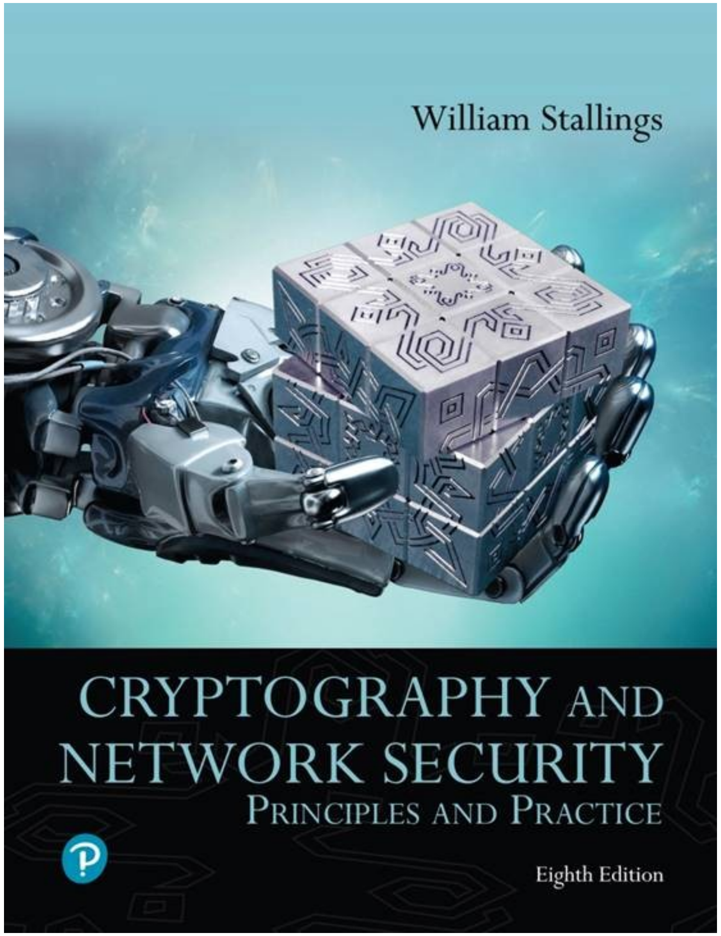
Cryptography and Network Security: Principles and Practice

Eighth Edition



Chapter 9

Public Key Cryptography and RSA

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.

Source: *Glossary of Key Information Security Terms*, NISTIR 7298.

