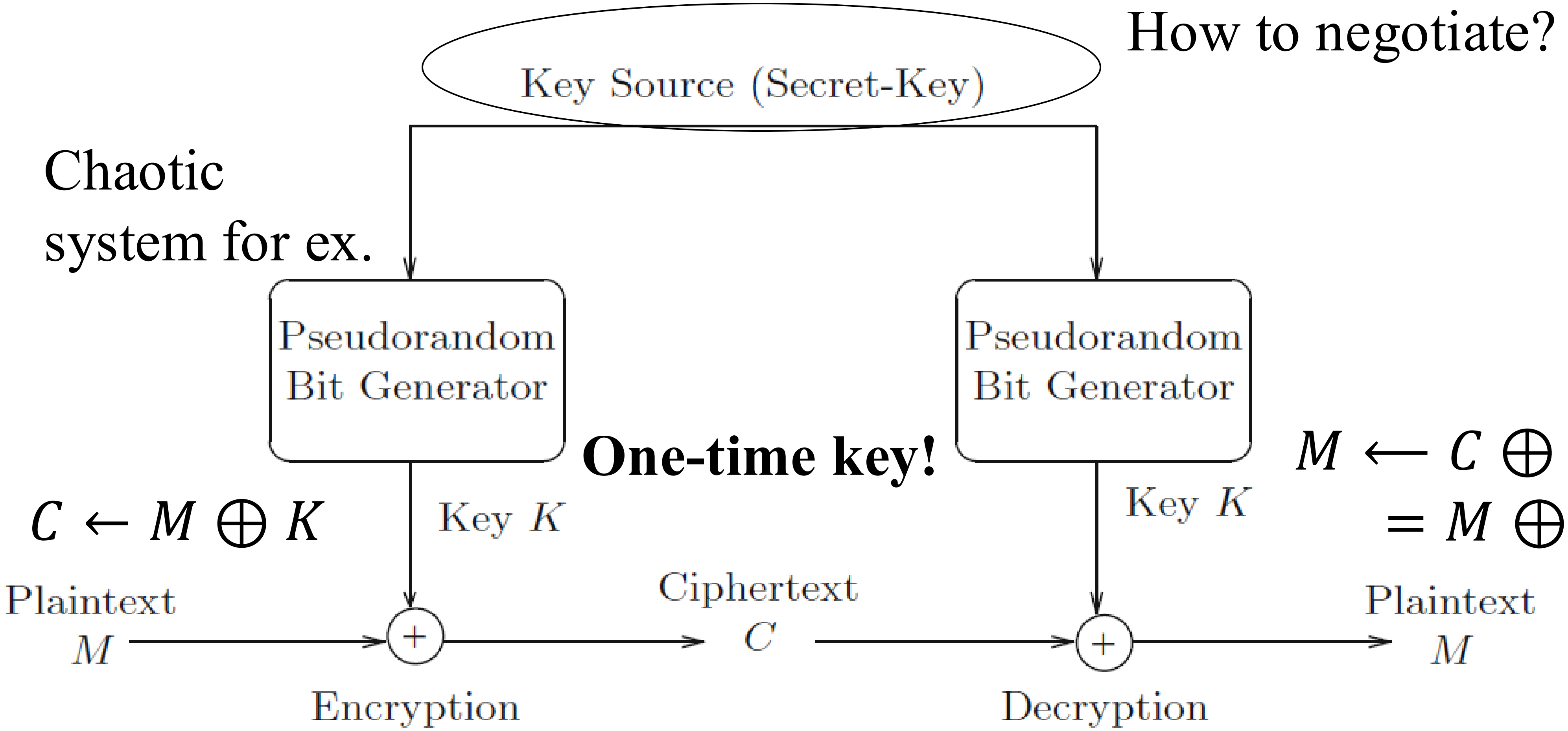
NT219- Cryptography

Week 6: Modern Asymmetric Ciphers

PhD. Ngoc-Tu Nguyen

tunn@uit.edu.vn

Stream Cipher Review



How to negotiate?

Chaotic system for ex.

One-time key!

$C \leftarrow M \oplus K$

$M \longleftarrow C \oplus K$
$= M \oplus K \oplus K$

Attacks (chosen plaintext attacks): $Known\,(M, C)$: $C \oplus M = M \oplus K \oplus M = K$

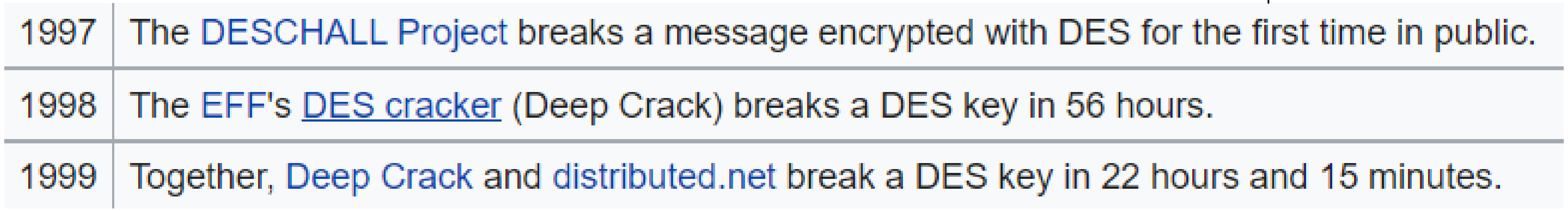
DES review

DES: 64-bits block cipher

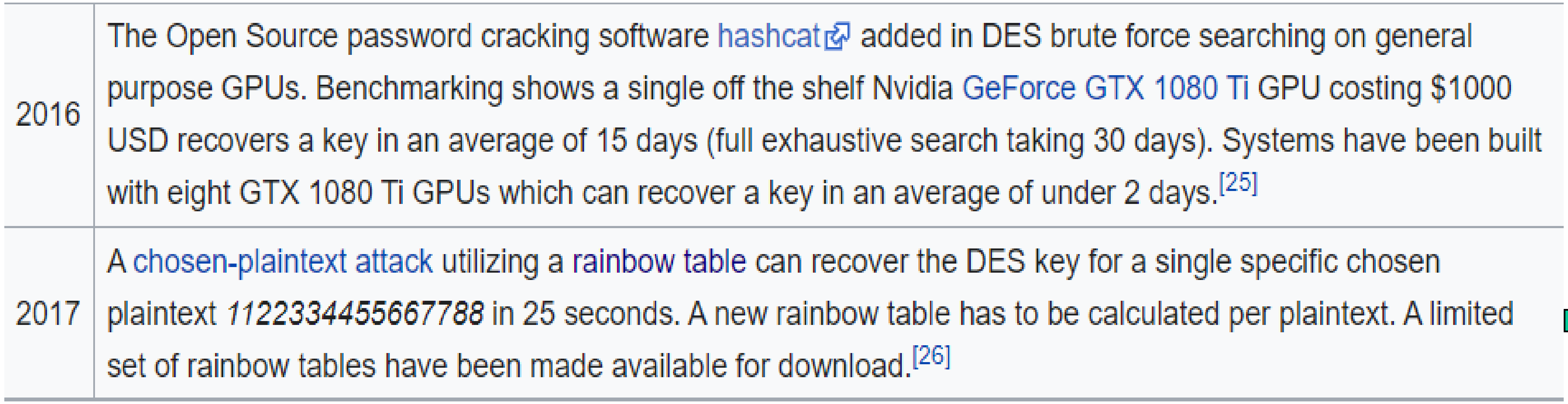
Key spaces: $\{0,1\}^{56} = 2^{56}$ possible keys

Brute Force attacks

Unsecure!



