Public-Key Cryptography

Outline

1. Public Key Encryption
2. Symmetric vs. Public-Key
3. RSA Public Key Encryption
4. RSA Key Construction
5. Optimizing Private Key Operations
6. RSA Security

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:

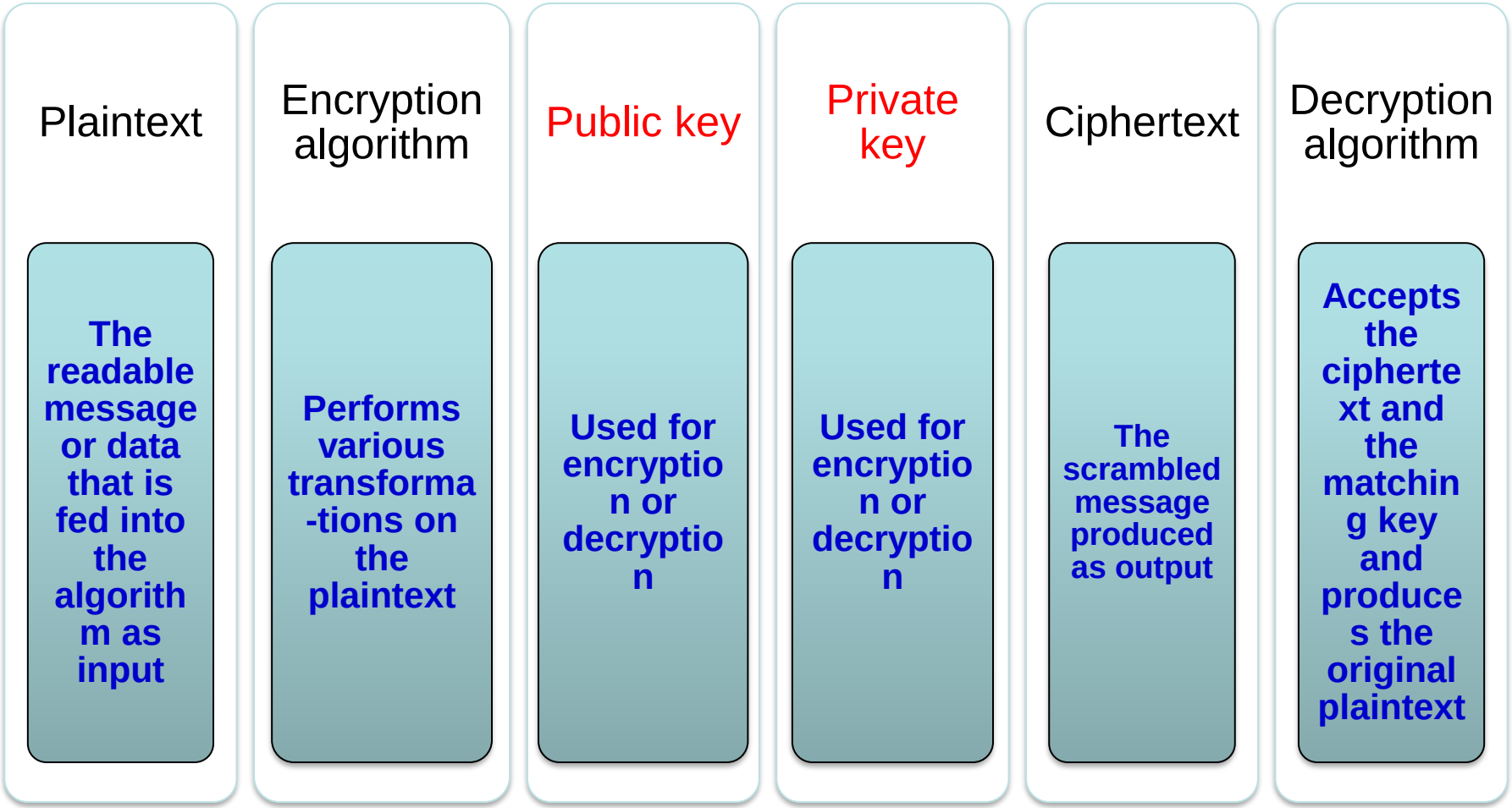
Key distribution

Digital signatures

- 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:



Symmetric and Asymmetric-key Cryptography

Symmetric and asymmetric-key cryptography will exist in parallel and continue to serve the community. We actually believe that they are complements of each other; the advantages of one can compensate for the disadvantages of the other.

Symmetric-key cryptography is based on sharing secrecy; asymmetric-key cryptography is based on personal secrecy.

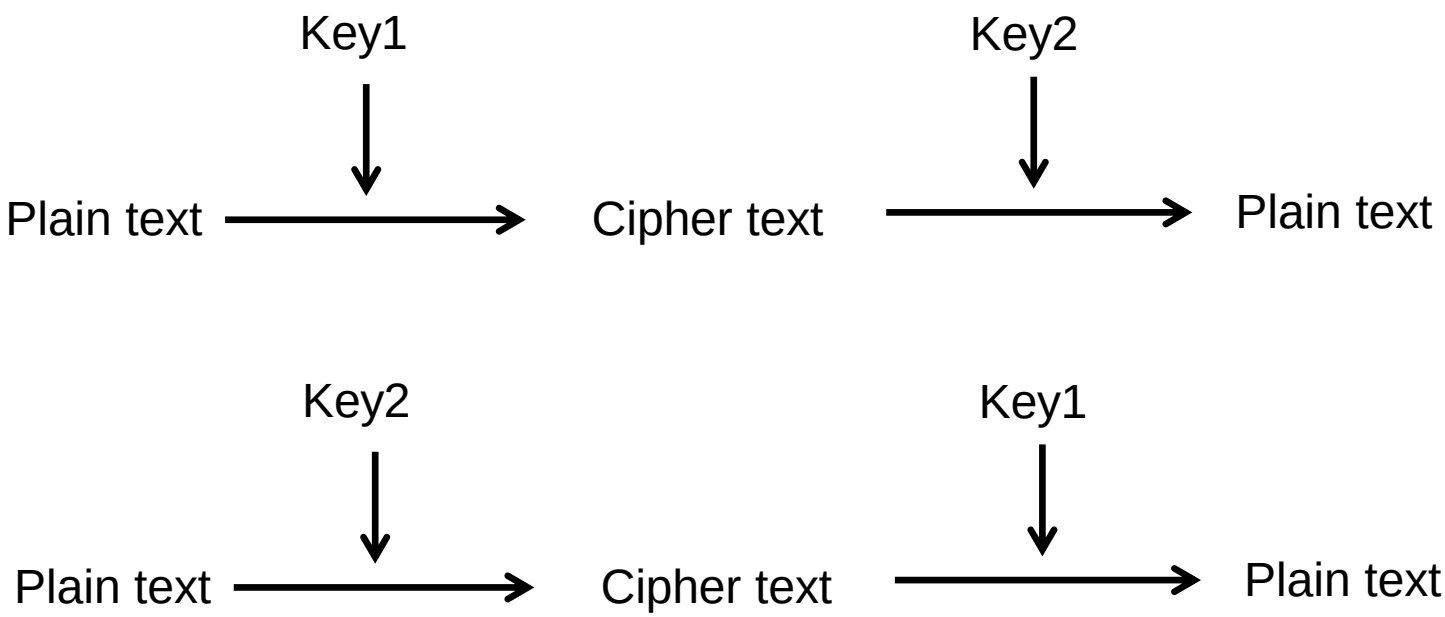
Need for Both

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.

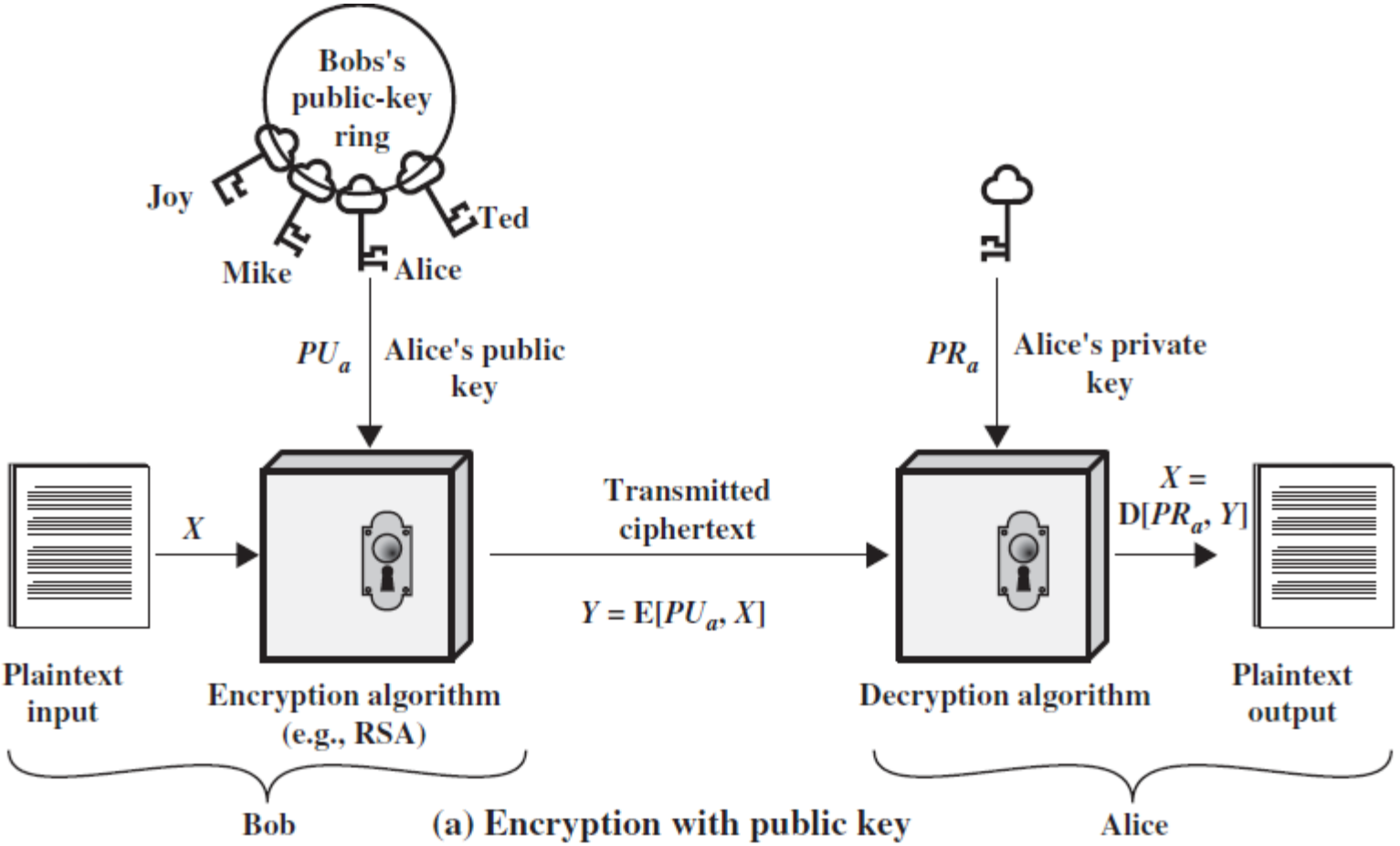
Public Key Cryptography

- Invented in 1975 by Diffie and Hellman at Stanford
- Encrypted_Message = Encrypt (Key1, Message)
- Message = Decrypt (Key2, Encrypted_Message)
- Keys are **interchangeable**
- One key is made **public** while the other is kept **private**
- Sender knows only public key of the receiver =>**Asymmetric**

Public Key Cryptography

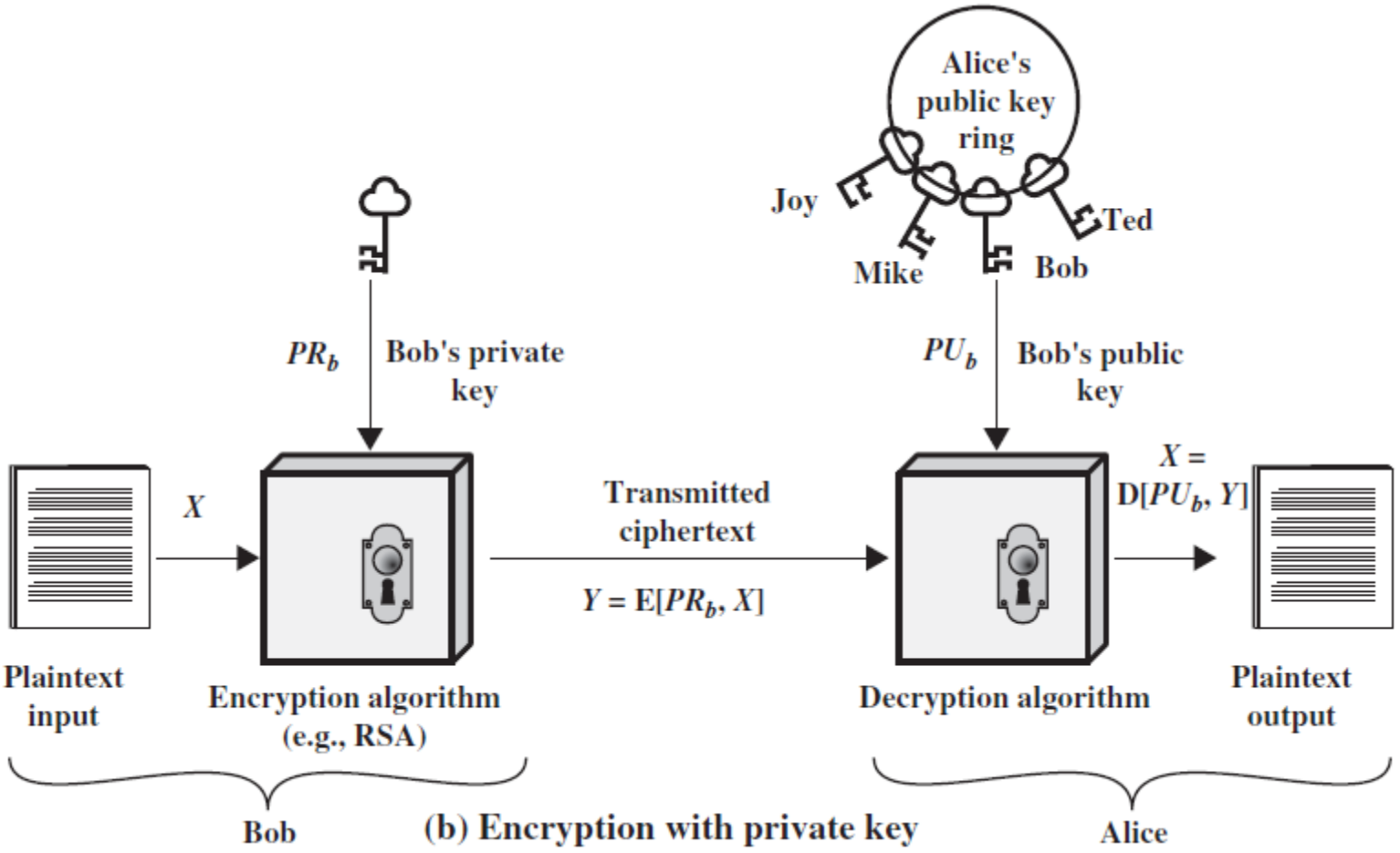


Encryption with Public Key



(a) Encryption with public key

Encryption with Private Key



(b) Encryption with private key

Why Public-Key Cryptography?

