FSI

LEI

2025/2026

T5 – Asymmetric ciphers and public-Key Encryption

Public-Key encryption: a revolution in the history of cryptography

- Symmetric encryption systems are based on complex solutions employing substitution and permutation
- Public-key cryptography provides a **radical** departure from all that has gone before: use mathematical functions rather than substitution and permutation operations
- Asymmetric cryptography uses two separate keys, rather than a single key as in symmetric encryption systems
- The use of two keys has profound consequences in the areas of confidentiality, key distribution and authentication.

Asymmetric Encryption - Terminology

Asymmetric Keys

Two related keys, a public key and a private key that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification.

Public Key Certificate

A digital document issued and digitally signed by the private key of a Certification Authority that binds the name of a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private key.

Public Key (Asymmetric) Cryptographic Algorithm

A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that deriving the private key from the public key is computationally infeasible.

Public Key Infrastructure (PKI)

A set of policies, processes, server platforms, software and workstations used for the purpose of administering certificates and public-private key pairs, including the ability to issue, maintain, and revoke public key certificates.

Asymmetric v/s Symmetric encryption

Symmetric v/s Asymmetric

Characteristic	Symmetric Key Cryptography	Asymmetric Key Cryptography
Key used for encryption / decryption	Same key is used for encryption and decryption	One key used for encryption and another, different key is used for decryption
Speed of encryption / decryption	Very fast	Slower
Size of resulting encrypted text	Usually same as or less than the original clear text size	More than the original clear text size
Key agreement / exchange	A big problem	No problem at all
Number of keys required as compared to the number of participants in the message exchange	Equals about the square of the number of participants, so scalability is an issue	Same as the number of participants, so scales up quite well
Usage	Mainly for encryption and decryption (confidentiality), cannot be used for digital signatures (integrity and non-repudiation checks)	Can be used for encryption and decryption (confidentiality) as well as for digital signatures (integrity and non-repudiation checks)

Principles of Public-Key Cryptosystems

- The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption:

Key distribution

- **How to have secure communications in general without having to trust a KDC with your key**

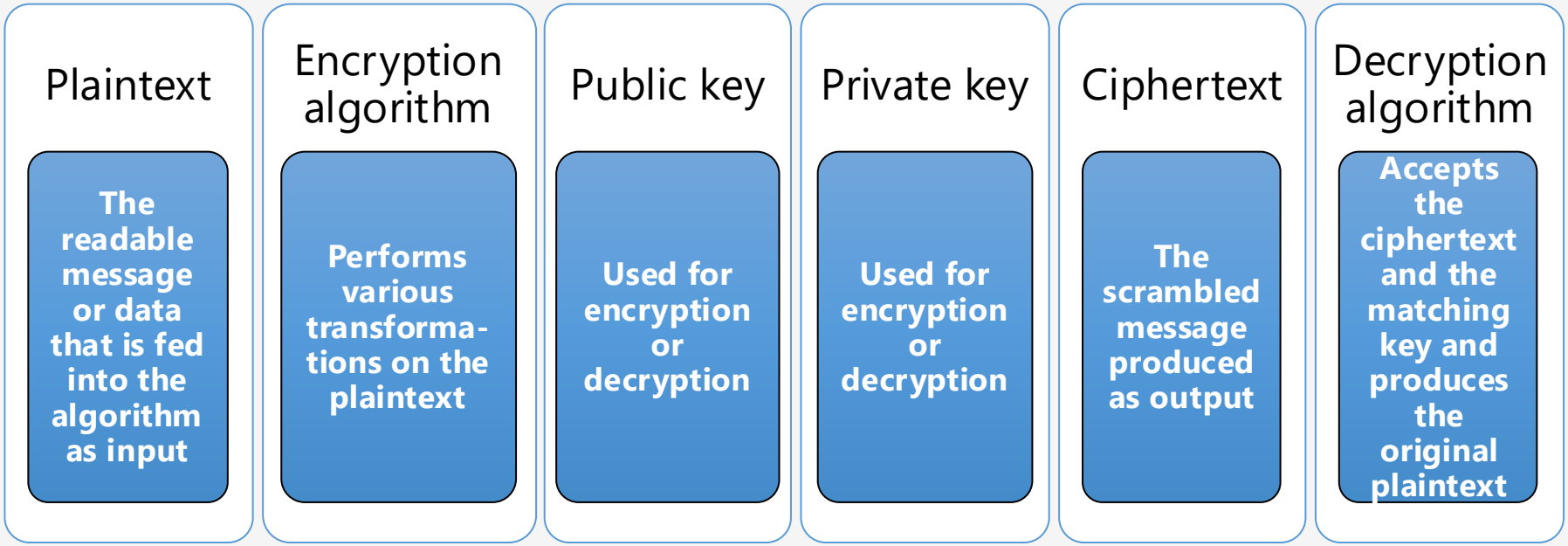
Digital signatures

- **How to verify that a message comes intact from the claimed sender**

- Whitfield Diffie and Martin Hellman from Stanford University achieved a breakthrough in 1976 by coming up with a method that addressed both problems and was radically different from all previous approaches to cryptography