1997	The DESCHALL Project breaks a message encrypted with DES for the first time in public.
1998	The EFF's DES cracker (Deep Crack) breaks a DES key in 56 hours.
1999	Together, Deep Crack and distributed.net break a DES key in 22 hours and 15 minutes.



2016	The Open Source password cracking software hashcat added in DES brute force searching on general purpose GPUs. Benchmarking shows a single off the shelf Nvidia GeForce GTX 1080 Ti GPU costing $1000 USD recovers a key in an average of 15 days (full exhaustive search taking 30 days). Systems have been built with eight GTX 1080 Ti GPUs which can recover a key in an average of under 2 days.[25]
2017	A chosen-plaintext attack utilizing a rainbow table can recover the DES key for a single specific chosen plaintext *1122334455667788* in 25 seconds. A new rainbow table has to be calculated per plaintext. A limited set of rainbow tables have been made available for download.[26]

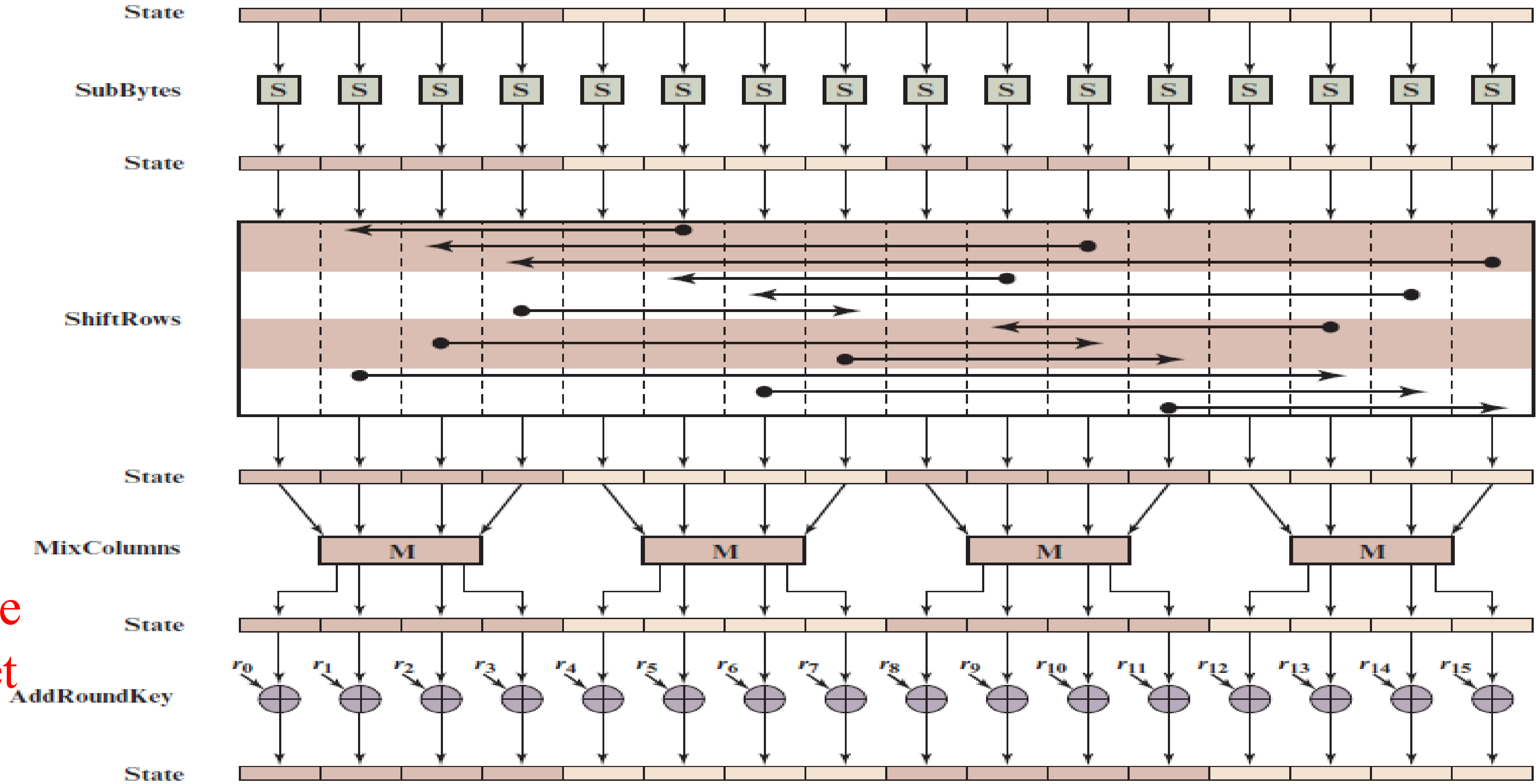
Choosen plaintext attacks!