- Developed to address two key issues:
  - **key distribution** – how to have secure communications in general without having to trust a KDC with your key
  - **digital signatures** – how to verify a message comes intact from the claimed sender

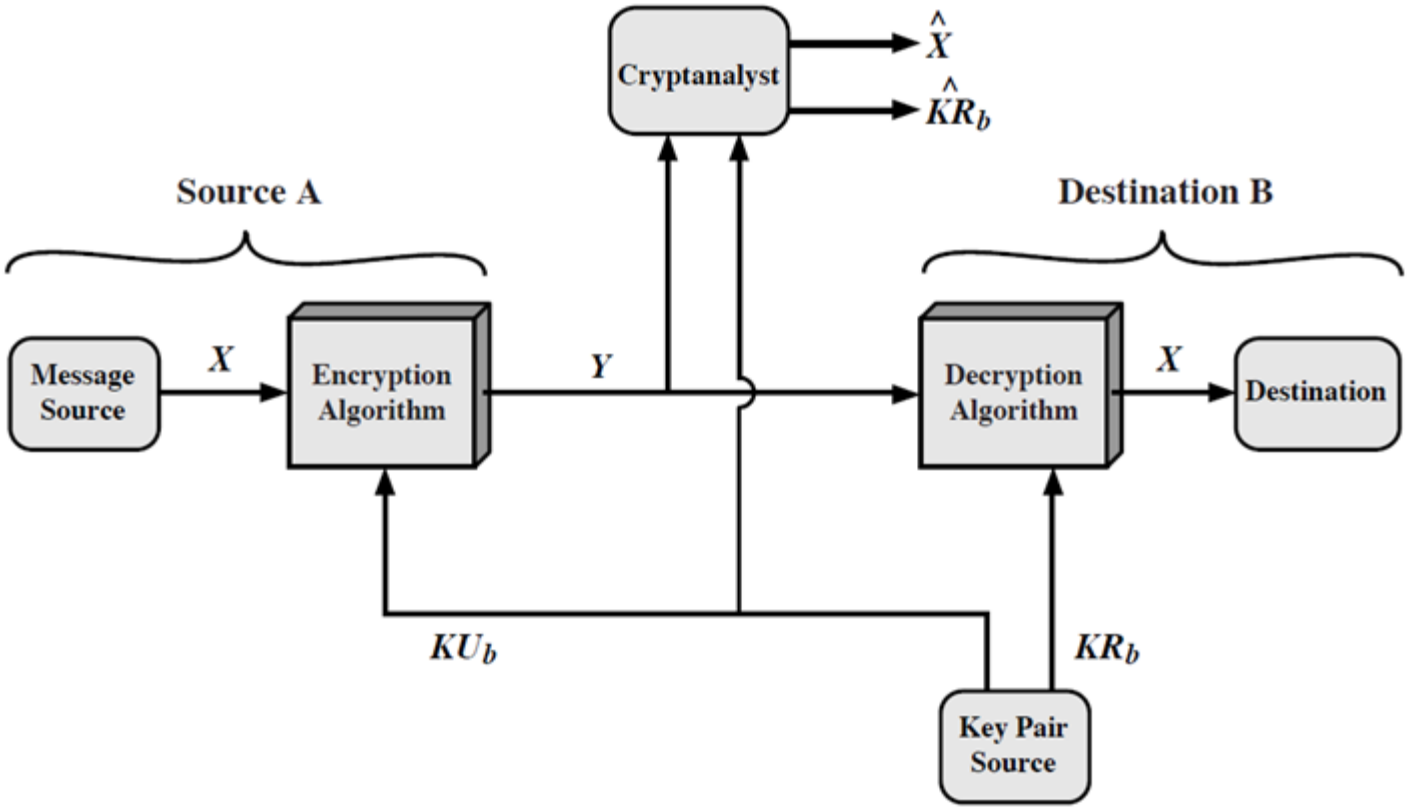
Conventional and Public-Key Encryption

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 Characteristics

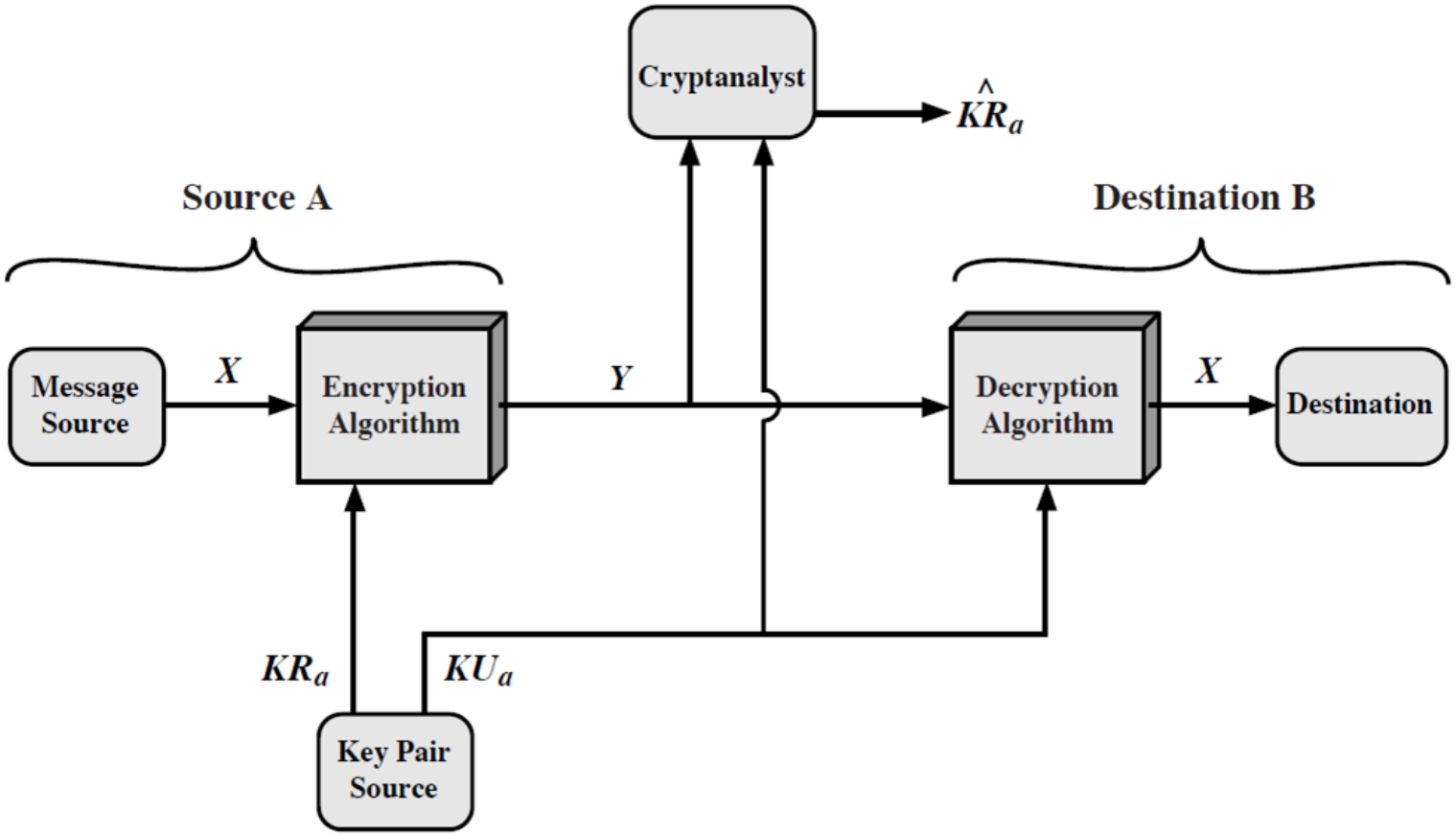
- Public-Key algorithms rely on two keys with the characteristics that it is:
  - computationally infeasible to find decryption key knowing only algorithm & encryption key
  - computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
  - either of the two related keys can be used for encryption, with the other used for decryption (in some schemes)

Public-Key: Secrecy



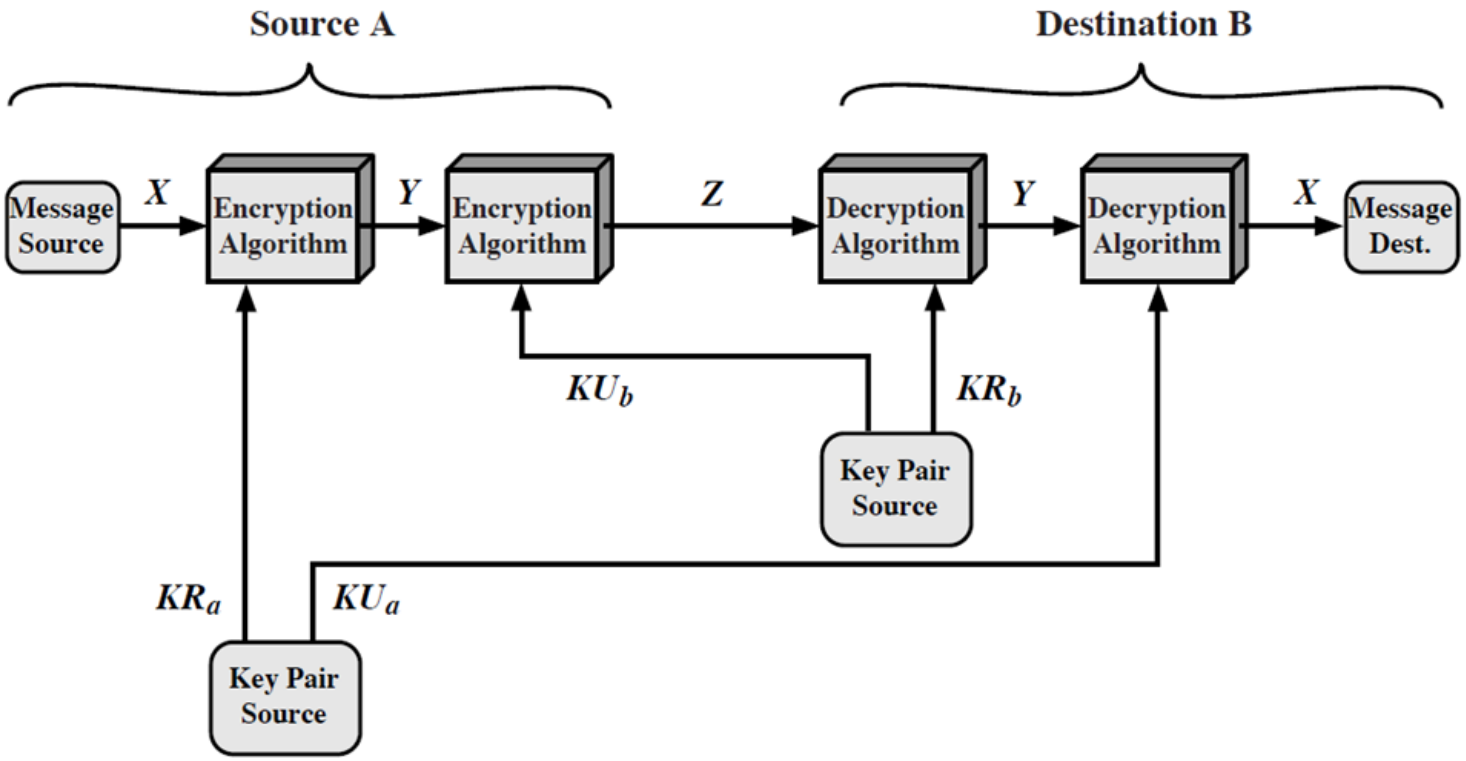
Public-Key Cryptosystem: Secrecy

Public-Key: Authentication



Public-Key Cryptosystem: Authentication

Public-Key: Secrecy and Authentication



Public-Key Cryptosystem: Secrecy and Authentication

Public-Key Applications

- Can classify uses into 3 categories:
  - **encryption/decryption** (provide secrecy)
  - **digital signatures** (provide authentication)
  - **key exchange** (of session keys)
- Some algorithms are suitable for all uses, others are specific to one

Public-Key Cryptography: Requirements

The main idea behind asymmetric-key cryptography is the concept of the trapdoor 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

Trapdoor One-Way Function

- 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

Security of Public Key Schemes

- Like private key schemes brute force exhaustive search attack is always theoretically possible
- But keys used are too large (>512bits)
- Security relies on a large enough difference in difficulty between easy (en/decrypt) and hard (cryptanalyse) problems
- More generally the hard problem is known, but is made hard enough to be impractical to break
- Requires the use of very large numbers
- Hence is slow compared to private key schemes

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

Public-Key Cryptanalysis (cont.)

- 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

Probable message attack

The public key is known- Encrypt all possible messages- Try to find a match between the ciphertext and one of the above encrypted messages

RSA Cryptosystem

- 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

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

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

RSA Algorithm (cont)

- 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 $PR=\{d,n\}$

RSA 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 $n = p \times q$	
Calcuate $\phi(n) = (p - 1)(q - 1)$	
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	$PR = \{d, n\}$

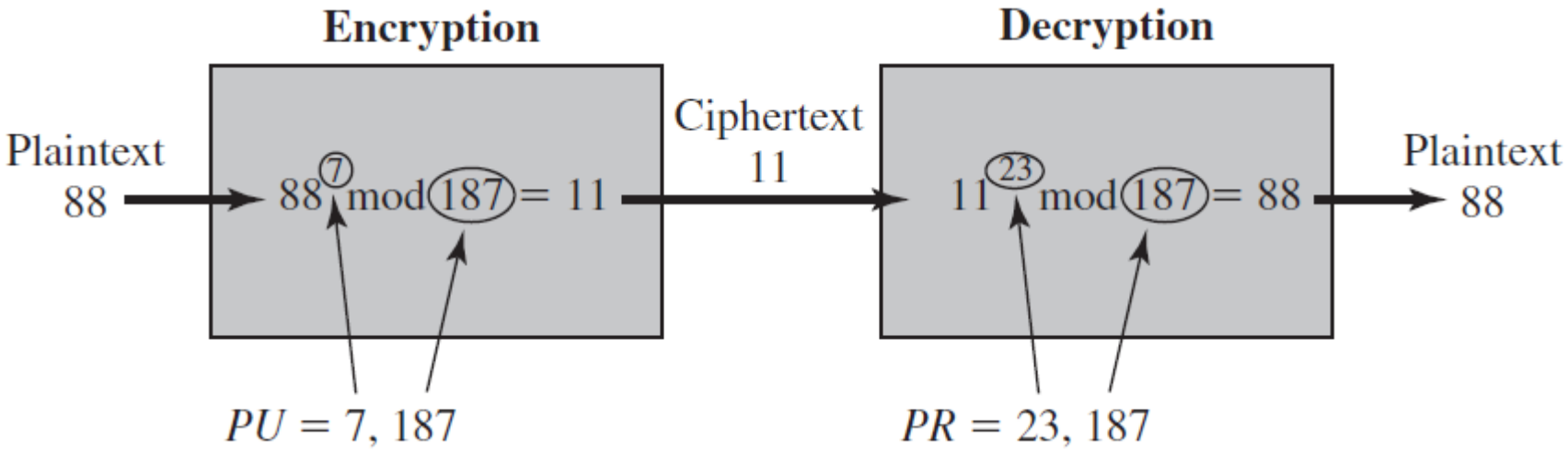
Encryption by Bob with Alice's Public Key