Public-Key Cryptosystems

- Asymmetric encryption uses a key for encryption and a different but related key for decryption
- With some algorithms (e.g. RSA) either of the keys can be used for encryption, with the other used for decryption
- It is computationally infeasible to determine the decryption key knowing only the algorithm and the encryption key
- A public-key encryption scheme has *6 ingredients*:



Public-Key Cryptography

- Encrypting with the sender's private key supports integrity and data sender's authentication. Example: digital signatures in in PGP
- Encrypting with the receiver's public key supports confidentiality. Example: encryption in PGP
- Observe the key pairs involved:
  - ✓ *PUa* and *PRa* (encryption with public key of receiver)
  - ✓ *PRb* and *PUb* (encryption of private key of sender)

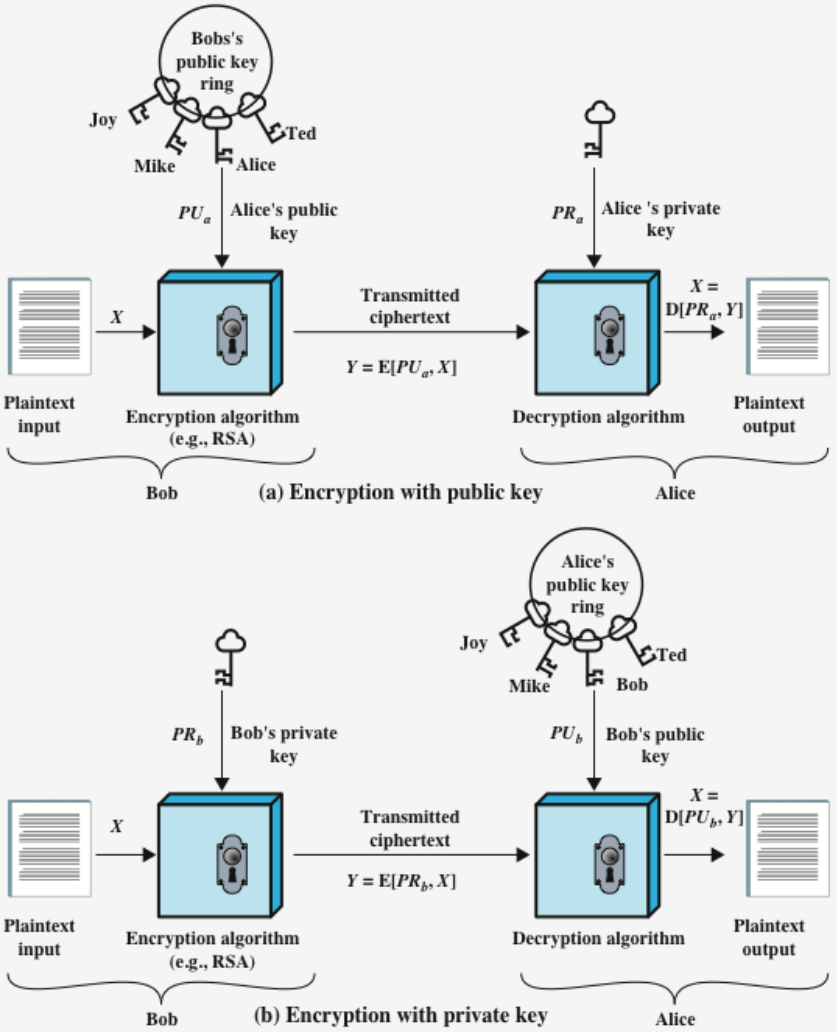
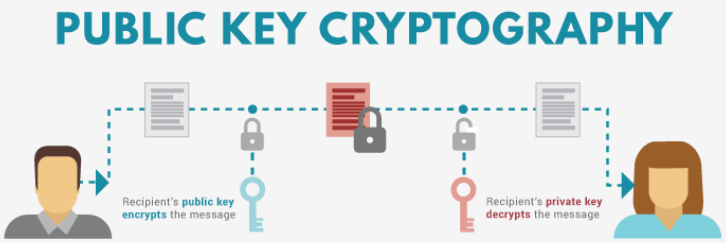


Figure 9.1 Public-Key Cryptography

Public-Key Cryptography



Essential steps involved:

- ✓ Each user generates a *pair of keys* to be used for encryption and decryption of messages
- ✓ Each user places one of the two keys in a *public register* or other assessable file: this is the public key, the companion is kept private
- ✓ Each user maintains a *collection of public keys* obtained from others
- ✓ If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key. Alice decrypts the message using its private key (only Alice should know this key)
- ✓ If Bob wishes to send a message that Alice can verify has not been modified in transit (integrity verification), in addition to being actually sent by Bob (sender authentication), he generates a digital signature using his own private key. The signature can be verified by Alice using Bob's public key

Conventional and Public-Key Encryption

- ✓ We refer to the key used in symmetric encryption as a secret key
- ✓ The two keys used for asymmetric encryption are referred to as the public key and the private key

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 same algorithm and 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. The key must be kept secret 2. It must be impossible or at least (computationally) 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 for Security:** 1. One of the 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

Public-Key Cryptosystem: Confidentiality

- ✓ Adversary knows E and D, will attempt to recover X and/or PR_b
- ✓ If focus is *only in this message,* the attempt will be to *estimate* X
- ✓ If goal is to *read future messages* the adversary will try to *estimate* PR_b

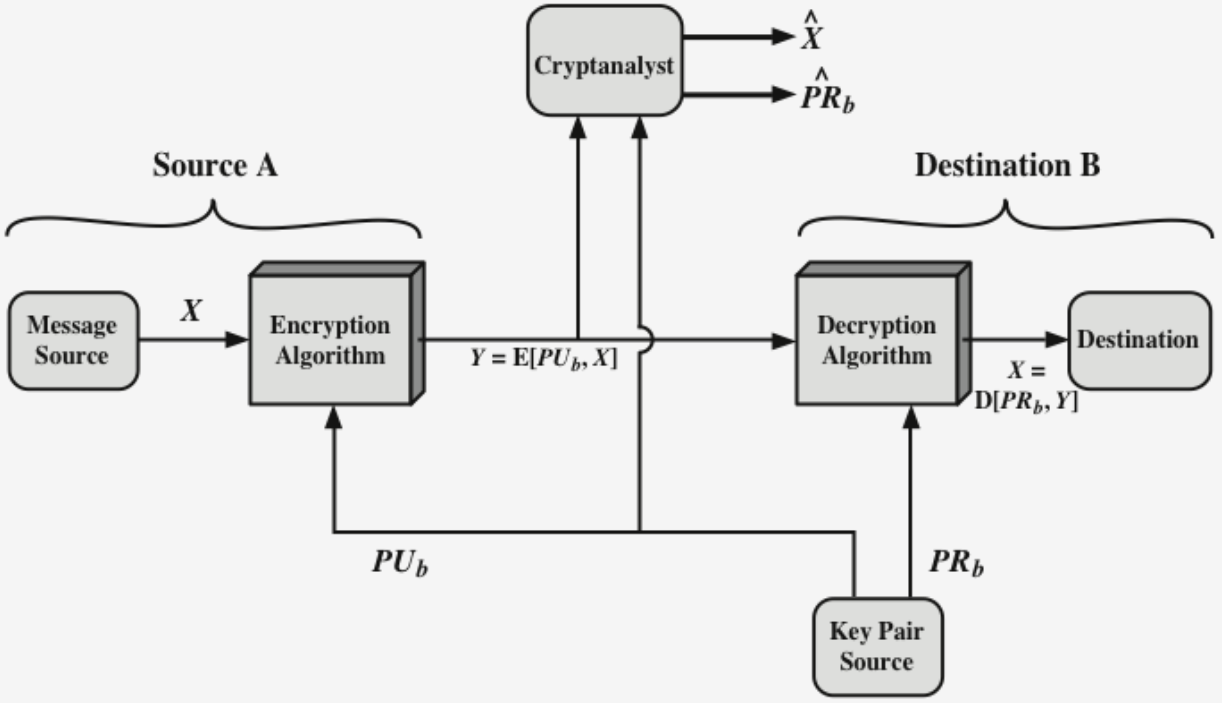


Figure 9.2 Public-Key Cryptosystem: Secrecy

Public-Key Cryptosystem: Authentication and integrity

- The *entire* encrypted message can serve as a digital signature
- It is more efficient (in storage and computational effort) to encrypt a resume (or hash) of the message instead, thus generating a MAC (*Message Authentication Code*)
- The hash (authenticator) must have the property that it is infeasible to change the document without changing the hash
- It is impossible to alter the message without using the sender's private key
- The authenticator encrypted with the sender's private key provides the signature (verifies origin and contents)