https://en.wikipedia.org/wiki/Data_Encryption_Standard

AES review

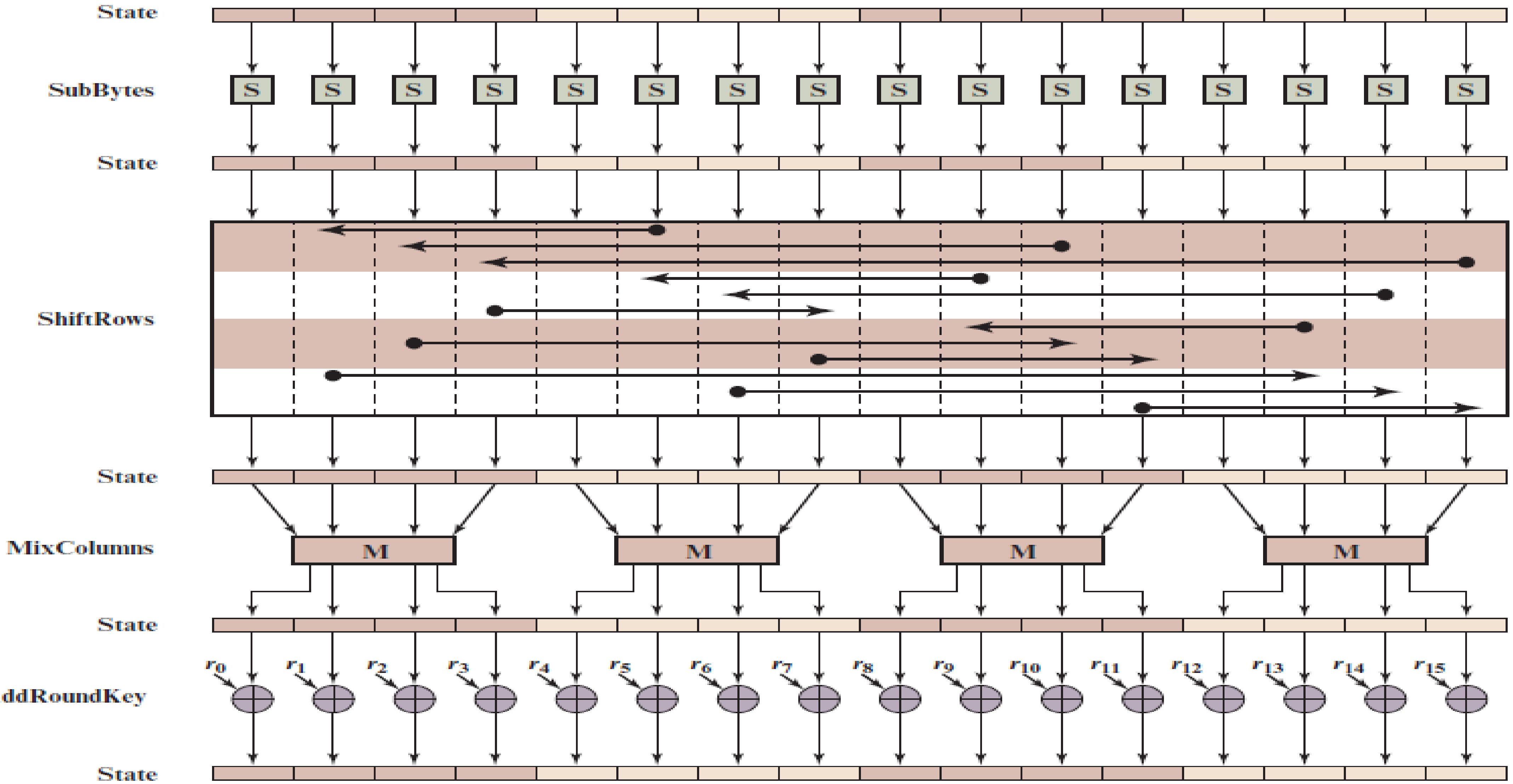
- **substitute-bytes** (sub)
  - Non-linear operation based on a defined **substitution box**
  - Used to resist cryptanalysis and other mathematical attacks
- **shift-rows** (shr)
  - Linear operation for producing **diffusion**
- **mix-columns** (mic)
  - Elementary operation also for producing **diffusion**
- **add-round-key** (ark) (128, 192, 256 bits)
  - Simple set of $\oplus$ operations on state matrices
  - Linear operation
  - Produces **confusion**

Dynamic AES?


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r_0 r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} r_{15}
State

Negotiate the secret key?

AES review


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r_0 r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} r_{15}
State

Negotiate the secret key?

Outline

- Why asymmetric cryptography?
- Factoring Based Cryptography (P1)
  - RSA
  - *Rabin*
- Logarithm Based Cryptography (P2)
- Elliptic Curve Cryptography (P3)
- Some advanced cryptography system (quantum resistance)

Why Public-Key Cryptosystems?

To overcome two of the most difficult problems associated with symmetric encryption:

- **Key distribution (key for sysmetric encryption)**
  - 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: proposed a method that addressed both problems (1976)

Moden Asymmetric ciphers

Symmetric cipher vs Asymmetric cipher

Hommophic, Searchable encryption,..

$ID_A(atributes)$ $ID_B(atributes)$

$\updownarrow$ $\swarrow\!\!\!\nearrow$

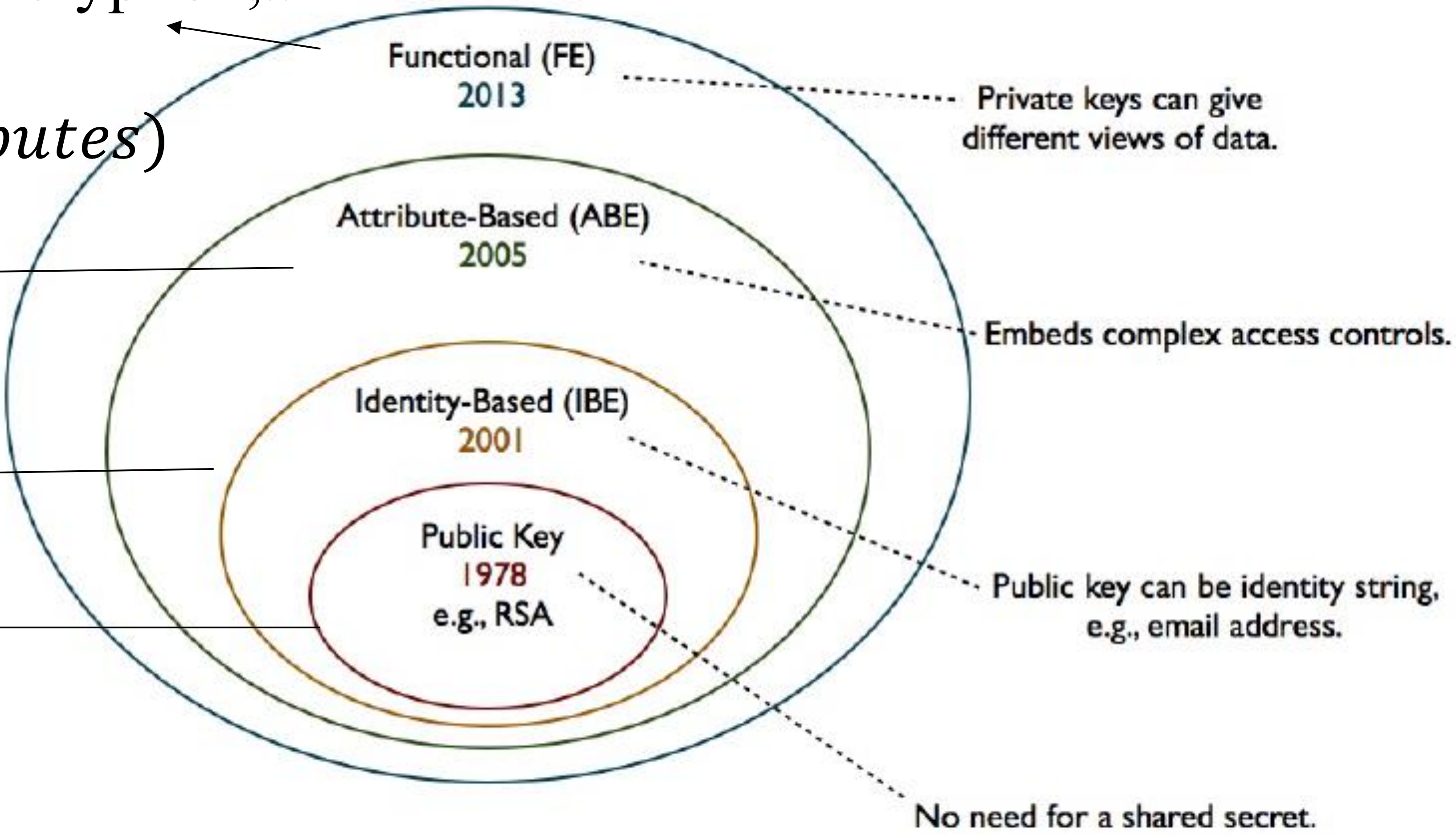
$(PK_A, \{SK_A, SK_B, ..\})$

ID

$\updownarrow$

(PK_A, SK_A)

(PK_A, SK_A)

