

Advanced data Security

Part I:

Cryptography and Network Security

Chapter5:

Public Key Cryptography and RSA

Prepared by:

Dr. Hamssa Hasrouny

Misconceptions Concerning Public-Key Encryption



- Public-key encryption **is more secure from cryptanalysis** than symmetric encryption
- Public-key encryption is a general-purpose technique that has made **symmetric encryption obsolete**
- 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

Table 9.1

Terminology Related to Asymmetric Encryption

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.

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 without having to trust a KDC(key distribution center) with your key**

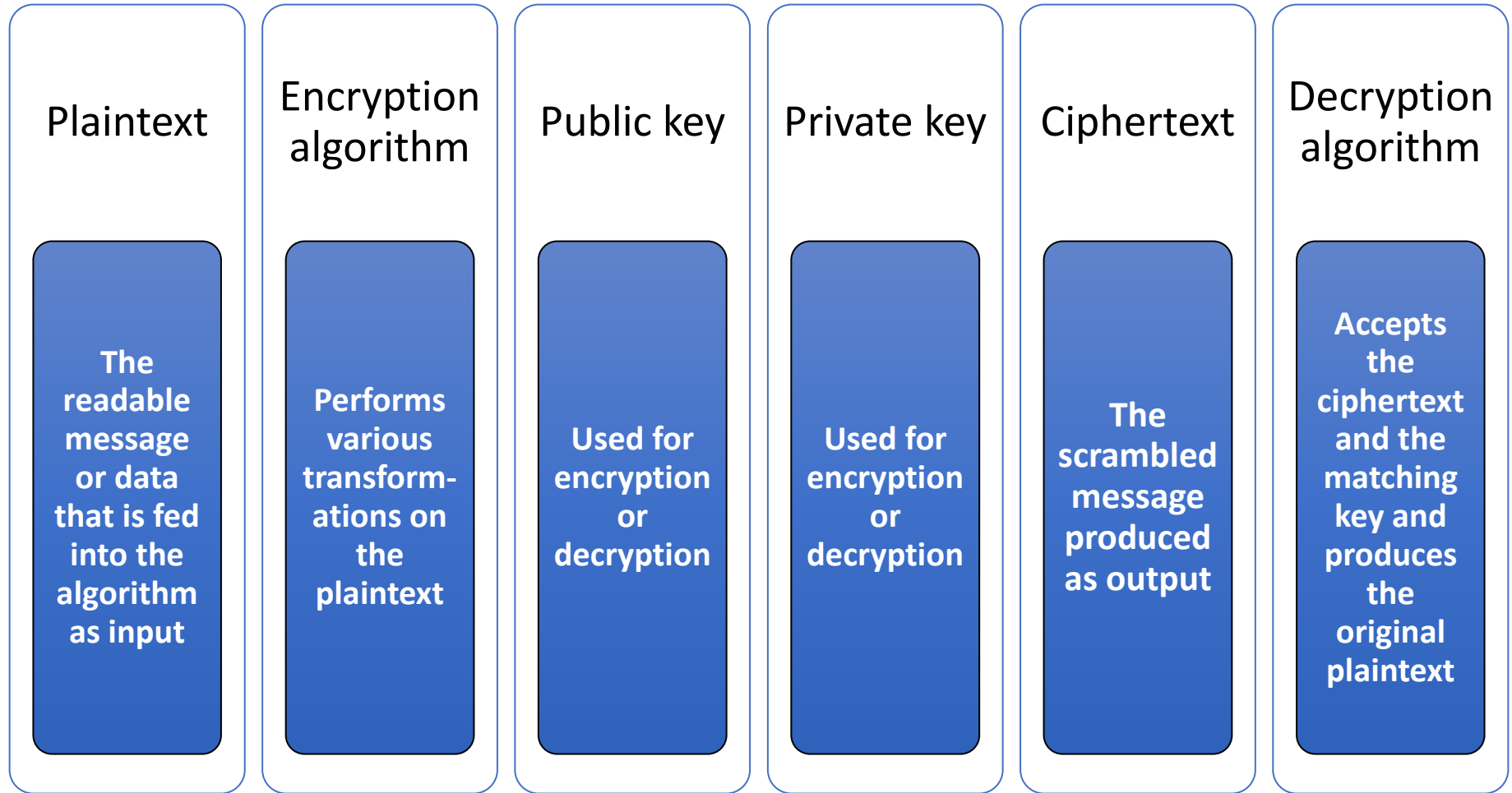
Digital signatures

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

- Whitfield Diffie and Martin Hellman 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

- A public-key encryption scheme has six ingredients:



Public-Key Cryptography

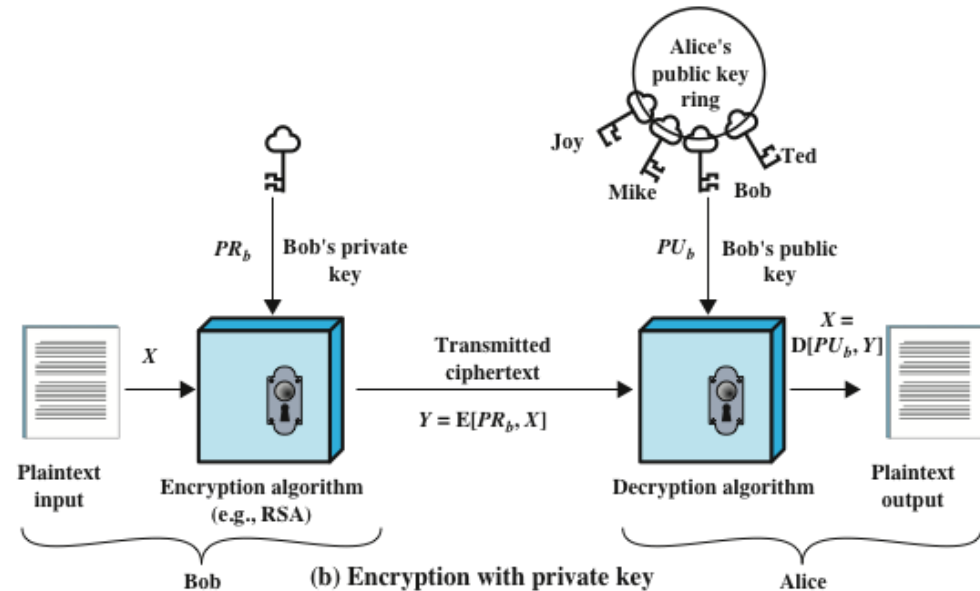
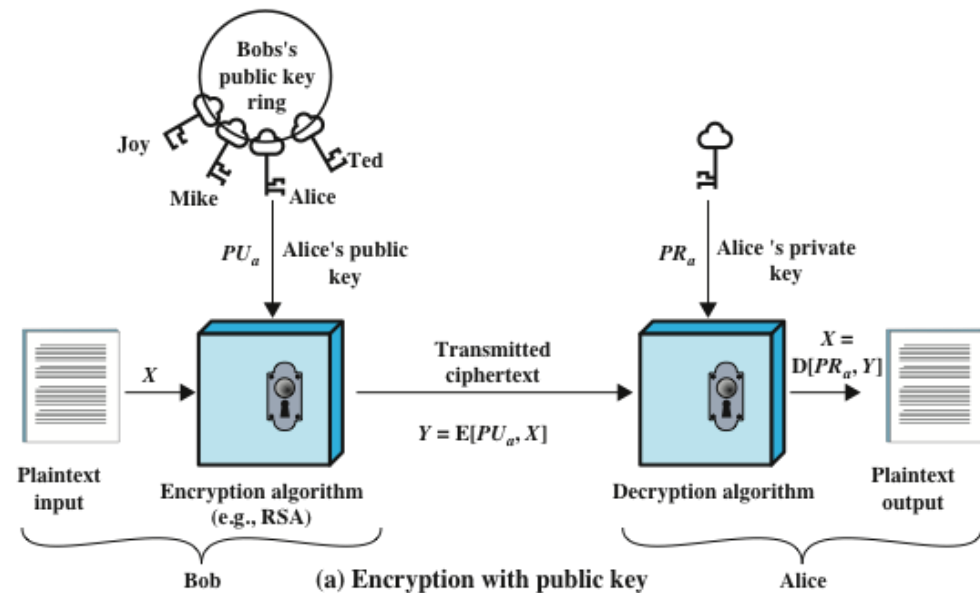
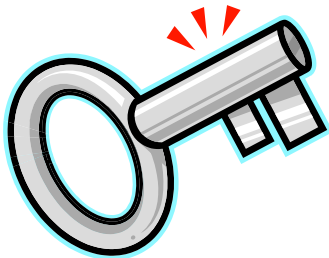


Figure 9.1 Public-Key Cryptography

Conventional and Public-Key Encryption

Conventional Encryption	Public-Key Encryption
<i>Needed to Work:</i> 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. <i>Needed for Security:</i> 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.	<i>Needed to Work:</i> 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). <i>Needed for Security:</i> 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.

the use of **public-key encryption** to provide **authentication and confidentiality**

