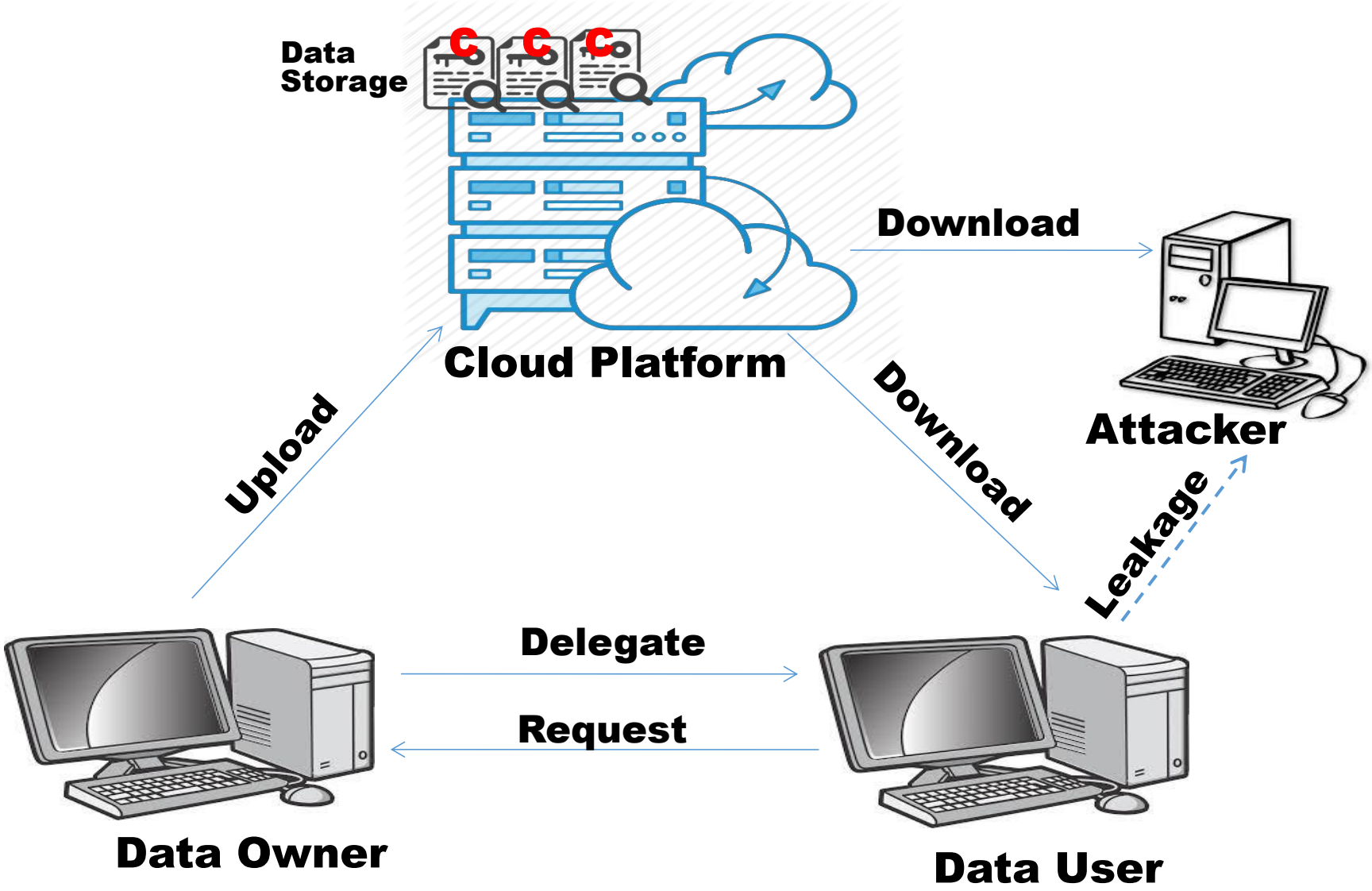
UNIT III
Securing Cloud Communication and API

Dr Mukti Padhya

Assistant Professor

NFSU, Gandhinagar

Cloud Services



Encryption

- Why Encryption
  - To achieve Confidentiality
  - Protection against unauthorized disclosure of information.

- What is Encryption?
- Types

Encryption

- can characterize Encryption system by:
  - ❖ type of encryption operations used
    - ➢ Substitution (each element of PT is mapped into another element)
    - ➢ Transposition (elements in PT are rearranged)
    - ➢ Product (combination of both)
  - ❖ number of keys used
    - ➢ single-key or private or symmetric
    - ➢ two-key or public or asymmetric
  - ❖ way in which plaintext is processed
    - ➢ block
    - ➢ stream