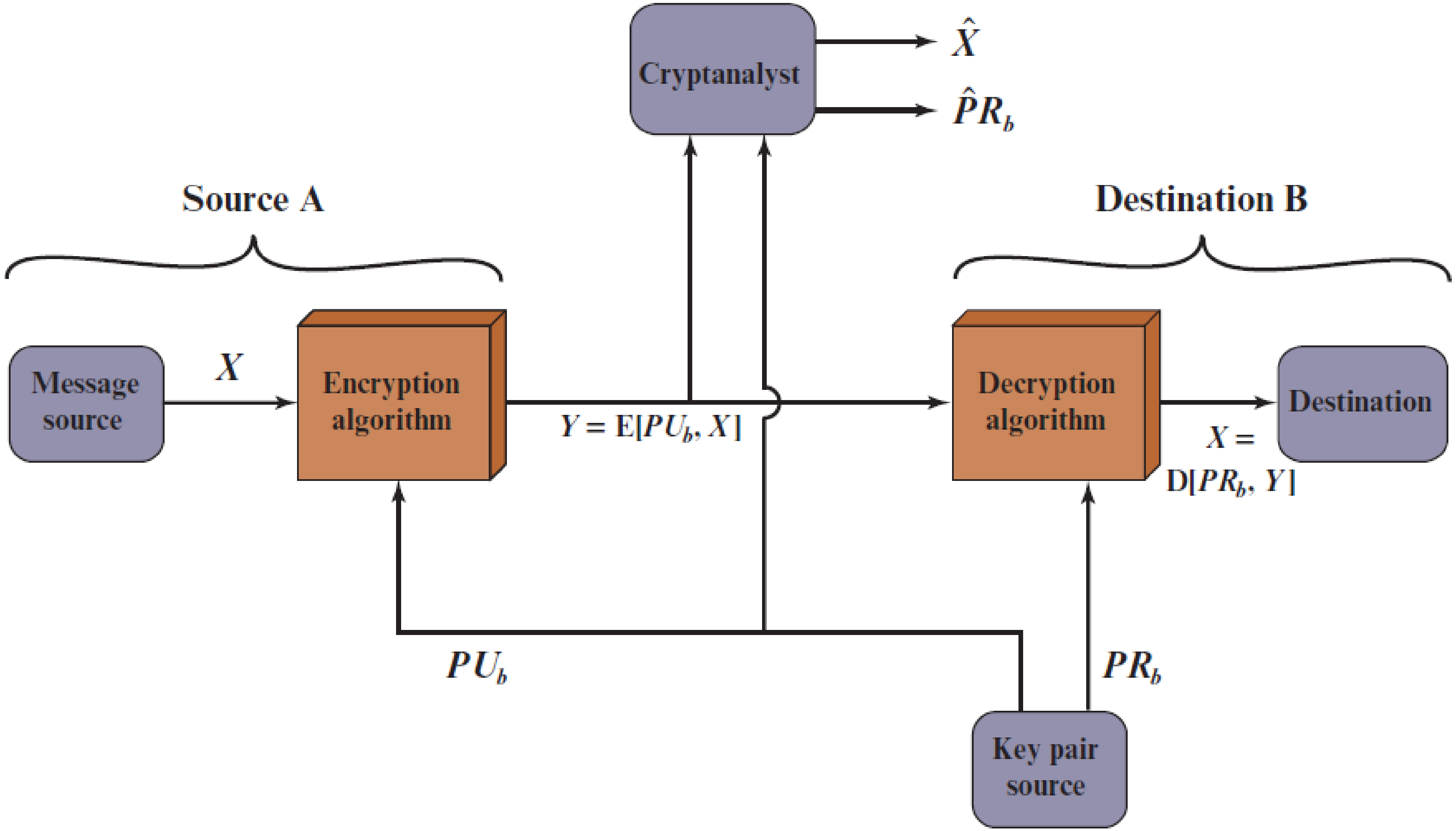

Functional (FE)
2013
Private keys can give different views of data.
Attribute-Based (ABE)
2005
Embeds complex access controls.
Identity-Based (IBE)
2001
Public key can be identity string, e.g., email address.
Public Key
1978
e.g., RSA
No need for a shared secret.

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

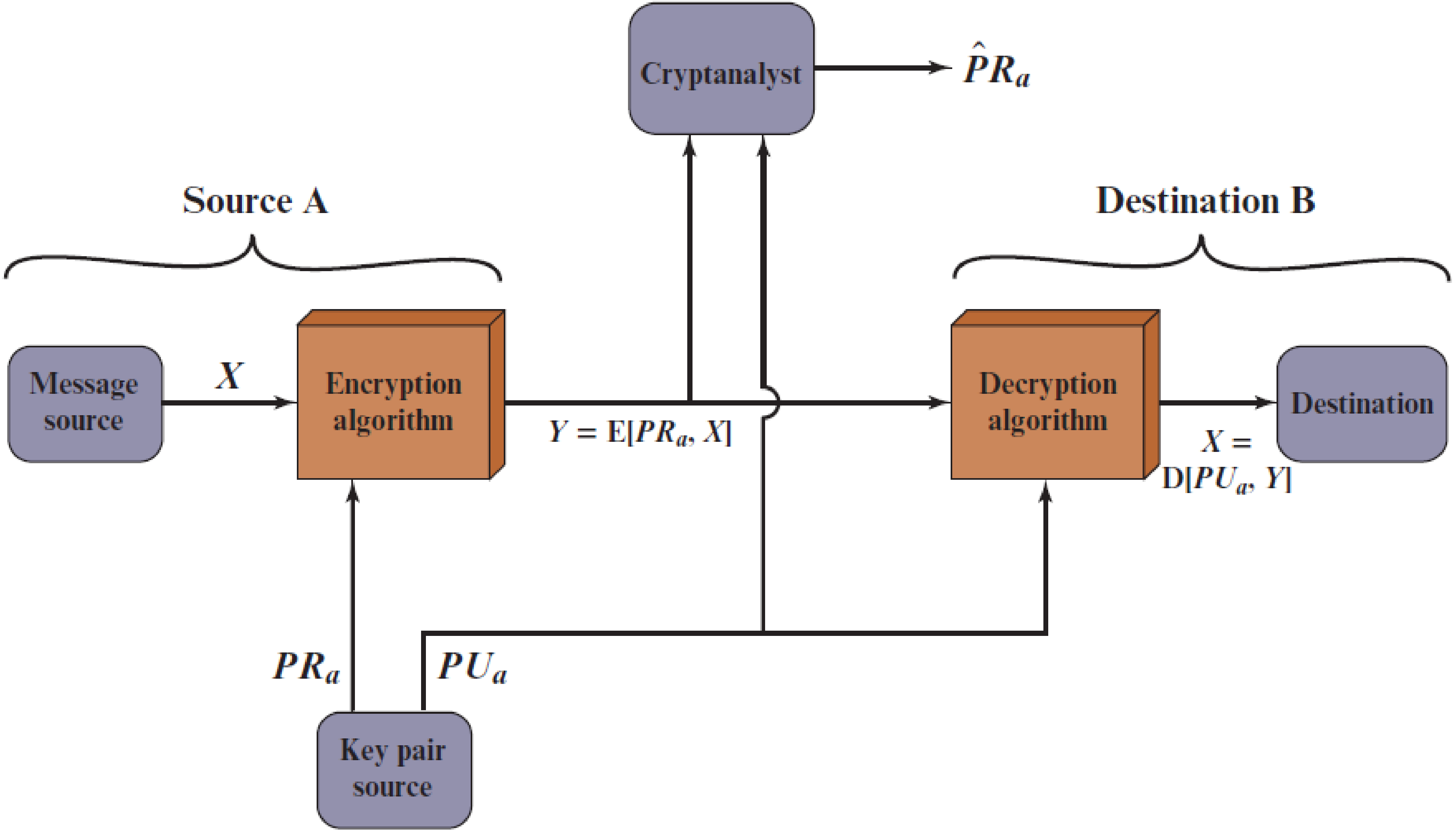
Public-Key Cryptosystem: Confidentiality

