Cryptography and Network Security

Eighth Edition

by William Stallings

Chapter 9

Public Key Cryptography and RSA

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 Key Distribution Center (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

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

Public-Key Cryptosystems

- A public-key encryption scheme has six ingredients:

Plaintext	Encryption algorithm	Public key	Private key	Ciphertext	Decryption algorithm
The readable message or data that is fed into the algorithm as input	Performs various transformations on the plaintext	Used for encryption or decryption	Used for encryption or decryption	The scrambled message produced as output	Accepts the ciphertext and the matching key and produces the original plaintext

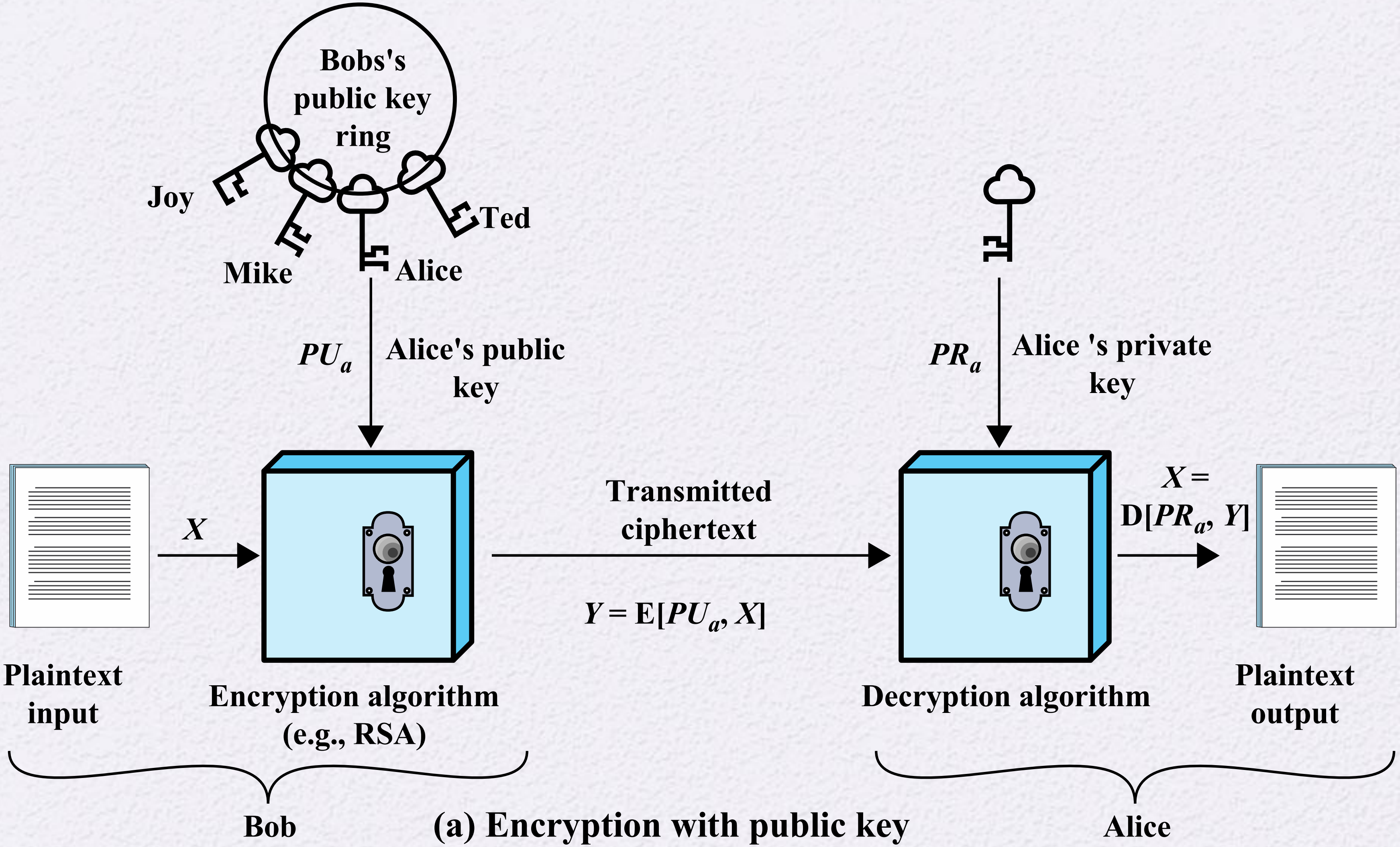


Figure 9.1 Public-Key Cryptography

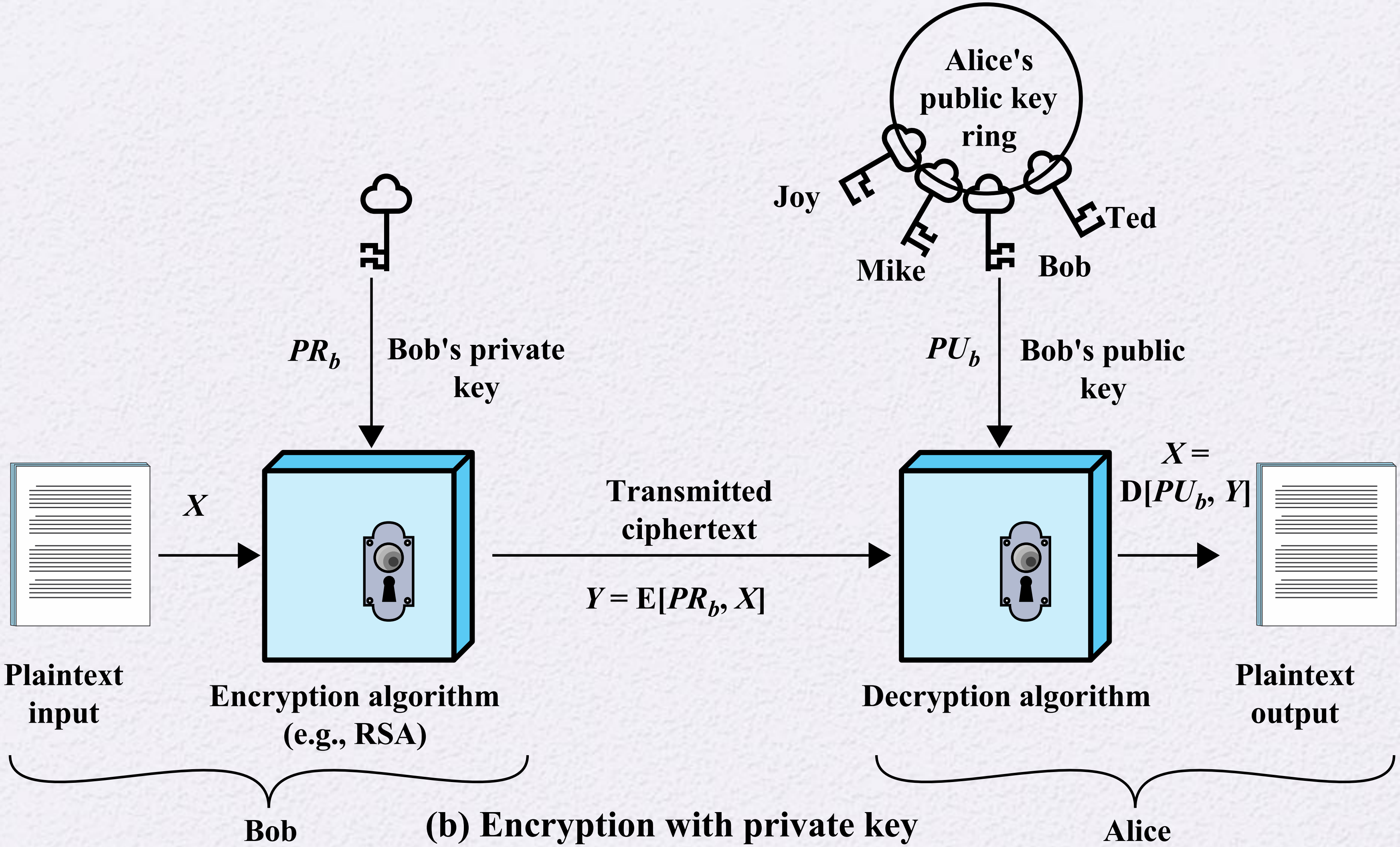
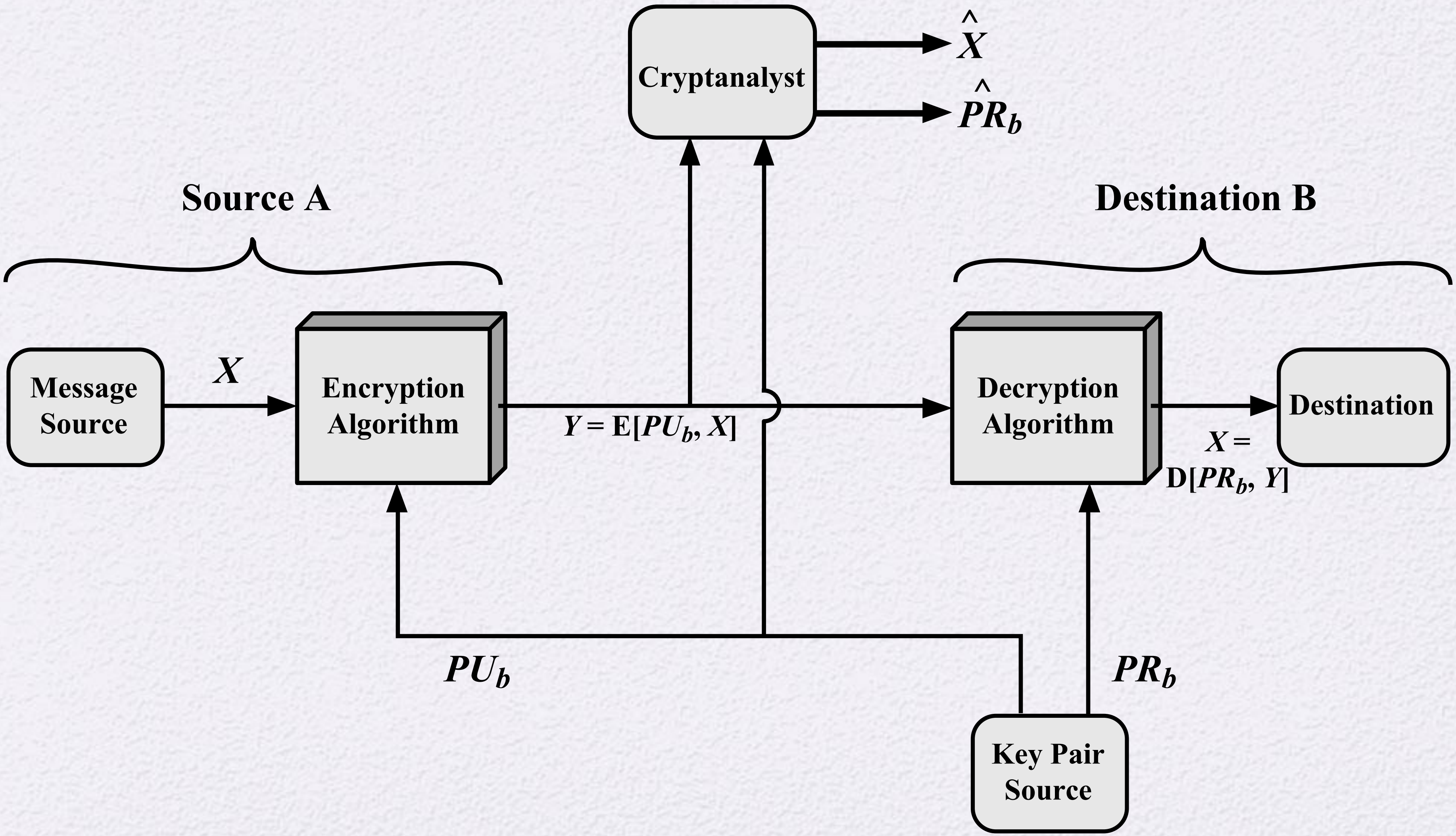


