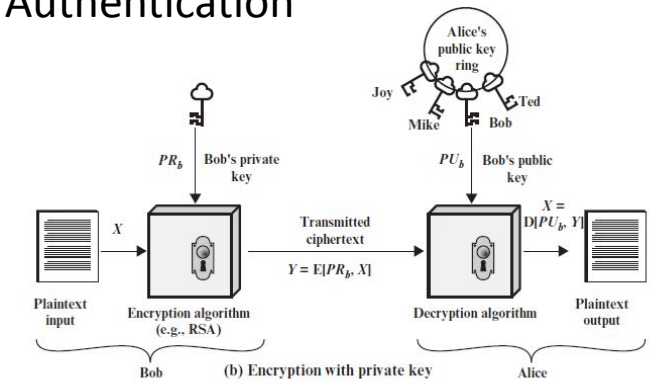


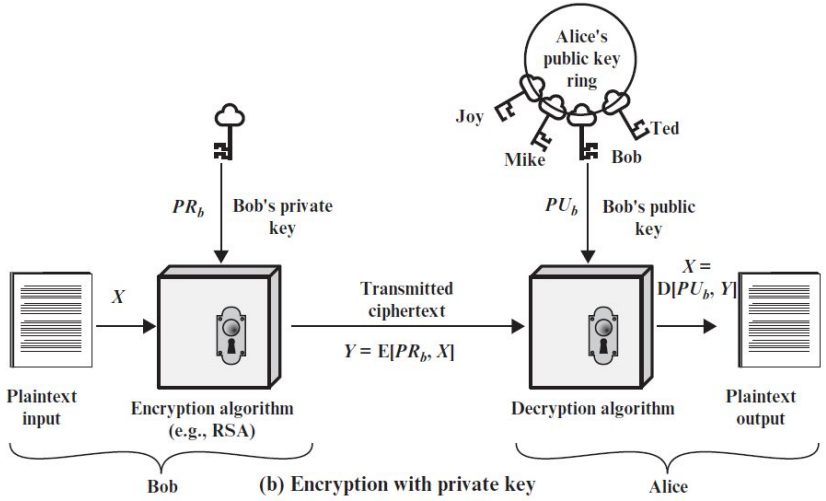
Figure 9.1 Public-Key Cryptography

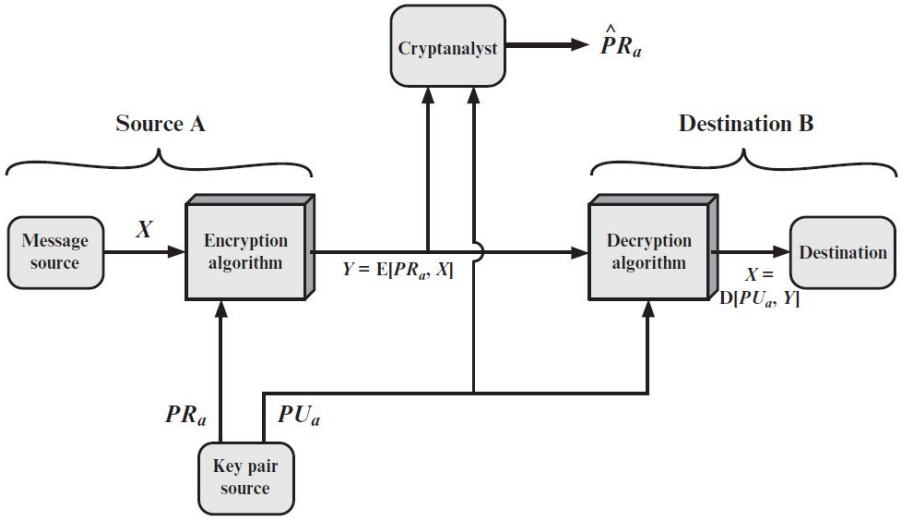
Public-Key Cryptosystems

- **How to keep the record of origin?**
- the **entire message is encrypted**, which, although **validating both author and contents**, requires a **great deal of storage**.
- Each document must be
  - **kept in plaintext** to be used for practical purposes.
  - A copy also must be stored **in ciphertext** so that the **origin and contents can be verified in case of a dispute**.
- A **more efficient** way of achieving the same results is to **encrypt a small block of bits** that is a **function of the document**. Such a block, called an **authenticator**, must have the property that it is **infeasible to change the document without changing the authenticator**.
- If the authenticator is **encrypted with the sender's private key**, it serves as a **signature that verifies origin, content, and sequencing**.