Public-Key Cryptosystem: Authentication and Secrecy

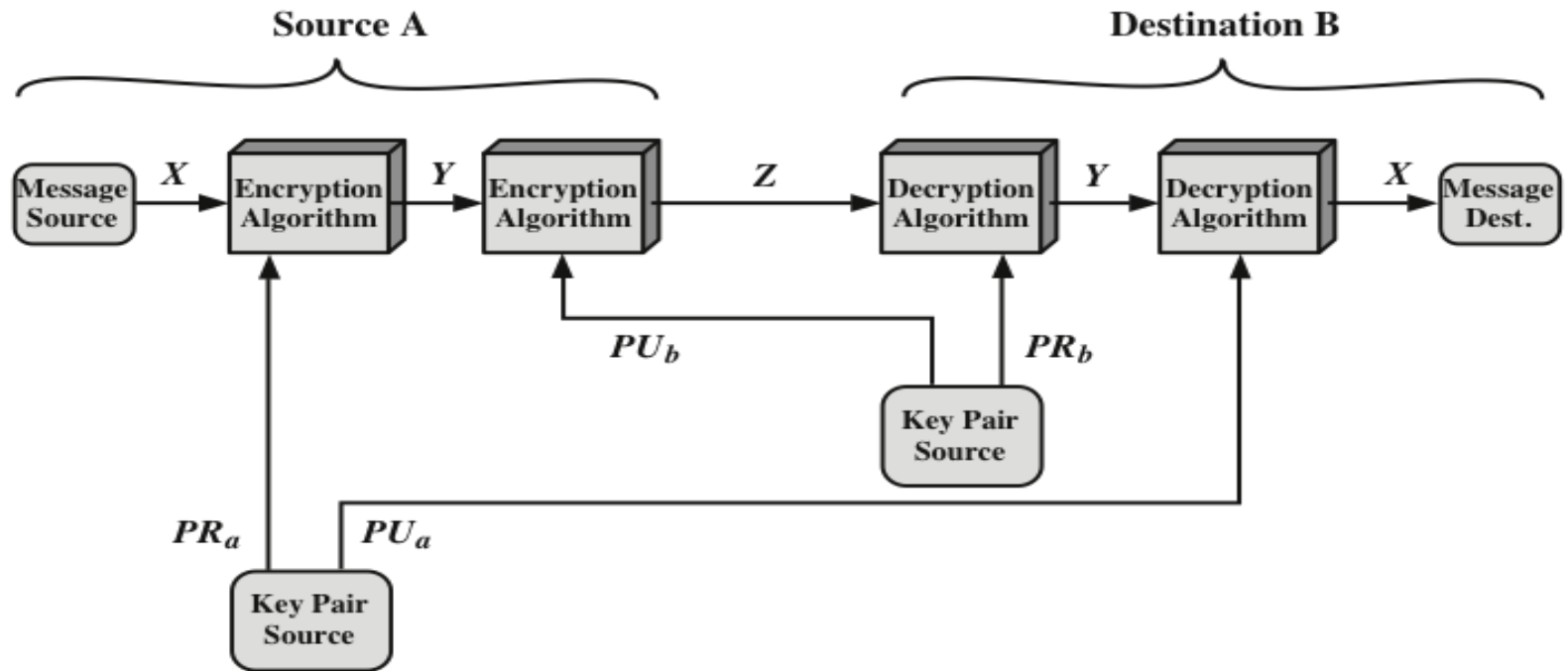
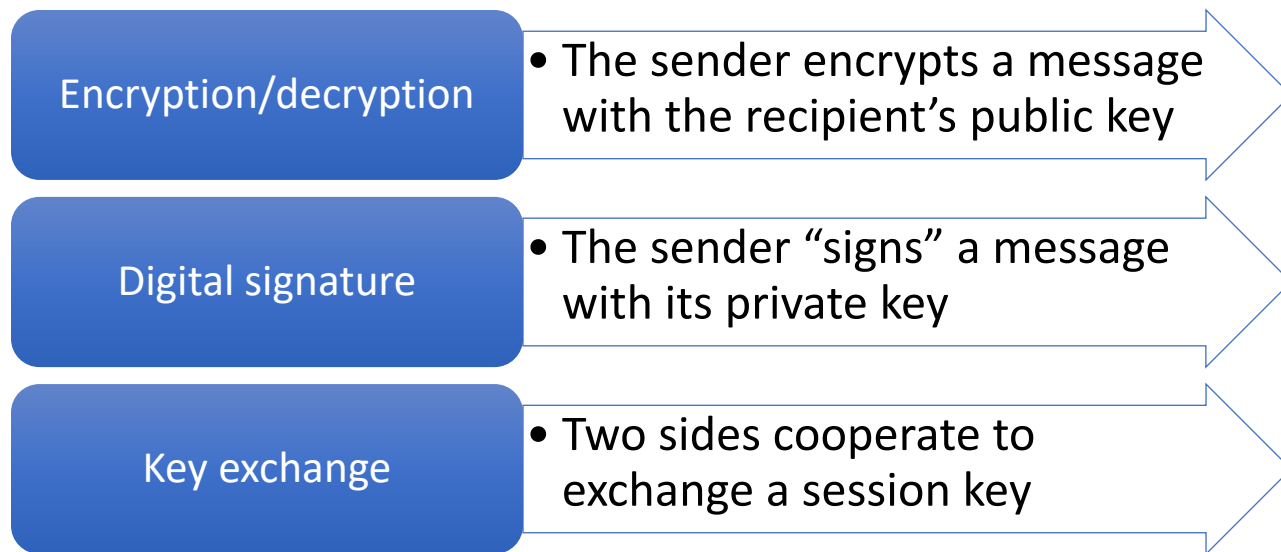


Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy

- It is, possible to provide both the **authentication function and confidentiality** by a **double use of the public-key scheme**
- disadvantage of this approach is that the public-key algorithm(complex), must be **exercised four times** rather than two **in each communication**.

Applications for Public-Key Cryptosystems

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



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

Table 9.3

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

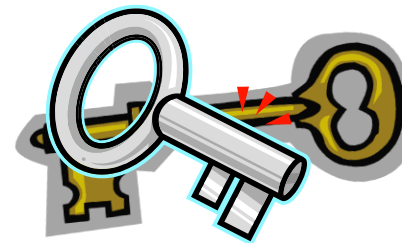
Table 9.3 Applications for Public-Key Cryptosystems

Public-Key Requirements

- Conditions that these algorithms must fulfill:
 - It is computationally easy for a party B to generate a pair (public-key PU_b , private key PR_b)
 - It is computationally easy for a sender A, knowing the public key and the message to be encrypted, to generate the corresponding ciphertext
 - It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message
 - 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
 - The two keys can be applied in either order