Figure 9.1 Public-Key Cryptography

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 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.

Public-Key Cryptosystem: Confidentiality



Public-Key Cryptosystem: Authentication

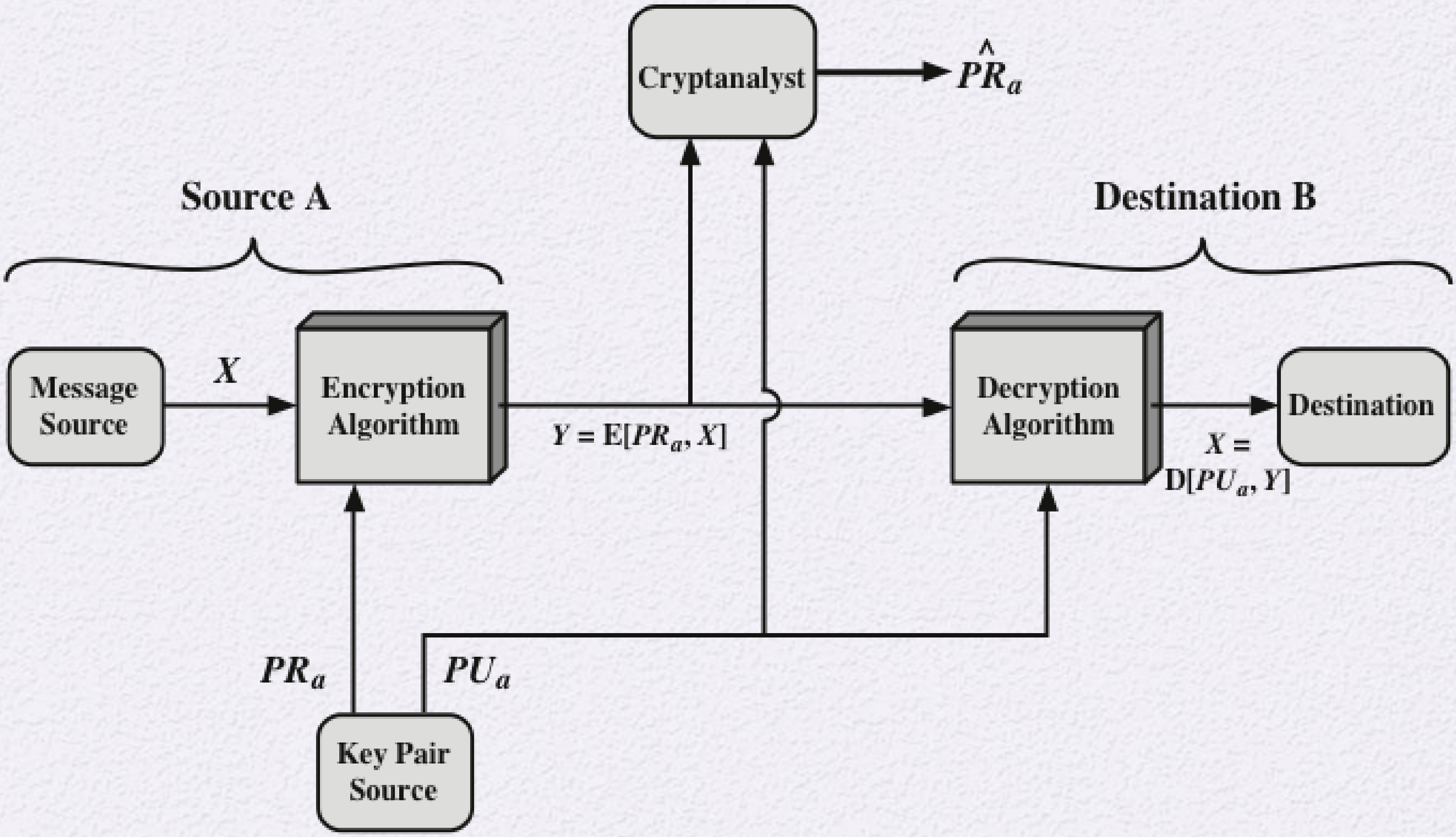


Figure 9.3 Public-Key Cryptosystem: Authentication

Public-Key Cryptosystem: Authentication and Secrecy

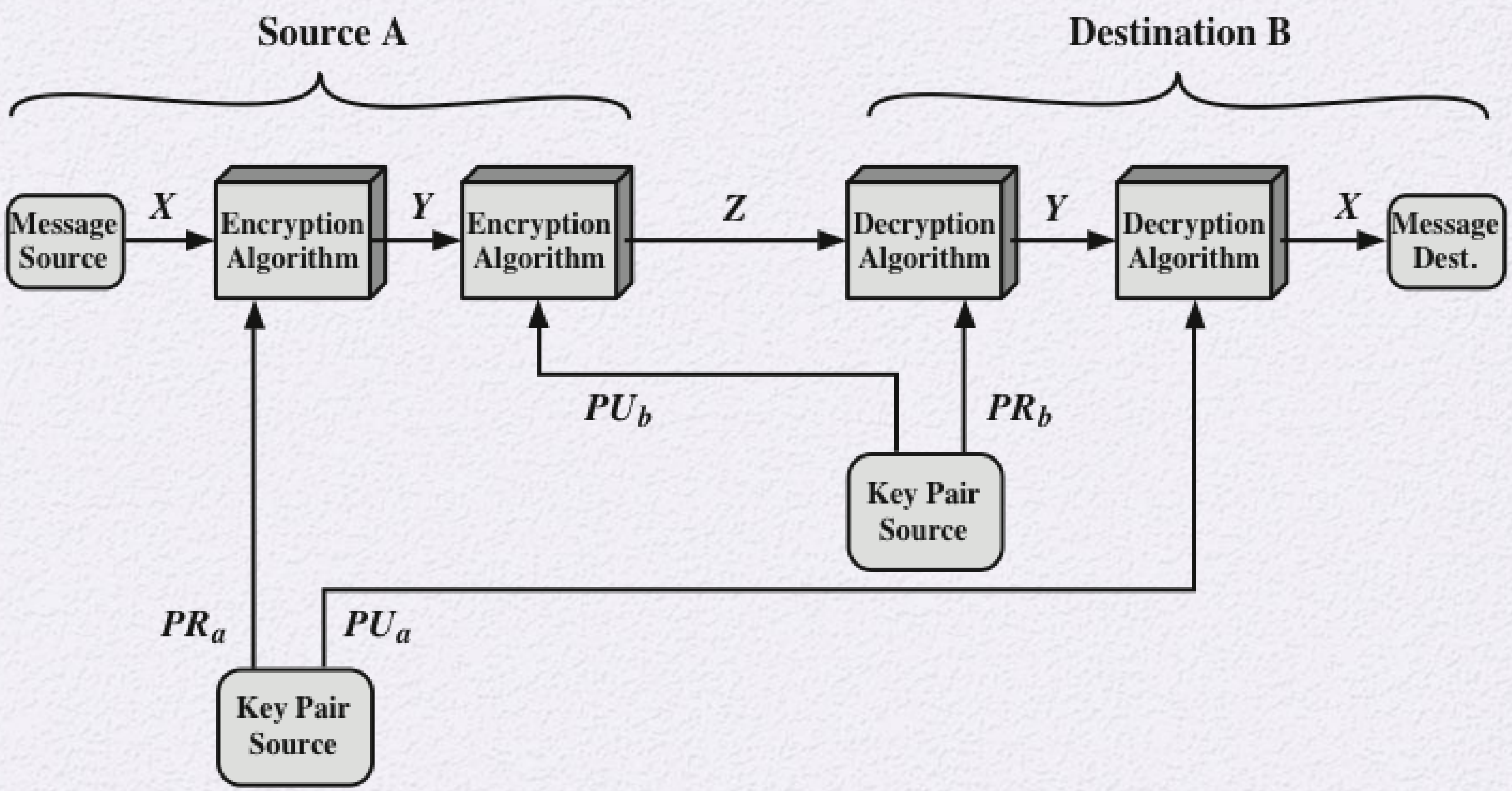


Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy

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

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

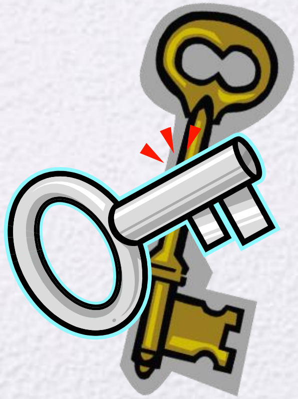
- Conditions that these algorithms must fulfill:
  - It is computationally easy for a party B to generate a pair (public-key PU_b, private key PR_b)
  - It is computationally easy for a sender A, knowing the public key and the message to be encrypted, to generate the corresponding ciphertext
  - It is 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 Requirements

- 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

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

