Asymmetric Cryptography, Operations and Security of RSA

ASYMMETRIC CRYPTOGRAPHY CONCEPTS

RSA ENCRYPTION, DECRYPTION, AND KEY GENERATION OPERATIONS

OAEP PADDING AND ATTACKS TO RSA

Learning Outcomes of This Topic

Describe the principles and requirements of a public-key cryptosystem

Perform the encryption, decryption and key generation operations of the RSA algorithm

Describe attack approaches to RSA

Asymmetric Cryptography

Overview

Overview of Asymmetric Cryptography

- Components of Asymmetric Cryptography
- Applications of Public-key Cryptosystems
- Requirements for Public Key Cryptosystems
- Attacks to Public Key Cryptography

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 communication** in general **without having to trust a key distribution centre** with your key
- Digital signature: How to **verify** that a message **comes intact from the claimed sender**

Public-Key Cryptosystems

Public-key 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**

Public-Key Cryptosystems

A public-key encryption scheme has six components

Plaintext
The readable message or data that is fed into the algorithm as input

Encryption algorithm
Performs various transformations on the plaintext using a key

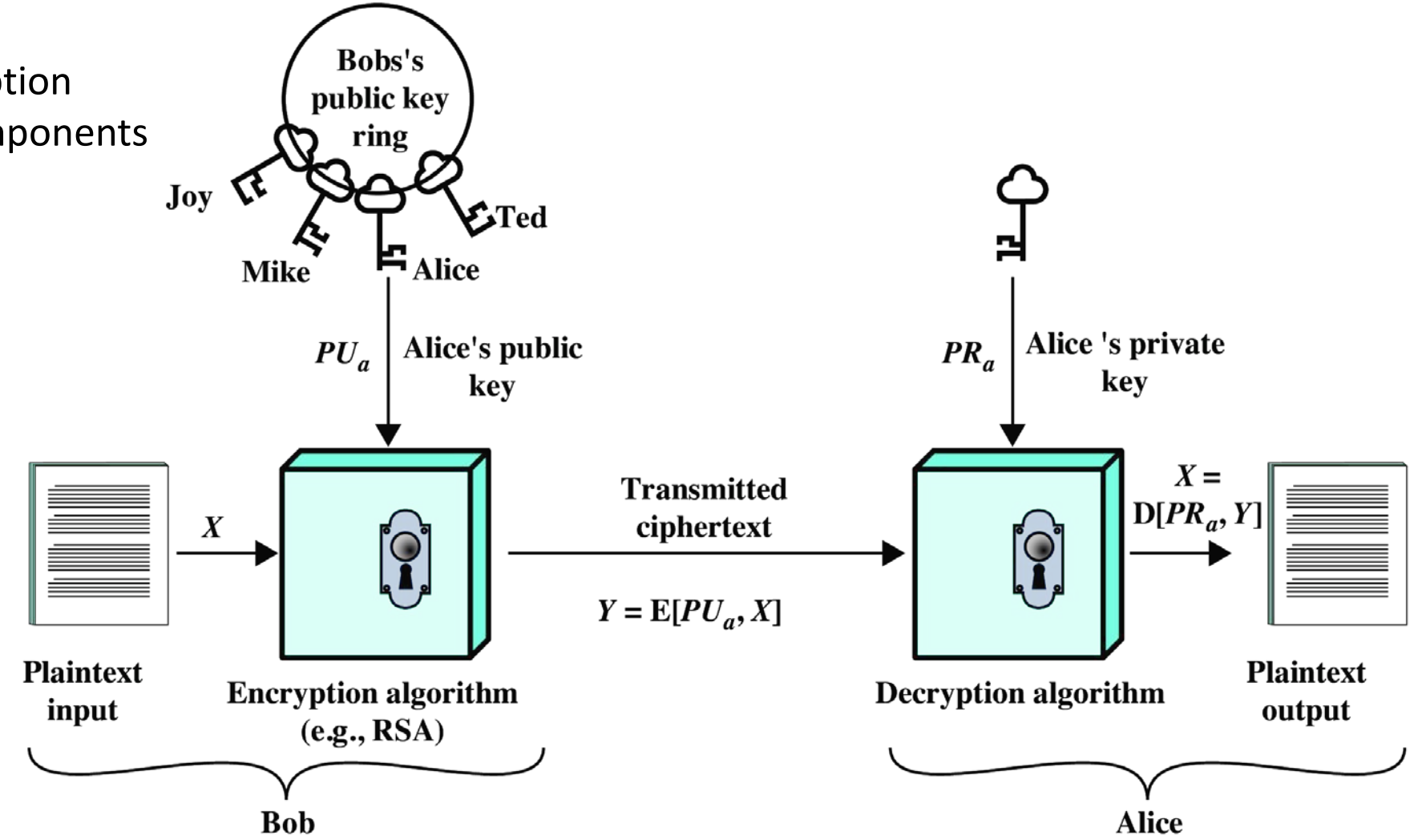
Public key
Used for encryption or digital signature verification (decryption), made public

Private key
Used for signing digital signature (encryption) or decryption, kept private

Ciphertext
The scrambled message produced as output

Decryption algorithm
Accepts the ciphertext and the matching key and produces the original plaintext