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

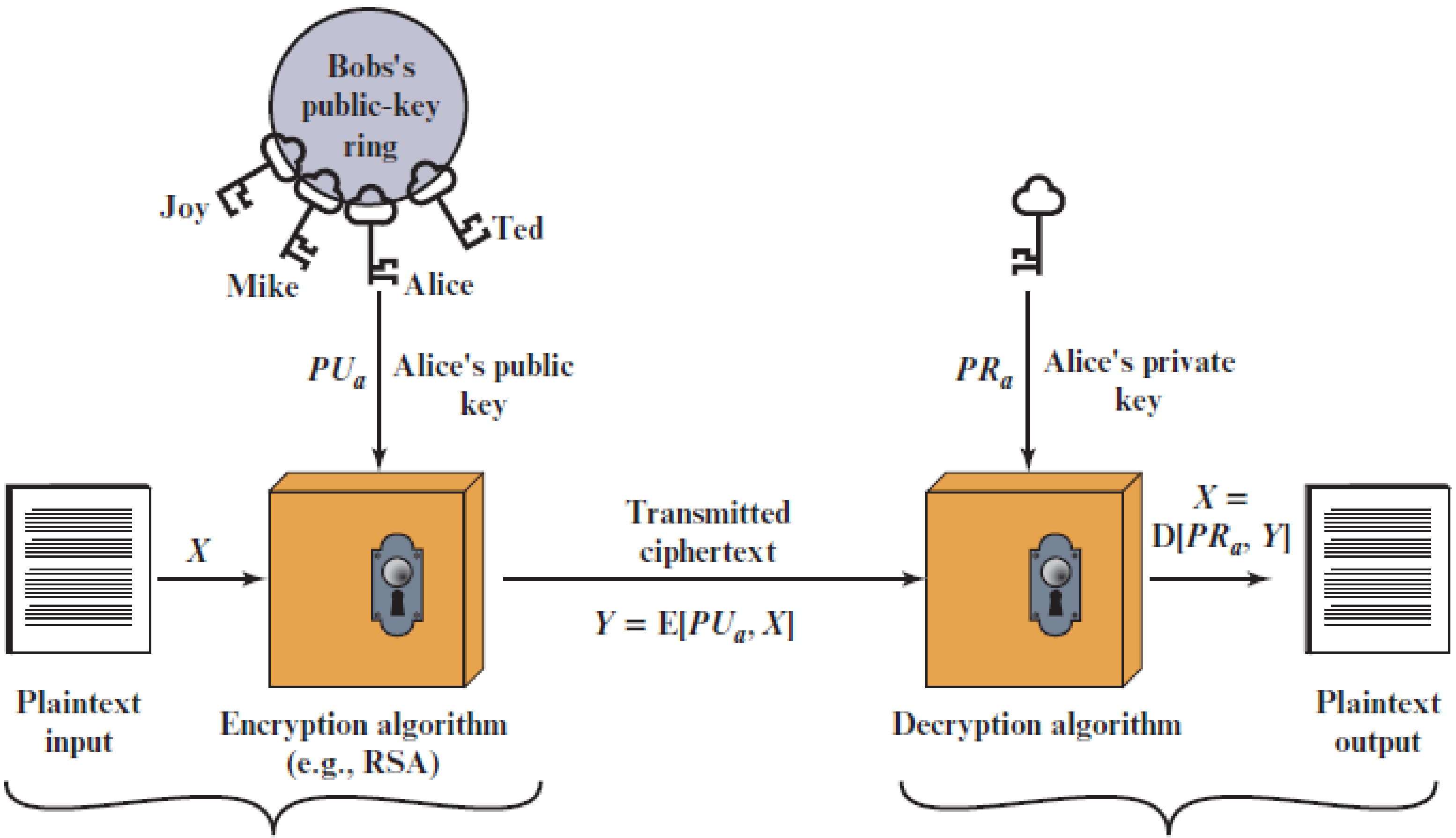
Principles of Public-Key Cryptosystems

- 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

- A public-key encryption scheme has six ingredients:
- Plaintext
  - **The readable message or data that is fed into the algorithm as input**
- Encryption algorithm
  - **Performs various transforma-tions on the plaintext**
- Public key
  - **Used for encryption or decryption**
- Private key
  - **Used for encryption or decryption**
- Ciphertext
  - **The scrambled message produced as output**
- Decryption algorithm
  - **Accepts the ciphertext and the matching key and produces the original plaintext**

Figure 9.1 Public-Key Cryptography (1 of 2)



(a) Encryption with public key

Figure 9.1 Public-Key Cryptography (2 of 2)

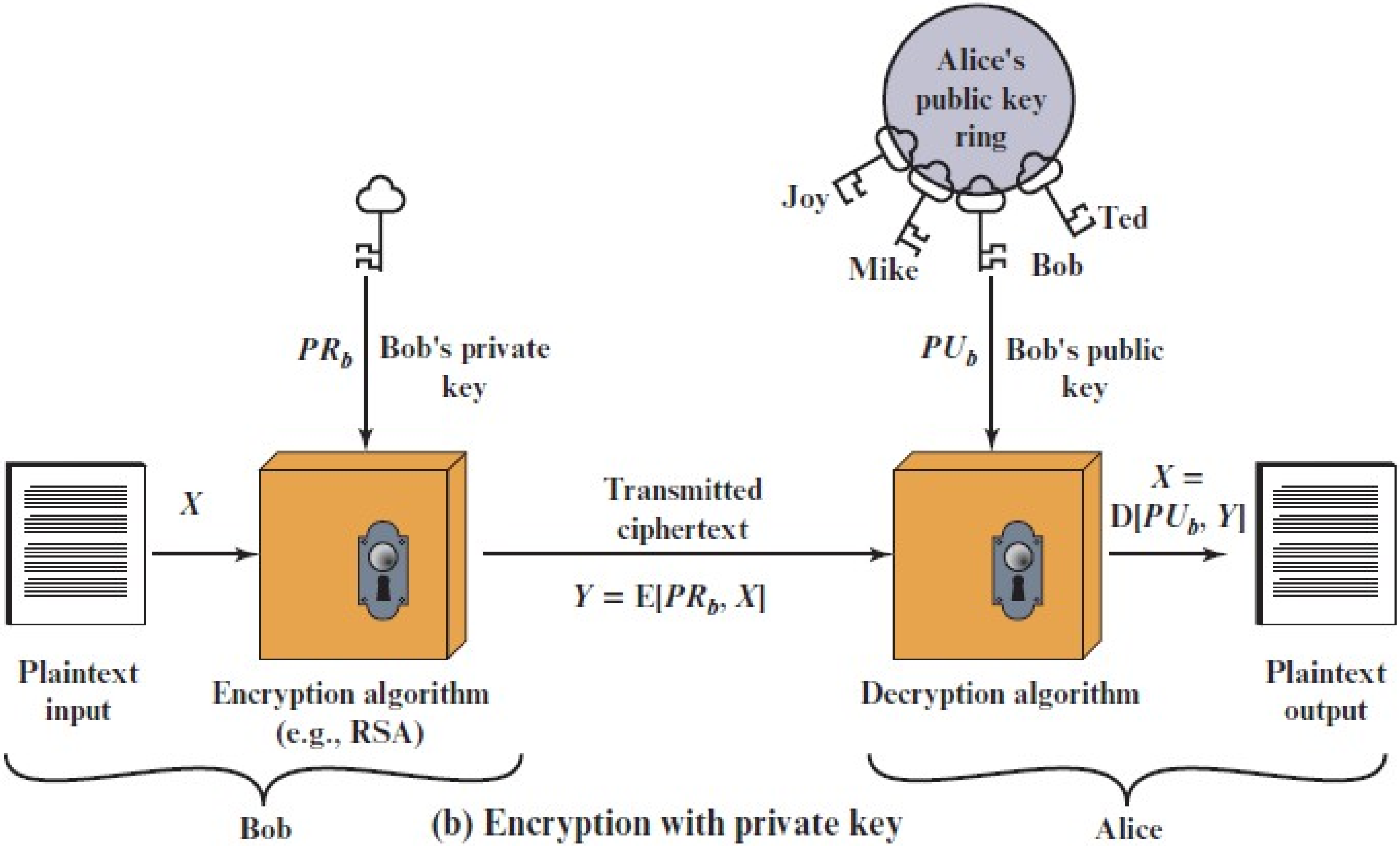
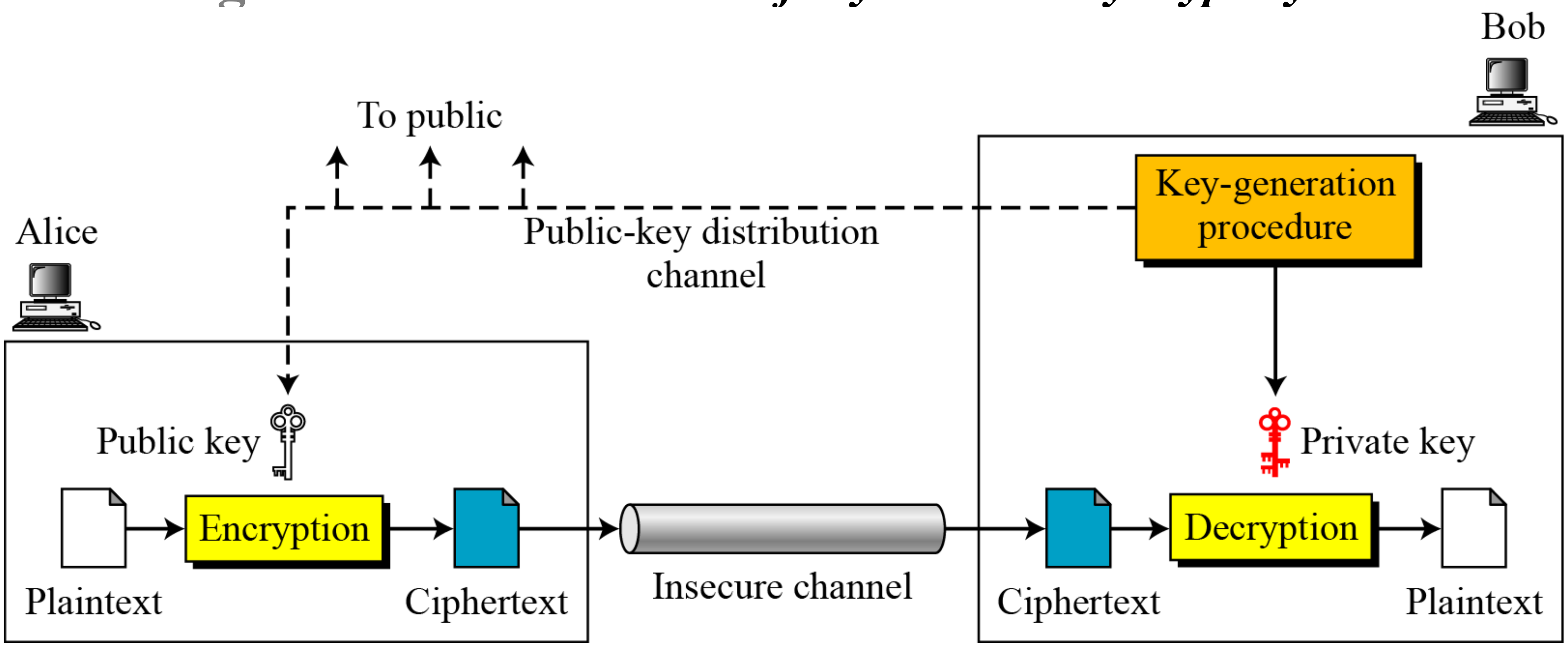