- Select prime number $\mathbf{p}, \boldsymbol{q}$ such that $\mathbf{p} \neq \boldsymbol{q}$
- Calculate $\boldsymbol{n} = \boldsymbol{p} \times \boldsymbol{q}$
- Calculate $\boldsymbol{\varphi}(\boldsymbol{n}) = (\boldsymbol{p} - \mathbf{1})(\boldsymbol{q} - \mathbf{1})$
- Select integer '$\boldsymbol{e}$' such that $\mathbf{GCD}(\boldsymbol{\varphi}(\boldsymbol{n}), \boldsymbol{e}) = \mathbf{1}$ and $1 < \boldsymbol{e} < \boldsymbol{\varphi}(\boldsymbol{n})$
- Calculate '$\boldsymbol{d}$' such that $\boldsymbol{e} \times \boldsymbol{d} \equiv \mathbf{1} \bmod \boldsymbol{\varphi}(\boldsymbol{n})$
- Public key $\boldsymbol{PU} = \{\boldsymbol{e}, \boldsymbol{n}\}$
- Private key $\boldsymbol{PR} = \{\boldsymbol{d}, \boldsymbol{n}\}$

RSA Algorithm

- Plaintext is encrypted in blocks **M** with each block having a binary value less than some number ***n***
- Encryption by Bob using Alice's Public key
  - Plaintext: $M < n$
  - Cipher text: $C = M^e \bmod n$
- Decryption by Alice using Alice's Private key
  - Cipher text: C
  - Plaintext: $M = C^d \bmod n$