A public-key encryption scheme has six components



Encryption with Alice's public key, decryption with Alice's private key

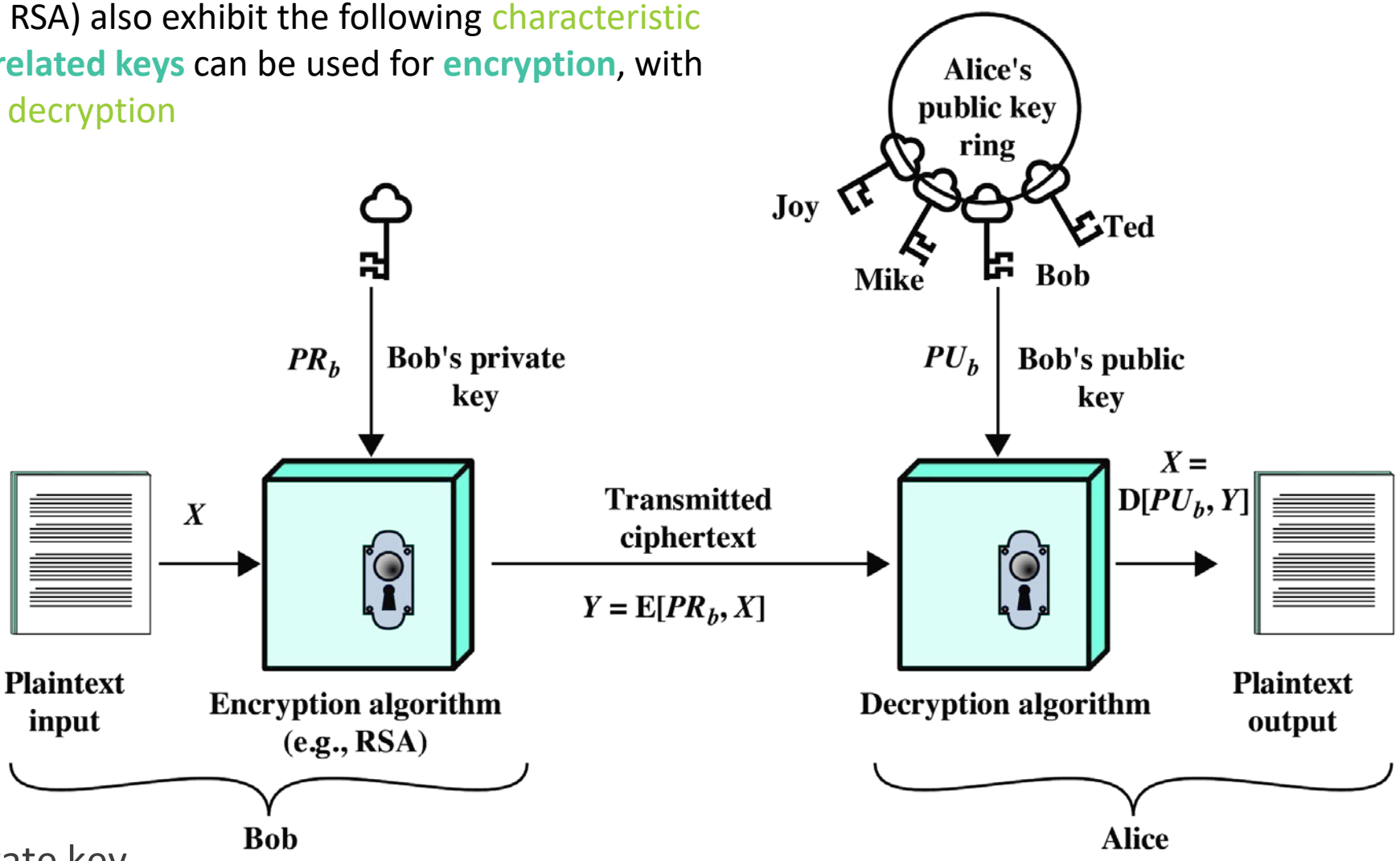
Public-Key Cryptosystems

The **essential steps** are as follows

- Each user generates a pair of keys to be used for the **encryption** and **decryption** of messages
- Each user places one of the two keys (i.e., the public key) in a **public register** or other accessible file; the companion key is kept private
- If Bob wishes to send a **confidential message** to Alice, Bob encrypts the message using Alice's public key
- 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

Some algorithms (e.g., RSA) also exhibit the following characteristic

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



Encryption with private key

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 (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

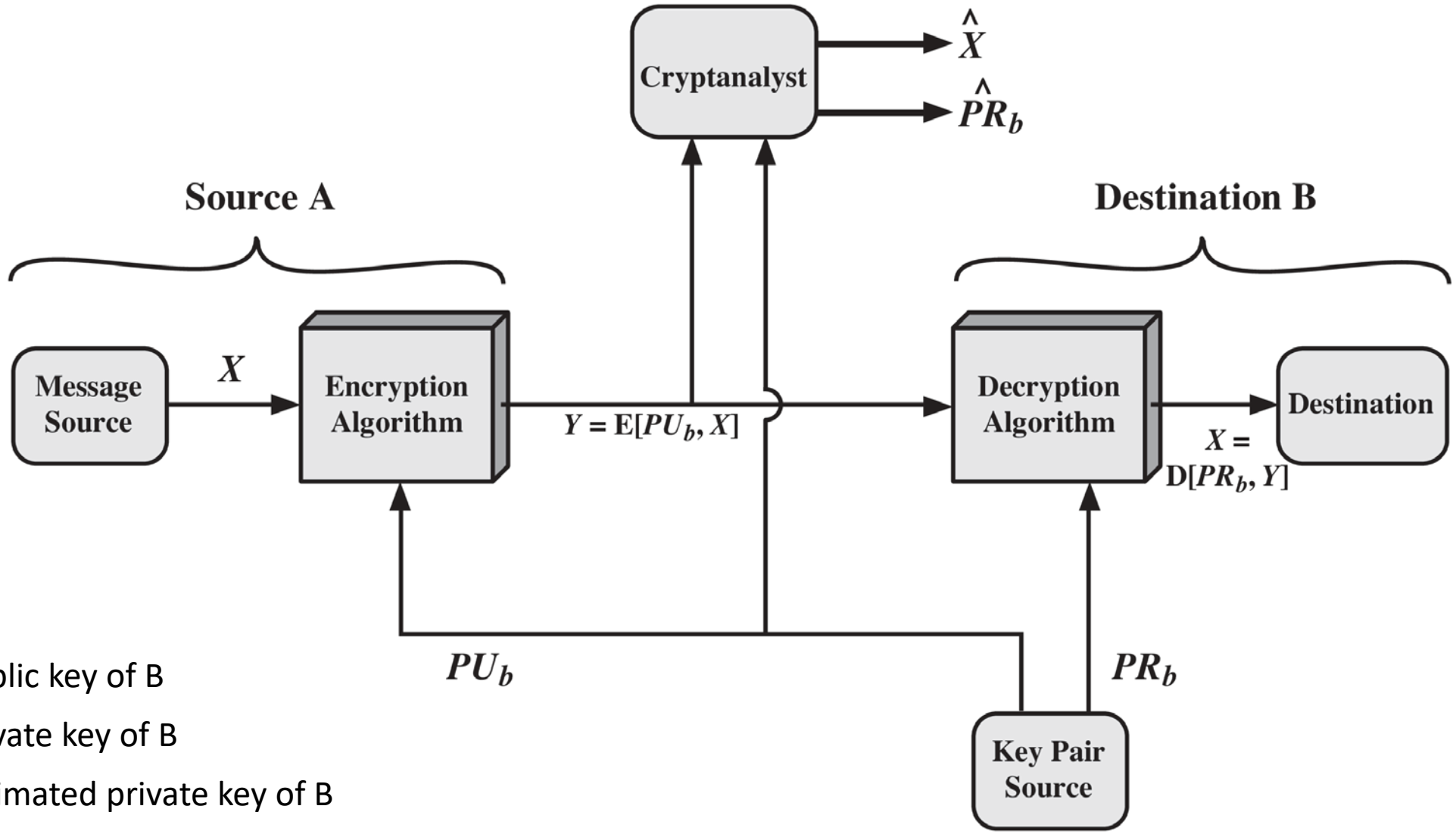
Public Key Certificate
A digital document issued and digitally signed by the private key of a Certification Authority that binds the name of a subscriber to a public key; the certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private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