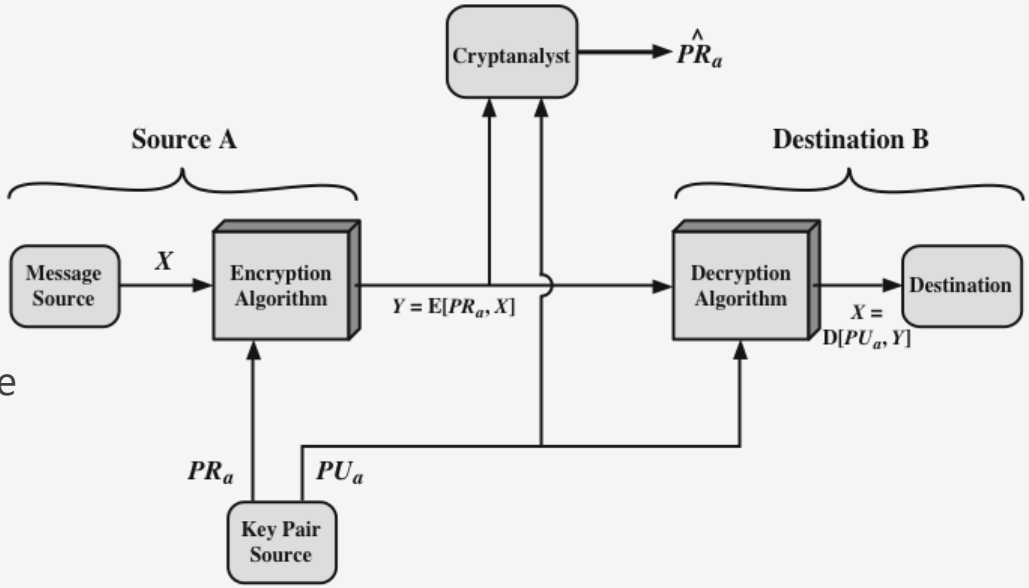


Figure 9.3 Public-Key Cryptosystem: Authentication

Public-Key Cryptosystem: Authentication, integrity and confidentiality

- Encrypt with the sender's private key to provide authentication and integrity (build a digital signature) and next encrypt with the receiver's public-key to provide confidentiality

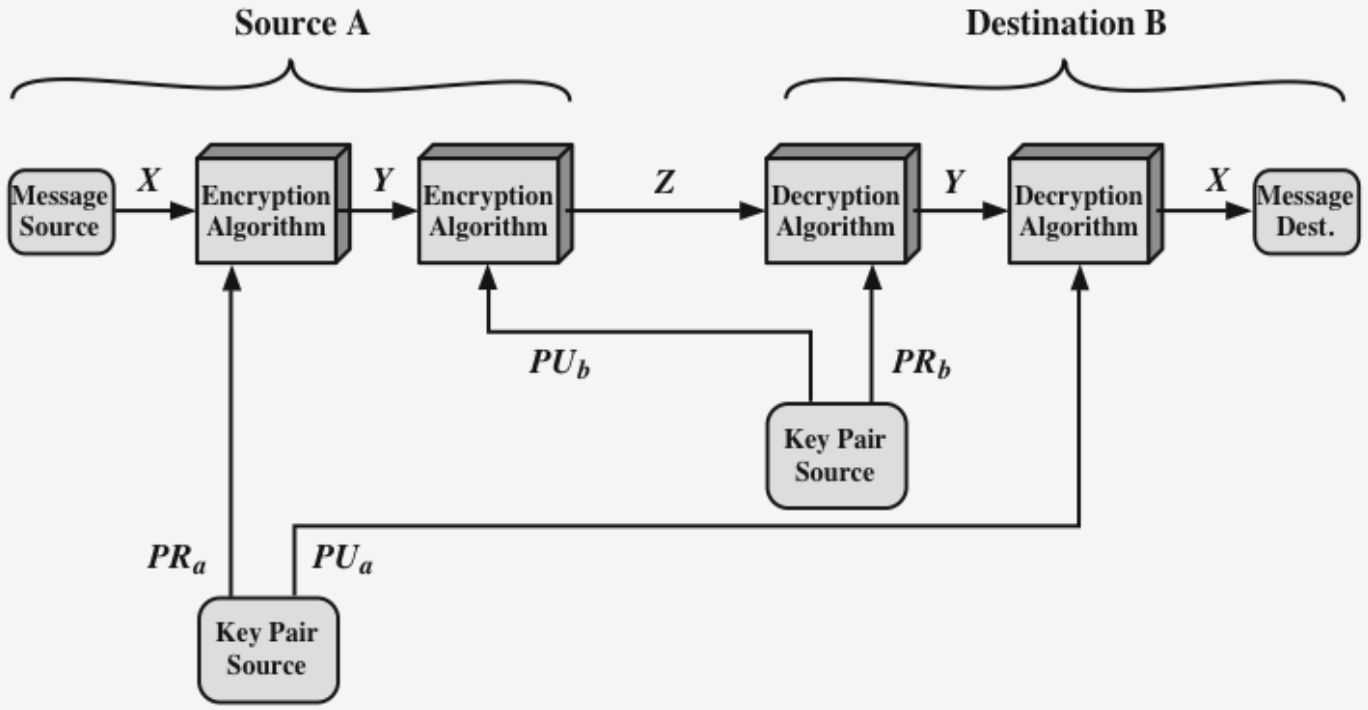
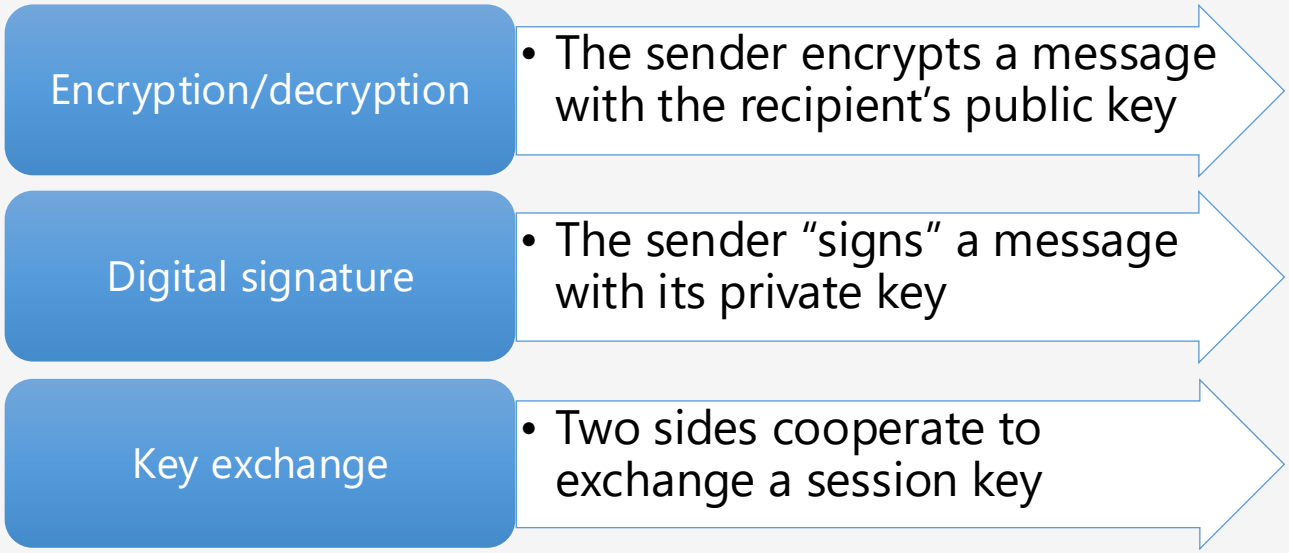


Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy

Applications for Public-Key Cryptosystems

- Public-key cryptosystems can be classified into three categories:



- In most practical applications: a session key is a secret key used for symmetric encryption during a short period of time

Applications for Public-Key Cryptosystems

- Some public-key encryption algorithms are suitable for all three applications, whereas others can be used only for one or two:

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

Requirements for Public-key Cryptography

Any Public-Key algorithm must fulfill some *fundamental conditions*:

- ✓ It is computationally easy to generate pairs of keys
- ✓ The two keys of the same pair can be applied in either order
- ✓ It is computationally easy to encrypt and decrypt using either the private or the public key
- ✓ It is computationally infeasible for an adversary, knowing the public key, to determine the private key
- ✓ It is computationally infeasible for an adversary, knowing the public key and a ciphertext, to recover the original message

- ❖ These are formidable requirements!
- ❖ In practice, only a few algorithms with such properties have been designed and received widespread acceptance over the years:

  RSA, Elliptic Curve Cryptography (ECC), Diffie-Hellman and DSS

Public-key Cryptanalysis

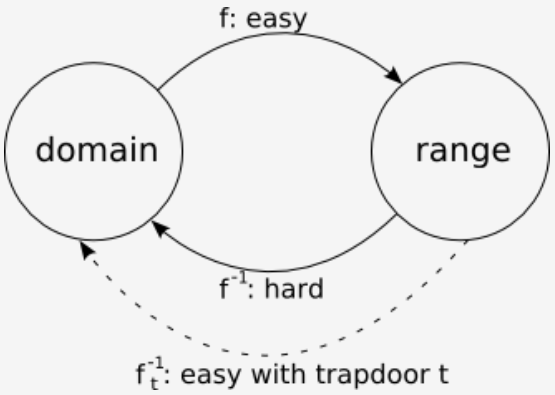
- A public-key encryption scheme is vulnerable to a brute-force attack
  - Countermeasure: as with symmetric encryption, use large keys
  - Key size must be small enough for practical encryption and decryption
  - Key sizes that have been proposed result in encryption and decryption speeds that are too slow for general-purpose encryption
  - In practice, public-key encryption is currently confined to key management and signature applications (conjugated with symmetric encryption in security systems, e.g: PGP)