Figure 9.1 Public-Key Cryptography

10.1.2 General Idea

Figure 10.2 *General idea of asymmetric-key cryptosystem*



10.1.2 Continued

Plaintext/Ciphertext

Unlike in symmetric-key cryptography, plaintext and ciphertext are treated as integers in asymmetric-key cryptography.

Encryption/Decryption

$$\boxed{C = f\,(K_{public}\,,\,P) \qquad P = g(K_{private}\,,\,C)}$$

10.1.3 Need for Both

There is a very important fact that is sometimes misunderstood: The advent of asymmetric-key cryptography does not eliminate the need for symmetric-key cryptography.

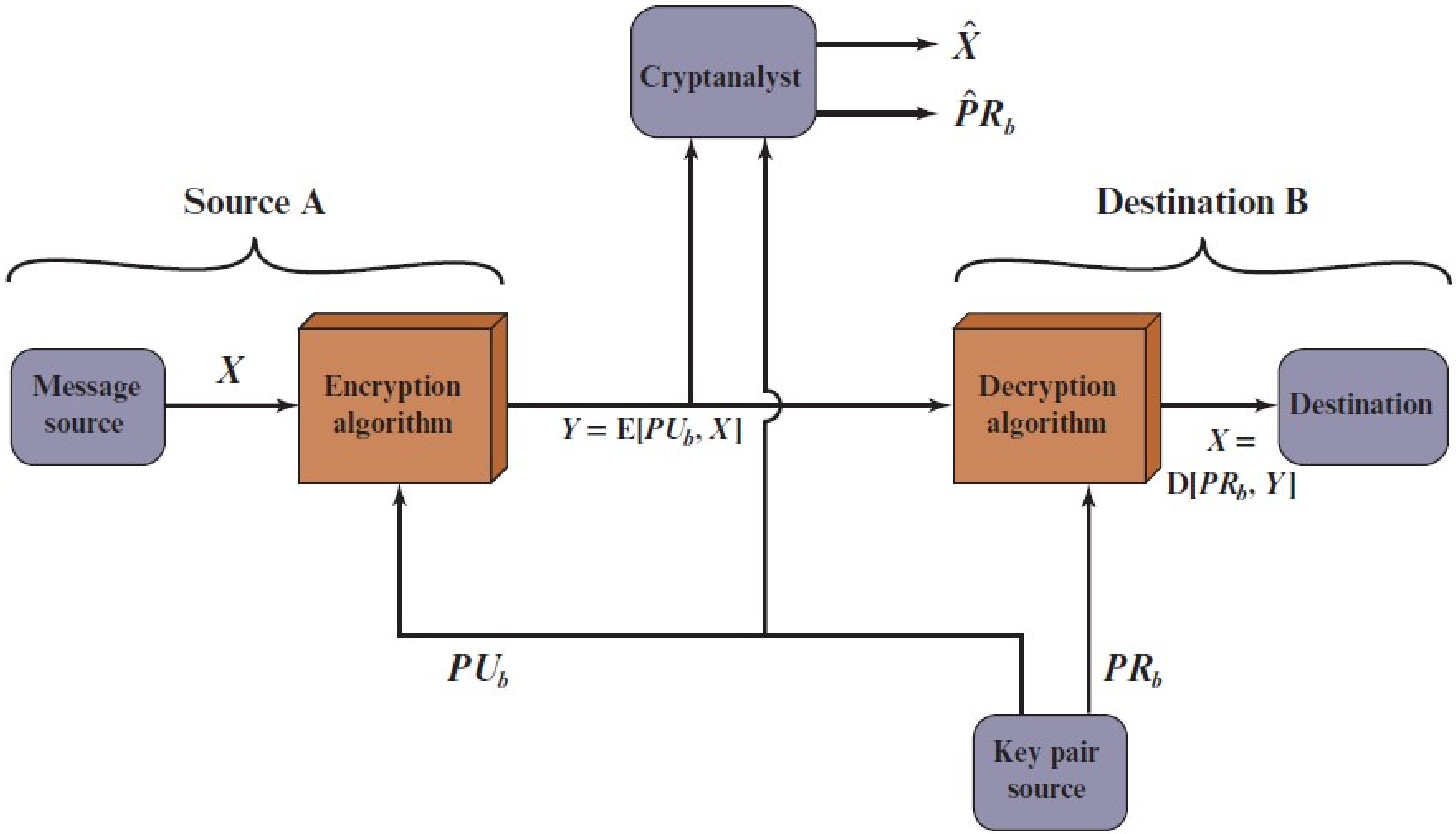
Table 9.2 Conventional and Public-key Encryption