Public Key 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

Applications of Public-Key Cryptosystems

An algorithm may allow for either of the two related keys be used for **encryption**, with the other being used for **decryption**

- Enables some different **cryptographic scheme** to be implemented
- In addition to just **confidentiality**, can be used to **provide authentication**
- When used for authentication, the encrypted message serves as a **digital signature**

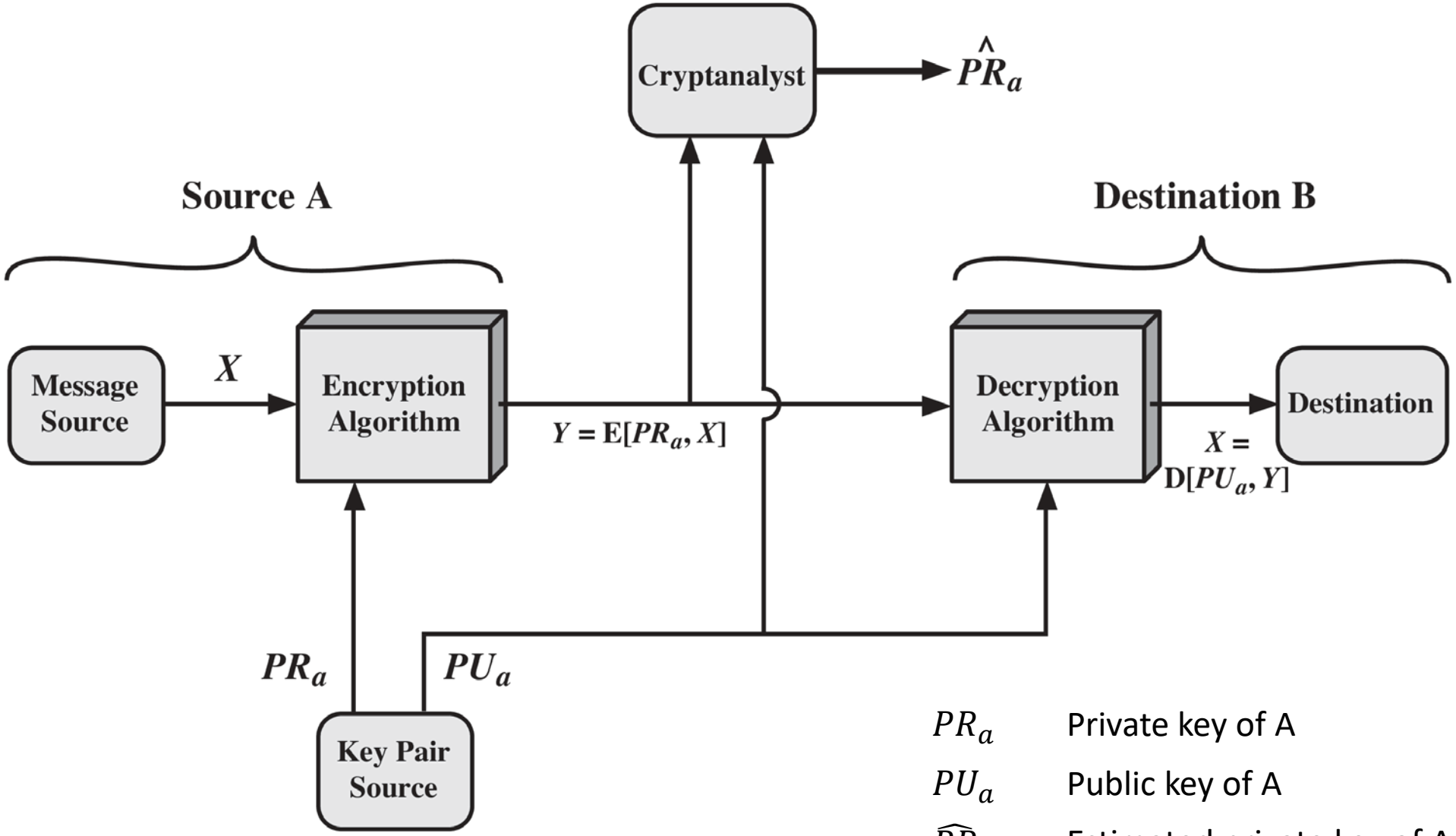


PU_b Public key of B

PR_b Private key of B

$\widehat{PR}_b$ Estimated private key of B

Public-key cryptosystem (confidentiality)



PR_a	Private key of A
PU_a	Public key of A
$\widehat{PR}_a$	Estimated private key of A

Public-key cryptosystem (authentication)

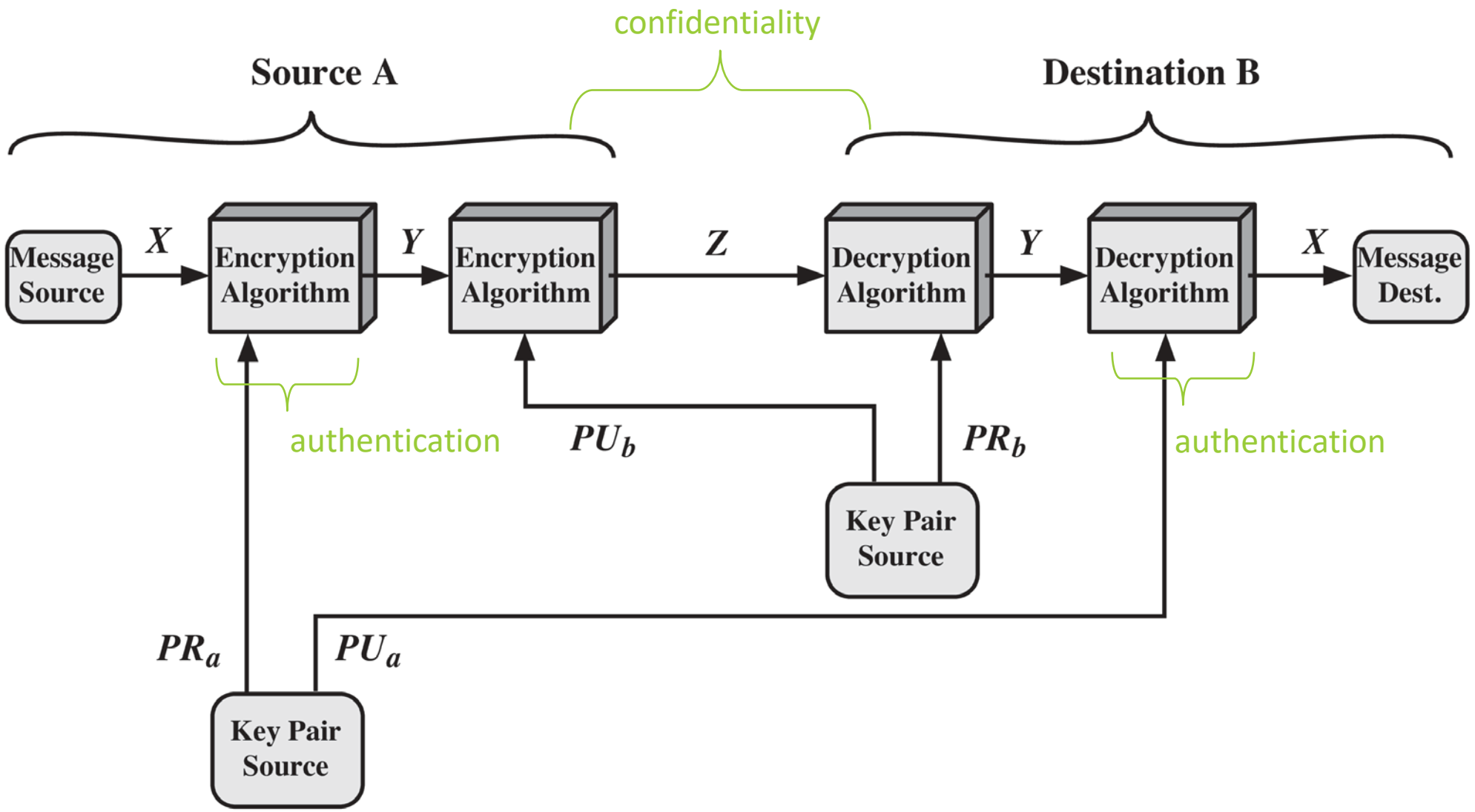
Applications of Public-Key Cryptosystems