Some Basic Terminology

plaintext - original message

ciphertext - coded message

cipher - algorithm for transforming plaintext to ciphertext

key - info used in cipher known only to sender/receiver

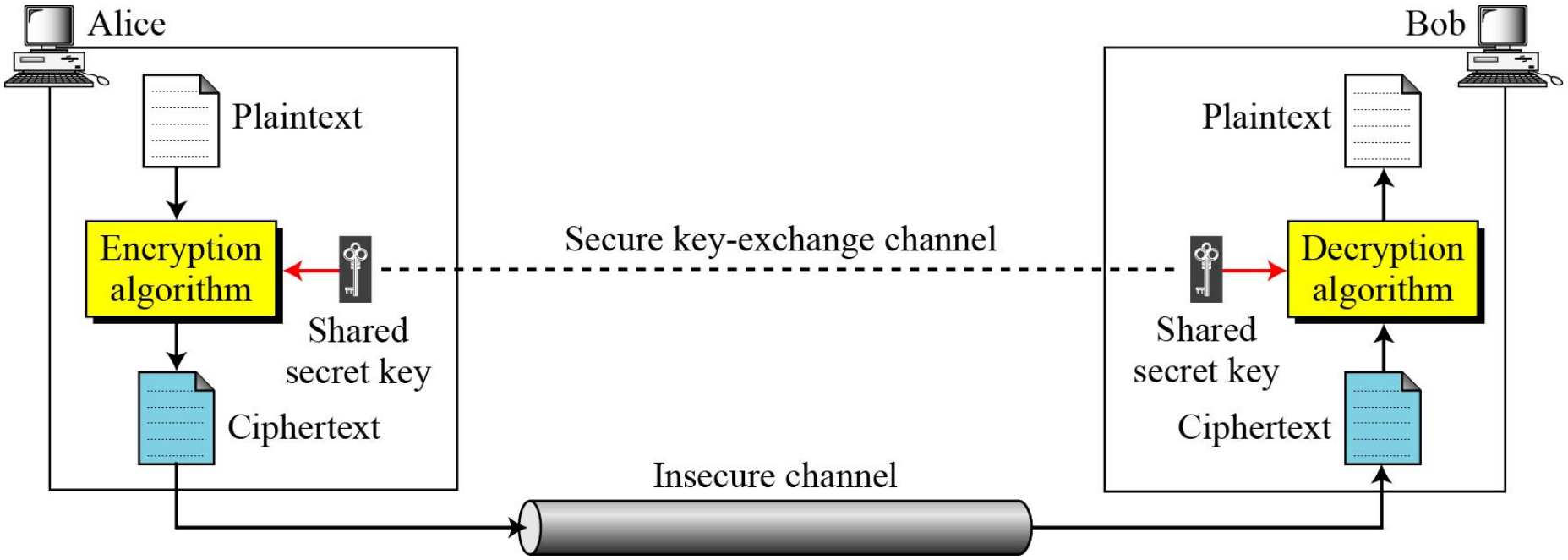
encipher (encrypt) - converting plaintext to ciphertext

decipher (decrypt) - recovering ciphertext from plaintext

cryptography - study of encryption principles/methods

cryptanalysis (codebreaking) - study of principles/ methods of deciphering ciphertext *without* knowing key

Symmetric-key cipher



Symmetric-key cipher

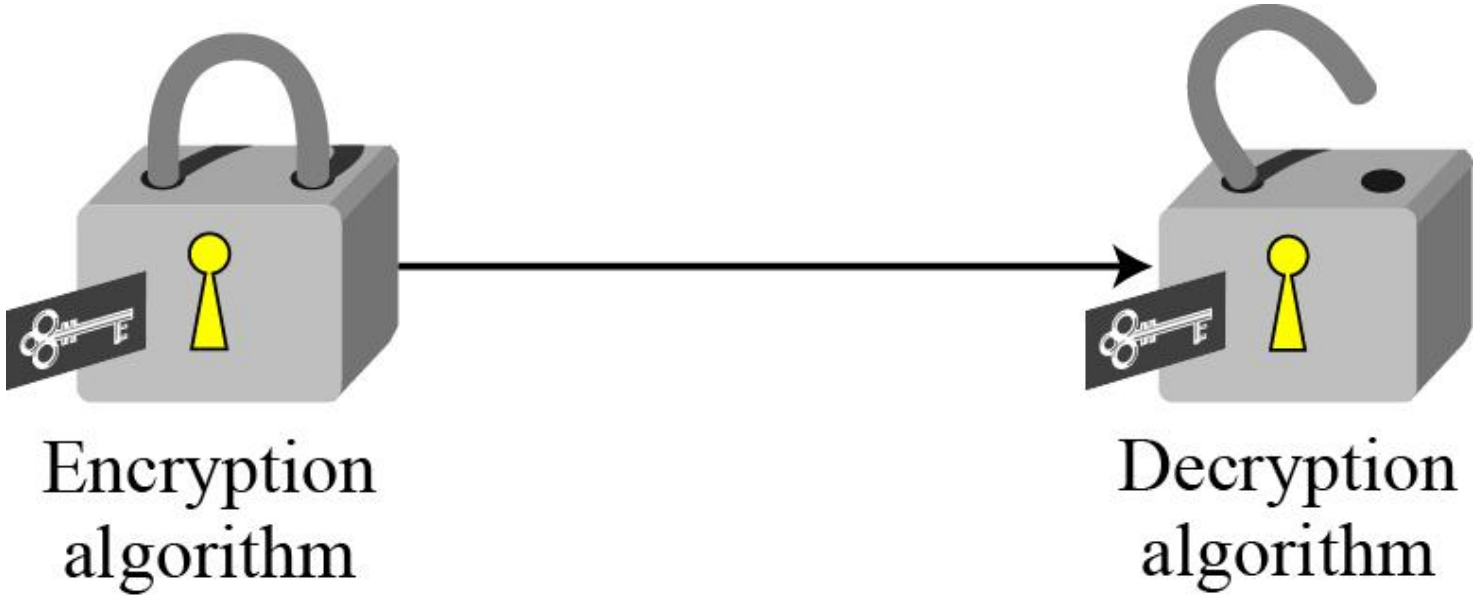
If P is the plaintext, C is the ciphertext, and K is the key,