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 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 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 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.

Pearson

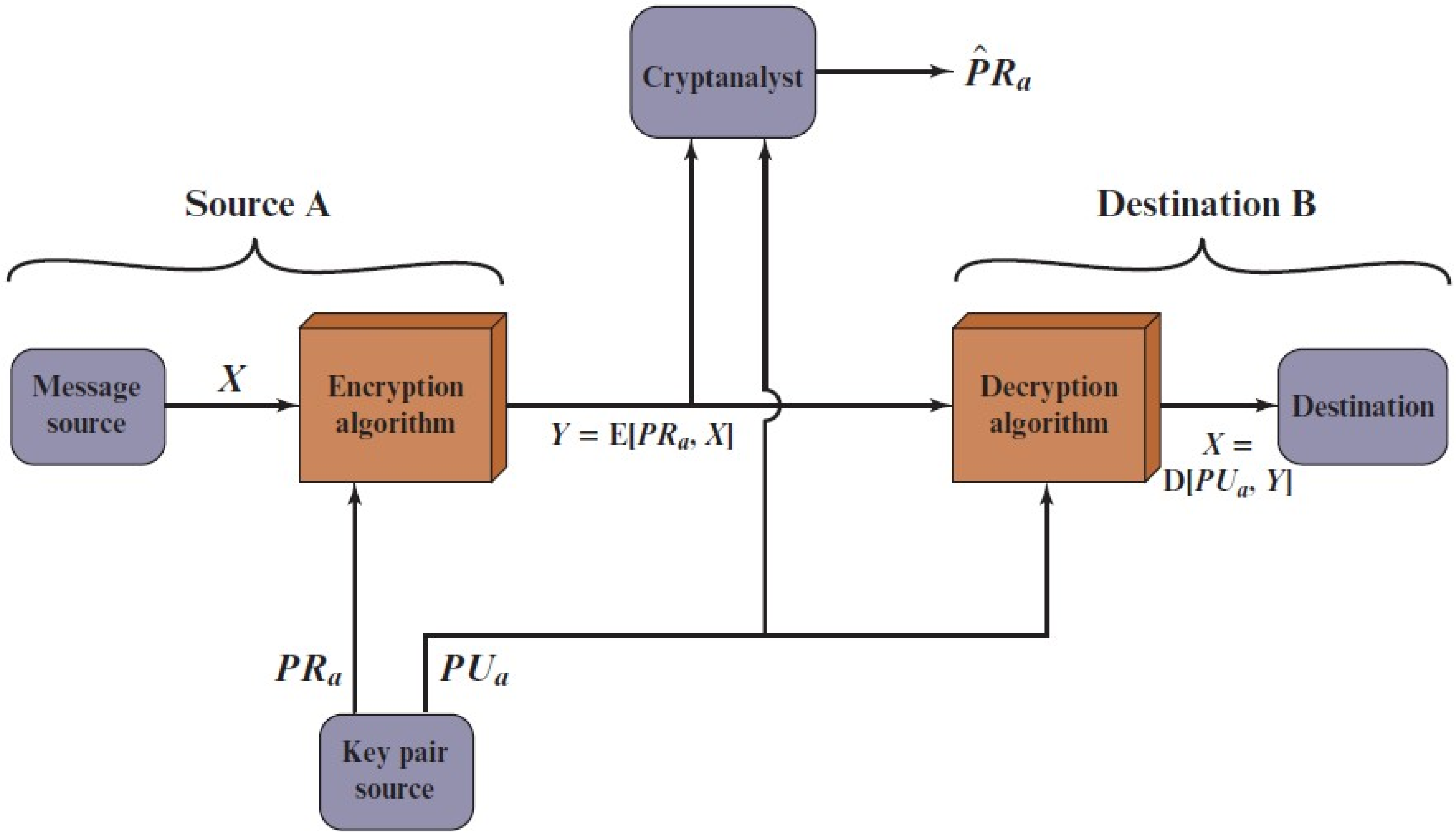
Public-Key Cryptosystem: Confidentiality