Public-Key Cryptanalysis

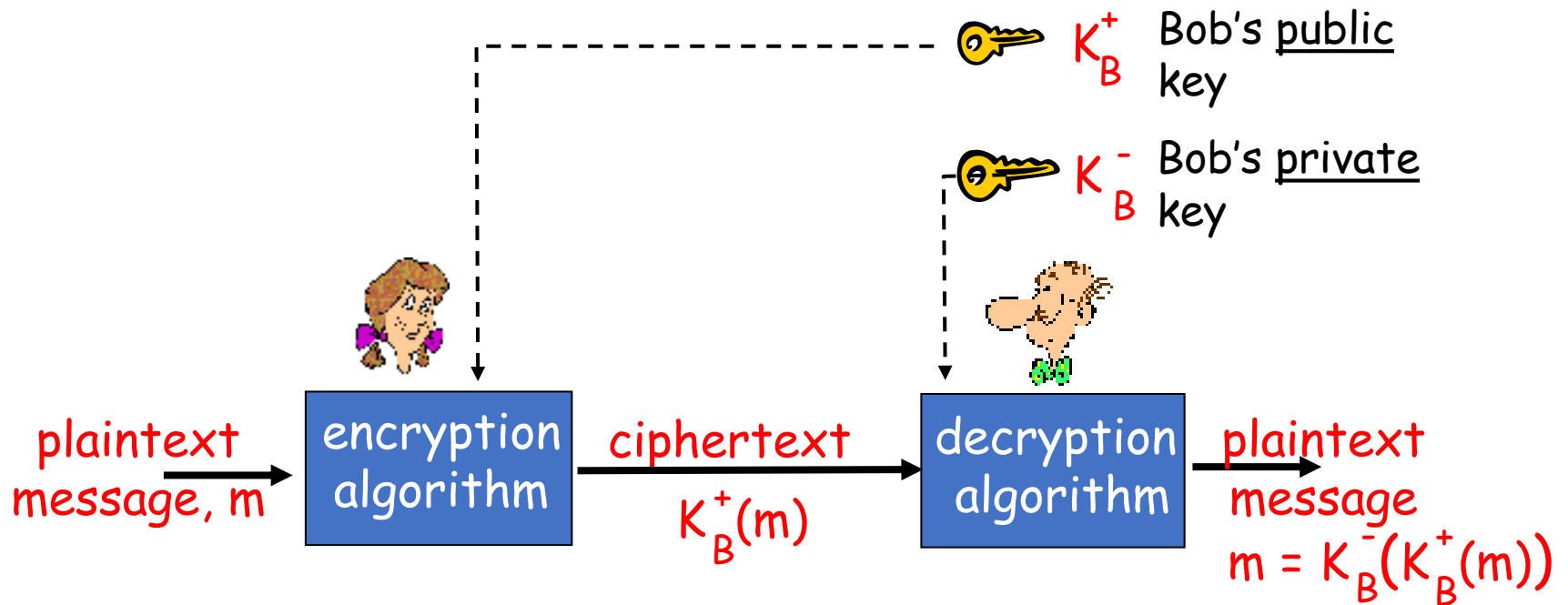
- A public-key encryption scheme is **vulnerable to a brute-force attack**
 - **Countermeasure:** use large keys
 - Key size must be small enough for practical encryption and decryption
 - Key sizes that have been proposed result in encryption/decryption speeds that are **too slow for general-purpose use**
 - **Public-key encryption is currently confined to key management and signature applications**
- 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
- Finally, there is a **probable-message attack**
 - This attack can be thwarted by appending some random bits to simple messages



Rivest-Shamir-Adleman (RSA) Scheme

- Developed in 1977 at MIT by Ron Rivest, Adi Shamir & Len Adleman
- Most widely used general-purpose approach to public-key encryption
- 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, or 309 decimal digits

Public key cryptography



Public key encryption algorithms

Requirements:

- 1 need K_B^+ and K_B^- such that

$$K_B^-(K_B^+(m)) = m$$

- 2 given public key K_B^+ , it should be impossible to compute private key K_B^-

RSA: Rivest, Shamir, Adelson algorithm

Prerequisite: modular arithmetic

- $x \bmod n$ = remainder of x when divide by n
- Facts:
 - $[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$
 - $[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$
 - $[(a \bmod n) * (b \bmod n)] \bmod n = (a*b) \bmod n$
- Thus $(a \bmod n)^d \bmod n = a^d \bmod n$
- Example: $x=14, n=10, d=2$:
- $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$
 $x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$

RSA: getting ready

- A message is a bit pattern.
- A bit pattern can be uniquely represented by an integer number.
- Thus encrypting a message is equivalent to encrypting a number.

Example

- $m = 10010001$. This message is uniquely represented by the decimal number 145.
- To encrypt m , we encrypt the corresponding number, which gives a new number (the cyphertext).

RSA: Creating public/private key pair

1. Choose two large prime numbers p, q .
(e.g., 1024 bits each)
2. Compute $n = pq$, $z = (p-1)(q-1)$
3. Choose e (with $e < n$) that has no common factors with z . (e, z are "relatively prime").
4. Choose d such that $ed-1$ is exactly divisible by z .
(in other words: $ed \bmod z = 1$).
5. Public key is (n, e) . Private key is (n, d) .

$\underbrace{\hspace{1.5cm}}_{K_B^+}$

$\underbrace{\hspace{1.5cm}}_{K_B^-}$

RSA: Encryption, decryption

0. Given (n,e) and (n,d) as computed above

1. To encrypt message m ($<n$), compute

$$c = m^e \bmod n$$

2. To decrypt received bit pattern, c , compute

$$m = c^d \bmod n$$

Magic
happens!

$$m = \underbrace{(m^e \bmod n)}_c^d \bmod n$$

RSA example 1:

Bob chooses $p=5$, $q=7$. Then $n=35$, $z=(p-1)(q-1)=24$.

$e=5$ (so e , z relatively prime).

$d=29$ (so $ed-1$ exactly divisible by z).