Public-Key Cryptosystems

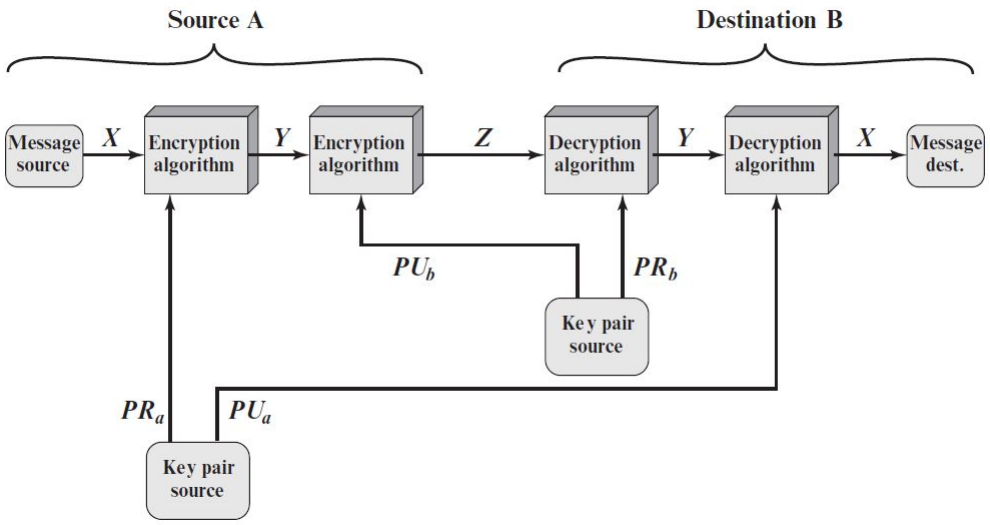
- It is important to emphasize that the encryption process depicted in Figures below does not provide **confidentiality**. That is, the message being sent is **safe from alteration** but not from eavesdropping.
- *any observer can decrypt the message by using the* ***sender's public key***





Public-Key Cryptosystems

- possible to provide both
  - the **authentication** function and
  - **confidentiality**
- by a **double use** of the public-key scheme (Figure):



$Y = E(PR_a, X))$

$Z = E(PU_b, Y)$

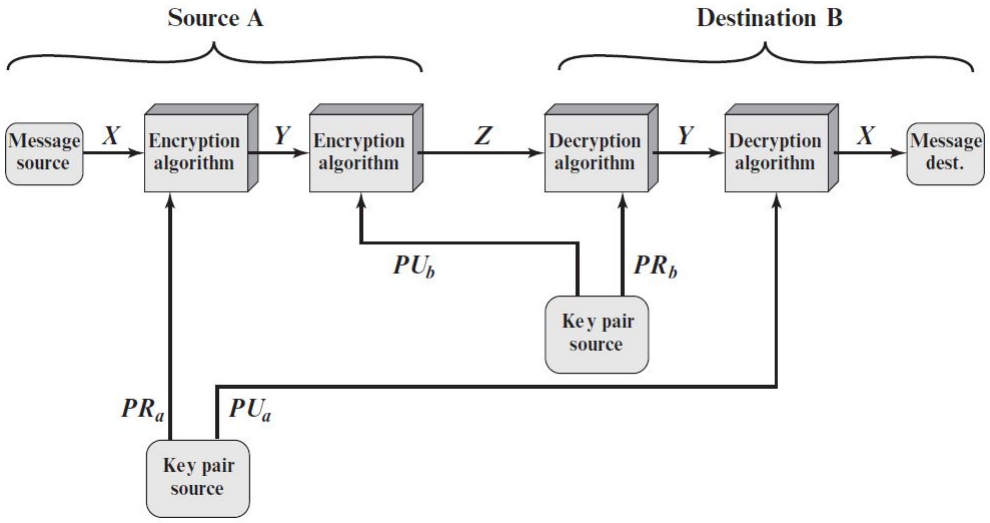
$Y = D(PR_b, Z)$

$X = D(PU_a, Y)$

Public-Key Cryptosystems

- First step - **encrypting a message**, using the **sender's private key**. This provides the **digital signature**.
- Next, we **encrypt again**, using the **receiver's public key**.
- The **final ciphertext** can be **decrypted** only by the **intended receiver,** who alone has the matching private key. Thus, **confidentiality** is provided.

- The **disadvantage** **of this approach** is that the public-key algorithm, **which is complex**, must be exercised **four times** rather than **two** in each communication.



Applications for public key cryptosystems

Classify the use of public-key cryptosystems into **three categories**

1. **Encryption/decryption**: The sender encrypts a message with the recipient's public key, and the recipient decrypts the message with the recipient's private key.