Note that the use for authentication alone **does not provide confidentiality**

- Anyone with the sender's public key can decrypt the message

Possible to provide both the authentication function and confidentiality

- Through **double use** of the public-key scheme, in a sign-then-encrypt or encrypt-then-sign procedure



Public-key cryptosystem (authentication + confidentiality)

Applications of Public-Key Cryptosystems

Public-key cryptosystems can be classified into three categories

- **Encryption / decryption** (provide secrecy): The sender encrypts a message with the recipient's public key
- **Digital signature** (provide authentication): The sender "signs" a message with its private key
- **Key exchange** (of session keys): Two sides cooperate to exchange a session key

Some public key algorithms are suitable for all uses, others are specific to one or two

Applications of 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

Aspects of Symmetric-key and Public-Key Encryption

Required for security for **conventional encryption**

- The key must be kept secret
- It must be impractical to decipher a message if the key is kept secret
- Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key

Required for security for **public-key encryption**

- One of the two keys must be kept secret
- It must be impractical to decipher a message if the decryption key is kept secret
- Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key

Attacks to Public-Key Cryptography – Brute Force Attack

Brute-force attack

- Use **large key size** to mitigate
- However 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
- Thus currently confined mainly to **key management** and **signature applications**

Attacks to Public-Key Cryptography - Cryptanalysis

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

Misconceptions Concerning Public-Key Encryption

*"Public-key encryption is **more secure** from cryptanalysis than symmetric encryption"*

- The security of any encryption scheme depends on the **length of the key** and the **computational work involved in breaking a cipher**
- Nothing in principle about either symmetric or public-key encryption that makes one superior to another from the point of view of resisting cryptanalysis

Misconceptions Concerning Public-Key Encryption

*"Public-key encryption is **a general-purpose technique** that has made symmetric encryption obsolete"*

- Because of the **computational overhead** of current public-key encryption schemes, there seems no foreseeable likelihood that symmetric encryption will be abandoned
- Often, both symmetric and asymmetric schemes are **used together**

Misconceptions Concerning Public-Key Encryption

*"**Key distribution is trivial** when using public-key encryption, compared to the cumbersome handshaking involved with key distribution centres for symmetric encryption"*

- Some form of **protocol** is typically needed, generally involving a central agent, and the procedures involved are neither simpler nor any more efficient than those required for symmetric encryption

RSA Algorithm

RSA operations

- Key pair generation
- Encryption & Decryption

Extended Euclidean algorithm

Public-Key Cryptography (Recap)

Public-key/two-key/asymmetric cryptography involves the use of two keys:

- A **public-key**, which may be known by anybody, and can be used to encrypt messages, and verify signatures
- A **private-key**, known only to the recipient, used to decrypt messages, and sign (create) signatures

It's **asymmetric** because

- Those who encrypt messages or verify signatures cannot decrypt messages or create signatures

Rivest-Shamir-Adleman (RSA) Scheme

Developed in 1977 at MIT by Ron Rivest, Adi Shamir, and 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 (i.e., they are in $\mathbb{Z}_n$)
- A typical size for n is ~~1024~~ **2048 bits** (or longer)

Plaintext is encrypted in **blocks**

- Each block having a **value** less than some number n
- Both sender and receiver must know the value of n

Description of the RSA Algorithm

RSA makes use of an expression with exponentials

RSA can be used for public-key encryption

For **plaintext** block M and **ciphertext** block C, encryption and decryption are of the following form

- $C = M^e \bmod n$
- $M = C^d \bmod n$

The **sender** knows the value of e, and **only the receiver** knows the value of d

- Public key $PU = \{e, n\}$
- Private key $PR = \{d, n\}$

Description of the RSA Algorithm

Stages in the RSA algorithm

- Key generation
- Encryption
- Decryption

RSA Key Generation

- Select $\boldsymbol{p, q} \in \mathbb{P}$ such that $p \neq q$
- Calculate $\boldsymbol{n = p \times q}$ and $\boldsymbol{\phi(n) = (p-1)(q-1)}$
- Select integer $\boldsymbol{e}$ such that $\gcd(\phi(n), e) = 1$ and $1 < e < \phi(n)$
- Calculate $\boldsymbol{d}$, where $d \equiv e^{-1} \left(\mathrm{mod}\ \phi(n)\right)$
- Public key $\boldsymbol{PU = \{e, n\}}$, private key $\boldsymbol{PR = \{d, n\}}$

- $\mathbb{P}$ - set of prime numbers
- Euler's totient function $\boldsymbol{\phi(n)}$ calculates $|\mathbb{Z}_n{}^*|$, where $\mathbb{Z}_n{}^* = \{x \in \mathbb{Z} \mid x < n \text{ and } \gcd(x, n) = 1\}$
- $\mathbb{Z}$ - set of integers