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

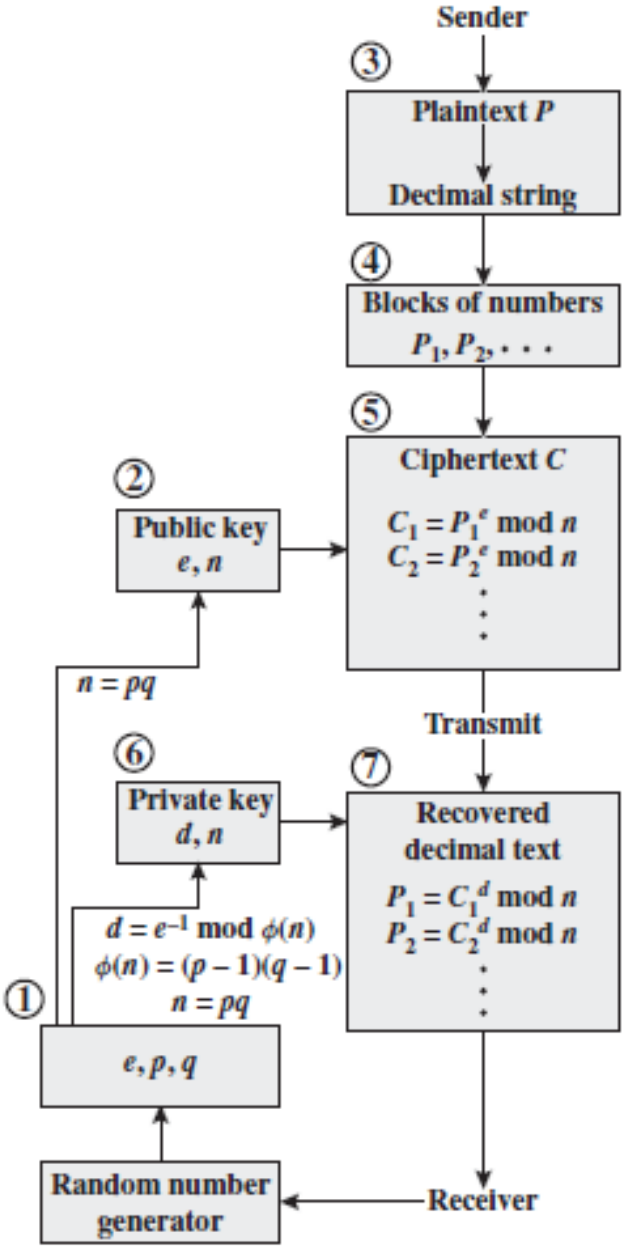
Decryption by Alice with Alice's Public Key

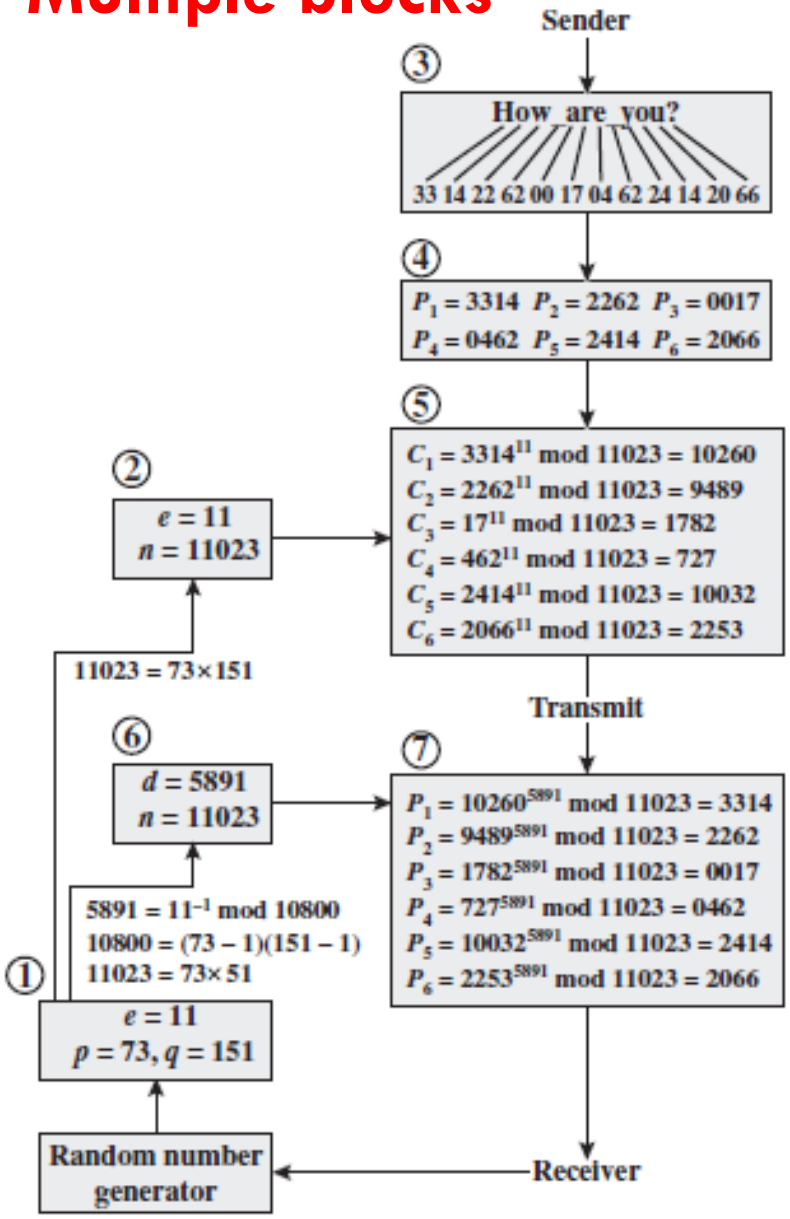
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

RSA: Example


Encryption
Decryption
Plaintext
88
88^7 mod 187 = 11
Ciphertext
11
11^23 mod 187 = 88
Plaintext
88
PU = 7, 187
PR = 23, 187

RSA: Processing of Multiple blocks





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

Exponentiation in Modular Arithmetic

$$a^b \bmod n$$

If we express b as a binary number $b_k b_{k-1} \ldots b_0$, then

$$b = \sum_{b_i \neq 0} 2^i$$

Therefore,

$$a^b = a^{\left(\begin{smallmatrix}\sum 2^i \\ b_i \neq 0\end{smallmatrix}\right)} = \prod_{b_i \neq 0} a^{(2^i)}$$

Algorithm for Computing a^b mod n

$$a^b \bmod n = \left[\prod_{b_i \neq 0} a^{(2^i)}\right] \bmod n = \left(\prod_{b_i \neq 0}\left[a^{(2^i)} \bmod n\right]\right) \bmod n$$

Note that
the variable c is not needed; it is included for explanatory purposes. The final value of c is the value of the exponent.

$$x^{11} = x^{1+2+8} = (x)(x^2)(x^8)$$

```
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
```

Algorithm for Computing a^b mod n

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

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

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

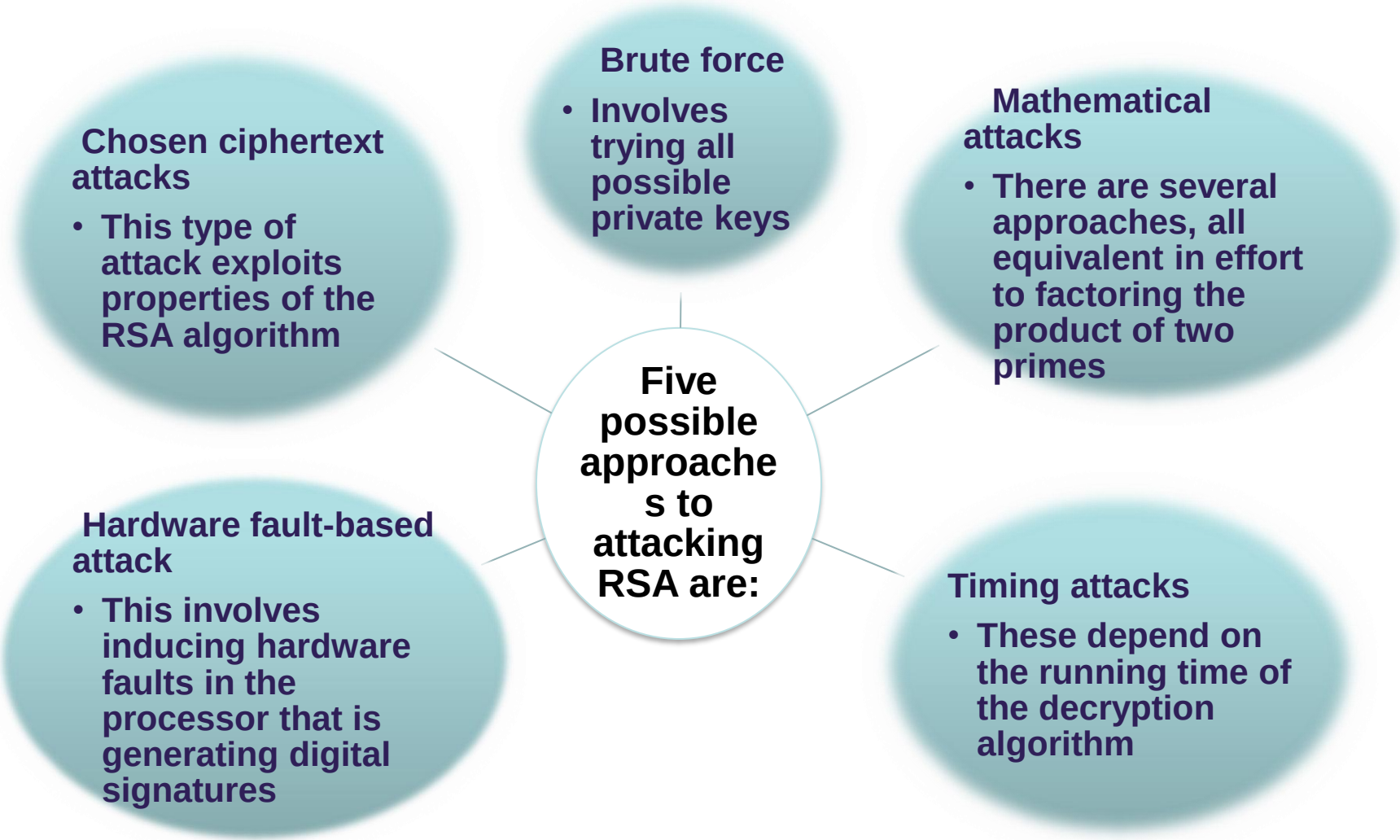
Key Generation

- 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

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

Security of RSA



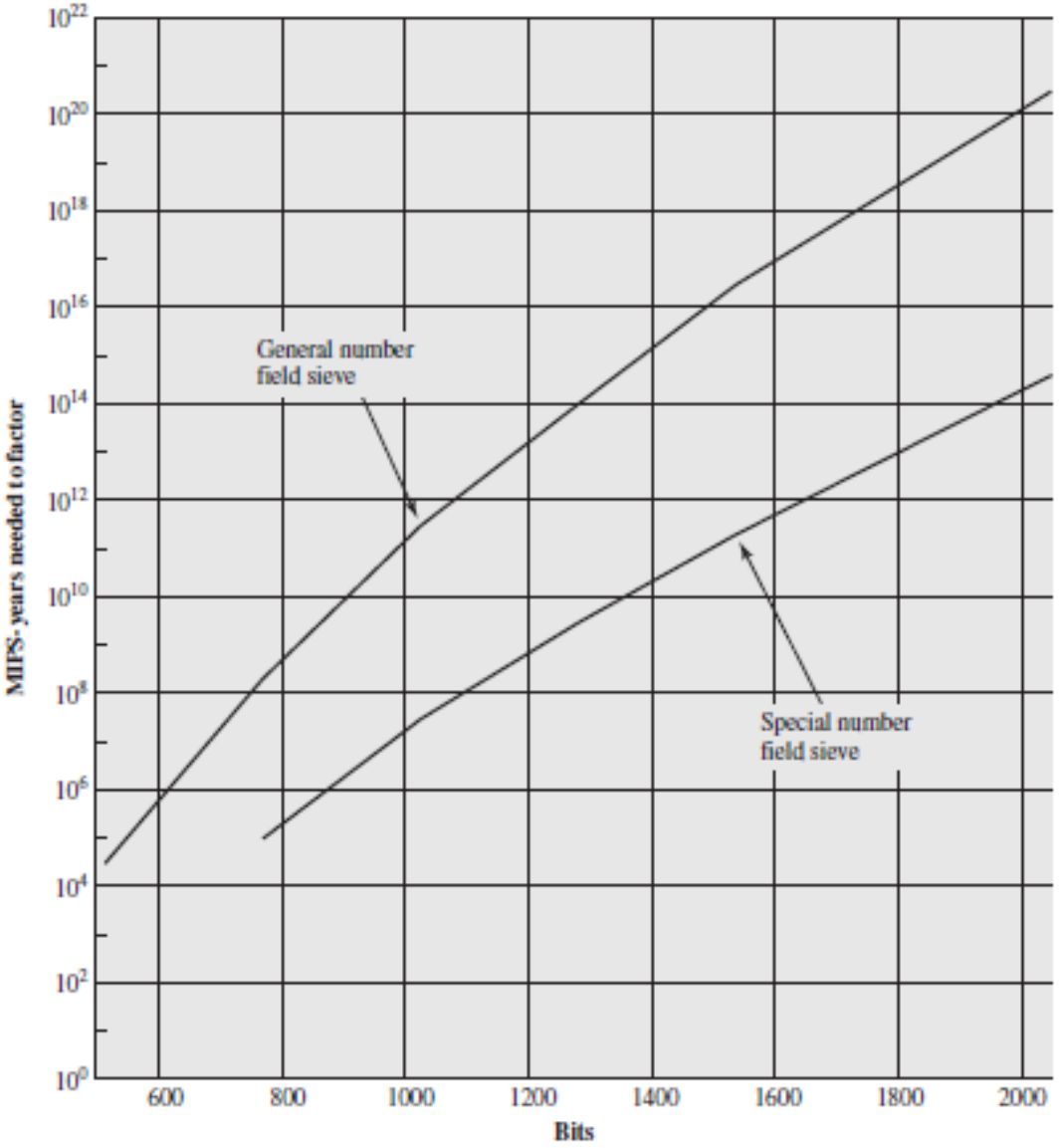
Factoring Problem

- We can identify three approaches to attacking RSA mathematically:
  - Factor *n* into its two prime factors. This enables calculation of $ø(n) = (p - 1) \times (q - 1)$, 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)

Progress in RSA Factorization

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

MIPS Years needed to Factor



Timing Attack

- 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

Counter Measures

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

- 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

Chosen CipherText Attack

- To counter such attacks, RSA Security Inc. recommends modifying the plaintext using a procedure known as *optimal asymmetric encryption padding* (OAEP)