Figure 9.2 Public-Key Cryptosystem: Confidentiality



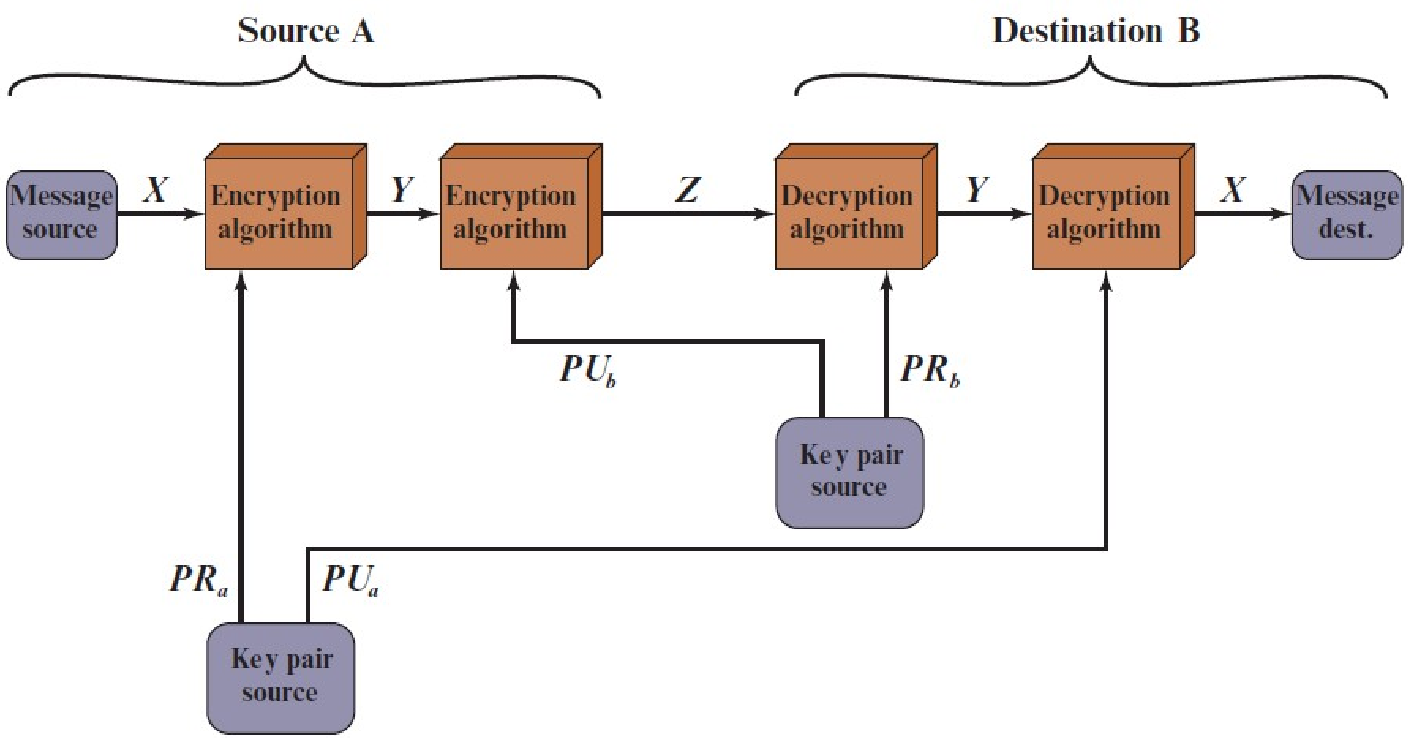
Public-Key Cryptosystem: Authentication

Figure 9.3 Public-Key Cryptosystem: Authentication



Public-Key Cryptosystem: Authentication and Secrecy

Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy



Applications for Public-Key Cryptosystems

- Public-key cryptosystems can be classified into three categories:
- Encryption/decryption
  - The sender encrypts a message with the recipient's public key
- Digital signature
  - The sender "signs" a message with its private key
- Key exchange
  - Two sides cooperate to exchange a session key
- 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

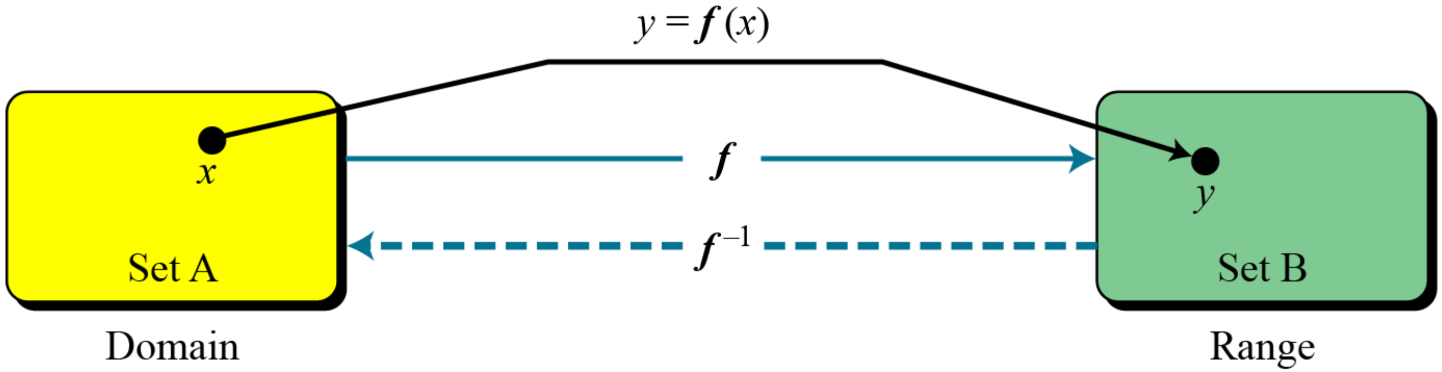
Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie–Hellman	No	No	Yes
DSS	No	Yes	No

10.1.4 Trapdoor One-Way Function

The main idea behind asymmetric-key cryptography is the concept of the trapdoor one-way function.

Functions

Figure 10.3 ***A function as rule mapping a domain to a range***



10.1.4 Continued

One-Way Function (OWF)

1. f is easy to compute.
2. f^{-1} is difficult to compute.

Trapdoor One-Way Function (TOWF)

3. Given y and a trapdoor, x can be computed easily.

10.1.4 Continued

Example 10. 1

When n is large, $n = p \times q$ is a one-way function. Given p and q , it is always easy to calculate n ; given n, it is very difficult to compute p and q. This is the factorization problem.

Example 10. 2

When n is large, the function $y = x^k \bmod n$ is a trapdoor one-way function. Given x, k, and n, it is easy to calculate y. Given y, k, and n, it is very difficult to calculate x. This is the discrete logarithm problem. However, if we know the trapdoor, k′ such that $k \times k' = 1 \bmod \phi(n)$, we can use $x = y^{k'} \bmod n$ to find x.