Public-Key Cryptosystem: Confidentiality



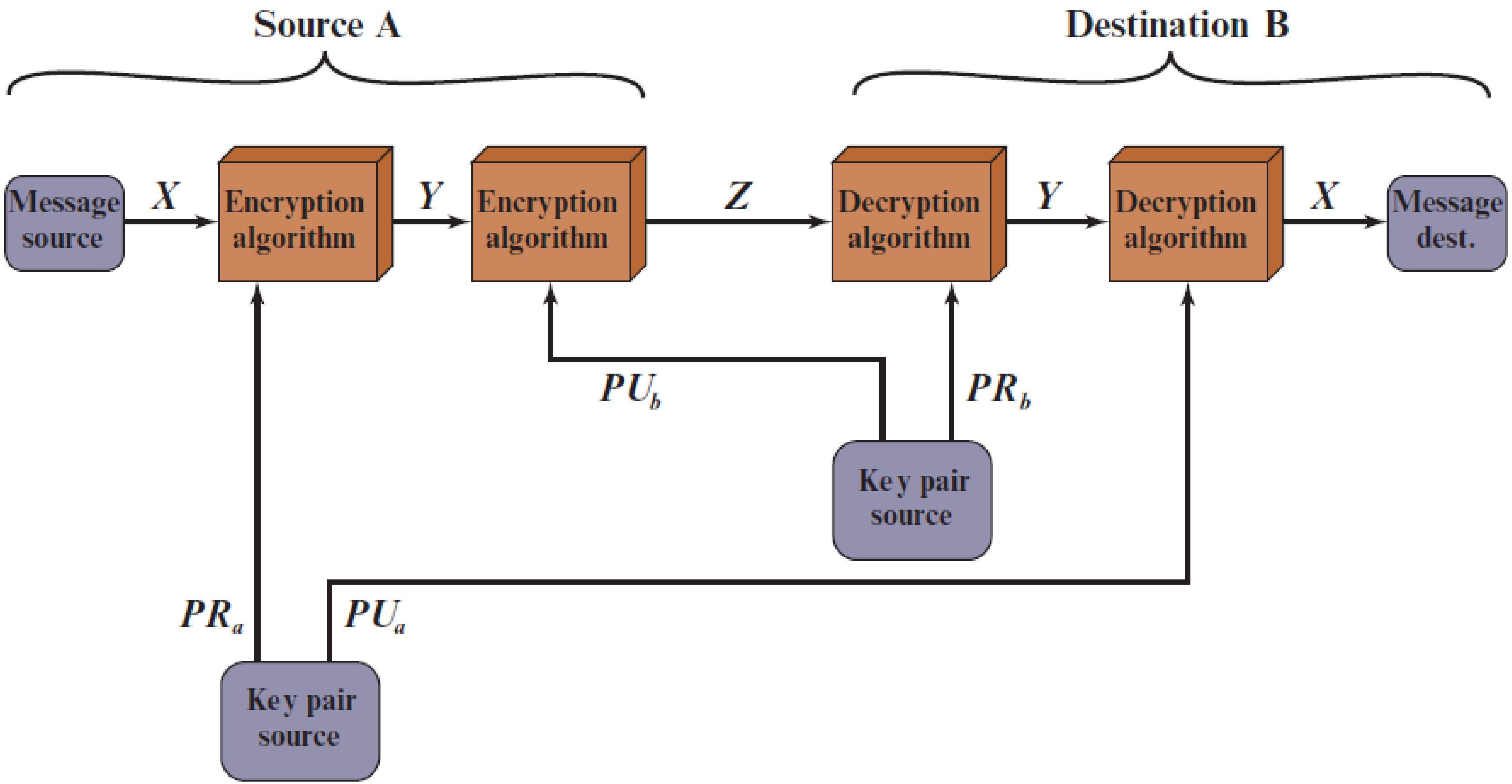
Public-Key Cryptosystem: Authentication

Public-Key Cryptosystem: Authentication



Public-Key Cryptosystem: Authentication and Secrecy

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

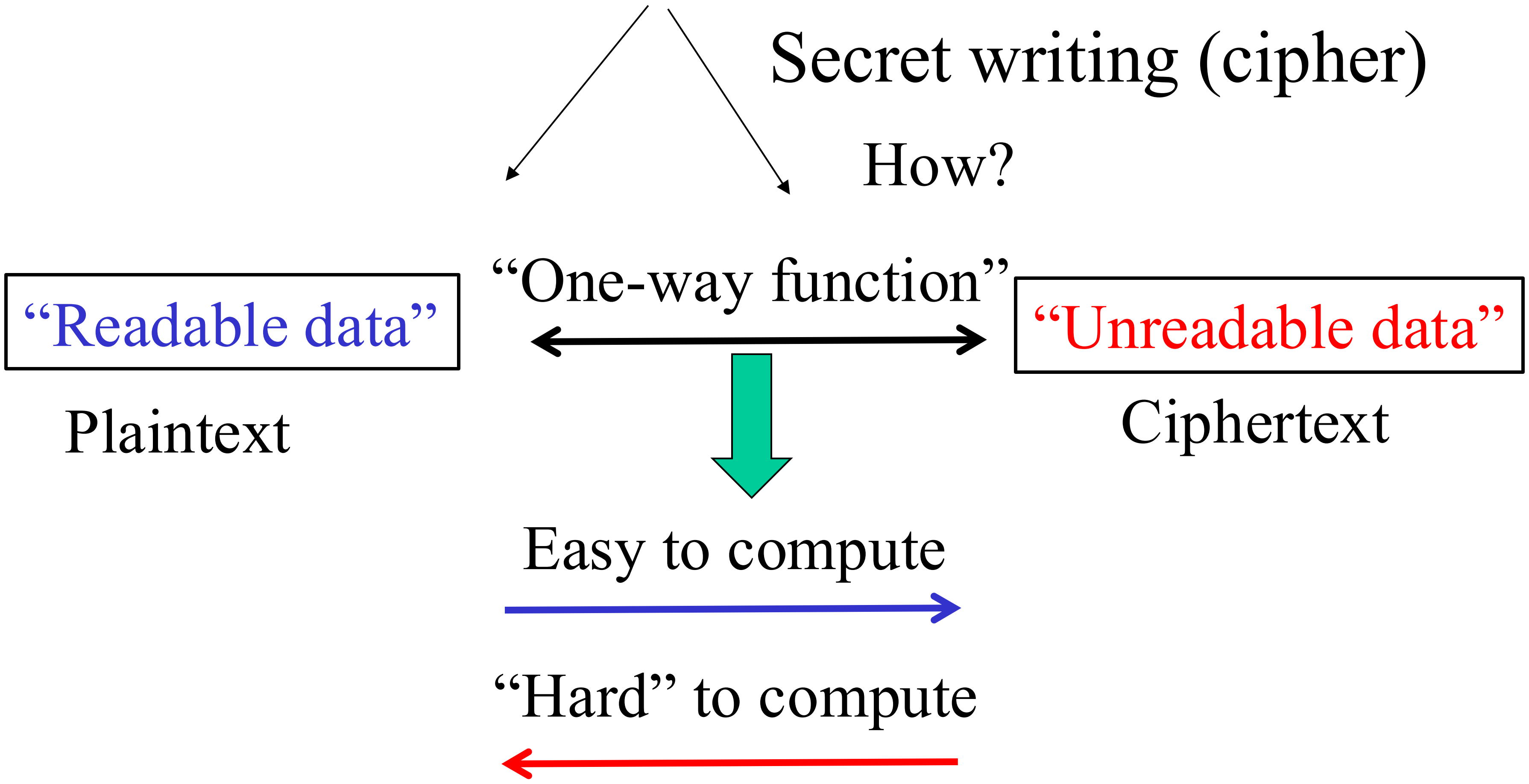
Some algorithms are suitable for all three applications, whereas others can be used only for one or two