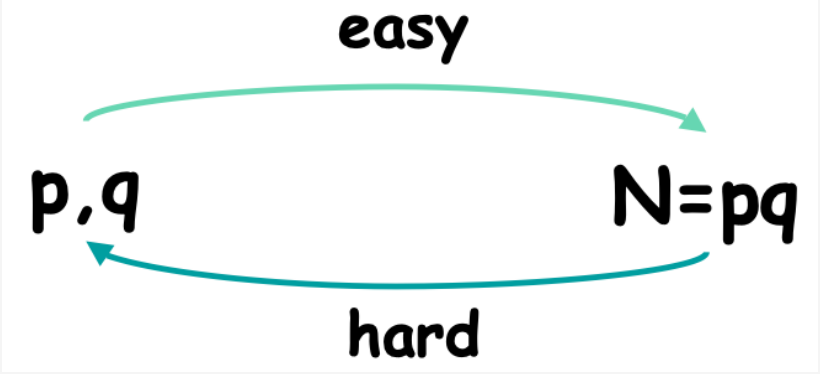
- Another form of attack is to find some way to compute the private key given the public key (*not accomplished so far!)*

Public-key Requirements

- A **trap-door one-way function** is used

- A trap-door one-way function f_k, is one such that:
  - $Y = f_k(X)$ easy, if k and X are known
  - $X = f_k^{-1}(Y)$ easy, if k and Y are known
  - $X = f_k^{-1}(Y)$ infeasible, if Y known but k not known
  - **A practical public-key scheme depends on a suitable trap-door one-way function**





RSA (Rivest-Shamir-Adleman) Algorithm

- Up until the 1970s, cryptography was based on symmetric keys
- RSA was developed in 1977 at MIT by Ron Rivest, Adi Shamir & Len Adleman
- Most popular public-key cryptosystem
- Is a cipher in which the plaintext and ciphertext are integers between 0 and n – 1 for some n
- A typical size for *n* is 1024 bits

RSA Algorithm

- Unlike symmetric algorithms (AES,3DES), public-key cryptosystems require the computation of a key pair (private and public)
- In symmetric algorithms we need a RNG or TRNG, while here there is math involved: RSA makes use of an expression with exponentials
- Plaintext is encrypted in blocks with each block having a binary value less than some number *n*
- The trapdoor is the knowledge of the private key d, which satisfies $ed \equiv 1 \bmod \varphi(n)$ (where $\varphi(n)$ is Euler's totient function of n)
- The security of RSA relies on the difficulty of factoring a large semiprime $n = p * q$. Without knowing p and q, it's very hard to compute $\varphi(n)$, and thus to derive d

The RSA Algorithm (setup phase in 5 steps)

1. Choose large primes ***p*** and ***q***
2. Multiply the two primes: ***n = p x q***
3. Calculate the Euler's totient function: ***Φ(n)=(p-1)x(q-1)***
4. Choose the public encryption key ***e*** *from* ***set*** **{1,2,…, *Φ(n)-1*}** such that gcd(e,*Φ(n)*) = 1 (in other words, e and *Φ(n)* are relatively primes or coprimes)

   Note: if e is not relatively prime to (p-1)(q-1), then it may not be possible to find a suitable private exponent d such that ed ≡ 1 mod (p-1)*(q-1), which is required for the **RSA trapdoor function to work**. It **ensures that a multiplicative inverse of e modulo $\varphi(n)$ exists**.

5. Compute the private key **d** such that d x e = 1 mod *Φ(n)* (the inverse of *e* will be our private key)

A few remarks:

- In practice *p* and *q* should be at least 2^{512} thus n >= 2^{1024}
- In practical system (e.g. PGP) we choose the length of *n* (for example using 2048 bits in RSA *p* and *q* will be 1024-bit in length)

Number theory: Euler's totient function

- An important concept in number theory
- Expressed using the Greek letter *phi* as: $\Phi(n)$
- The Euler's totient function counts the positive integers up to a given integer *n* that are relatively prime to *n* (have no common factors other than 1)
- In other words, is the number of integers k in the range $1 \leq k \leq n$ for which the greatest common divisor gcd(n,k) is 1

Examples:

$\Phi(1)=1$: 1 is the only integer in the range from 1 to n and gcd(1,1)=1

$\Phi(4)=2$: numbers 1,3 are all relatively prime to 4. other numbers in range are not since gcd(4,2)=2 and gcd(4,4)=4)

$\Phi(9)=6$: numbers 1,2,4,5,7,8 are all relatively prime to 9. other numbers in range are not since gcd(9,3)=3,gcd(9,6)=3 and gcd(9,9)=9