2. **Digital signature**: The sender "**signs" a message with its private key**. Signing is achieved by a cryptographic algorithm applied to the **message or to a small block** of data that is a **function of the message**.

3. **Key exchange:** Two sides cooperate to exchange a session key, which is a secret key for symmetric encryption generated for use for a particular transaction (or session) and valid for a short period of time. Several different approaches are possible, involving the private key(s) of one or both parties

Applications for public key cryptosystems

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie–Hellman	No	No	Yes
DSS	No	Yes	No

RSA- Rivest–Shamir–Adleman
DSS is **Digital Signature Standard**

Requirements of Public Key Cryptography

Depends on a **cryptographic algorithm** **based on two related keys**.

Diffie and Hellman postulated this system without demonstrating that such algorithms exist.

However, they did lay out the **conditions that such algorithms must fulfill** [DIFF76b].

1. It is **computationally easy** for a party B to **generate a key pair** (public key PU_b, private key PR_b).
2. It is **computationally easy for a sender** A, knowing the public key and the message to be **encrypted**, M, to **generate the corresponding ciphertext**:

   $C = E(PU_b, M)$
3. It is **computationally easy for the receiver B** to **decrypt** the resulting ciphertext using the private key to recover the original message:

   $M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$
4. It is **computationally infeasible** for an **adversary**, knowing the **public key**, PU_b, to **determine the private key, PR_b.**
5. It is computationally **infeasible** for an adversary, knowing the public key, PU_b, and a ciphertext, C, to **recover the original message**, M.
6. The **two keys** can be **applied in either order**

   $M = D[PU_b, E(PR_b, M)] = D[PR_b, E(PU_b, M)]$

Requirements of Public Key Cryptography

- These are **formidable requirements**, as evidenced by the fact that **only a few algorithms** (**RSA, elliptic curve cryptography, Diffie—Hellman, DSS**) have **received widespread** acceptance in the several decades since the concept of public-key cryptography was proposed.
- Before elaborating on **why the requirements are so formidable**, let us first recast them.

- The **requirements boil down to the need** for a **trap-door one-way function**.

1. A one-way function is **one that maps a domain** into a range such that **every function value** has a **unique inverse,**
2. with the condition that the **calculation of the function is easy**, whereas **the calculation of the inverse is infeasible:**

$Y = f(X)$ **easy**

$X = f^{-1}(Y)$ **infeasible**

Requirements of Public Key Cryptography

Define / meaning of the terms:

- Easy
- Infeasible

- Before explain the meaning of **trap-door one-way function**

Cryptography

Public Key Cryptography & RSA

M S Vilkhu

04 Nov 2024(C1/C3)

02 Nov 2024(C1/C3/C5)

-- Nov 2024(C1/C3)

-08 Nov 2024(C5)

Requirements of Public Key Cryptography

- **Easy** is defined to mean
  - a **problem that can be solved** in **polynomial time** as a **function of input length**.
- if the length of the input is **n bits**, then the **time to compute** the function is **proportional to n^a**, where **a** is a fixed constant.
- Such algorithms are said to **belong to the class P**.

What is class P?

Requirements of Public Key Cryptography

- **Class P**

- **Class P** in **computational complexity theory** refers to a set of decision problems that can be solved by a deterministic Turing machine in **polynomial time**. These problems are considered to be **efficiently solvable**,
- **Deterministic Algorithm**: Class P includes problems that **can be solved** by a **deterministic machine** (i.e., an algorithm that behaves predictably for a given input).

- **Examples**:
  - Sorting a list of numbers (e.g., using algorithms like quicksort or mergesort).
  - Finding the greatest common divisor (GCD) of two numbers.
  - Matrix multiplication.

Requirements of Public Key Cryptography

- **infeasible**
- The term **infeasible** is a much fuzzier concept.
- In general, we can say a **problem is infeasible** if the **effort to solve it grows faster than polynomial time** as a **function of input size**.
- For example,
- if the length of the input is **n bits** and the **time to compute** the function is proportional to 2^n the **problem is considered infeasible**. Unfortunately, it is difficult to determine if a particular algorithm exhibits this complexity.
- Furthermore, **traditional** notions **of computational complexity** focus on the **worst-case or average-case complexity** of an algorithm. These measures are **inadequate for cryptography**, which requires that it be **infeasible to invert a function** for virtually **all inputs**, **not for** the worst case or even average case.