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

Public-Key Requirements

- Cryptology= Cryptography + Cryptanalysis



Secret writing (cipher)

How?

“One-way function”

“Readable data”

Plaintext

“Unreadable data”

Ciphertext

Easy to compute

“Hard” to compute

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)

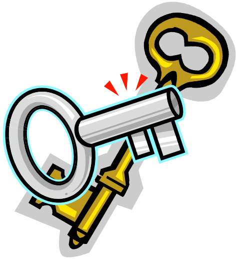
- **Need a trap-door one-way function**
  - A one-way function is one that maps a domain into a range such that every function value has a unique inverse, with the condition that the calculation of the function is easy, whereas the calculation of the inverse is infeasible
    - $Y = f(X)$ easy
    - $X = f^{-1}(Y)$ 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**

Outline

- Why asymmetric cryptography?
- Factoring Based Cryptography (P1)
  - RSA
  - *Rabin*
- Logarithm Based Cryptography (P2)
- Elliptic Curve Cryptography (P3)
- Some advanced cryptography system (quantum resistance)

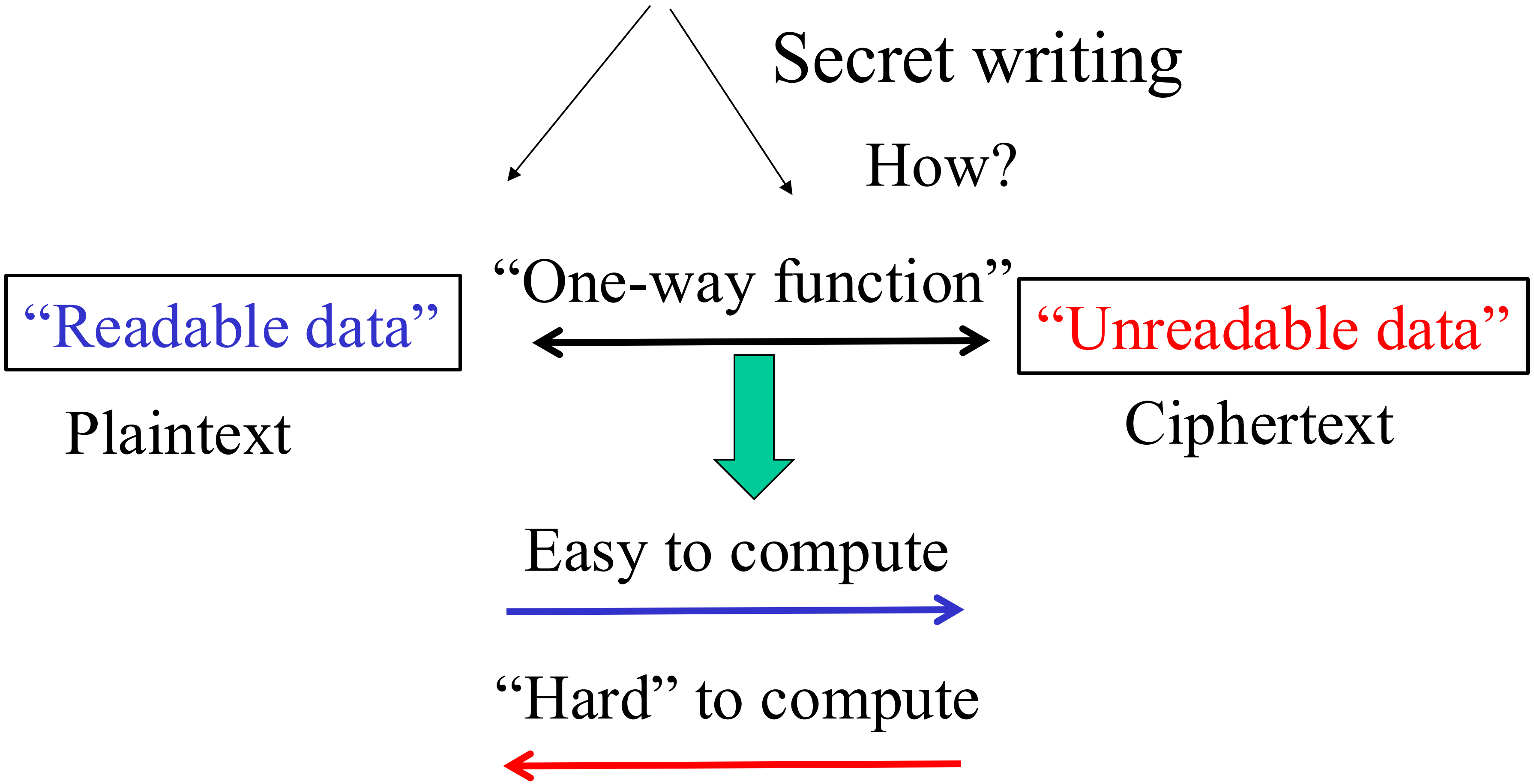
Rivest-Shamir-Adleman (RSA) Algorithm

- 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 3072 bits



Cryptograph review

- Cryptology= Cryptography + Cryptanalysis



Secret writing

How?

“One-way function”

“Readable data”

Plaintext

“Unreadable data”

Ciphertext

Easy to compute

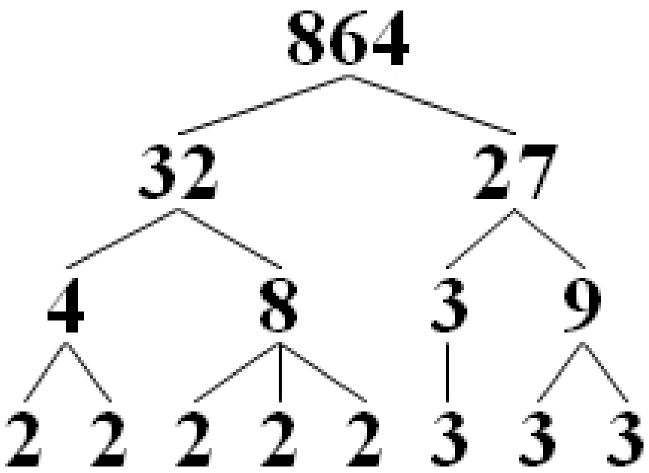
“Hard” to compute

Prime factorization problem

Factorize number

$N = 864$

$= 2^5 \times 3^3$



Input: n-bits composite number N

Output: $N = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_k^{\alpha_k}, \alpha_k \in$ ¥*

No classical algorithm has been published that can factor all integers in polynomial time.

https://en.wikipedia.org/wiki/Integer_factorization

Prime factorization problem

“Prime factorization one-way function!”