Attacks on RSA

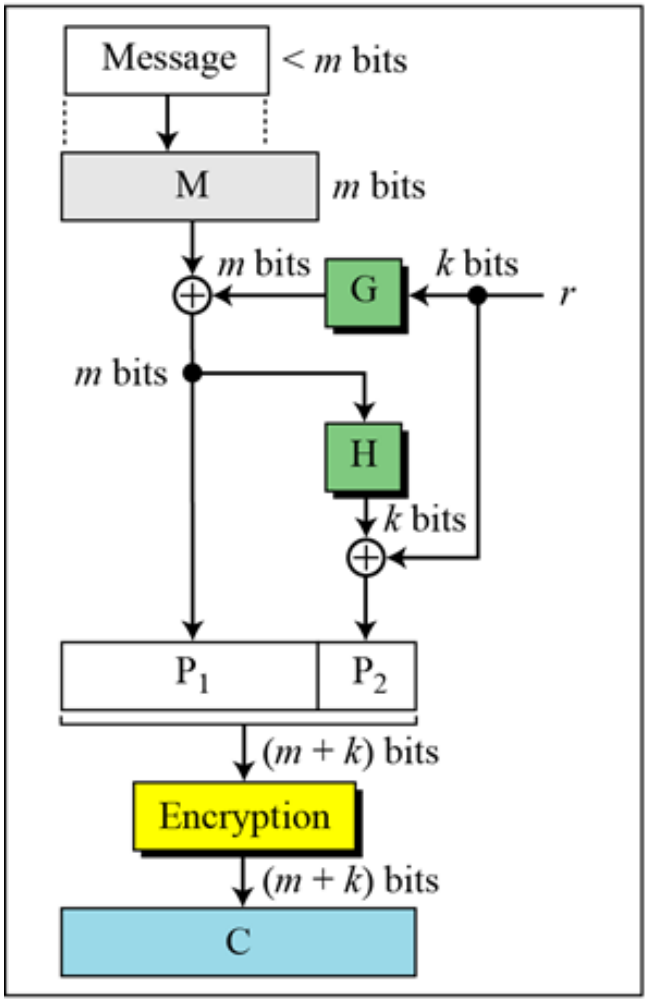
Attack bsed on the multiplicative property of RSA. Let Alice creates $C=P^e \bmod n$ and sends it to Bob. Bob will decrypt an arbitrary ciphertext for Eve other than C. Eve intercepts C and do following steps to get P

a. Eve chooses a random integer X in Z_n^*.
b. Eve calculates $Y=C\times X^e \bmod n$.
c. Eve sends Y to Bob for decryption and get $Z= Y^d \bmod n$;
   This step is an instance of a chosen-ciphertext attack.
d. Eve can easily find P because

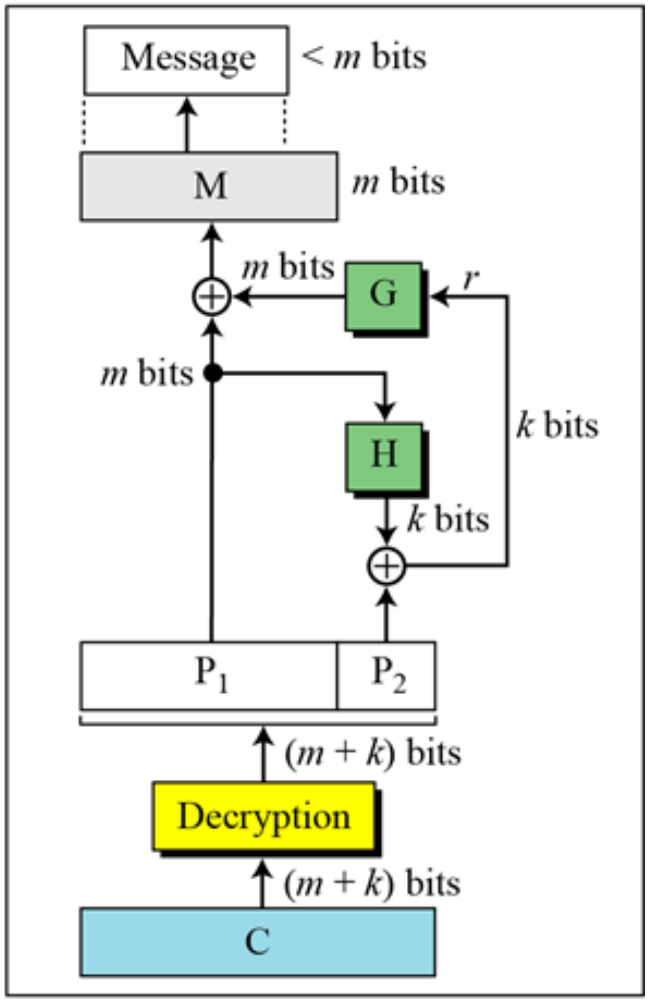
$$Z= Y^d \bmod n=(C\times X^e)^d \bmod n$$
$$=(C^d \times X^{ed}) \bmod n=(C^d \times X) \bmod n$$
$$=(P \times X) \bmod n \rightarrow P=Z\times X^{-1} \bmod n$$

Optimal Asymmetric Encryption Padding (OAEP)

M: Padded message
r: One-time random number
P: Plaintext ($P_1 \| P_2$)
C: Ciphertext
G: Public function (*k*-bit to *m*-bit)
H: Public function (*m*-bit to *k*-bit)



Sender



Receiver

Continued

Proof of RSA

If $n = p \times q$, $a < n$, and k is an integer, then $a^{k \times \phi(n)+1} \equiv a \pmod{n}$.

$P_1 = C^d \bmod n = (P^e \bmod n)^d \bmod n = P^{ed} \bmod n$

$ed = k\phi(n) + 1$ // d and e are inverses modulo $\phi(n)$

$P_1 = P^{ed} \bmod n \rightarrow P_1 = P^{k\phi(n)+1} \bmod n$

$P_1 = P^{k\phi(n)+1} \bmod n = P \bmod n$ // Euler's theorem (second version)

Some Trivial Examples

Example

Bob chooses 7 and 11 as p and q and calculates n = 77. The value of f(n) = (7 − 1)(11 − 1) or 60. Now he chooses two exponents, e and d, from Z_{60}*. If he chooses e to be 13, then d is 37. Note that e × d mod 60 = 1 (they are inverses of each Now imagine that Alice wants to send the plaintext 5 to Bob. She uses the public exponent 13 to encrypt 5.

Plaintext: 5	$C = 5^{13} = 26 \bmod 77$	Ciphertext: 26

Bob receives the ciphertext 26 and uses the private key 37 to decipher the ciphertext:

Ciphertext: 26	$P = 26^{37} = 5 \bmod 77$	Plaintext: 5

Some Trivial Examples

Example

Now assume that another person, John, wants to send a message to Bob. John can use the same public key announced by Bob (probably on his website), 13; John's plaintext is 63. John calculates the following:

Plaintext: 63	$C = 63^{13} = 28 \bmod 77$	Ciphertext: 28

Bob receives the ciphertext 28 and uses his private key 37 to decipher the ciphertext:

Ciphertext: 28	$P = 28^{37} = 63 \bmod 77$	Plaintext: 63

Some Trivial Examples

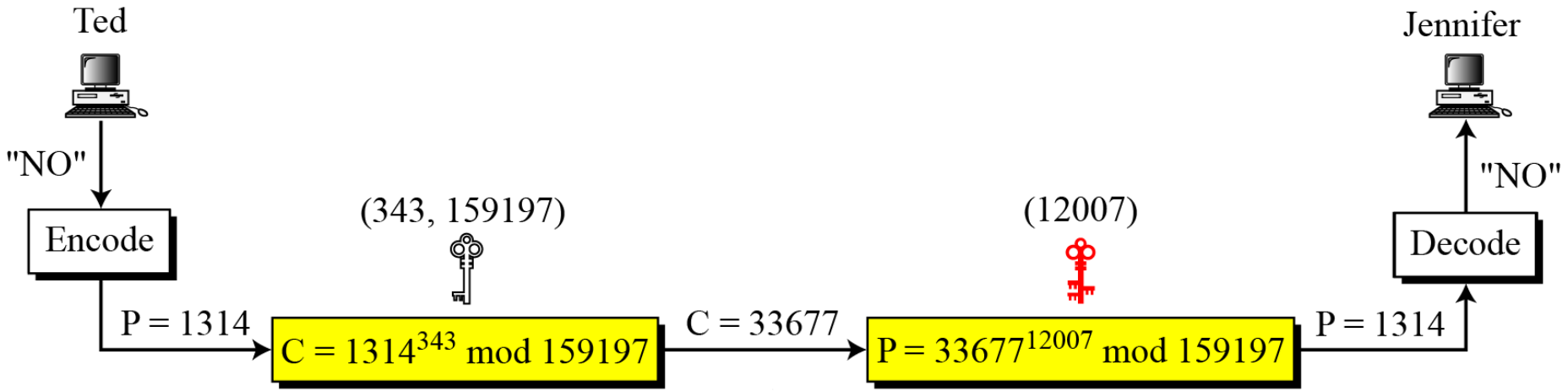
Example

Jennifer creates a pair of keys for herself. She chooses $p = 397$ and $q = 401$. She calculates $n = 159197$. She then calculates $f(n) = 158400$. She then chooses $e = 343$ and $d = 12007$. Show how Ted can send a message to Jennifer if he knows e and n.

Suppose Ted wants to send the message "NO" to Jennifer. He changes each character to a number (from 00 to 25), with each character coded as two digits. He then concatenates the two coded characters and gets a four-digit number. The plaintext is 1314. Figure 10.7 shows the process.

Continued

Figure Encryption and decryption in Example



Attacks on RSA

Attack bsed on the multiplicative property of RSA. Let Alice creates $C=P^e \bmod n$ and sends it to Bob. Bob will decrypt an arbitrary ciphertext for Eve other than C. Eve intercepts C and do following steps to get P

a. Eve chooses a random integer X in Z_n^*.
b. Eve calculates $Y = C \times X^e \bmod n$.
c. Eve sends Y to Bob for decryption and get $Z = Y^d \bmod n$; This step is an instance of a chosen-ciphertext attack.
d. Eve can easily find P because

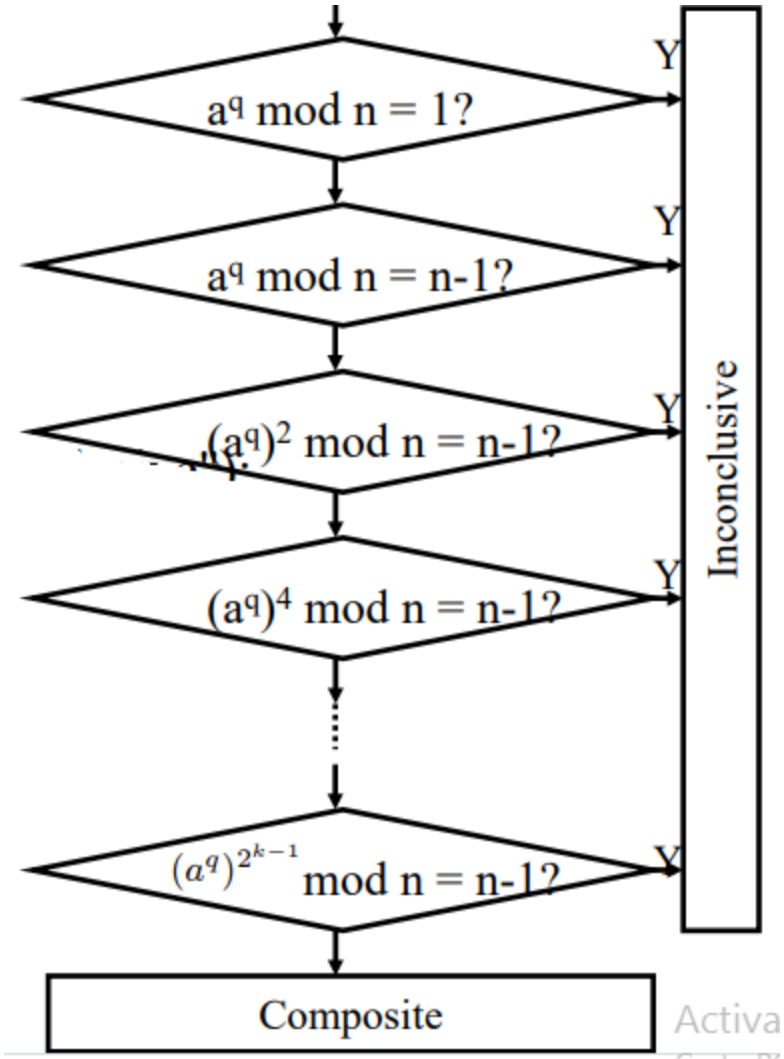
$$Z = Y^d \bmod n = (C \times X^e)^d \bmod n = (C^d \times X^{ed}) \bmod n = (C^d \times X) \bmod n = (P \times X) \bmod n$$
$$Z = (P \times X) \bmod n \quad \rightarrow \quad P = Z \times X^{-1} \bmod n$$

Miller-Rabin Algorithm for Primality Test

The procedure TEST takes a candidate integer n as input and returns the result composite if n is definitely not a prime, and the result inconclusive if n may or may not be a prime.

```
TEST (n)
1. Find integers k, q, with k > 0, q odd, so that
   (n − 1 = 2^k q);
2. Select a random integer a, 1 < a < n − 1;
3. if a^q mod n = 1 then return("inconclusive");
4. for j = 0 to k − 1 do
5. if a^(2^j q) mod n = n − 1 then return("inconclusive");
6. return("composite");
```

Miller-Rabin Algorithm for Primality Test



Miller-Rabin Algorithm for Primality Test

Test 29 for primality

- $29\text{-}1 = 28 = 2^2 \times 7 = 2^k q \Rightarrow k=2, q=7$
- Let a = 10
  - $10^7 \bmod 29 = 17$
  - $10^{2\times 7} \bmod 29 = 17^2 \bmod 29 = 28 \Rightarrow$ Inconclusive

Test 221 for primality

- $221\text{-}1=220=2^2 \times 55$
- Let a=5
  - $5^{55} \bmod 221 = 112$
  - $5^{2\times 55} \bmod 221 = 112^2 \bmod 221 = 168 \Rightarrow$ Composite

Chinese Remainder Theorem (cont.)

The Chinese remainder theorem (CRT) is used to solve a set of congruent equations with one variable but different moduli, which are relatively prime, as shown below:

$$
\begin{aligned}
x &\equiv a_1 \pmod{m_1} \\
x &\equiv a_2 \pmod{m_2} \\
&\cdots \\
x &\equiv a_k \pmod{m_k}
\end{aligned}
$$

Chinese Remainder Theorem (cont.)

Example

The following is an example of a set of equations with different moduli:

$$x \equiv 2 \pmod 3$$
$$x \equiv 3 \pmod 5$$
$$x \equiv 2 \pmod 7$$

The solution to this set of equations is given in the next section; for the moment, note that the answer to this set of equations is $x = 23$. This value satisfies all equations: $23 \equiv 2 \pmod 3$, $23 \equiv 3 \pmod 5$, and $23 \equiv 2 \pmod 7$.

Chinese Remainder Theorem (cont.)

Solution To Chinese Remainder Theorem

1. Find $M = m_1 \times m_2 \times \ldots \times m_k$. This is the common modulus.
2. Find $M_1 = M/m_1$, $M_2 = M/m_2$, …, $M_k = M/m_k$.
3. Find the multiplicative inverse of M_1, M_2, …, M_k using the corresponding moduli (m_1, m_2, …, m_k). Call the inverses M_1^{-1}, M_2^{-1}, …, M_k^{-1}.
4. The solution to the simultaneous equations is

$$x = (a_1 \times M_1 \times M_1^{-1} + a_2 \times M_2 \times M_2^{-1} + \cdots + a_k \times M_k \times M_k^{-1}) \bmod M$$

Chinese Remainder Theorem (cont.)

Example

Find the solution to the simultaneous equations:

$$x \equiv 2 \pmod 3$$
$$x \equiv 3 \pmod 5$$
$$x \equiv 2 \pmod 7$$

Solution

We follow the four steps.

1. $M = 3 \times 5 \times 7 = 105$

2. $M_1 = 105 / 3 = 35$, $M_2 = 105 / 5 = 21$, $M_3 = 105 / 7 = 15$

3. The inverses are $M_1^{-1} = 2$, $M_2^{-1} = 1$, $M_3^{-1} = 1$

4. $x = (2 \times 35 \times 2 + 3 \times 21 \times 1 + 2 \times 15 \times 1) \bmod 105 = 23 \bmod 105$

Chinese Remainder Theorem (cont.)