Encrypting 8-bit messages.

encrypt:	<u>bit pattern</u>	<u>m</u>	<u>m^e</u>	<u>$c = m^e \bmod n$</u>
	00001100	12	248832	17

decrypt:	<u>c</u>	<u>c^d</u>	<u>$m = c^d \bmod n$</u>
	17	481968572106750915091411825223071697	12

RSA example 2:

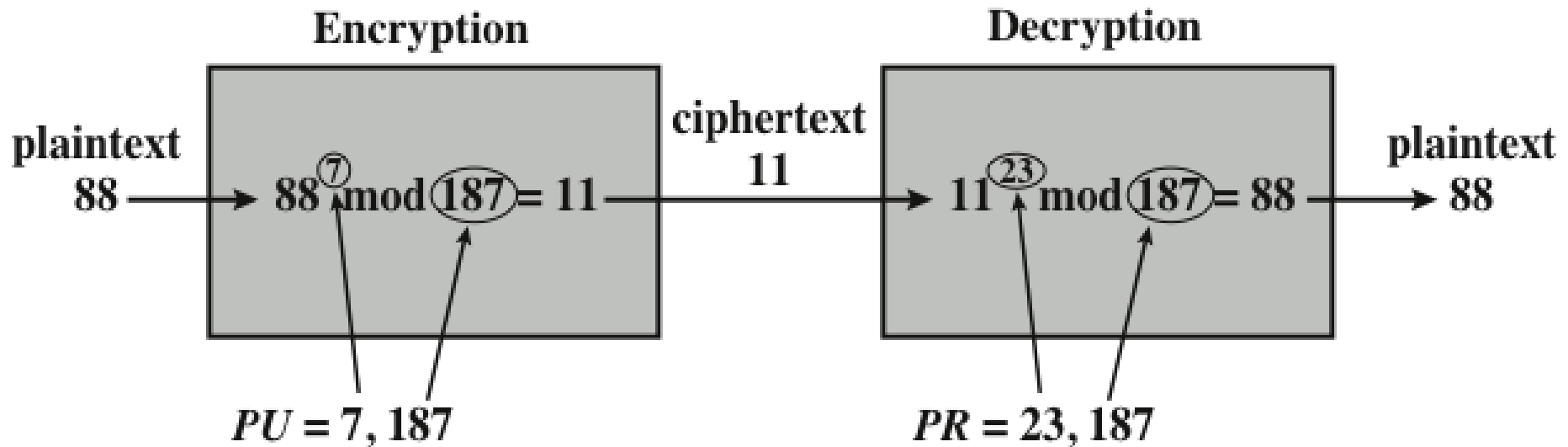


Figure 9.6 Example of RSA Algorithm

RSA: another important property

The following property will be *very* useful later:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

use public key
first, followed
by private key

use private key
first, followed
by public key

Result is the same!

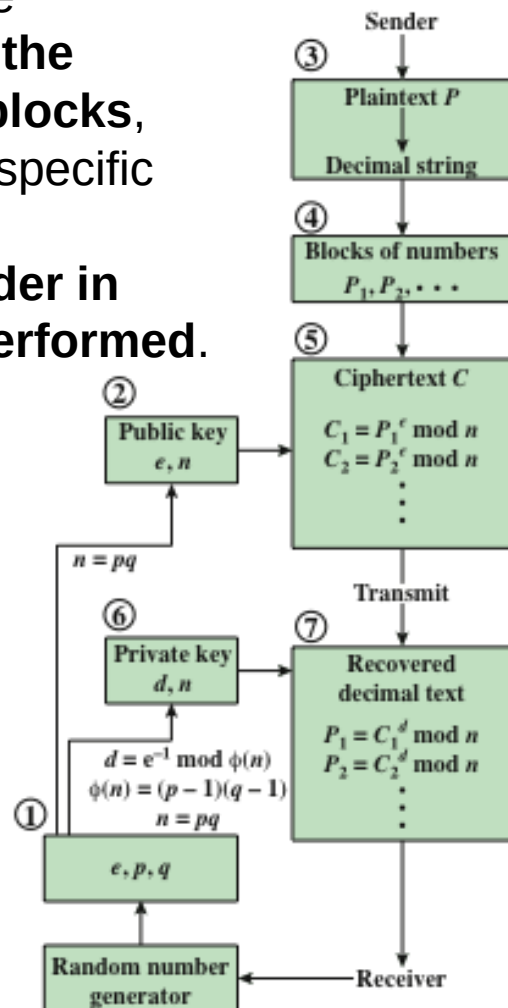
Why is RSA Secure?

- Suppose you know Bob's public key (n,e) . How hard is it to determine d ?
- Essentially need to find factors of n without knowing the two factors p and q .
- Fact: factoring a big number is hard.