Input: large prime number p, q and a large number e

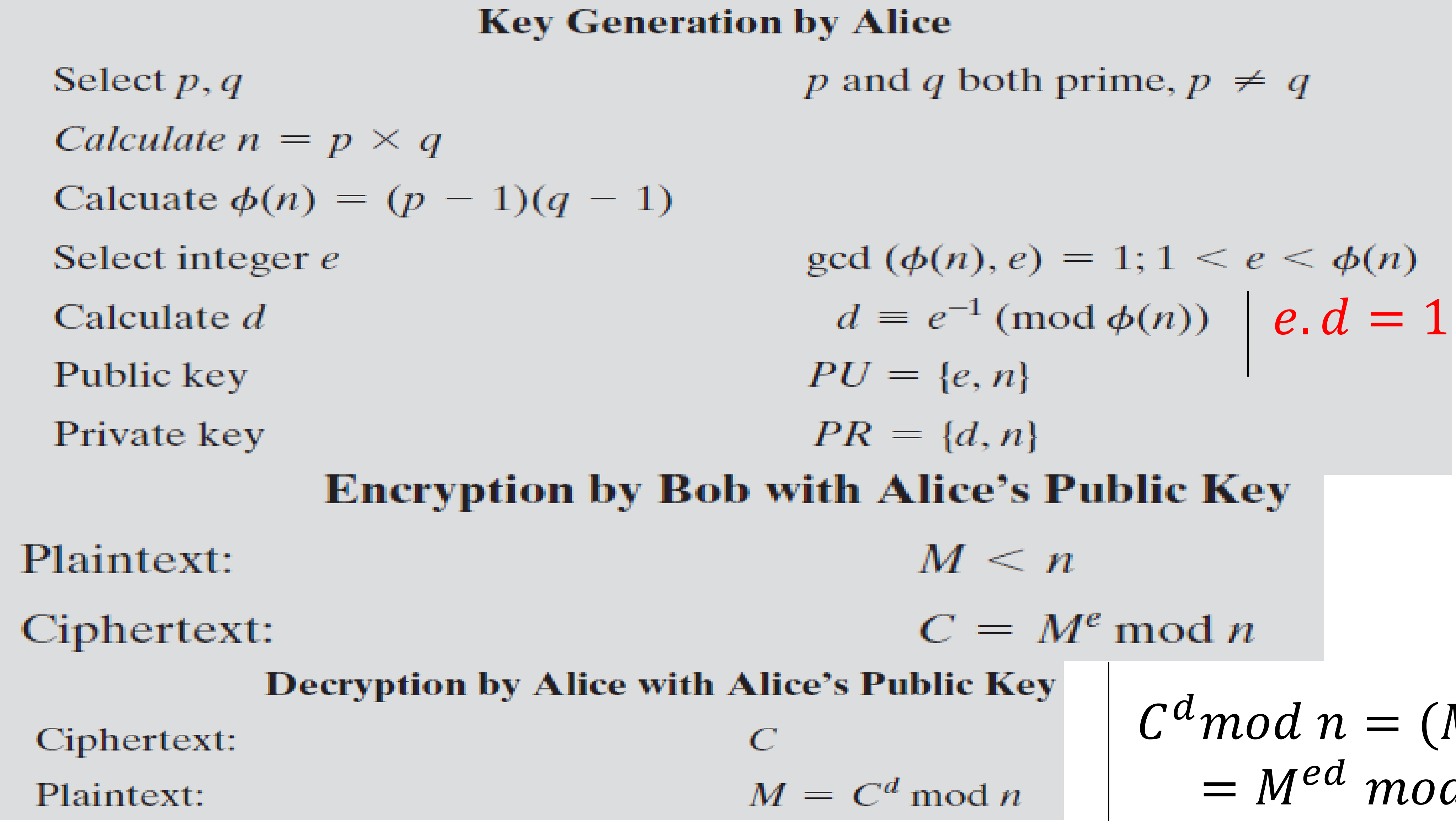
$$\xrightarrow{\text{Easy to compute}} \quad \begin{cases} n = p.q \\ d = e^{-1} \bmod (p-1)(q-1) \\ C = M^{\mathrm{e}} \bmod n \end{cases}$$

Input: n, e, C

$$\begin{cases} n = p.q \leftarrow p, q \\ d = e^{-1} \bmod (p-1)(q-1) \end{cases} \quad \xleftarrow{\text{“Hard” to compute}}$$

$$\boxed{C^{d} \bmod n = \mathrm{M}^{e.d} \bmod \mathrm{n} = \mathrm{M}^{e.d \bmod (p-1)(q-1)} \bmod \mathrm{n} = \mathrm{M}}$$

The RSA Algorithm



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:	$M = C^d \bmod n$

$$e.d = 1\ mod\ \phi(n)$$

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

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

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

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

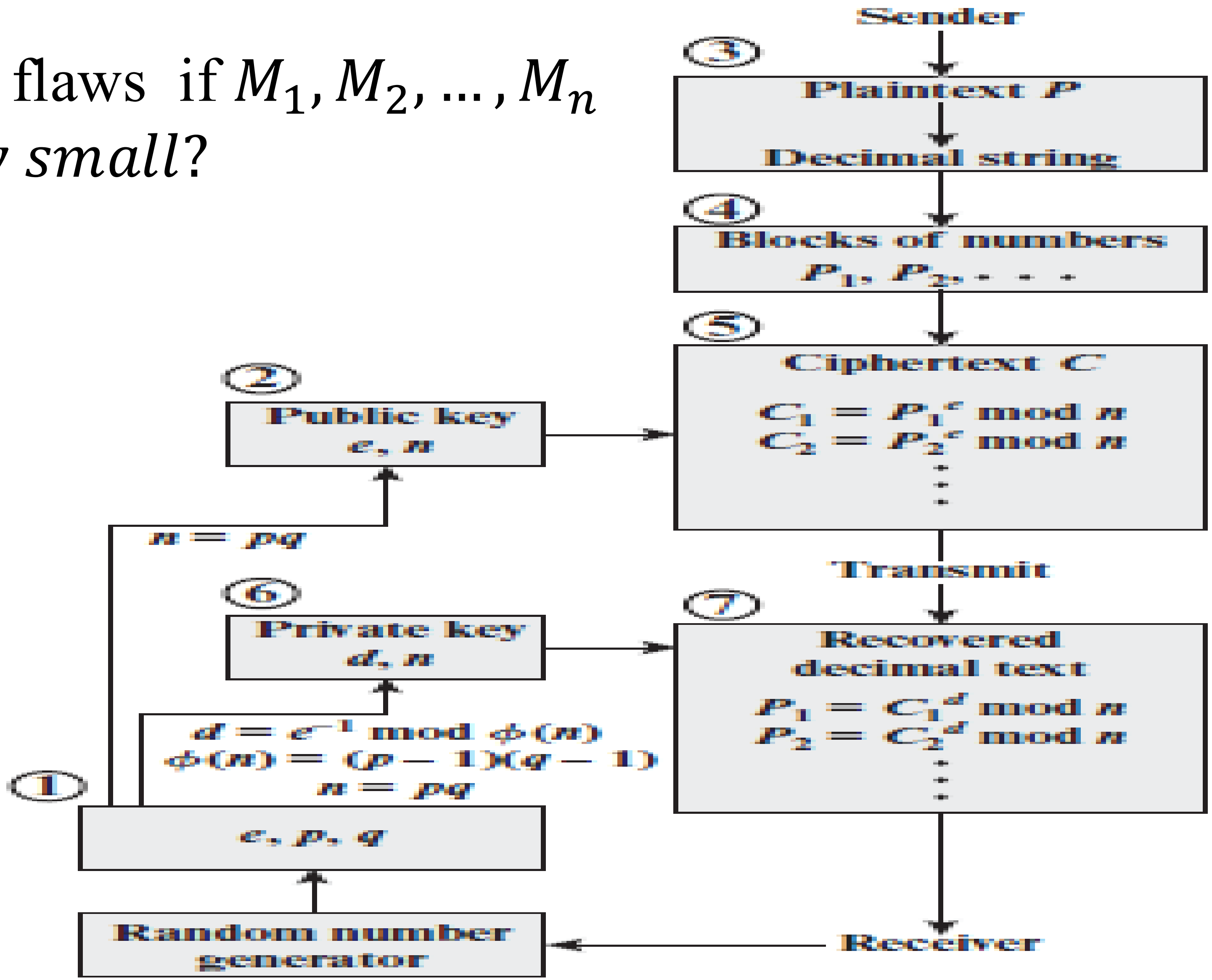
- Both sender and receiver must know the value of *n*
- The sender knows the value of *e,* and only the receiver knows the value of *d*
- This is a public-key encryption algorithm with a public key of *PU={e,n}* and a private key of *PR={d,n}*