Example

Find an integer that has a remainder of 3 when divided by 7 and 13, but is divisible by 12.

Solution

This is a CRT problem. We can form three equations and solve them to find the value of x.

$$x = 3 \bmod 7$$
$$x = 3 \bmod 13$$
$$x = 0 \bmod 12$$

If we follow the four steps, we find x = 276. We can check that 276 = 3 mod 7, 276 = 3 mod 13 and 276 is divisible by 12 (the quotient is 23 and the remainder is zero).

Chinese Remainder Theorem (cont.)

- mod M = $m_1 m_2 .. m_k$
- Chinese Remainder theorem lets us work in each moduli m_i separately
- Since computational cost is proportional to size, this is faster

$$A \bmod M = \sum_{i=1}^{k} (A \bmod m_i) \frac{M}{m_i} \left(\left[\frac{M}{m_i} \right]^{-1} \bmod m_i \right)$$

Chinese Remainder Theorem (cont.)

$$A \bmod M = \sum_{i=1}^{k} (A \bmod m_i) \frac{M}{m_i} \left(\left[\frac{M}{m_i} \right]^{-1} \bmod m_i \right)$$

$35^{-1} = x \bmod 3$
$35x = 1 \bmod 3 \Rightarrow x = 2$
$21x = 1 \bmod 5 \Rightarrow x = 1$
$15x = 1 \bmod 7 \Rightarrow x = 1$

Example: 452 mod 105

$= (452 \bmod 3)(105/3)\{(105/3)^{-1} \bmod 3\}$
$+(452 \bmod 5)(105/5)\{(105/5)^{-1} \bmod 5\}$
$+(452 \bmod 7)(105/7)\{(105/7)^{-1} \bmod 7\}$
$= 2 \times 35 \times (35^{-1} \bmod 3) + 2x21 \times (21^{-1} \bmod 5) + 4 \times 15 \times (15^{-1} \bmod 7)$
$= 2 \times 35 \times 2 + 2 \times 21 \times 1 + 4 \times 15 \times 1$
$= (140+42+60) \bmod 105 = 242 \bmod 105 = 32$

Chinese Remainder Theorem (cont.)

$x = a_1 \bmod m_1$

$x = a_2 \bmod m_2$

$x = a_k \bmod m_k$

where $m_1, m_2, \ldots, m_k$ are relatively prime is found as follows:

$M = m_1 m_2 \ldots m_k$ then

$$x = \sum_{i=1}^{k} a_i \frac{M}{m_i} \left(\left[\frac{M}{m_i} \right]^{-1} \bmod m_i \right)$$

Chinese Remainder Theorem (cont.)

$$
\begin{aligned}
x &= 1 \bmod 7 \\
x &= 2 \bmod 8 \\
x &= 3 \bmod 9 \\
N &= 7 \times 8 \times 9 = 504 \\
x &= \left(1 \times \tfrac{504}{7} \times \left[\tfrac{504}{7}\right]_7^{-1} + 2 \times \tfrac{504}{8} \times \left[\tfrac{504}{8}\right]_8^{-1}\right. \\
&\quad \left. +3 \times \tfrac{504}{9} \times \left[\tfrac{504}{9}\right]_9^{-1}\right) \bmod 7 \times 8 \times 9 \\
&= (1 \times 72 \times (72^{-1} \bmod 7) + 2 \times 63 \times (63^{-1} \bmod 8) \\
&\quad +3 \times 56 \times (56^{-1} \bmod 9)) \bmod 504 \\
&= (1 \times 72 \times 4 + 2 \times 63 \times 7 + 3 \times 56 \times 5) \bmod 504 \\
&= (288 + 882 + 840) \bmod 504 \\
&= 2010 \bmod 504 \\
&= 498
\end{aligned}
$$

Fermat's Little Theorem

Given a prime number p:

$$a^{p-1} = 1 \pmod{p}$$

For all integers $a \neq p$

Or

$$a^{p} = a \pmod{p}$$

Example:

- $1^4 \bmod 5 = 1$
- $2^4 \bmod 5 = 1$
- $3^4 \bmod 5 = 1$
- $4^4 \bmod 5 = 1$

Euler's Theorem

A generalisation of Fermat's Theorem

$$a^{\phi(n)} = 1 \pmod{n}$$

- for any a, n where $\gcd(a,n)=1$

Example:

$a=3$; $n=10$; $\phi(10)=4$;

hence $3^4 = 81 = 1 \bmod 10$

$a=2$; $n=11$; $\phi(11)=10$;

hence $2^{10} = 1024 = 1 \bmod 11$

Also have: $a^{\phi(n)+1} = a \pmod{n}$

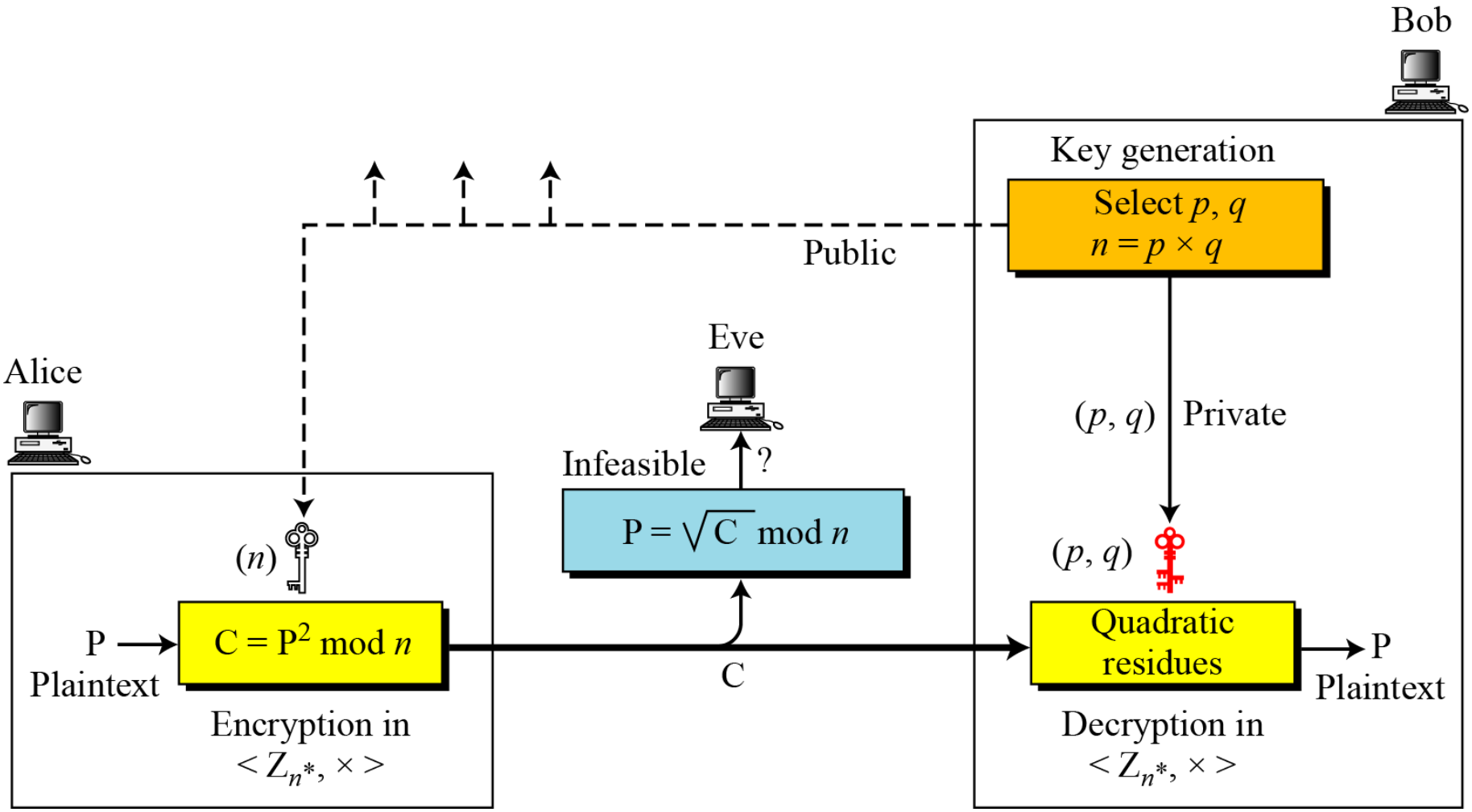
RABIN CRYPTOSYSTEM

The Rabin cryptosystem can be thought of as an RSA cryptosystem in which the value of e and d are fixed. The encryption is $C \equiv P^2 \pmod{n}$ *and the decryption is* $P \equiv C^{1/2} \pmod{n}$.

Topics :

1 **Procedure**
2 **Security of the Rabin System**

Continued


Bob
Key generation
Select p, q
$n = p \times q$
Public
Alice
Eve
(p, q) Private
Infeasible
?
$P = \sqrt{C} \bmod n$
(n)
(p, q)
P
Plaintext
$C = P^2 \bmod n$
C
Quadratic residues
P
Plaintext
Encryption in
$< Z_n^*, \times >$
Decryption in
$< Z_n^*, \times >$

Continued

Algorithm *Key generation for Rabin cryptosystem*

Rabin_Key_Generation

{

Choose two large primes p and q in the form $4k + 3$ and $p \neq q$.

$n \leftarrow p \times q$

Public_key $\leftarrow$ n // To be announced publicly

Private_key $\leftarrow$ (q, n) // To be kept secret

return Public_key and Private_key

}

Continued

Encryption

Algorithm *Encryption in Rabin cryptosystem*

```
Rabin_Encryption (n, P)          // n is the public key; P is the ciphertext from Z_n*
{
    C  ←  P^2 mod n              // C is the ciphertext
    return C
}
```

Continued

Decryption

Algorithm *Decryption in Rabin cryptosystem*

Rabin_Decryption (p, q, C) // C is the ciphertext; p and q are private keys

{

$a_1 \leftarrow +(C^{(p+1)/4}) \bmod p$

$a_2 \leftarrow -(C^{(p+1)/4}) \bmod p$

$b_1 \leftarrow +(C^{(q+1)/4}) \bmod q$

$b_2 \leftarrow -(C^{(q+1)/4}) \bmod q$

// The algorithm for the Chinese remainder algorithm is called four times.

$P_1 \leftarrow$ Chinese_Remainder (a_1, b_1, p, q)

$P_2 \leftarrow$ Chinese_Remainder (a_1, b_2, p, q)

$P_3 \leftarrow$ Chinese_Remainder (a_2, b_1, p, q)

$P_4 \leftarrow$ Chinese_Remainder (a_2, b_2, p, q)

return P_1, P_2, P_3, and P_4

}

The Rabin cryptosystem is not deterministic: Decryption creates four plaintexts.

Continued

Example

Here is a very trivial example to show the idea.

1. **Bob selects $p = 23$ and $q = 7$. Note that both are congruent to 3 mod 4.**
2. **Bob calculates $n = p \times q = 161$.**
3. **Bob announces n publicly; he keeps p and q private.**
4. **Alice wants to send the plaintext $P = 24$. Note that 161 and 24 are relatively prime; 24 is in Z_{161}^*. She calculates $C = 24^2 = 93 \bmod 161$, and sends the ciphertext 93 to Bob.**

Continued

Example

5. **Bob receives 93 and calculates four values:**

   $a_1 = +(93^{(23+1)/4}) \bmod 23 = 1 \bmod 23$

   $a_2 = -(93^{(23+1)/4}) \bmod 23 = 22 \bmod 23$

   $b_1 = +(93^{(7+1)/4}) \bmod 7 = 4 \bmod 7$

   $b_2 = -(93^{(7+1)/4}) \bmod 7 = 3 \bmod 7$

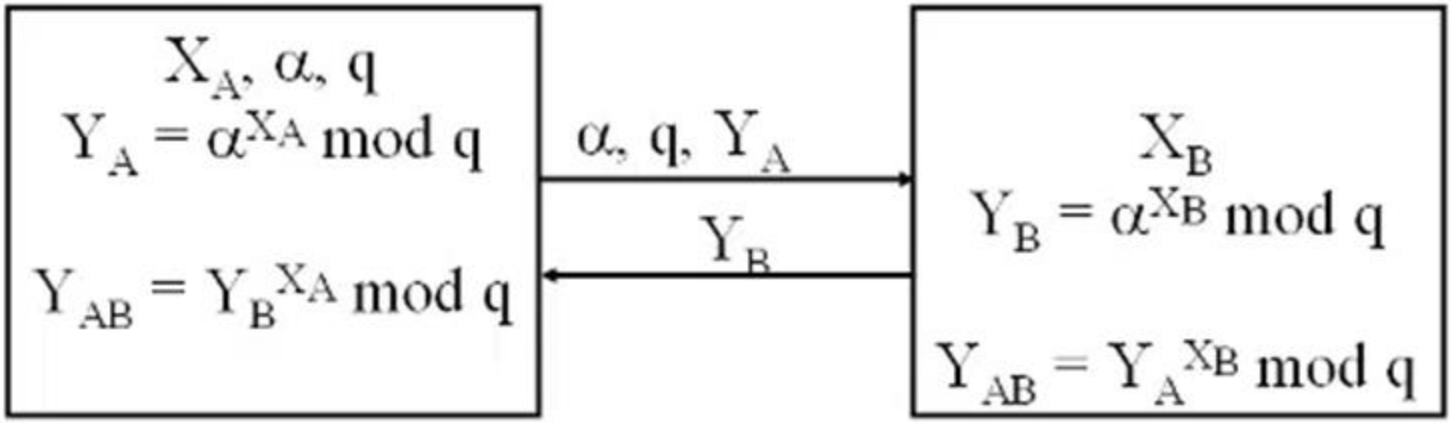
6. **Bob takes four possible answers, (a_1, b_1), (a_1, b_2), (a_2, b_1), and (a_2, b_2), and uses the Chinese remainder theorem to find four possible plaintexts: 116, 24, 137, and 45. Note that only the second answer is Alice's plaintext.**

Diffie-Hellman Key Exchange

- First public-key type scheme proposed
- By Diffie & Hellman in 1976 along with the exposition of public key concepts
  - note: now know that Williamson (UK CESG) secretly proposed the concept in 1970
- Is a practical method for public exchange of a secret key
- Used in a number of commercial products