RSA Key Generation Example

1. Select primes: $p=17$ & $q=11$
2. Compute $n = pq = 17 \times 11 = 187$
3. Compute $\phi(n)=(p-1)(q-1)=16 \times 10=160$
4. Select e: $\gcd(e,160)=1$; choose $e=7$
5. Determine d: $de=1 \bmod 160$ and $d < 160$ Value is d=23 since $23 \times 7=161= 1 \times 160+1$
6. Publish public key PU={7,187}
7. Keep secret private key PR={23,187}

RSA Encryption and Decryption

Suppose that Alice published her public key $PU_A = \{e, n\}$

- Bob wants to send a message M to Alice
- Bob calculates $C = M^e \bmod n$, and transmits C to Alice
- Alice decrypts the ciphertext by calculating $M = C^d \bmod n$

RSA En/Decryption example

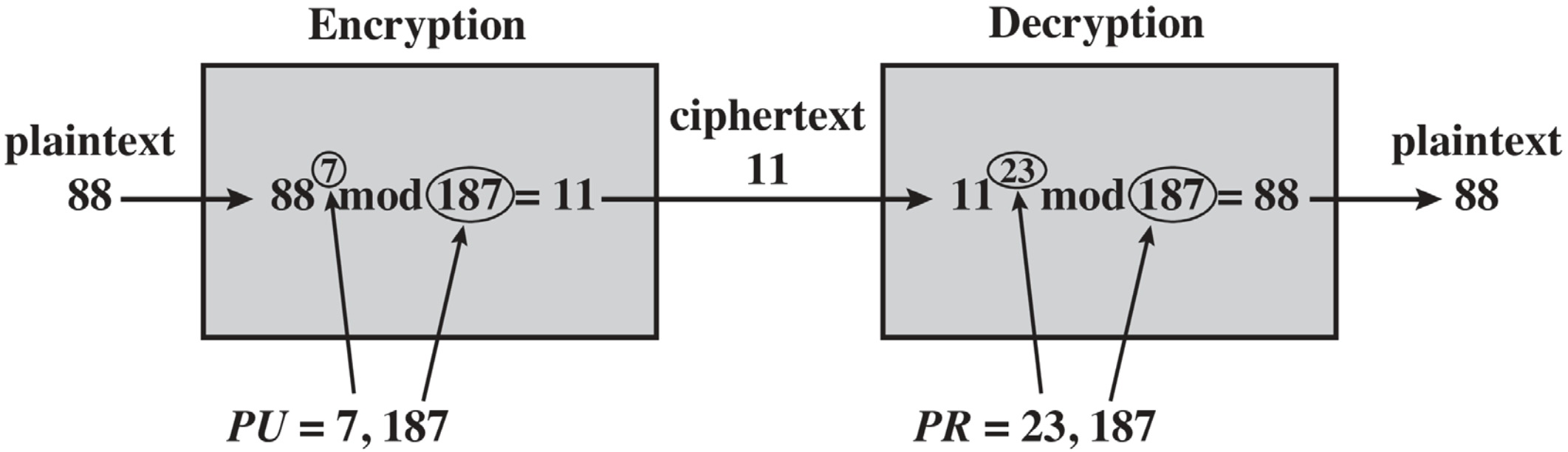
Given message $M = 88$ $(88<187)$

Encryption:

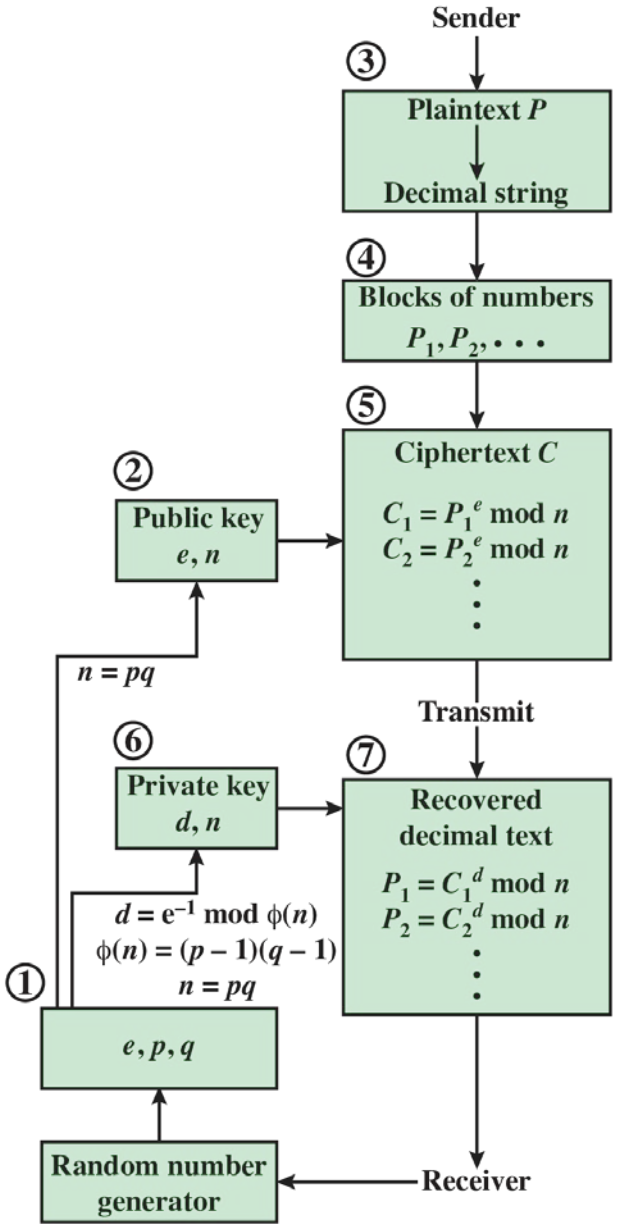
$$C = 88^{7} \bmod 187 = 11$$

Decryption:

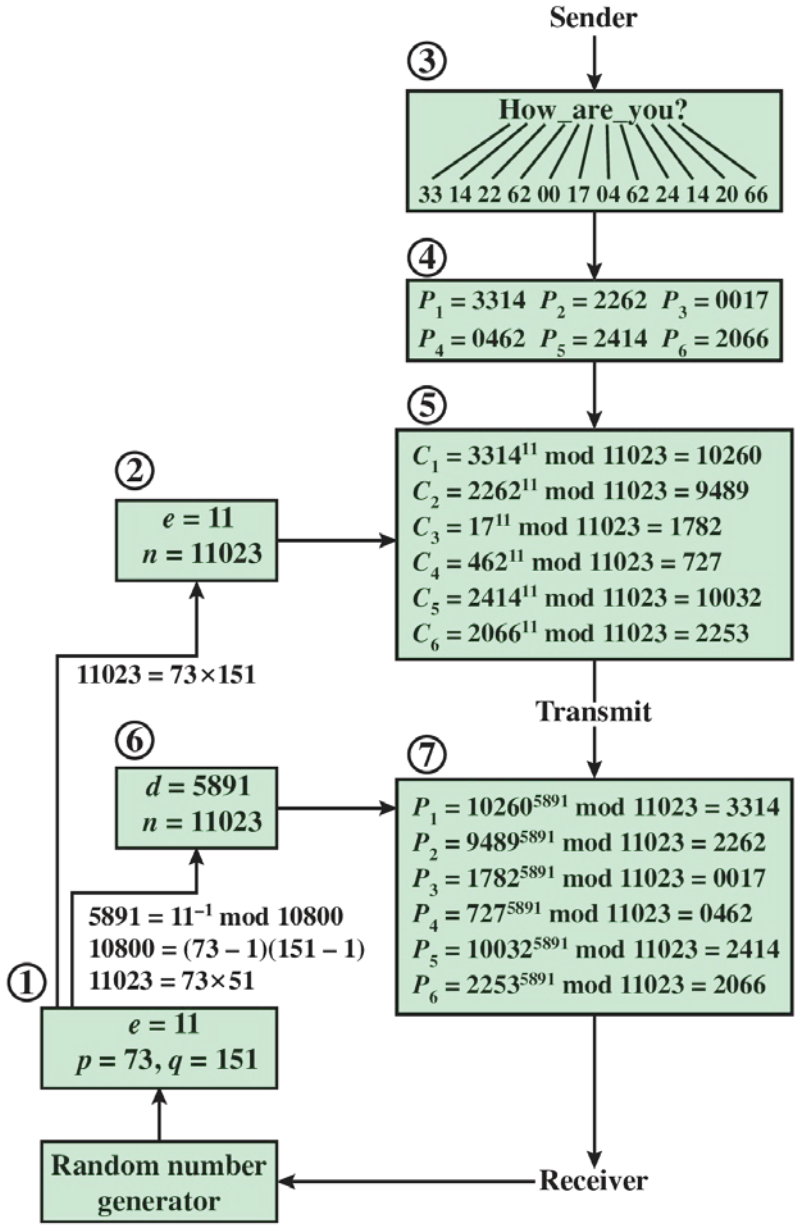
$$M = 11^{23} \bmod 187 = 88$$



Example of the RSA algorithm



RSA processing of multiple blocks



RSA processing of multiple blocks (example)

** The decimal string conversion is for illustration here.*

Calculating d using the Extended Euclidean Algorithm

The value d is the multiplicative inverse of e in $|(Z_N)^*|$ (modulo φ (N))

- i.e., $d \equiv e^{-1} \pmod{\phi(n)}$

Given e, the value of d can be calculated using the **extended Euclidean algorithm**

In the **extended Euclidean algorithm**, for $a, b \in \mathbb{Z}$, the algorithm computes

- The greatest common divisor $\gcd(a, b)$
- $x, y \in \mathbb{Z}$ that satisfies the following equation

$$ax + by = \gcd(a, b)$$

Round, i	r_i	q_i	x_i	y_i
-1	$r_{-1} = a$		1	0
0	$r_0 = b$		0	1
1	$a \bmod b$	$\lfloor a/b \rfloor$	$x_{-1} - q_1 x_0 = 1$	$y_{-1} - q_1 y_0 = -q_1$
2	$b \bmod r_1$	$\lfloor b/r_1 \rfloor$	$x_0 - q_2 x_1$	$y_0 - q_2 y_1$
3	$r_1 \bmod r_2$	$\lfloor r_1/r_2 \rfloor$	$x_1 - q_3 x_1$	$y_1 - q_3 y_1$
		⋮		
$n-1$	$r_{n-3} \bmod r_{n-2}$	$\lfloor r_{n-3}/r_{n-2} \rfloor$	$x_{n-3} - q_{n-1} x_{n-2}$	$y_{n-3} - q_{n-1} y_{n-2}$
n	$r_{n-2} \bmod r_{n-1}$	$\lfloor r_{n-2}/r_{n-1} \rfloor$	$x_{n-2} - q_n x_{n-1}$	$y_{n-2} - q_n y_{n-1}$
$n+1$	$r_{n-1} \bmod r_n = 0$	$\lfloor r_{n-1}/r_n \rfloor$		

Extended Euclidean algorithm calculations

$\therefore \gcd(a, b) = r_n; x = x_n; y = y_n$

Calculating d

To use the **extended Euclidean algorithm** to compute d, run the algorithm with

- $a = \phi(n) = (p-1)(q-1)$
- $b = e$

Once the algorithm **terminates**, the value of $d = y$

Consider an example where $p = 17$, $q = 41$, and $e = 11$

- $\phi(n) = 640$

Round, i	r_i	q_i	x_i	y_i
-1	$a = \phi(n) = 640$		1	0
0	$b = e = 11$ mod		0 -	1
1	2 mod	58 *	1 -	-58
2	1 mod	5 *	-5	291
3	0	2		

$\therefore \gcd(a, b) = \gcd(\phi(n), e) = 1; x = -5; y = 291$

$\phi(n)\, x + e\, d = (640)(-5) + (11)(291) = -3200 + 3201 = 1$

$d = y = 291$

$$x_i = x_{i-2} - q_i x_{i-1};$$
$$y_i = y_{i-2} - q_i y_{i-1}.$$

```
Round 1: x1 = 1 - 58*0 =  1; y1 = 0 - 58*1 = -58.
Round 2: x2 = 0 -  5*1 = -5; y2 = 1 - 5*(-58) = 291.
```

Computing the d value for an RSA key pair

Calculating d

From the algorithm run, it can be seen that $y = 291$

- d is thus **291**

This can be verified by calculating $de \bmod \phi(n) = (291)(11) \bmod 640 = 1$

- Thus showing that $291 \equiv 11^{-1} \bmod 640$

RSA with Low Public Exponent

To speed up RSA encryption use a small e: $c = m^e \pmod N$

Recommended value: **$e=65537=2^{16}+1$**

Asymmetry of RSA: fast encryption / slow decryption

Key Lengths

Security of public key system should be comparable to security of symmetric cipher:

Cipher key-size	RSA Modulus size
80 bits	1024 bits
128 bits	3072 bits
256 bits (AES)	**15360** bits

Introduction to Number Theory

Greatest Common Divisor

Modular Arithmetic

Modular Inversion

Euler’s Generalization of Fermat’s Theorem

Greatest common divisor

Def: For integers x, y: **gcd(x, y)** denotes the greatest common divisor of x, y

Fact: For all integers x, y there exist integers a, b such that

$$a \cdot x + b \cdot y = \gcd(x,y)$$

a,b can be found efficiently using the extended Euclidean algorithm.