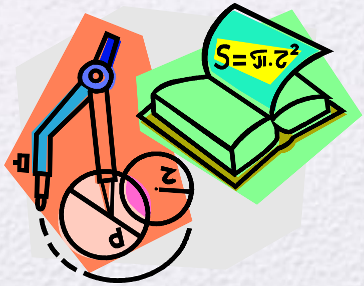
$$= (M^e)^d \bmod n$$

$$= M^{ed} \bmod n$$

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*



Key Generation by Alice

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

Example of RSA Algorithm

- Perform Encryption for plaintext 88 using the RSA algorithm with the values p = 11, q = 17, and e =7.
- We have prime number $\mathbf{p} = \mathbf{11}, \boldsymbol{q} = \mathbf{17}$ such that $\mathbf{p} \neq \boldsymbol{q}$
- Calculate $\boldsymbol{n} = \boldsymbol{p} \times \boldsymbol{q} = \mathbf{11} \times \mathbf{17} = \mathbf{187}$
- Calculate $\boldsymbol{\varphi}(\boldsymbol{n}) = (\boldsymbol{p} - \mathbf{1})(\boldsymbol{q} - \mathbf{1}) = (\mathbf{11} - \mathbf{1})(\mathbf{17} - \mathbf{1}) = \mathbf{160}$
- Given integer '$\boldsymbol{e} = \mathbf{7}$' such that $\mathbf{GCD}(\mathbf{160}, \mathbf{7}) = \mathbf{1}$ and $1 < \mathbf{7} < \mathbf{40}$

Example of RSA Algorithm

- Perform Encryption for plaintext 88 using the RSA algorithm with the values p = 11, q = 17, and e =7.

- Calculate 'd' such that $\boldsymbol{e \times d \equiv 1 \bmod \varphi(n)}$
  - $\boldsymbol{7 \times d \equiv 1 \bmod 160}$ (use extended Euclidean algorithm to find $\boldsymbol{d}$)
  - $\boldsymbol{d = 23}$

- Public key $\boldsymbol{PU = \{e, n\} = \{7, 187\}}$

- Private key $\boldsymbol{PR = \{d, n\} = \{23, 187\}}$

Example of RSA Algorithm

- Perform Encryption for plaintext 88 using the RSA algorithm with the values p = 11, q = 17, and e =7.

- $\boldsymbol{M = 88 < 187}$

- $\boldsymbol{C = M^e \bmod n = 88^7 \bmod 187 = 11}$

Example of RSA Algorithm

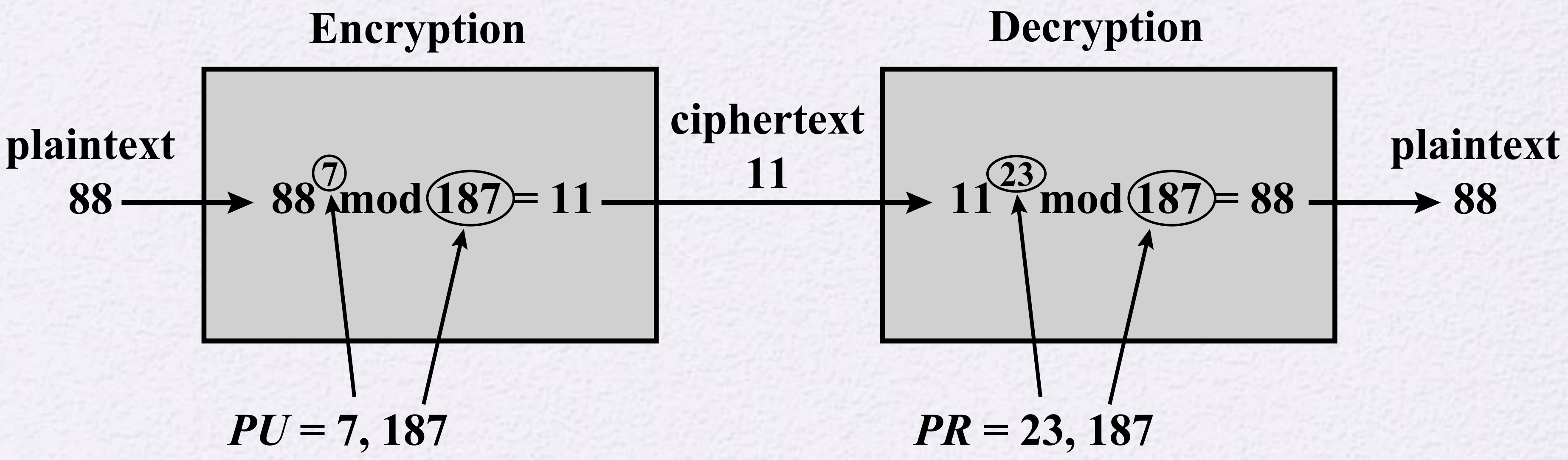


Figure 9.6 Example of RSA Algorithm

RSA Real World Example

- **p**
  1213107243921127189732367153161244042847242763370141092563454931230196437304208561932419736532241686654101705736136521417171171379797429933487106282980354 1