Diffie-Hellman Key Exchange

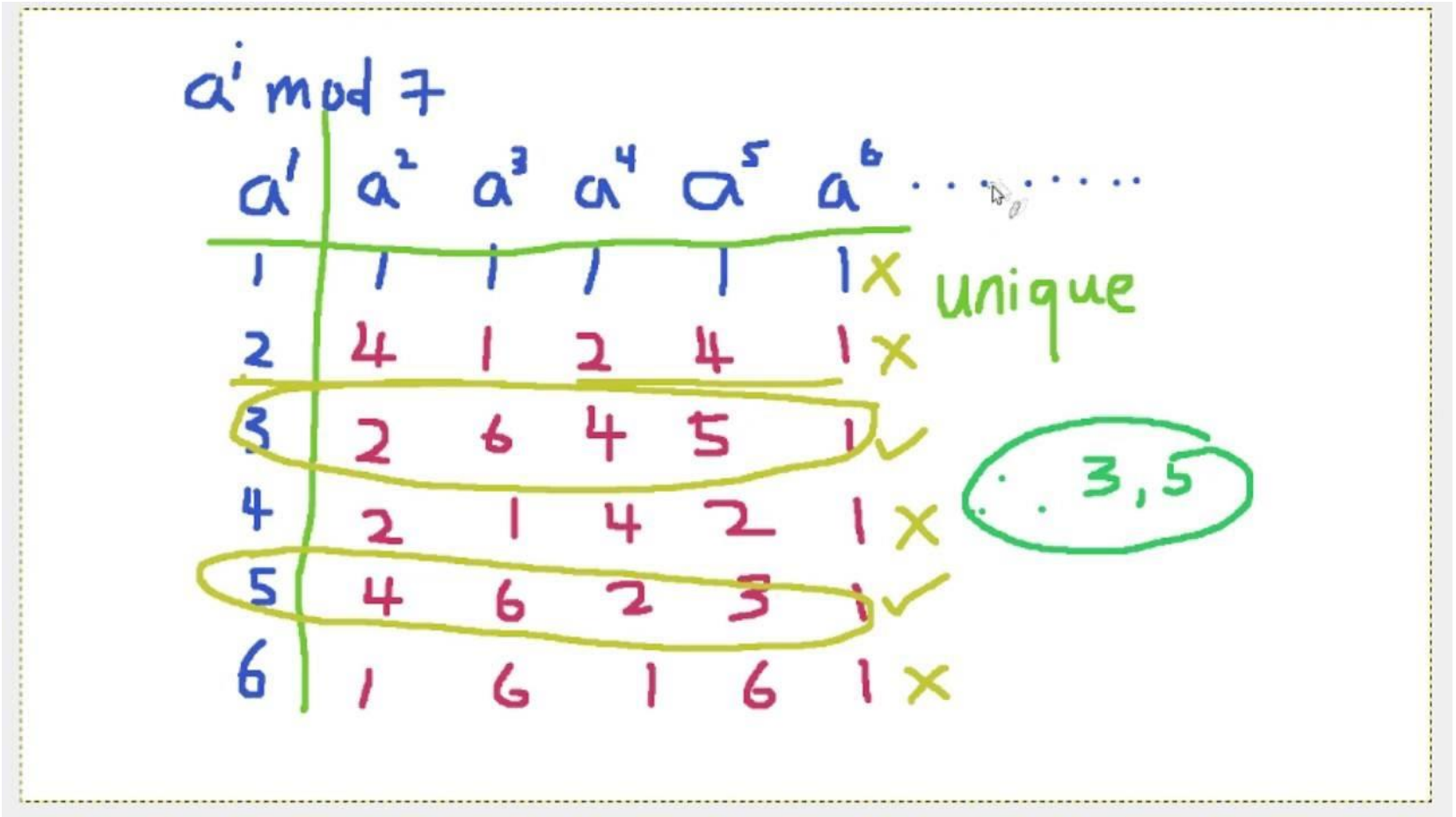
- Allows two party to agree on a secret key using a public channel
- A selects q=large prime, and α=a primitive root of q
- A selects a random # X_A, B selects another random # X_B



$$Y_{AB} = \alpha^{X_A X_B} \bmod q$$

- Eavesdropper can see Y_A, α, q but cannot compute X_A
- Computing X_A requires discrete logarithm - a difficult problem

Diffie-Hellman Key Exchange

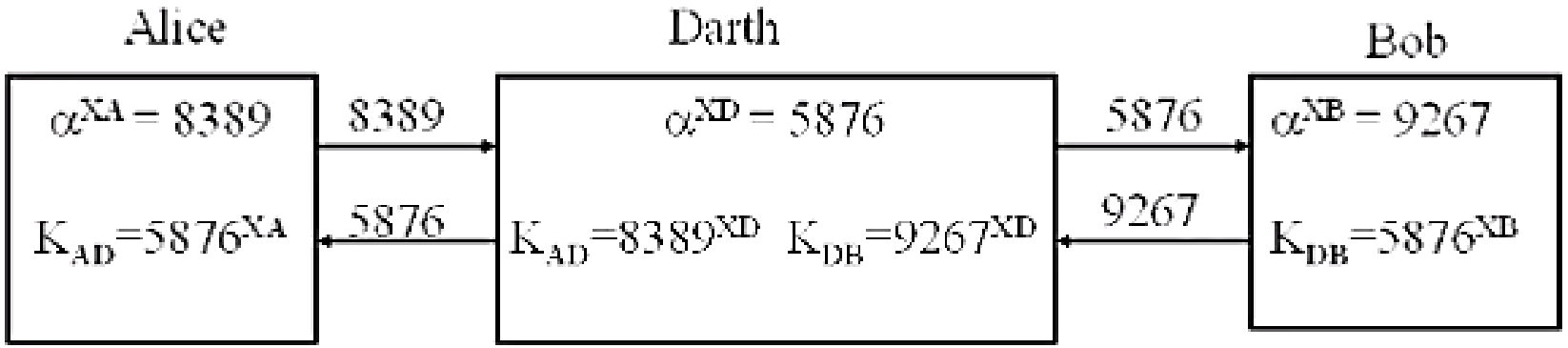


Diffie-Hellman Key Exchange (cont.)

- Example: α=5, q=19
  - A selects 6 and sends $5^6 \bmod 19 = 7$
  - B selects 7 and sends $5^7 \bmod 19 = 16$
  - A computes $K = 16^6 \bmod 19 = 7$
  - B computes $K = 7^7 \bmod 19 = 7$
- Preferably (q-1)/2 should also be a prime.
- Such primes are called safe prime.

Man-in-the-Middle Attack

- Diffie-Hellman does not provide authentication



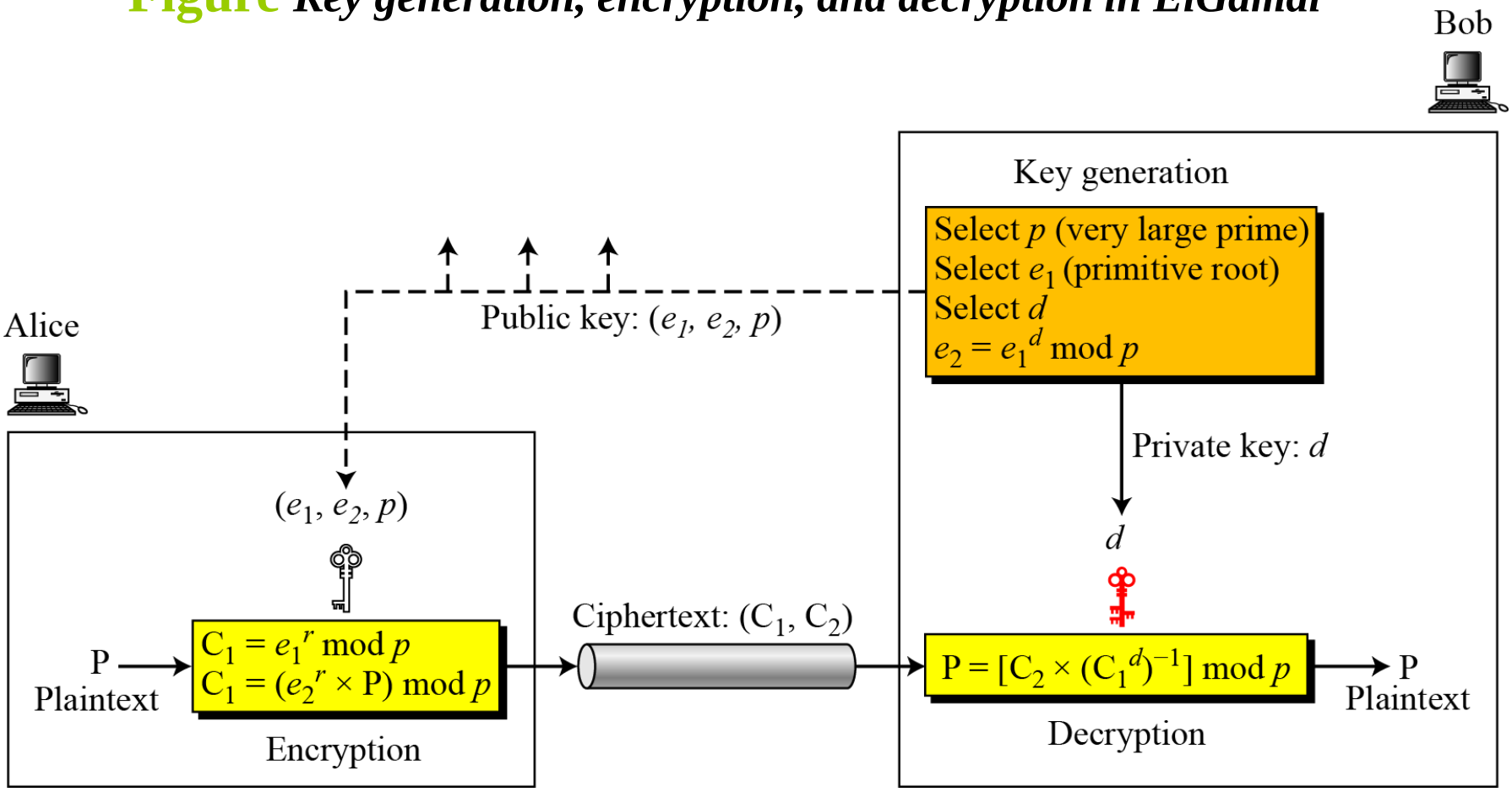
- X can then intercept, decrypt, re-encrypt, forward all messages between Alice & Bob
- You can use RSA authentication and other alternatives

ELGAMAL CRYPTOSYSTEM

Besides RSA and Rabin, another public-key cryptosystem is ElGamal. ElGamal is based on the discrete logarithm problem.

Procedure

Figure ***Key generation, encryption, and decryption in ElGamal***



Continued

Example

Here is a trivial example. Bob chooses p = 11 and e_1 = 2. and d = 3 $e_2 = e_1^d = 8$. So the public keys are (2, 8, 11) and the private key is 3. Alice chooses r = 4 and calculates C1 and C2 for the plaintext 7.

Plaintext: 7

$C_1 = e_1^r \bmod 11 = 16 \bmod 11 = 5 \bmod 11$

$C_2 = (P \times e_2^r) \bmod 11 = (7 \times 4096) \bmod 11 = 6 \bmod 11$

Ciphertext: (5, 6)

Bob receives the ciphertexts (5 and 6) and calculates the plaintext

$[C_2 \times (C_1^d)^{-1}] \bmod 11 = 6 \times (5^3)^{-1} \bmod 11 = 6 \times 3 \bmod 11 = 7 \bmod 11$

Plaintext: 7

Digital Signature

COMPARISON

Let us begin by looking at the differences between conventional signatures and digital signatures.

Topics discussed in this section:

Inclusion
Verification Method
Relationship
Duplicity

Inclusion

A conventional signature is included in the document; it is part of the document. But when we sign a document digitally, we send the signature as a separate document.

Verification Method

For a conventional signature, when the recipient receives a document, he/she compares the signature on the document with the signature on file. For a digital signature, the recipient receives the message and the signature. The recipient needs to apply a verification technique to the combination of the message and the signature to verify the authenticity.

Relationship

For a conventional signature, there is normally a one-to-many relationship between a signature and documents (same signature to many documents). For a digital signature, there is a one-to-one relationship between a signature and a message.

Duplicity

In conventional signature, a copy of the signed document can be distinguished from the original one on file. In digital signature, there is no such distinction unless there is a factor of time on the document.

PROCESS

Figure in the next slide shows the digital signature process. The sender uses a signing algorithm to sign the message. The message and the signature are sent to the receiver. The receiver receives the message and the signature and applies the verifying algorithm to the combination. If the result is true, the message is accepted; otherwise, it is rejected.