Public-Key Requirements (1 of 2)

- 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 Requirements (2 of 2)

- Need a trap-door one-way function
  - A one-way function is one that maps a domain into a range such that every function value has a unique inverse, 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
- A trap-door one-way function is a family of invertible functions f_k, 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

Pearson

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) Algorithm

- 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

RSA Algorithm

- 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*
- Encryption and decryption are of the following form, for some plaintext block *M* and ciphertext block C

$C = M^e \bmod n$

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

- Both sender and receiver must know the value of *n*
- The sender knows the value of *e*, and only the receiver knows the value of *d*
- This is a public-key encryption algorithm with a public key of *PU={e,n}* and a private key of *PR={d,n}*

Algorithm Requirements

- For this algorithm to be satisfactory for public-key encryption, the following requirements must be met:
  1. It is possible to find values of e, d, n such that $M^{ed} \bmod n = M$ for all $M < n$
  2. It is relatively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M < n$
  3. It is infeasible to determine d given e and n

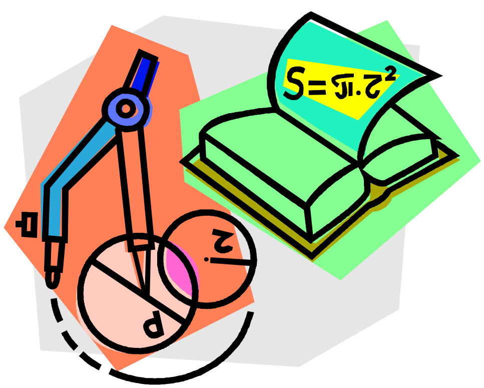


Figure 9.5 The RSA Algorithm

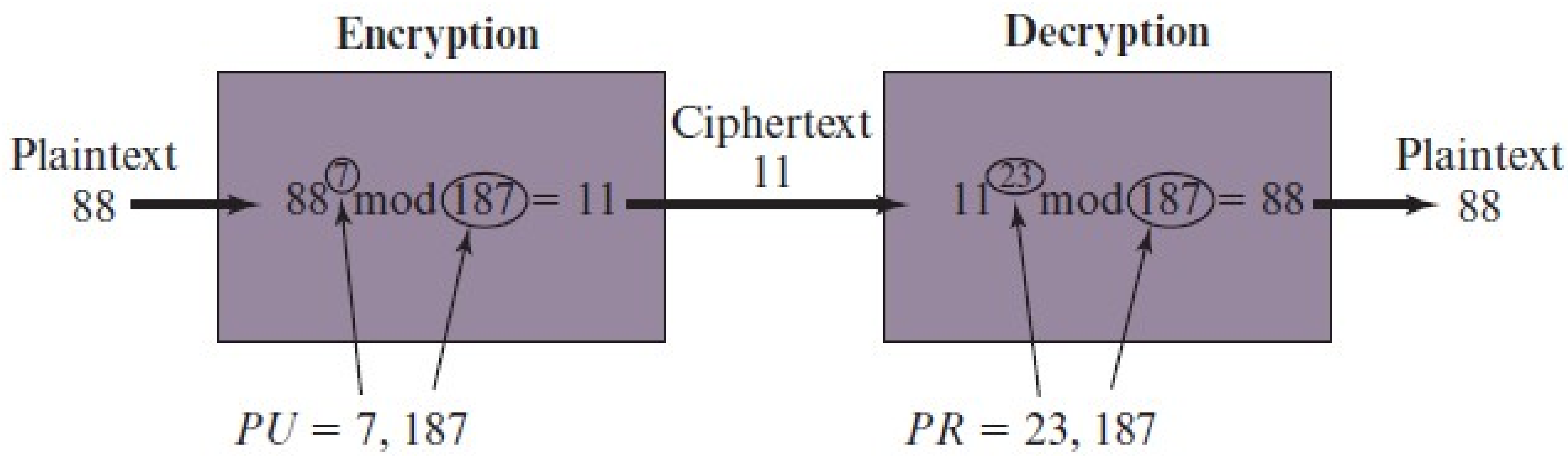
Key Generation by Alice	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

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:	$M = C^d \bmod n$

Example of RS A Algorithm

Figure 9.6 Example of RS A Algorithm



10.2.6 Continued

Example 10. 8

Here is a more realistic example. We choose a 512-bit p and q, calculate n and $\phi(n)$, then choose e and test for relative primeness with $\phi(n)$. We then calculate d. Finally, we show the results of encryption and decryption. The integer p is a 159-digit number.

$p =$	96130345313583504574191581280615427909309845594996215822583150879647940455056470638491257160180347503120986666064924201918087806674210960633542199266612009

$q =$	120601919572314469182767942044508960015559250546370339360617983217314821484837646592153894532091752252732268301071206956046025138871455249690003596600456017

10.2.6 Continued

Example 10. 8 *Continued*

The modulus $n = p \times q$. It has 309 digits.

$n =$	1159350417396761496889250986461588752377145737545414477548552613761478854083263508172768788159683251684688493006254857641112501624145523391829271625076567727274600970827141277304349605005563472745666280600999240371029914244722922157727985317270338393813346926841373276220009666766718318310883734208234443709 53

$\phi(n) = (p - 1)(q - 1)$ has 309 digits.

$\phi(n) =$	1159350417396761496889250986461588752377145737545414477548552613761478854083263508172768788159683251684688493006254857641112501624145523391829271625076567510542336084929167520344826279881175547876570139234444057169895817281960982263610754672118646121713591073586406140088851702653772772644673410662438576641 28

10.2.6 *Continued*

Example 10. 8 *Continued*

Bob chooses e = 35535 (the ideal is 65537) and tests it to make sure it is relatively prime with φ(n). He then finds the inverse of *e* modulo φ(n) and calls it *d*.