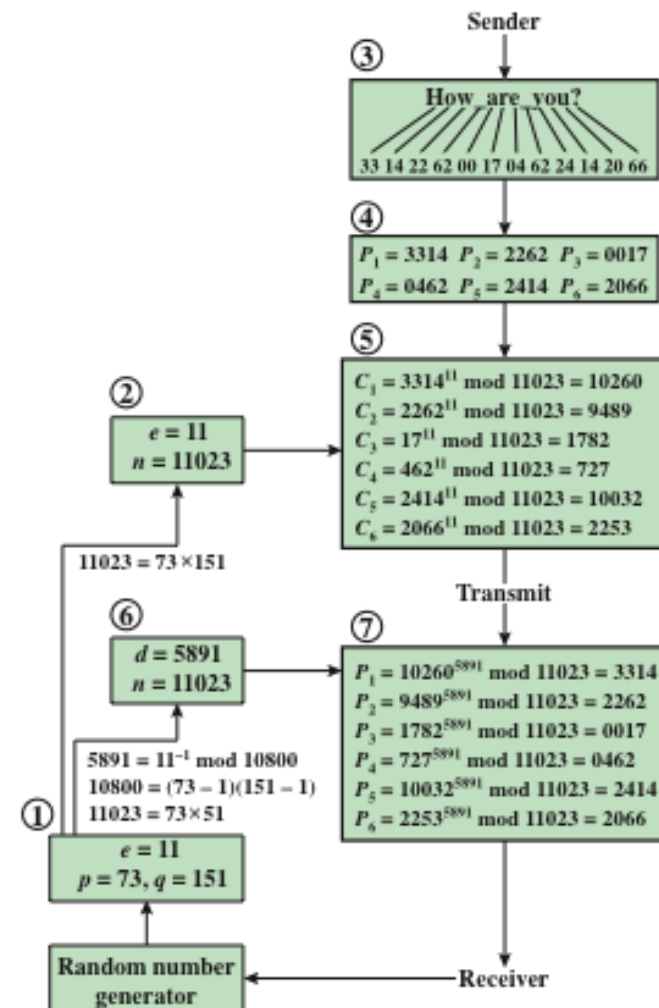
Generating RSA keys

- Have to find big primes p and q
- Approach: make good guess then apply testing rules (see Kaufman)

Figure 9.7a illustrates the sequence of events for the encryption of multiple blocks, and **Figure 9.7b** gives a specific example. The **circled numbers** indicate the order in which operations are performed.



(a) General approach



(b) Example

Figure 9.7 RSA Processing of Multiple Blocks

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	$d = e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod{n}$

Decryption by Alice with Alice's Private Key	
Ciphertext:	C
Plaintext:	$M = C^d \pmod{n}$

Figure 9.5 The RSA Algorithm

Exponentiation in Modular Arithmetic

- Both encryption and decryption in RSA involve raising an integer to an integer power, mod n
- Can make use of a property of modular arithmetic:
$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$
- With RSA you are dealing with potentially large exponents so efficiency of exponentiation is a consideration

Efficient Operation Using the Public Key

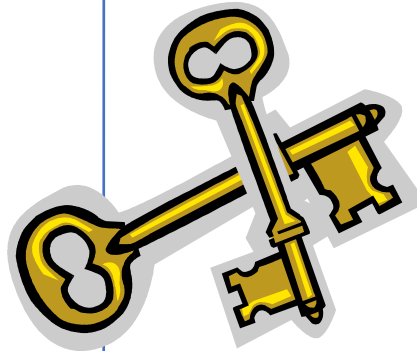
- To speed up the operation of the RSA algorithm using the public key, **a specific choice of e is usually made**
- The most common choice is **65537** ($2^{16} + 1$)
 - Two other popular choices **are $e=3$ and $e=17$**
 - Each of these choices has only two 1 bits, so the number of multiplications required to perform exponentiation is minimized
 - With a very small public key, such as $e = 3$, RSA **becomes vulnerable to a simple attack**

Efficient Operation Using the Private Key

- Decryption uses exponentiation to power d
 - A small value of d is vulnerable to a brute-force attack and to other forms of cryptanalysis
- Can use the Chinese Remainder Theorem (CRT) **to speed up computation**
 - The quantities $d \bmod (p - 1)$ and $d \bmod (q - 1)$ can be precalculated
 - End result is that the calculation is approximately four times as fast as evaluating $M = C^d \bmod n$ directly

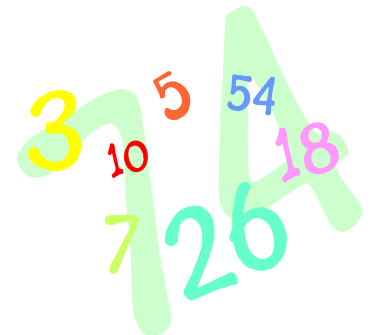
Key Generation

- Before the application of the public-key cryptosystem each participant must generate a pair of keys:
 - Determine two prime numbers p and q
 - Select either e or d and calculate the other
- Because the value of $n = pq$ will be known to any potential adversary, primes must be chosen **from a sufficiently large set**
 - The method used for finding large primes must be reasonably efficient