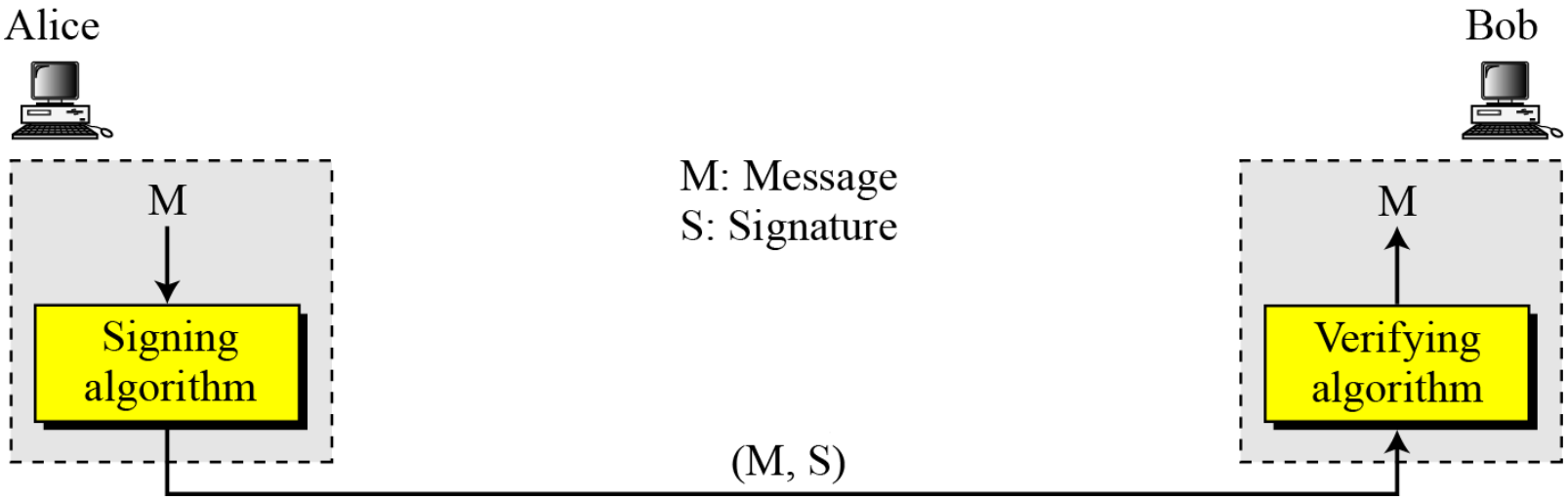
Topics

Need for Keys

Signing the Digest

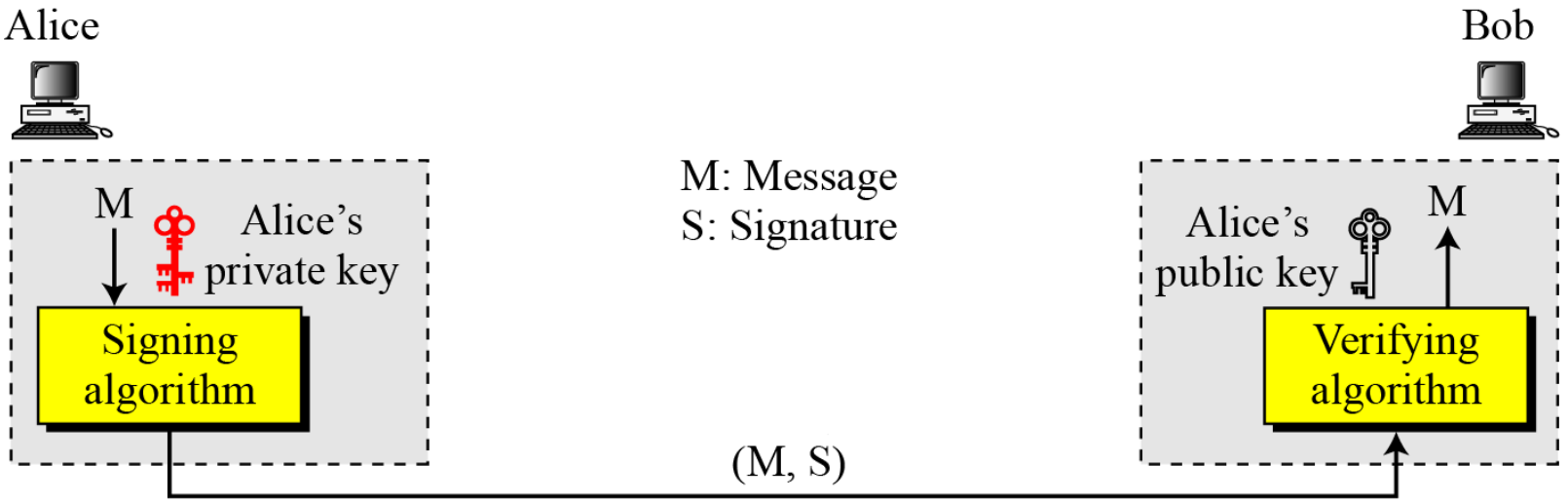
Continued

Figure *Digital signature process*



Need for Keys

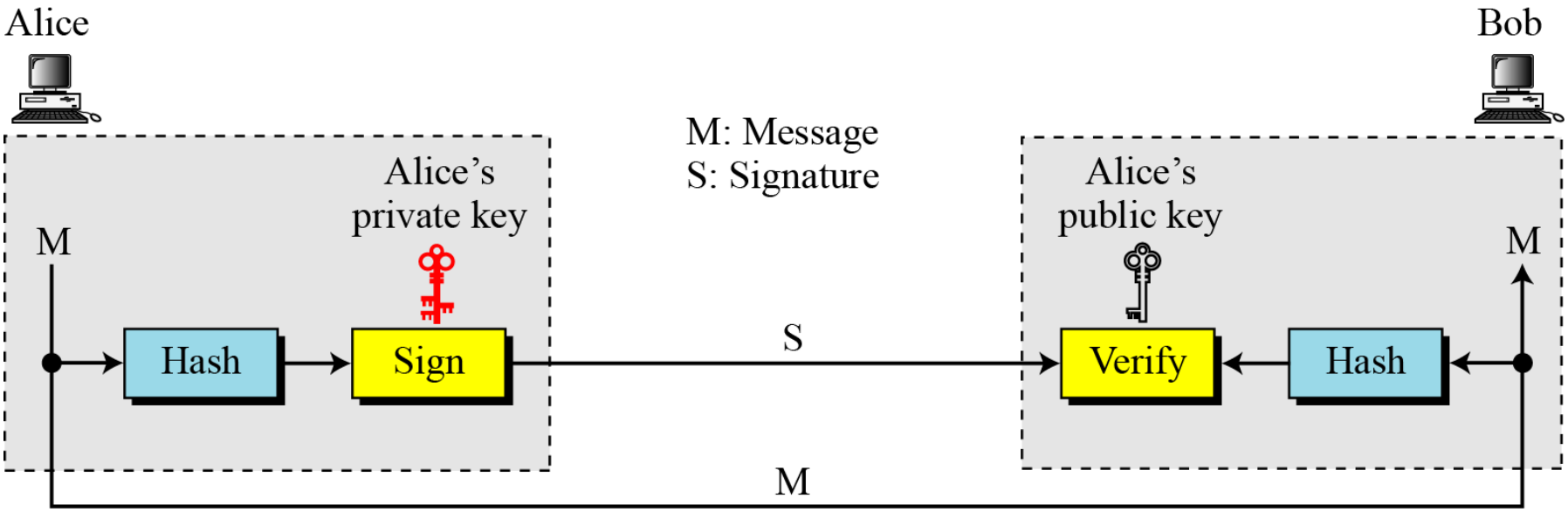
Figure Adding key to the digital signature process



A digital signature needs a public-key system. The signer signs with her private key; the verifier verifies with the signer's public key.

Signing the Digest

Figure Signing the digest



Attacks

- In the order of Increasing severity.
- C=Attacker, A=Victim

1. **Key-only attack**: C only knows A's public key
2. **Known message attack**: C has a set of messages, signatures
3. **Generic chosen message attack**: C obtains A's signatures on messages selected without knowledge of A's public key
4. **Directed chosen message attack**: C obtains A's signatures on messages selected after knowing A's public key
5. **Adaptive chosen message attack**: C may request signatures on messages depending upon previous message-signature pairs

Forgeries

1. **Total break**: C knows A's private key
2. **Universal forgery**: C can generate A's signatures on any message
3. **Selective forgery**: C can generate A's signature for a particular message chosen by C
4. **Existential forgery**: C can generate A's signature for a message not chosen by C

Digital Signature Requirements

- Must depend on the message signed
- Must use information unique to sender
  - To prevent both forgery and denial
- Must be relatively easy to produce
- Must be relatively easy to recognize & verify
  Directed ⇒ Recipient can verify
  Arbitrated ⇒ Anyone can verify
- Be computationally infeasible to forge
  - With new message for existing digital signature
  - With fraudulent digital signature for given message
- Be able to retain a copy of the signature in storage

SERVICES

We discussed several security services including message confidentiality, message authentication, message integrity, and nonrepudiation. A digital signature can directly provide the last three; for message confidentiality we still need encryption/decryption.

Topics

Message Authentication
Message Integrity
Nonrepudiation
Confidentiality

Message Authentication

A secure digital signature scheme, like a secure conventional signature can provide message authentication.

A digital signature provides message authentication.

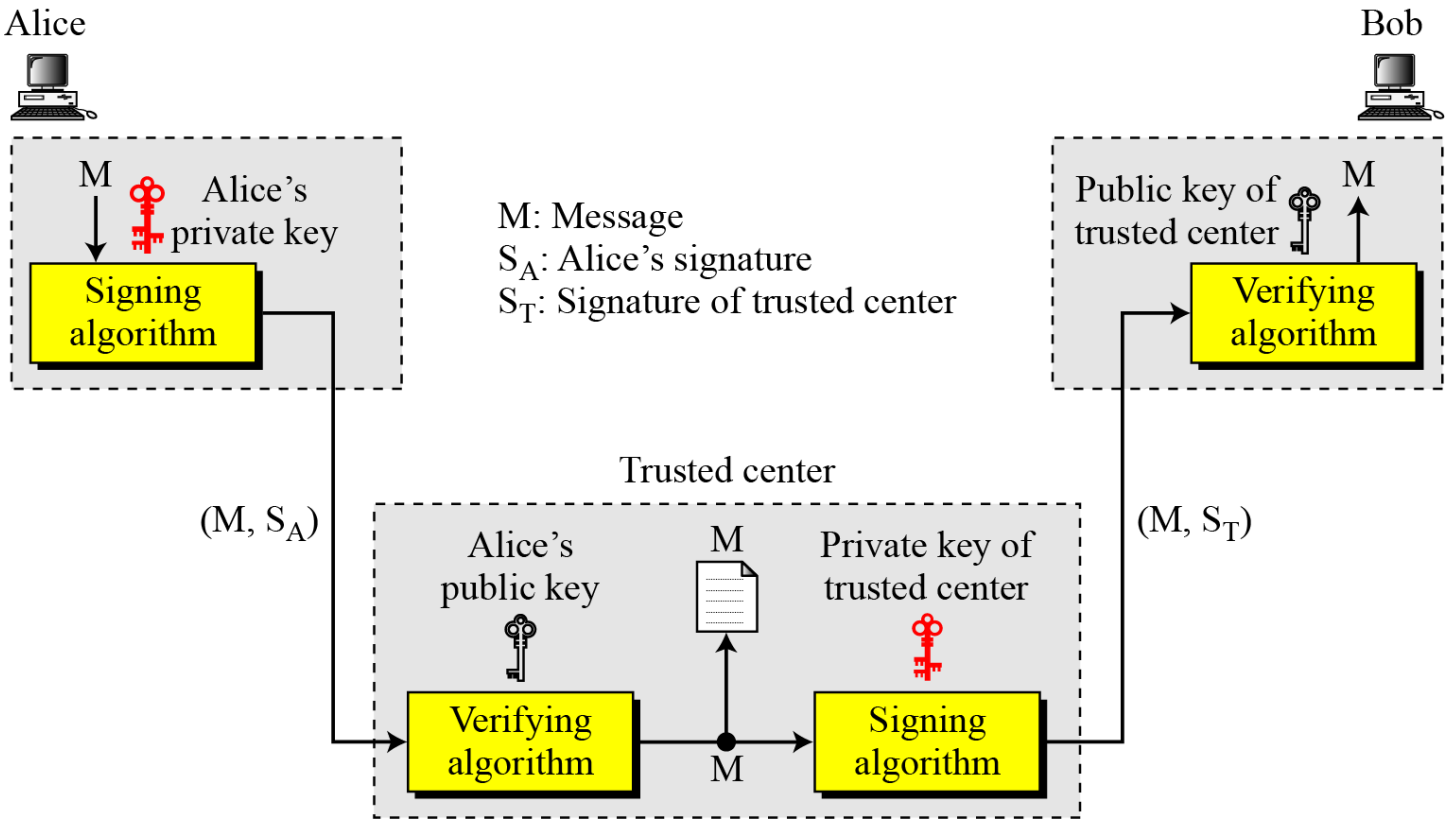
Message Integrity

The integrity of the message is preserved even if we sign the whole message because we cannot get the same signature if the message is changed.

A digital signature provides message integrity.

Nonrepudiation

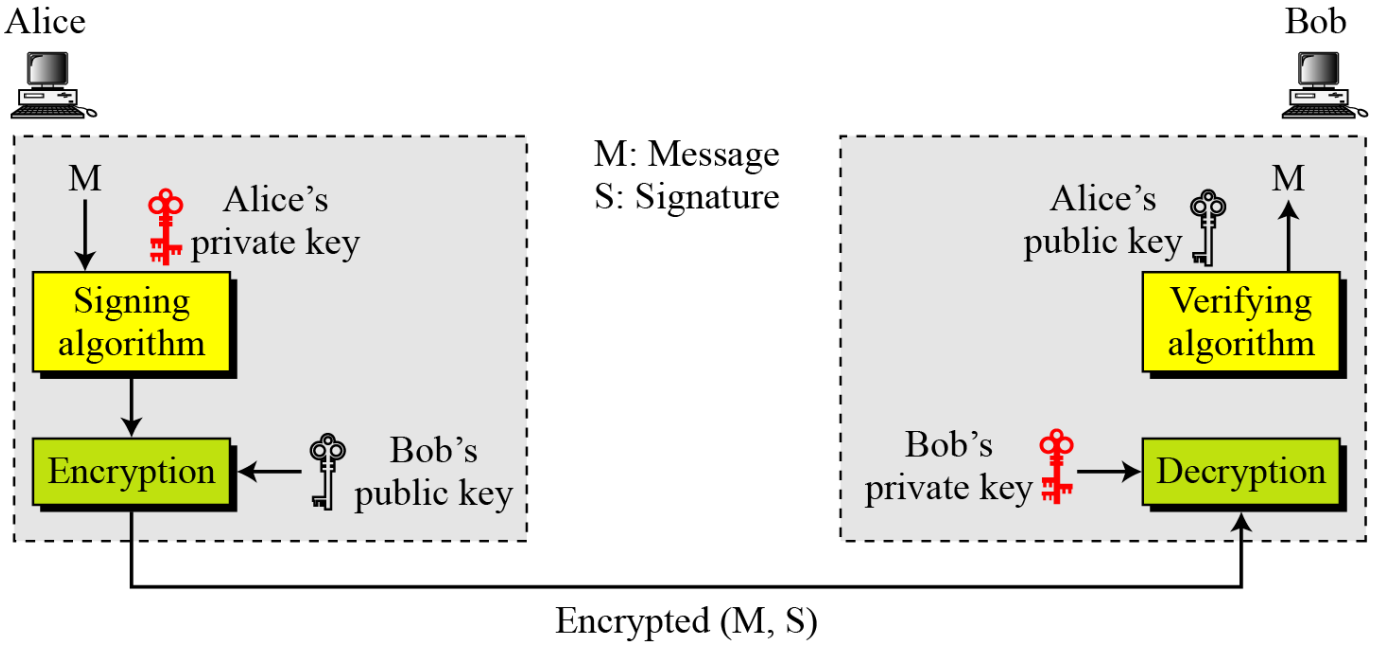
Figure *Using a trusted center for nonrepudiation*



Nonrepudiation can be provided using a trusted party.

Confidentiality

Figure *Adding confidentiality to a digital signature scheme*



A digital signature does not provide privacy. If there is a need for privacy, another layer of encryption/decryption must be applied.

DIGITAL SIGNATURE SCHEMES

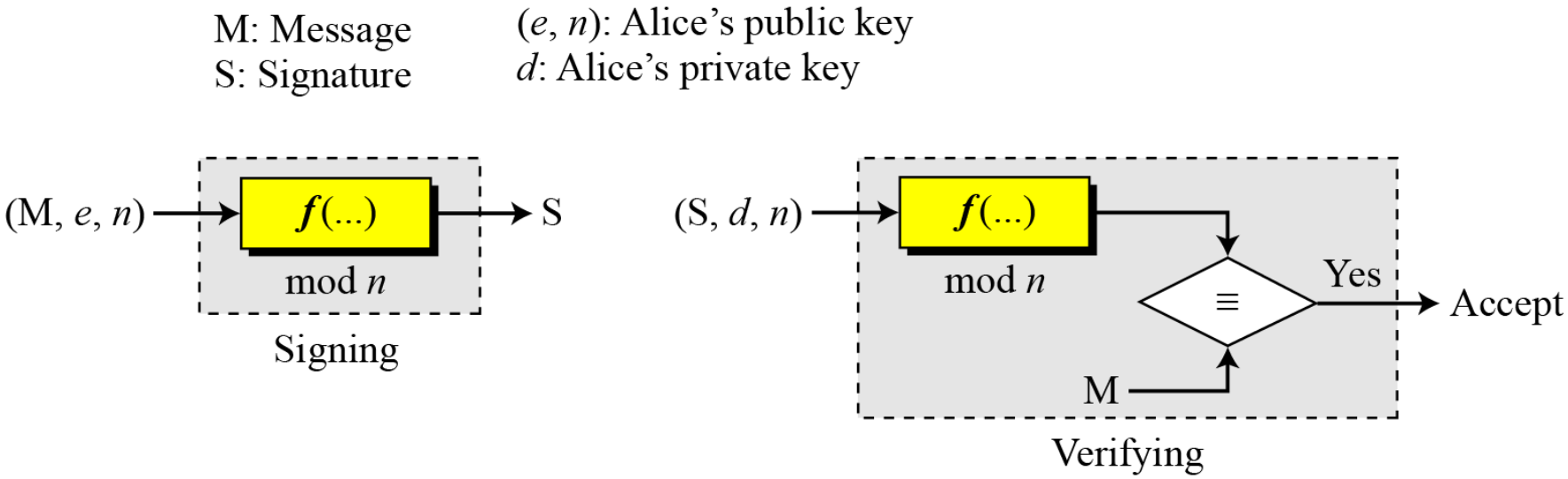
Several digital signature schemes have evolved during the last few decades. Some of them have been implemented.