$e =$	35535
$d =$	580083028600377639360936612896779175946690620896509621804228661113 805938528223587317062869100300217108590443384021707298690876006115 306202524959884448047568240966247081485817130463240644077704833134 010850947385295645071936774061197326557424237217617674620776371642 0760033708533328853214470885955136670294831

10.2.6 *Continued*

Example 10.8 *Continued*

Alice wants to send the message "THIS IS A TEST", which can be changed to a numeric value using the 00−26 encoding scheme (26 is the space character).

P =	1907081826081826002619041819

The ciphertext calculated by Alice is C = P^{e}, which is

C =	47530912364622682720636555061054518094237179607049171652323924305445296061319932856661784341835911415119741125200568297979457173603610127821884789274156609048002350719071527718591497518846588863210114835410336165789846796838676373376577746562507928052114814184404814184430812773059004692874248559166462108656

10.2.6 *Continued*

Example 10. 8 *Continued*

Bob can recover the plaintext from the ciphertext using $P = C^d$, which is

P =	1907081826081826002619041819

The recovered plaintext is "THIS IS A TEST" after decoding.

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

Figure 9.8 Algorithm for Computing a^b mod n

Note: The integer b is expressed as a binary number $b_k b_{k-1} \ldots b_0$

```
c ← 0; f ← 1
for i ← k downto 0
    do  c ← 2 × c
        f ← (f × f) mod n
    if  b_i = 1
        then c ← c + 1
             f ← (f × a) mod n
return f
```

Exponentiation

Fast Exponentiation

Figure 9.6 *The idea behind the square-and-multiply method*

$$y = a^{\boxed{x_{n_b-1} \times 2^{n_b-1} + x_{n_b-2} \times 2^{n_b-2} + \cdots + x_1 \times 2^1 + x_0 \times 2^0}}$$

in which x_i is 0 or 1

$$\downarrow$$

$$y = \boxed{\boxed{a^{2^{n_b-1}}} \text{ or } \boxed{1}} \times \boxed{\boxed{a^{2^{n_b-2}}} \text{ or } \boxed{1}} \times \cdots \times \boxed{\boxed{a^2} \text{ or } \boxed{1}} \times \boxed{\boxed{a} \text{ or } \boxed{1}}$$

Example:

$$\boxed{y = a^9 = a^{1001_2} = a^8 \times 1 \times 1 \times a}$$

Continued

Algorithm 9.7 *Pseudocode for square-and-multiply algorithm*

```
Square_and_Multiply (a, x, n)
{
    y ← 1
    for (i ← 0 to n_b − 1)                  // n_b is the number of bits in x
    {
        if (x_i = 1)    y ← a × y  mod n     // multiply only if the bit is 1

        a ← a^2 mod n                        // squaring is not needed in the last iteration
    }
    return y
}
```

Continued

Example 9.45

Figure 9.7 shows the process for calculating $y = a^x$ using the Algorithm 9.7 (for simplicity, the modulus is not shown). In this case, $x = 22 = (10110)_2$ in binary. The exponent has five bits.

Figure 9.7 *Demonstration of calculation of a^{22} using square-and-multiply method*

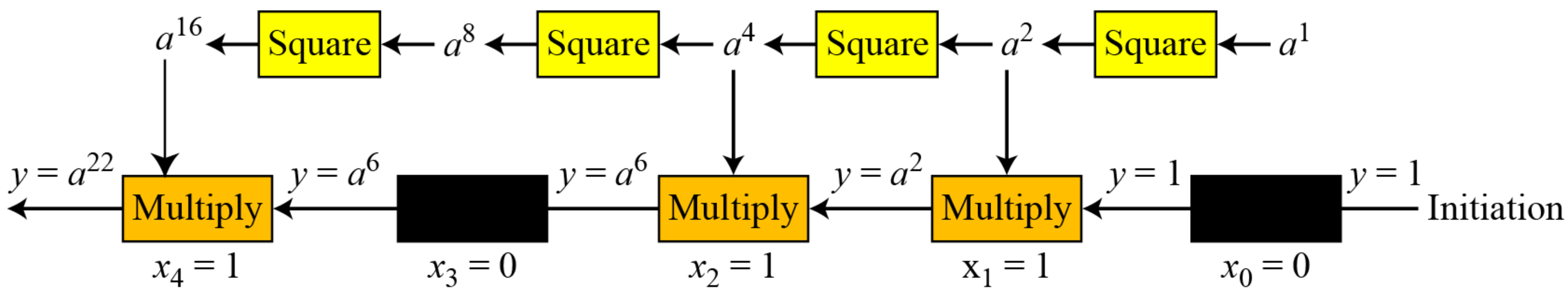


Table 9.4 Result of the Fast Modular Exponentiation Algorithm for a^b mod n, where $a = 7$, $b = 560 = 1000110000$, and $n = 561$

I	9	8	7	6	5	4	3	2	1	0
B_i	1	0	0	0	1	1	0	0	0	0
C	1	2	4	8	17	35	70	140	280	560
F	7	49	157	526	160	241	298	166	67	1

Efficient Operation Using the Public Key

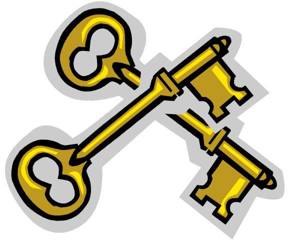
- To speed up the operation of the RSA algorithm using the public key, a specific choice of *e* is usually made
- The most common choice is 65537 ($2^{16} + 1$)
  - Two other popular choices are e=3 and e=17
  - Each of these choices has only two 1 bits, so the number of multiplications required to perform 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 mod $(p - 1)$ and d mod $(q - 1)$ can be precalculated
  - End result is that the calculation is approximately four times as fast as evaluating $M = C^d \bmod n$ directly