Requirements of Public Key Cryptography

- Definition: **trap-door one-way function**,
  - **easy** to **calculate in one direction** and
  - **infeasible** **to calculate in the other direction unless certain additional information is known**.
- With the **additional information** the inverse can be **calculated in polynomial time**. We can summarize as follows:
- A **trap- door one-way function** is a family of invertible functions f_k, such that

$$
\begin{aligned}
Y &= f_k(X) && \text{easy, if } k \text{ and } X \text{ are known} \\
X &= f_k^{-1}(Y) && \text{easy, if } k \text{ and } Y \text{ are known} \\
X &= f_k^{-1}(Y) && \text{infeasible, if } Y \text{ is known but } k \text{ is not known}
\end{aligned}
$$

Thus, the development of a practical public-key scheme depends on discovery of a suitable **trap-door one-way function**.

Polynomial time : Explanation

info only

- **Polynomial time** refers to the **time complexity of an algorithm**,
- expressed in **terms of the size of the input**, where the **maximum number of operations** the algorithm performs **grows at most as a polynomial function of the input size**.

- **Explanation:**
- If an algorithm runs in polynomial time, it means that for an input of size n, the runtime can be expressed as $O(n^k)$ where k is a constant and n is the input size.
- Examples include $O(n^2)$ (quadratic time), $O(n^3)$ (cubic time), or O(n) (linear time).
- Polynomial time is considered **efficient** or **tractable**, as the runtime **does not grow exponentially** with the input size, making the algorithm **feasible** to run even for reasonably large inputs.

Polynomial time : Explanation

info only

- **Why is Polynomial Time Important?**
- **Efficiency Benchmark**: Polynomial time is used as a **benchmark to classify whether** a problem is **computationally feasible**. If an algorithm can **solve** a **problem in polynomial time**, it is generally considered **manageable**, even if the input size grows.
- **Complexity Classes**:
  - **P (Polynomial time)**: The class of decision problems that can be solved by a deterministic Turing machine in polynomial time. These are problems considered "easy" or "efficient" to solve.
  - **NP (Nondeterministic Polynomial time)**: The class of decision problems for which a solution can be *verified* in polynomial time by a deterministic Turing machine.

Polynomial time : Explanation

info only

The **time complexity** of this algorithm is O(n) ,which is **polynomial time** because the number of operations **scales linearly with the size of the input array.**

In contrast, **exponential time** algorithms, such as those with time complexities like $O(2^n)$ or O(n!) become **infeasible** for large inputs because their runtime grows much faster than polynomial time algorithms.

Attacks on Public-Key Cryptanalysis

Attack on public key cryptography

1. **is vulnerable to a brute-force attack.**
2. find some way to **compute the private key given the public key**
3. **probable-message attack**.

Attacks on Public-Key Cryptanalysis

1. brute-force attack.

- As with symmetric encryption, a public-key encryption scheme **is vulnerable to a brute-force attack.**
- The **countermeasure** is the same: **Use large keys**.
- However, there is a **tradeoff to be considered**. Public-key systems **depend on the use of some sort of invertible mathematical function**. The complexity of calculating these functions **may not scale linearly with the number of bits in the key** but grow more rapidly than that.
- Thus, the **key size** **must be large enough to make brute-force attack impractical** but **small enough for practical encryption and decryption.**
- In practice, the **key sizes that have been proposed do make brute-force attack impractical** but result in encryption/decryption **speeds that are too slow** for general-purpose use.
- Instead, public-key encryption is currently **confined to key management and signature applications.**

Attacks on Public-Key Cryptanalysis

2. compute the private key given the public key.

- Another form of attack is to find some way to **compute the private key given the public key**.
- To date, it **has not been mathematically proven** that this form of **attack is infeasible for a particular public-key algorithm**.
- Thus, any given algorithm, including the **widely used RSA algorithm**, is **suspect**.
- The history of cryptanalysis shows that a **problem that seems insoluble** from one perspective can be found to have a solution if **looked at in an entirely different way**.

Attacks on Public-Key Cryptanalysis

3. probable-message attack

- Finally, **there is a form of attack** that is **peculiar** to public-key systems.
- This is, in essence, a **probable-message attack**.
- Suppose, for example, that a **message were to be sent that consisted solely of a 56-bit DES key.**
- An adversary **could encrypt all possible 56-bit DES keys using the public key** and could **discover the encrypted key by matching the transmitted ciphertext**.
- Thus, no matter how large the key size of the public-key scheme, the **attack is reduced to a brute-force attack on a 56-bit key.**
- This attack can be **thwarted** by **appending** some **random bits** to such simple messages.

Thank You