Number theory: Euler's totient function (2)

As it turns out, it is really easy to compute $\Phi(n)$ for an integer if we are able to know the prime factorization for the integer (thus, the two primes that "compose" the integer)

Given two (large) primes p and q, and n = p x q
Φ(n)= (p-1)x(q-1)

Example:

Given $n = p \times q = 3 \times 5 = 15$
$\Phi(15)=(3-1)x(5-1)=8$

In fact, $\Phi(15)=8$ *(numbers 1,2,4,7,8,11,13,14 are all relatively prime to 15. You may confirm that 3,5,6,9,10,12 and 15 are not)*

The RSA Algorithm (encryption and decryption)

Encryption:

given **(e,n)** (the public key) and the message **M** *from set* {1,2,…, *n-1*}

$C = M^e \bmod n$

Decryption:

given **C** (ciphertext) and **(d,n)** (the private key)

$M = C^d \bmod n$ = $(M^e)^d \bmod n = M^{ed} \bmod n$

- The **public key is (e,n)** and the **private key is (d,n)**
- This holds if *e* and *d* are multiplicative inverses modulo Φ(n)

RSA Algorithm: a simple application example

1. Select two prime numbers, **p=11** and **q=3**
2. Calculate **n=pq=11x3=33**
3. Calculate **Φ(33)=(p-1)(q-1)=10x2=20**
4. Select e such that e is relative prime to 20 and less than 20, we choose **e=3** (other values may be candidate to be chosen)
5. Determine d such that de=1 (mod 20). The correct value is **d = 7**, because
   dx3 = 1 (mod 20)
   7x3 = 21 = 1(mod 20) = (1x20)+1

Public key: PU={3,33}
Private key: PR={7,33}

Example application for message: 7

Encryption: $C = M^e \bmod n = 7^3 \bmod 33 = 343 \bmod 33 = 13$

Decryption: $M = C^d \bmod n = 13^7 \bmod 33 = 62748510 \bmod 33 = 7$

The RSA Algorithm: what guarantees security?

- The security relies on the fact that the attacker does not have *d* (he knowns the message and also *e*)
- Why cannot the attacker simply compute *d*?
- For that he needs to compute $\Phi(n)$, but to obtain *n* he needs *p* and *q*

✓ This is the process of integer factorization (determine which prime numbers divide a given positive integer)
✓ This is a very hard problem!

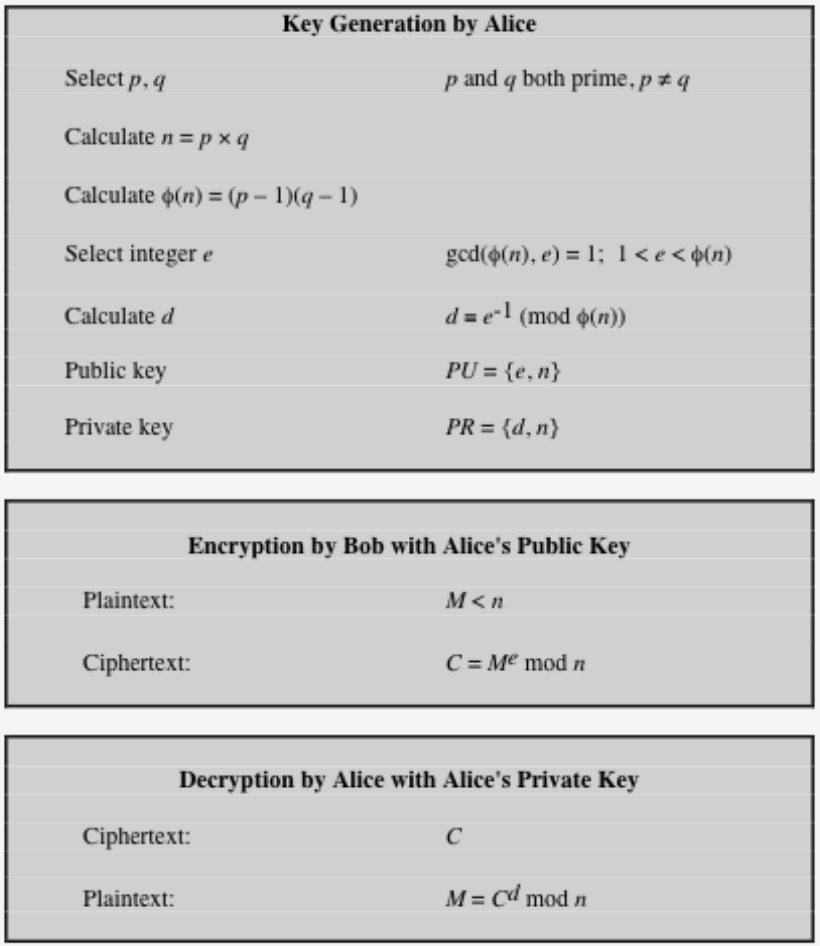
✓ For a 1024 bit number (around 300 digits) it cannot be done!