- **q**
  120275242554787488859562207937345121287333878036820754336538999839551798509887978998691469008091316111533468170508320960221601463663463918 12470987105415233

- **n**
  145906768007583323230186939349070635292401872375357164399581871019873438799005358938369571402670149802121818086292467422828157022922076746906543401224889672472407926969987100581290103199317858753663710862357656510507883714297115637342788911463535102712032765166518411726859837988672111837205085526346618740053

RSA Real World Example

- **ϕ(n)**

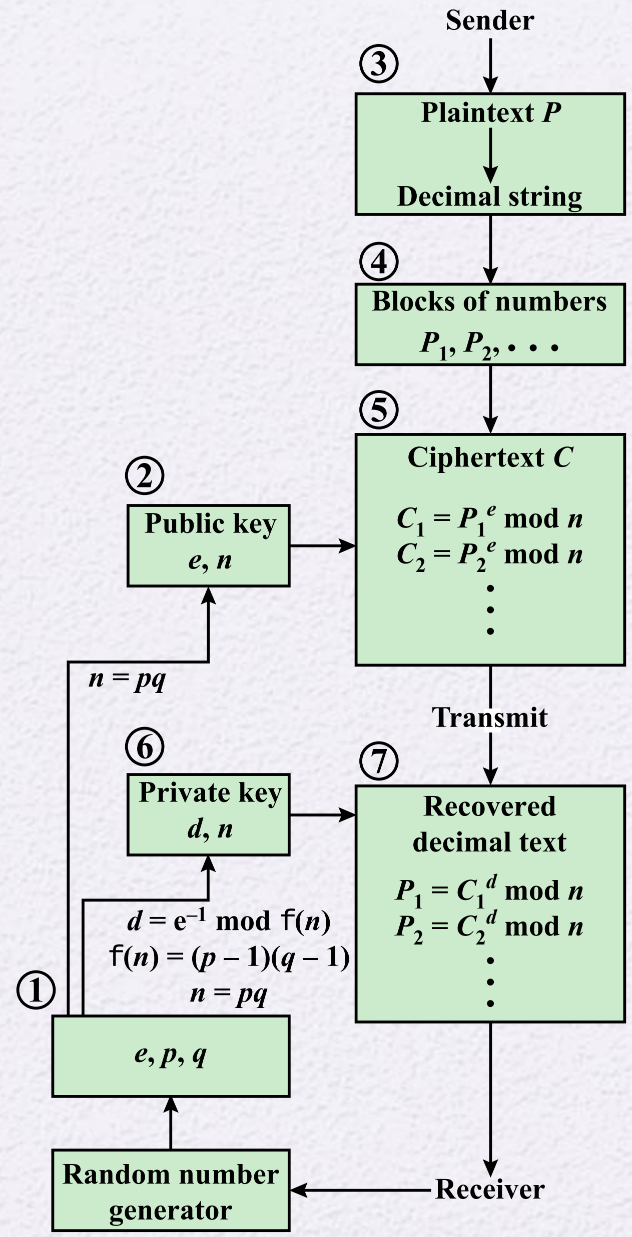
1459067680075833232301869393490706352924018723753571643995818710198734387990053589383695714026701498021218180862924674228281570229220767469065434012248896483138112322799663173013977785236530154784827347887129722205858745715289160645926971811926897116355507080264399952954964411681194751651393818429668352128
0

- **e** 65537 (a most common choice)

- **d**
  89489425009274444368228545921773093919669586065884257445497854456487674839
  62981839093494197326287961679797060891728367987549933157416111385408881327
  54881105882471930775825272784379065040156806234235500672400424666656542323
  83502922215493623289472138866445818789127946123407807725702626644091036502
  372545139713