Example: gcd(12, 18) = 6 $\quad$ $2 \times 12 - 1 \times 18 = 6$

If gcd(x,y)=1 then x and y are **relatively prime**

Modular Arithmetic

Notations

- N denotes a positive integer
- p denotes a prime number

$\mathbb{Z}_N = \{0,1,2,\ldots,N-1\}$

Addition and multiplication modulo N can be performed in $\mathbb{Z}_N$

Examples: let N = 12

9 + 8 = 5 in $\mathbb{Z}_{12}$

5 × 7 = 11 in $\mathbb{Z}_{12}$

5 − 7 = 10 in $\mathbb{Z}_{12}$

Arithmetic in $\mathbb{Z}_N$ follows the distributive law,

Example: $x\cdot(y+z) = x\cdot y + x\cdot z$ in $\mathbb{Z}_N$

Modular Inversion

Over the rational numbers, the inverse of 2 is ½ . What about over $\mathbb{Z}_N$?

Def: The **inverse** of x in $\mathbb{Z}_N$ is an element y in $\mathbb{Z}_N$ such that $x \cdot y = 1$ in $\mathbb{Z}_N$

y is denoted x^{-1} .

Example: let N be an odd integer. The inverse of 2 in $\mathbb{Z}_N$ is $\frac{N+1}{2}$

$$2 \cdot \left(\frac{N+1}{2}\right) = N + 1 = 1 \; in \; \mathbb{Z}_N$$

Which elements have an inverse in $\mathbb{Z}_N$?

Lemma: x in $\mathbb{Z}_N$ has an inverse if and only if gcd(x,N) = 1

More Notation

Def: $\mathbb{Z}_N^*$ = (set of invertible elements in $\mathbb{Z}_N$)

= $\{ \; x \in \mathbb{Z}_N \; : \; \gcd(x,N) = 1 \}$

Examples:

1. For prime p, $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\} = \{1, 2, \ldots, p-1\}$
2. $\mathbb{Z}_{12}^*$ = { 1, 5, 7, 11}

For x in $\mathbb{Z}_N^*$, can find x^{-1} using extended Euclid algorithm.

Fermat's Theorem (1640)

Thm: Let p be a prime

$$\forall\, x \in (Z_p)^* : \quad x^{p-1} = 1 \text{ in } Z_p$$

Example: p=5. $3^4 = 81 = 1$ in Z_5

So: $x \in (Z_p)^* \Rightarrow x \cdot x^{p-2} = 1 \Rightarrow x^{-1} = x^{p-2}$ in Z_p

This is another way to compute inverses, but it is less efficient than Euclid's method.

Euler's Generalization of Fermat's Theorem (1736)

Def: For an integer N define $\varphi(N) = \left| (Z_N)^* \right|$ (Euler's φ func.)

Examples: $\varphi(12) = \left| \{1,5,7,11\} \right| = 4$; $\varphi(p) = p-1$

For $N=p\cdot q$: $\varphi(N) = N-p-q+1 = (p-1)(q-1)$

Thm (Euler): $\forall\, x \in (Z_N)^*$: $x^{\varphi(N)} = 1$ in Z_N

Example: $5^{\varphi(12)} = 5^4 = 625 = 1$ in Z_{12}

Generalization of Fermat's theorem. Basis of the RSA cryptosystem

RSA Security

Attacks to RSA

- Brute force
- Mathematical attacks
- Timing attacks
- Hardware fault-based attack
- Chosen ciphertext attacks

OAEP padding scheme overview

Security of RSA

Brute force
Involves trying all possible private keys

Mathematical attacks
Several approaches, all equivalent in effort to factoring the product of two primes

Timing attacks
Depend on the running time of the decryption algorithm

Hardware fault-based attack
Involves inducing hardware faults in the processor that is generating digital signatures

Chosen ciphertext attacks
Exploits properties of the RSA algorithm

Factoring Problem

Three approaches to attacking RSA mathematically

Factor n into its two prime factors p, q
Enables calculation of $\phi(n) = (p - 1)(q - 1)$, which in turn enables determination of $d \equiv e^{-1}\left(mod\ \phi(n)\right)$

Determine $\phi(n)$ directly without first determining p and q
Enables determination of $d \equiv e^{-1}\left(mod\ \phi(n)\right)$

Determine d directly without first determining $\phi(n)$

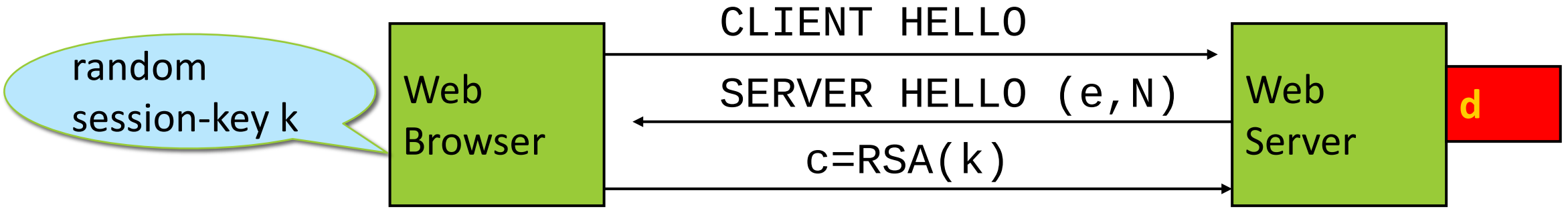
Number of Decimal Digits	Number of Bits	Date Achieved
100	332	April 1991
110	365	April 1992
120	398	June 1993
129	428	April 1994
130	431	April 1996
140	465	February 1999
155	512	August 1999
160	530	April 2003
174	576	December 2003
200	663	May 2005
193	640	November 2005
232	768	December 2009

Progress in RSA factorization

Security Strength	Symmetric Key Algorithm	DSA, Diffie-Hellman (Finite-Field Cryptography)	RSA (Integer Factorization Cryptography)	Elliptic Curve Cryptography
$\leq$ **80**	2TDEA	$L = 1024$ $N = 160$	1024	$160 - 223$
112	3TDEA	$L = 2048$ $N = 224$	2048	$224 - 255$
128	AES-128	$L = 3072$ $N = 256$	3072	$256 - 383$
192	AES-192	$L = 7680$ $N = 384$	7680	$384 - 511$
256	AES-256	$L = 15360$ $N = 512$	15360	$512 +$

Comparable maximum strength, in bits (NIST SP800-57)

Meet in the Middle Attack to textbook RSA



Suppose k is 64 bits: $k \in \{0,\ldots,2^{64}-1\}$. Eve sees: $c = k^e$ in Z_N

If $\mathbf{k = k_1 \cdot k_2}$ where $k_1, k_2 < 2^{34}$ then $\mathbf{c/k_1^e = k_2^e}$ in Z_N