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*

RSA Processing of Multiple Blocks

What are flaws if $M_1, M_2, \ldots, M_n$ $are\ very\ small$?



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

Algorithm for Computing a^b mod n

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

$$a^b = a^{(b_k b_{k-1} \dots b_0)}$$

$$= a^{(2^k b_k + \dots + 2^2.b_2 + 2.b_1 + b_0)}$$

$$= \prod_{i=0}^{k} a^{b_i.2^i} = \prod_{i=0}^{k} (\underbrace{a^{b_i} . a^{.2^i}})$$

$c = 2^i$

$$f_i = a^{.2^i}$$

$$f_{i+1} = a^{.2^{i+1}} = a^{2.2^i}$$

$$= (a^{2^i})^2 = (f_i)^2$$

$$f_i = a^c$$

$c = 2^i + 1$

$$f_{i+1} = (f_i)^2 . a$$

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

Algorithm for Computing a^b mod n

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

$$7^{560} = 7^{(1000110000)_2} = 7^{2^{10}}.7^{2^5}.7^{2^4}$$

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 (C R T) 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 $M = C^d$ mod 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

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

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

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

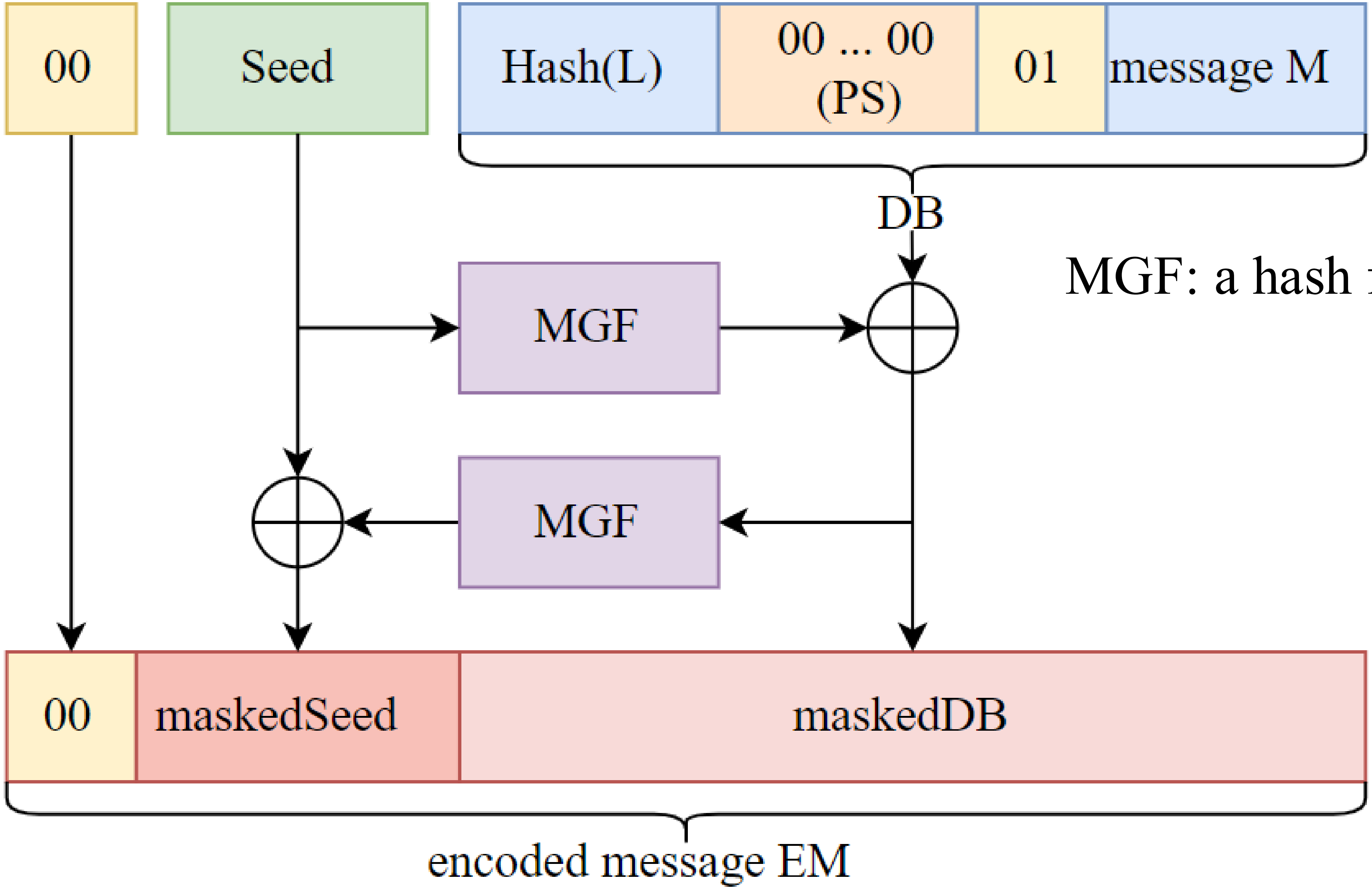
Fault-Based Attack

- An attack on a processor that is generating R S A 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 R S A
  - 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 (C C A)

- 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)

Encryption Using Optimal Asymmetric Encryption Padding (OAEP)



MGF: a hash function

Misconceptions about 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

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.

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 R S A algorithm
- Understand the timing attack
- Summarize the relevant issues related to the complexity of algorithms