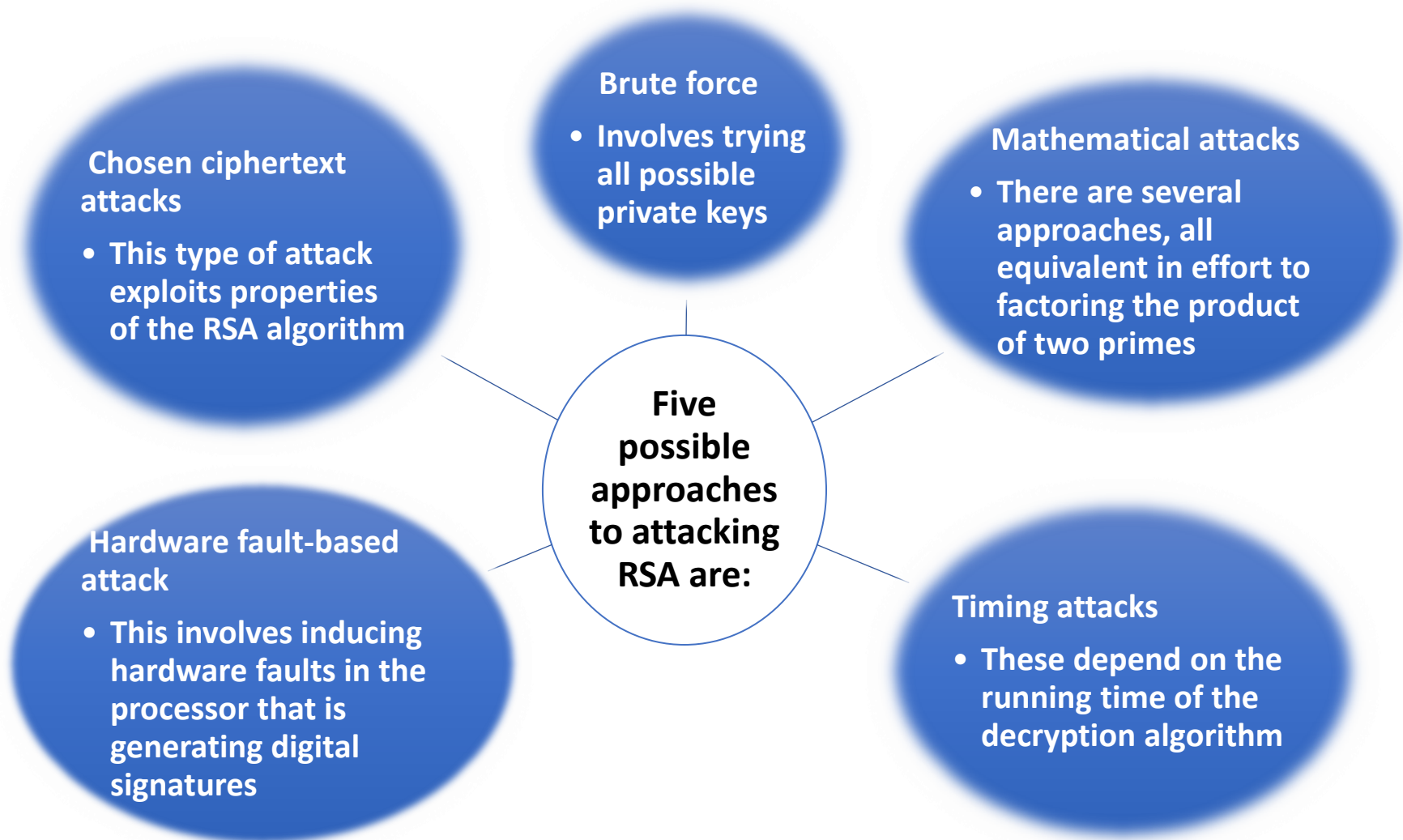


Procedure for Picking a Prime Number

- Pick an odd integer n at random
- Pick an integer $a < n$ at random
- Perform the **probabilistic primality** test with a as a parameter. If n fails the test, reject the value n and go to step 1
- If n has passed a sufficient number of tests, accept n ; otherwise, go to step 2



The Security of RSA



Factoring Problem

- We can identify three approaches to attacking RSA mathematically:
 - Factor n into its two prime factors. This enables calculation of $\phi(n) = (p - 1) \times (q - 1)$, which in turn enables determination of $d = e^{-1} \pmod{\phi(n)}$
 - Determine $\phi(n)$ directly without first determining p and q . Again this enables determination of $d = e^{-1} \pmod{\phi(n)}$
 - Determine d directly without first determining $\phi(n)$

Number of Decimal Digits	Number of Bits	Date Achieved
100	332	April 1991
110	365	April 1992
120	398	June 1993
129	428	April 1994
130	431	April 1996
140	465	February 1999
155	512	August 1999
160	530	April 2003
174	576	December 2003
200	663	May 2005
193	640	November 2005
232	768	December 2009

Table 9.5 Progress in RSA Factorization

Timing Attacks

- Paul Kocher, a cryptographic consultant, demonstrated that a snooper can determine a private key **by keeping track of how long a computer takes to decipher messages**
- Are applicable not just to RSA but to other public-key cryptography systems
- Are alarming for two reasons:
 - It comes from a completely unexpected direction
 - It is a ciphertext-only attack



Countermeasures

Constant exponentiation time

- **Ensure that all exponentiations take the same amount of time before returning a result; this is a simple fix but does degrade performance**

Random delay

- **Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack**

Blinding

- **Multiply the ciphertext by a random number before performing exponentiation; this process prevents the attacker from knowing what ciphertext bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack**

Fault-Based Attack

- An attack on a processor that is generating RSA digital signatures
 - Induces faults in the signature computation by reducing the power to the processor
 - The faults cause the software to produce invalid signatures which can then be analyzed by the attacker to recover the private key
- The attack algorithm involves inducing single-bit errors and observing the results
- While worthy of consideration, this attack does not appear to be a serious threat to RSA
 - It requires that **the attacker have physical access to the target machine and is able to directly control the input power to the processor**

Chosen Ciphertext Attack (CCA)

- The adversary chooses a number of ciphertexts and is then given the corresponding plaintexts, decrypted with the target's private key
 - Thus the adversary could select a plaintext, encrypt it with the target's public key, and then be able to get the plaintext back by having it decrypted with the private key
 - The adversary exploits properties of RSA and selects blocks of data that, when processed using the target's private key, yield information needed for cryptanalysis
- To counter such attacks, RSA Security Inc. recommends modifying the plaintext using a procedure known as optimal asymmetric encryption padding (OAEP)

Diffie-Hellman Key Exchange

- First published public-key algorithm
- A number of commercial products employ this key exchange technique
- Purpose is to enable two users to securely exchange a key that can then be used for subsequent symmetric encryption of messages
- The algorithm itself is **limited to the exchange of secret values**
- Its effectiveness depends on the difficulty of computing discrete logarithms



Alice



Bob

Alice and Bob share a prime q and α , such that $\alpha < q$ and α is a primitive root of q

Alice generates a private key X_A such that $X_A < q$

Alice calculates a public key $Y_A = \alpha^{X_A} \bmod q$

Alice receives Bob's public key Y_B in plaintext

Alice calculates shared secret key $K = (Y_B)^{X_A} \bmod q$



Alice and Bob share a prime q and α , such that $\alpha < q$ and α is a primitive root of q