Step 1: build table: $c/1^e, c/2^e, c/3^e, \ldots, c/2^{34e}$. time: 2^{34}

Step 2: for $k_2 = 0,\ldots, 2^{34}$ test if k_2^e is in table. time: 2^{34}

Output matching (k_1, k_2). Total attack time: $\approx 2^{40} << 2^{64}$

Timing Attack

A snooper can determine a private key by keeping track of how long a computer takes to decipher messages

- Applicable **not just to RSA** but to other public-key cryptography systems

Alarming for two reasons

- Comes from a completely **unexpected direction** (i.e. a side-channel attack)
- Is a **ciphertext-only attack**

Countermeasures Against Timing Attack

Constant exponentiation time

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

Random delay

- Adding a **random delay** to the exponentiation algorithm to confuse the timing attack

Blinding

- Multiply the ciphertext by **a random number** before performing exponentiation; prevents the attacker from knowing what ciphertext bits are being processed inside the computer

Chosen Ciphertext Attack

In a chosen ciphertext attack

- Adversary can get the **decrypted plaintext** of **some chosen ciphertext**

It is of interest to note that RSA displays the **following property**

$$\mathrm{E}(PU, M_1) \times \mathrm{E}(PU, M_2) = \mathrm{E}(PU, M_1 \times M_2)$$

i.e. $(M_1{}^e \; mod \; n)\; (M_2{}^e \; mod \; n) = (M_1 \times M_2)^e \; mod \; n$

or $C_1 \times C_2 = \mathrm{E}(PU, M_1 \times M_2)$

E.g. if $M_1 = 5$, $M_2 = 2$, $e = 3$, $n = 22$, then

$(M_1{}^e \; mod \; n)\; (M_2{}^e \; mod \; n) = (5^3 \; mod \; 22)\; (2^3 \; mod \; 22) = 15 \; \times 8 \; mod \; 22 = 10$

$(M_1 \times M_2)^e \; mod \; n = (5 \times 2)^3 \; mod \; 22 = 10$

Chosen Ciphertext Attack

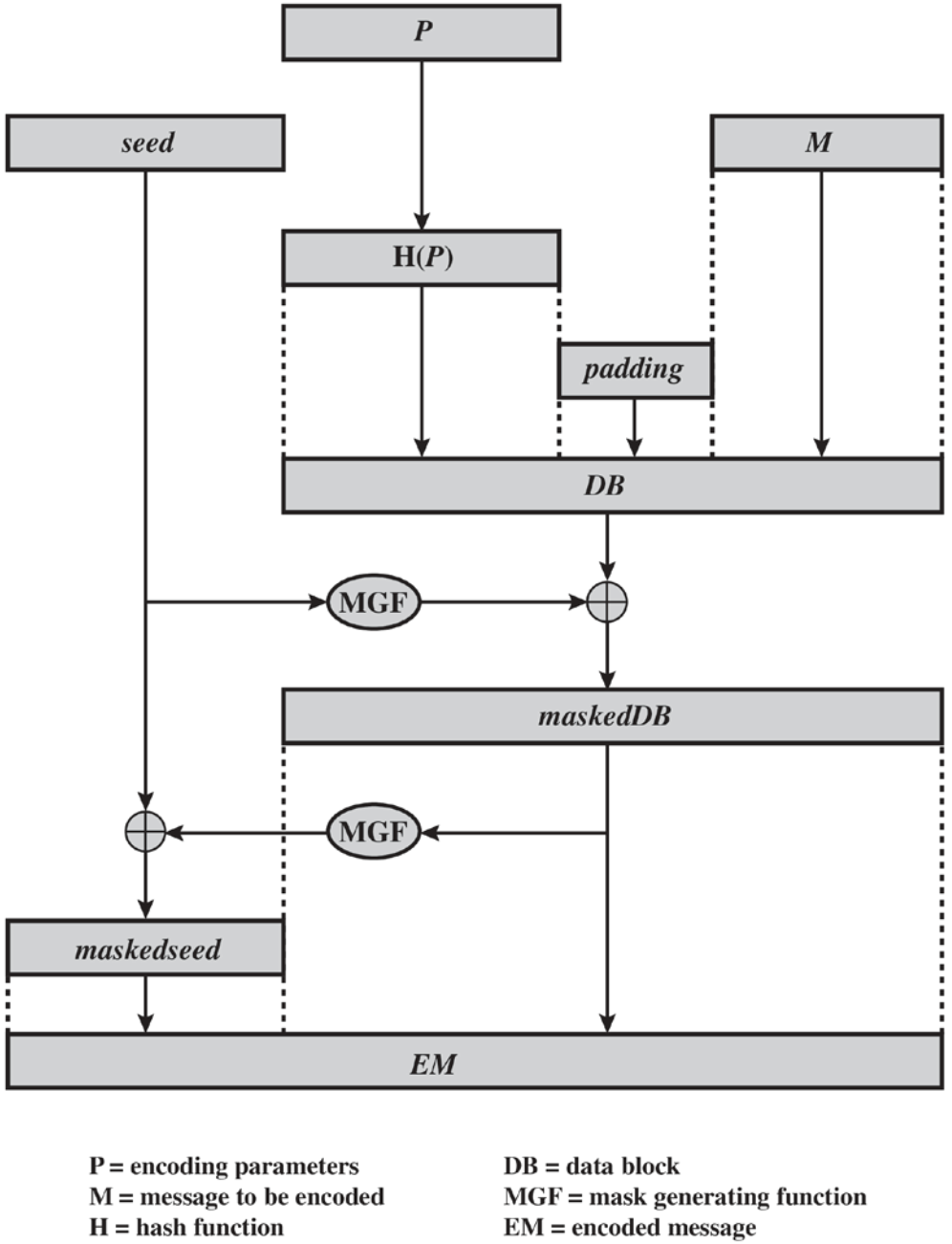
An example simple attack

- Assume attacker has $C = M^e \bmod n$
- Attacker computes $X = C \times 2^e \bmod n$
- Attacker submits X as chosen ciphertext, receives $Y = X^d \bmod n$
- Attacker now has $Y = 2M \bmod n$, can deduce M

Chosen Ciphertext Attack

To counter CCA, padding is suggested

- Specifically use **optimal asymmetric encryption padding** (OAEP)
- Check message padding on decryption, reject ciphertext if invalid.
- In practice: use SHA-256 for MGF.



Encryption using OAEP

Summary

Public-key cryptosystems

- Applications for public-key cryptosystems
- Requirements for public-key cryptography

The RSA algorithm

- Description of the algorithm
- Key pair generation
- Security of RSA

Review Questions

- What are the role of public and private keys?
- What are the three broad categories of applications of public-key cryptosystems?
- What requirements must a public-key cryptosystem fulfil to be a secure algorithm?
- List the different approaches to attack the RSA algorithm
- Describe the countermeasures to be used against timing attacks