Topics

1. RSA Digital Signature Scheme
2. ElGamal Digital Signature Scheme
3. Schnorr Digital Signature Scheme
4. Digital Signature Standard (DSS)

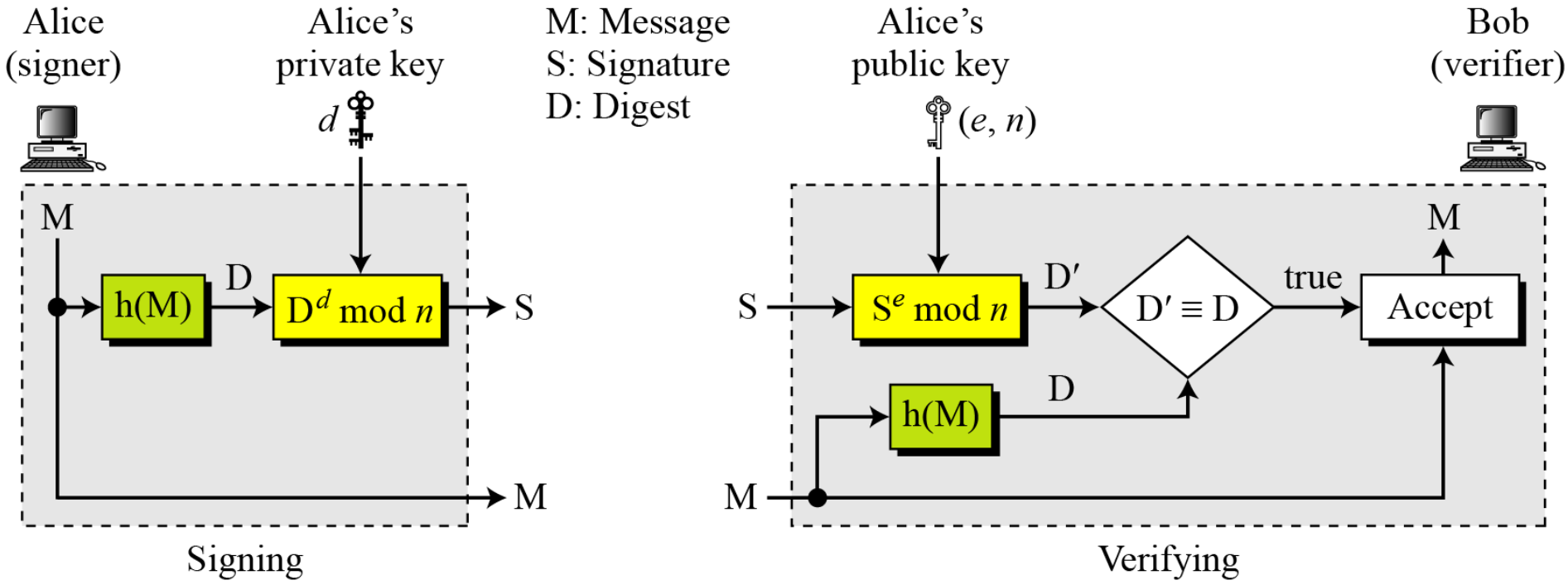
RSA Digital Signature Scheme

Figure General idea behind the RSA digital signature scheme



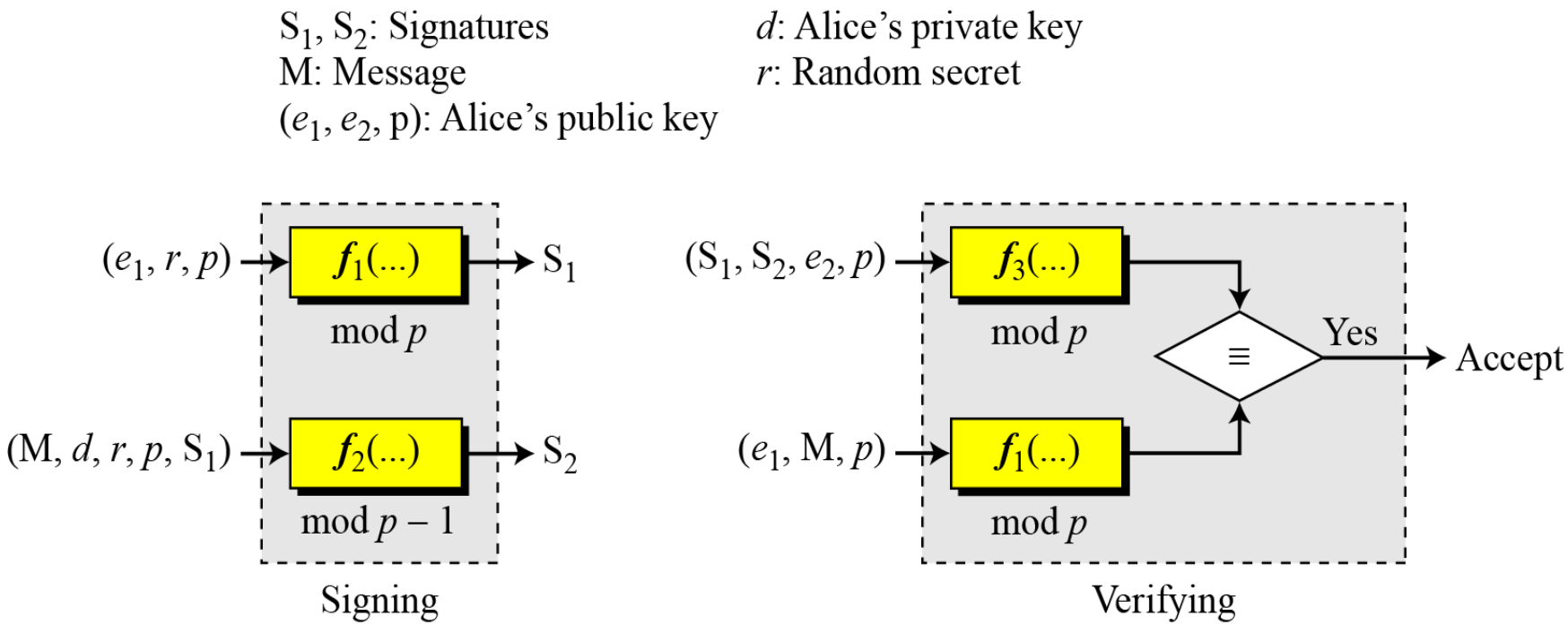
Continued

Signing and with Message digest



Continued

ElGamal signature Scheme

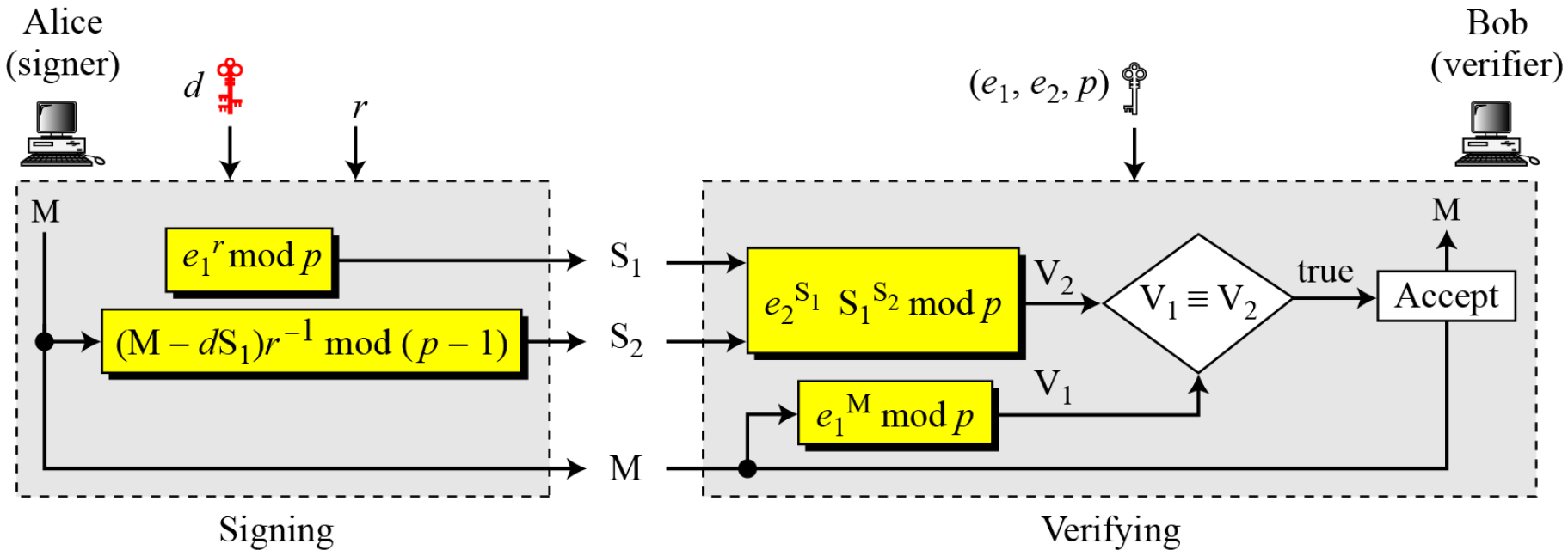


Continued

ElGamal signature Scheme

M: Message
S_1, S_2: Signatures
V_1, V_2: Verifications

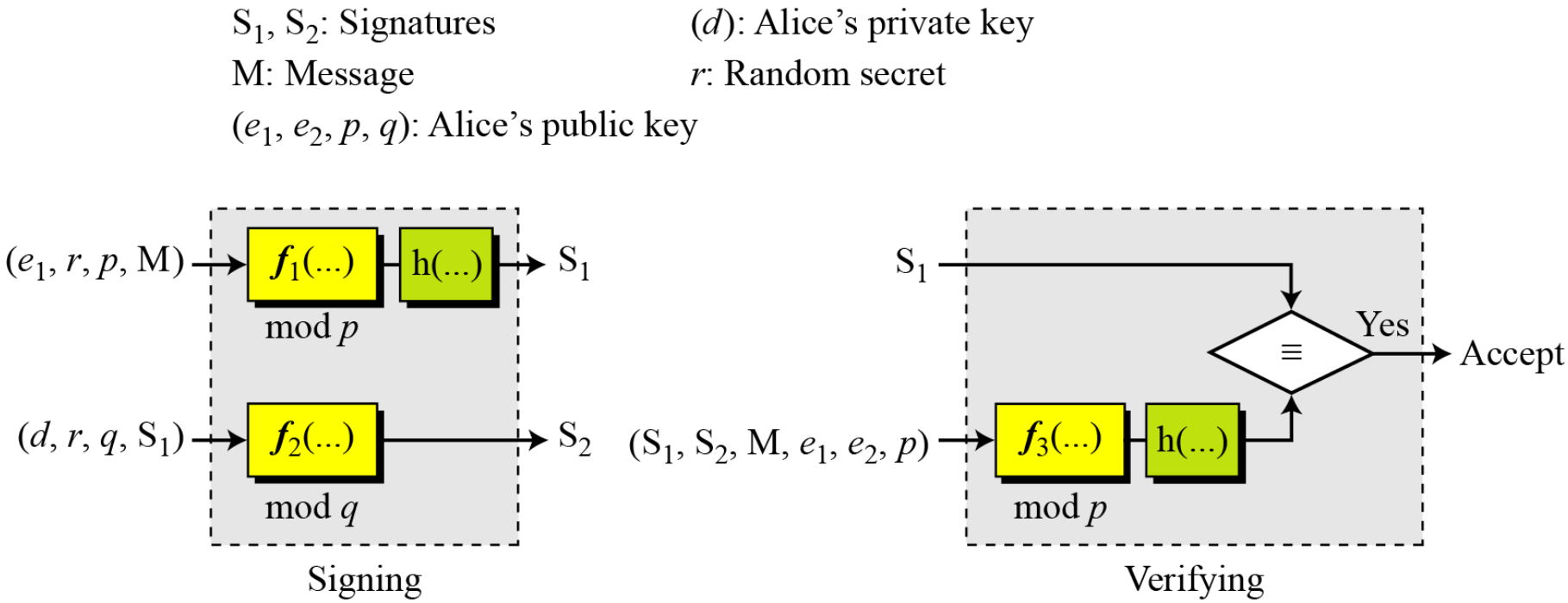
r: Random secret
d: Alice's private key
(e_1, e_2, p): Alice's public key



Schnorr Digital Signature Scheme

Figure *General idea behind the Schnorr digital signature scheme.*

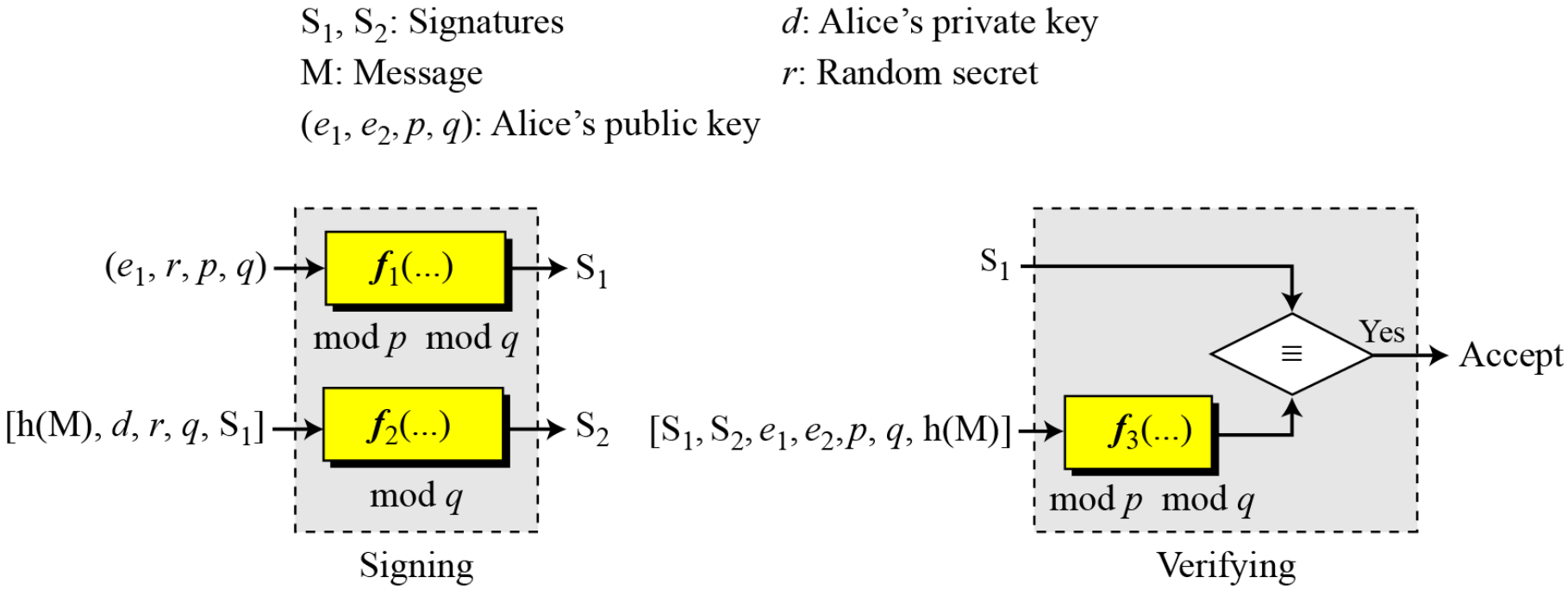
Based on ElGamal: reduce signature size



Digital Signature Standard (DSS) (NIST 1994)

Figure *General idea behind DSS scheme*

Based on ElGamal and Schnorr : reduce signature size



Continued

DSS Versus RSA

Computation of DSS signatures is faster than computation of RSA signatures when using the same p.

DSS Versus ElGamal

DSS signatures are smaller than ElGamal signatures because q is smaller than p.

Summary

- Public-key cryptosystems
- Applications for public-key cryptosystems
- Requirements for public-key cryptography
- Public-key cryptanalysis
- The RSA algorithm
  - Description of the algorithm
  - Computational aspects
  - Security of RSA