Figure 9.7

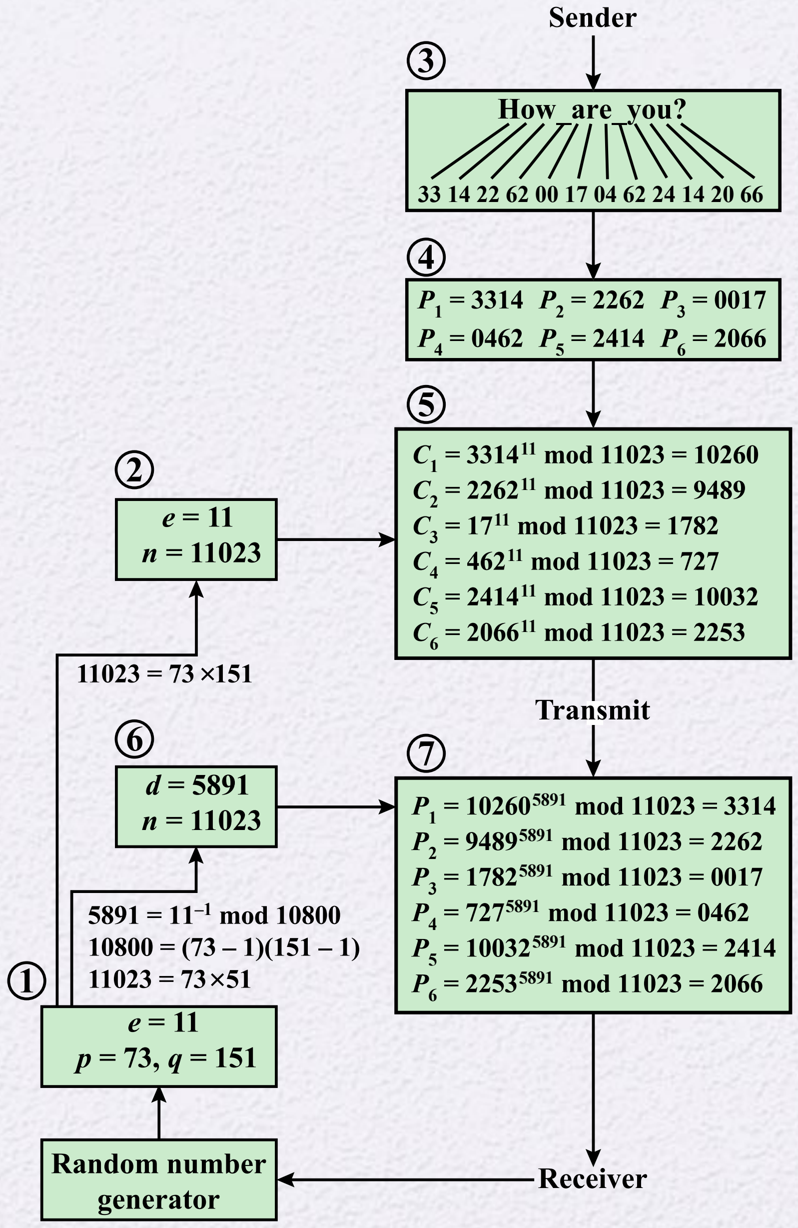
RSA Processing of Multiple Blocks



(a) General approach

Figure 9.7

RSA Processing of Multiple Blocks



(b) Example

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

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)

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

Extra Example of RSA Algorithm

- Perform Encryption for the plaintext 20 using the RSA algorithm with the values p = 5, q = 11, and e =13.

- We have prime number $\mathbf{p = 5}, \boldsymbol{q} = \mathbf{11}$ such that $\mathbf{p} \neq \boldsymbol{q}$

- Calculate $\boldsymbol{n} = \boldsymbol{p} \times \boldsymbol{q} = \mathbf{5} \times \mathbf{11} = \mathbf{55}$

- Calculate $\boldsymbol{\varphi}(\boldsymbol{n}) = (\boldsymbol{p} - \mathbf{1})(\boldsymbol{q} - \mathbf{1}) = (\mathbf{5} - \mathbf{1})(\mathbf{11} - \mathbf{1}) = \mathbf{40}$

- Given integer '$\boldsymbol{e} = \mathbf{13}$' such that $\mathbf{GCD}(\mathbf{40}, \mathbf{13}) = \mathbf{1}$ and $1 < \mathbf{13} < \mathbf{40}$

Extra Example of RSA Algorithm

- Perform Encryption for the plaintext 20 using the RSA algorithm with the values p = 5, q = 11, and e =13.

- Calculate 'd' such that $\boldsymbol{e \times d \equiv 1 \bmod \varphi(n)}$
  - $\boldsymbol{13 \times d \equiv 1 \bmod 40}$ (use extended Euclidean algorithm to find $\boldsymbol{d}$)
  - $\boldsymbol{d = 37}$

- Public key $\boldsymbol{PU = \{e, n\} = \{13, 55\}}$

- Private key $\boldsymbol{PR = \{d, n\} = \{37, 55\}}$

Extra Example of RSA Algorithm

- Perform Encryption for the plaintext 20 using the RSA algorithm with the values p = 5, q = 11, and e =13.

- $\boldsymbol{M = 20 < 55}$

- $\boldsymbol{C = M^e \bmod n = 20^{13} \bmod 55 = 25}$