Bob generates a private key X_B such that $X_B < q$

Bob calculates a public key $Y_B = \alpha^{X_B} \bmod q$

Bob receives Alice's public key Y_A in plaintext

Bob calculates shared secret key $K = (Y_A)^{X_B} \bmod q$

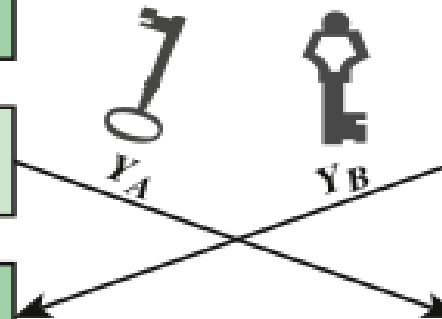


Figure 10.1 Diffie-Hellman Key Exchange

Key Exchange Protocols

- Users could create random private/public Diffie-Hellman keys each time they communicate
- Users could create a known private/public Diffie-Hellman key and publish in a directory, then consulted and used to securely communicate with them
 - Vulnerable to Man-in-the-Middle-Attack
 - Authentication of the keys is needed

ge protocol is
ch an attack
not
e
is
be
ne use of
s and public-

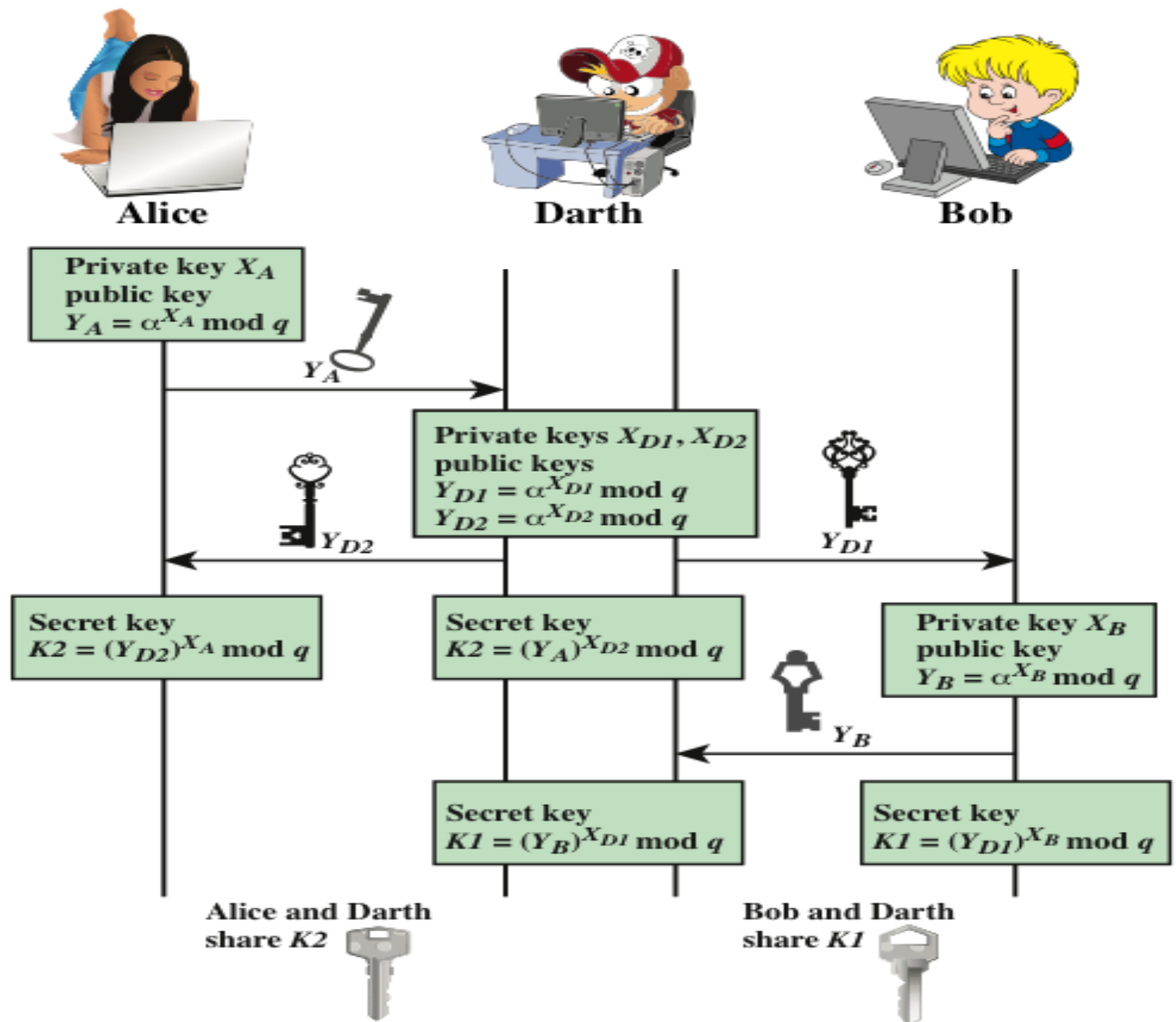


Figure 10.2 Man-in-the-Middle Attack

Elliptic Curve Arithmetic

- Most of the products and standards that use public-key cryptography for encryption and digital signatures use RSA
 - The key length for secure RSA use has increased over recent years and this has put a heavier processing load on applications using RSA
- Elliptic curve cryptography (ECC) is showing up in standardization efforts including the IEEE P1363 Standard for Public-Key Cryptography
- Principal attraction of ECC is that it appears to offer equal security for a far smaller key size
- Confidence level in ECC is not yet as high as that in RSA