✓ The current record for integer factorization using general-purpose algorithms is the factorization of RSA-250 (in 2020), a 250-digit (829-bit) semiprime number

RSA Algorithm: challenge

Given two prime numbers, p=17 and q=11, demonstrate how the RSA algorithm may be applied to encrypt plaintext represented by the number 88, by applying the previously discussed steps:

- setup
- encryption
- decryption



Key Generation by Alice

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key

Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Private Key

Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Figure 9.5 The RSA Algorithm

RSA Algorithm: challenge (2)

Considere que a Alice e o Bob pretendem comunicar de forma segura usando encriptação assimétrica recorrendo ao algoritmo RSA. Considere igualmente que a Alice escolhe como números primos os valores "p=3" e "q=7", e o Bob os valores "p=2" e "q=5". Calcule a chave pública e privada para a Alice e para o Bob. Em seguida, encripte uma mensagem representada pelo valor "m=9" utilizando a chave pública de Bob.

Misconceptions Concerning Public-Key Encryption

- Public-key encryption is more secure from cryptanalysis than symmetric encryption?
  - No, as with any system, its security depends on the length of the key and the computational work required to break the cipher
- Is public-key encryption a general-purpose technique that has made symmetric encryption obsolete?
  - No, since encryption with public-key systems is much more slow than with symmetric encryption. Public-key encryption is thus mostly used by key management and signatures
- There is a feeling that key distribution is trivial when using public-key encryption, compared to the cumbersome handshaking involved with key distribution centers for symmetric encryption
  - Even with public-key encryption we still need some for of protocol

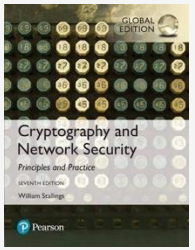
Summary

Public-key cryptography

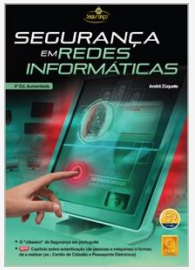
- ✓ Public-key cryptosystems
- ✓ Public-key encryption and decryption
- ✓ How RSA works

Bibliography

Cryptography and network security, Stallings, Pearson, 2017, Chapter 9: Public-Key Cryptography and RSA; Chapter 2.4 Prime Numbers and 2.5 Euler's Theorems



Segurança em Redes Informáticas, Capítulo 2: Criptografia, Capítulo 3: Gestão de Chaves públicas



RSA Algorithm: example (solution)

1. Select two prime numbers, p=17 and q=11
2. Calculate n=pq=17x11=187
3. Calculate (p-1)(q-1)=16x10=160
4. Select e such that e is relative prime to 160 and less than 160, we choose e=7 (please note that other values may be candidate to be chosen)
5. Determine d such that de=1 (mod 160) and d < 160. The correct value is d = 23, because 23x7=161=(1x160)+1
6. PU={7,187} and PR={23,187}

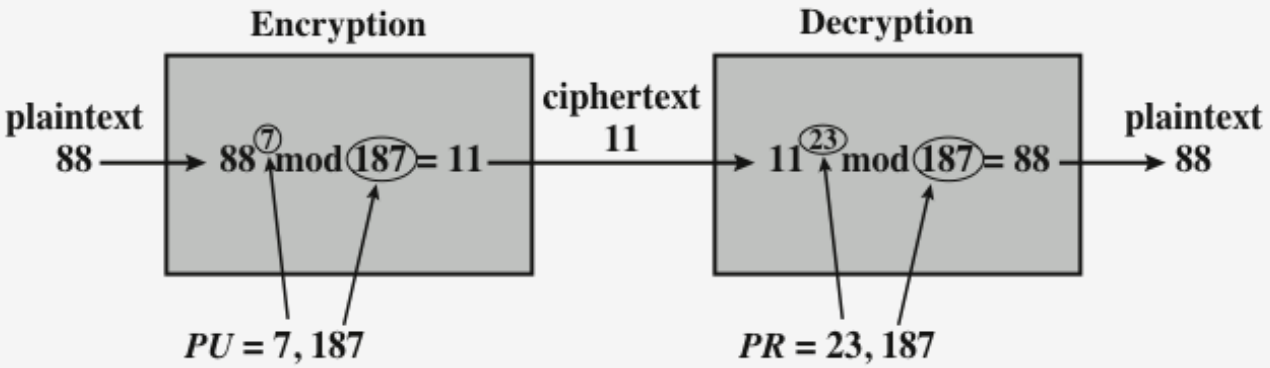


Figure 9.6 Example of RSA Algorithm