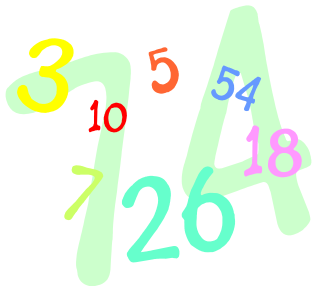
Key Generation

- 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

- Five possible approaches to attacking RSA are:
  - Brute force
    - Involves trying all possible private keys
  - Mathematical attacks
    - There are several approaches, all equivalent in effort to factoring the product of two primes
  - Timing attacks
    - These depend on the running time of the decryption algorithm
  - Hardware fault-based attack
    - This involves inducing hardware faults in the processor that is generating digital signatures
  - Chosen ciphertext attacks
    - This type of attack exploits properties of the RSA algorithm

10.2.1 Introduction

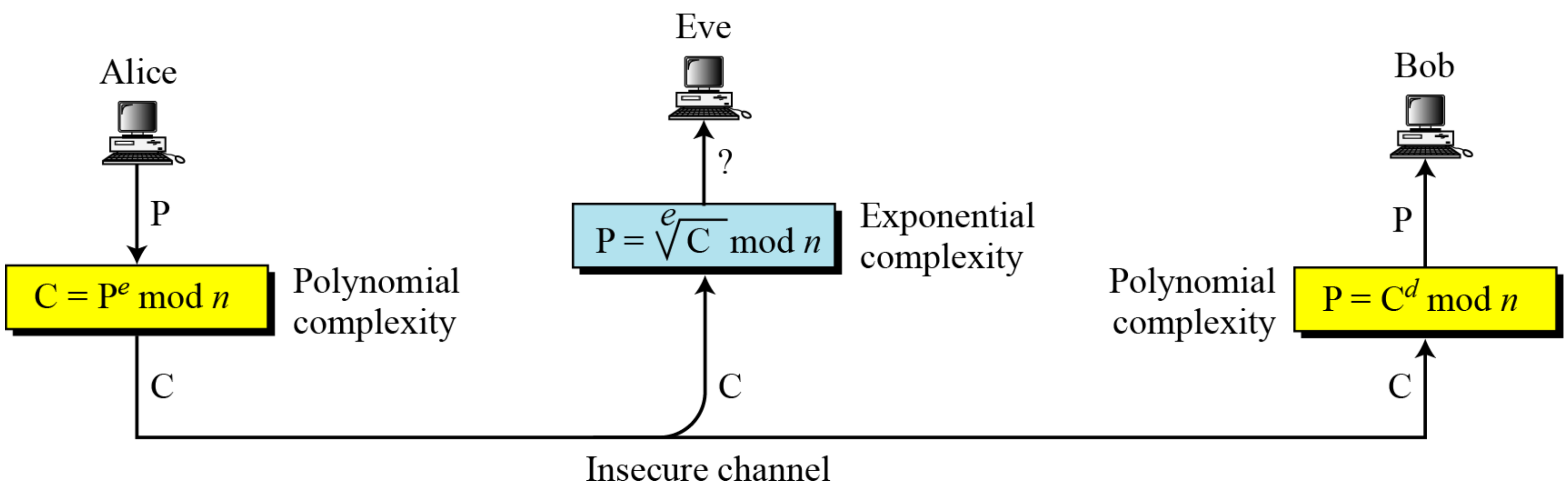


Figure 10.5 *Complexity of operations in RSA*

RSA uses modular exponentiation for encryption/decryption; To attack it, Eve needs to calculate $\sqrt[e]{C} \bmod n$.

Factoring Problem

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

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

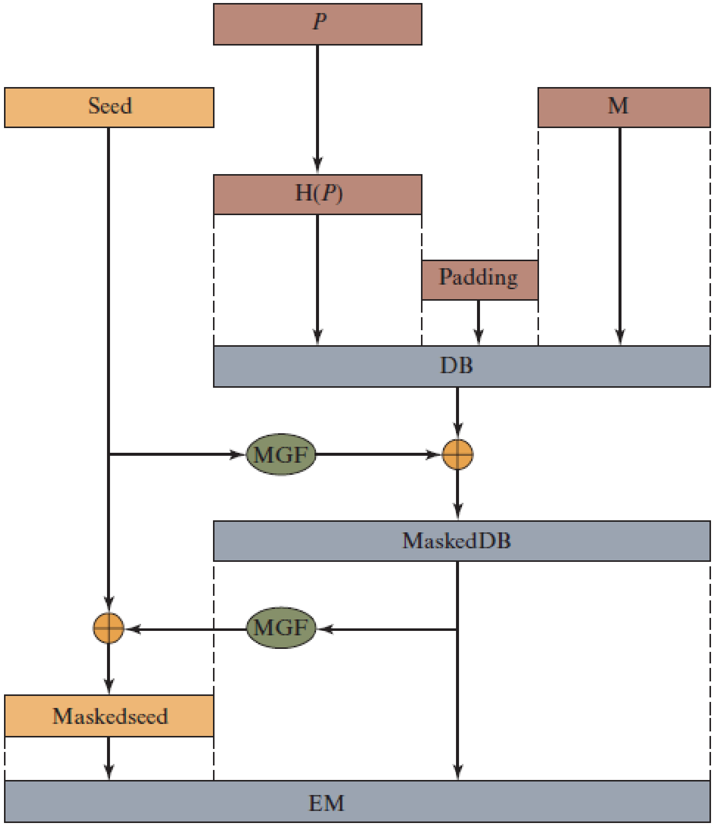
- An attack on a processor that is generating RSA digital signatures
  - Induces faults in the signature computation by reducing the power to the processor
  - The faults cause the software to produce invalid signatures which can then be analyzed by the attacker to recover the private key
- The attack 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)

Pearson

Figure 9.9 Encryption Using Optimal Asymmetric Encryption Padding (OAEP)



P = encoding parameters
M = message to be encoded
H = hash function
DB = data block
MGF = mask generating function
EM = encoded message

Summary

- 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

Copyright