Encryption: $C = E_k(P)$ Decryption: $P = D_k(C)$

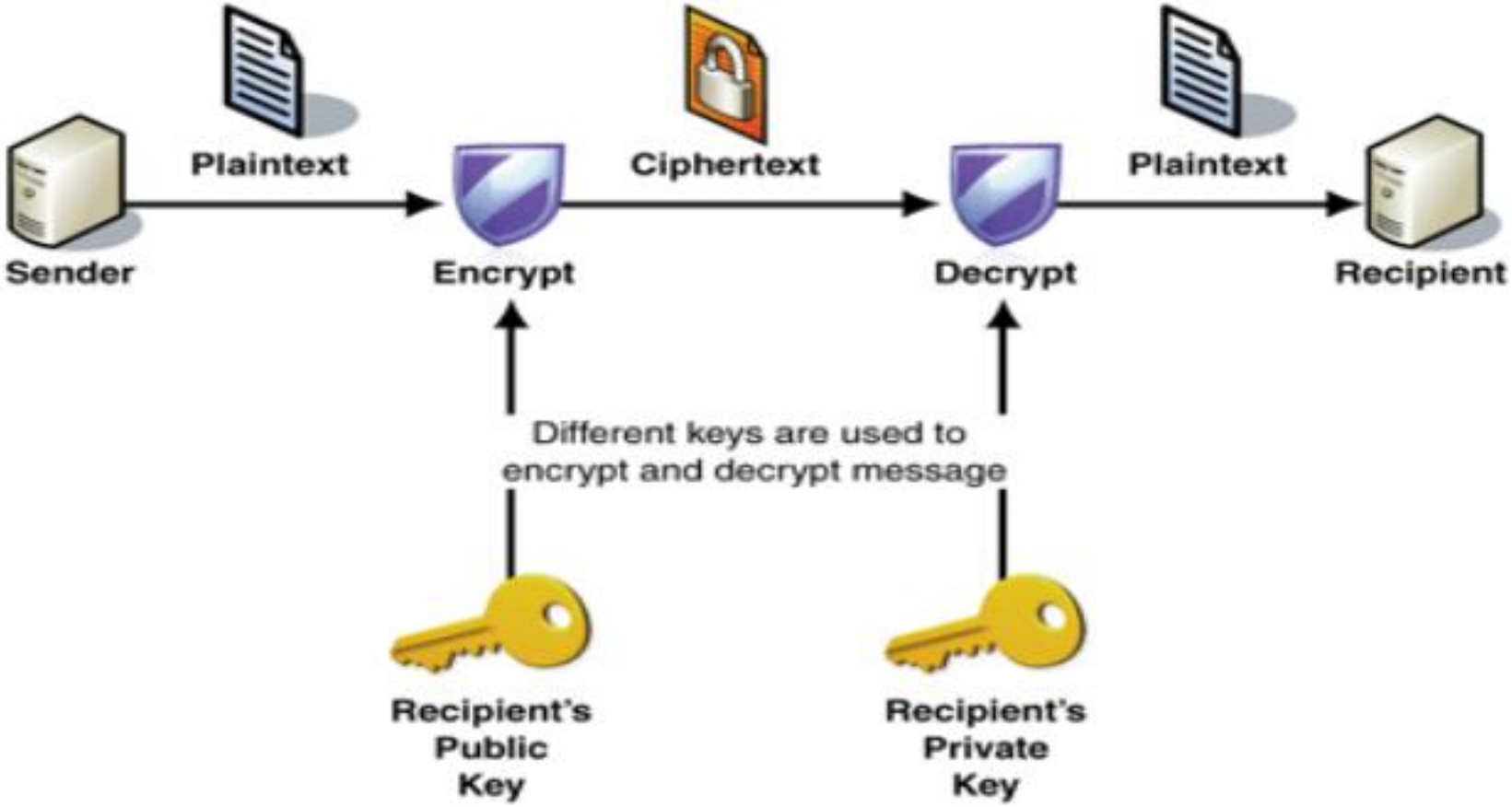
In which, $D_k(E_k(x)) = E_k(D_k(x)) = x$

Symmetric-key cipher

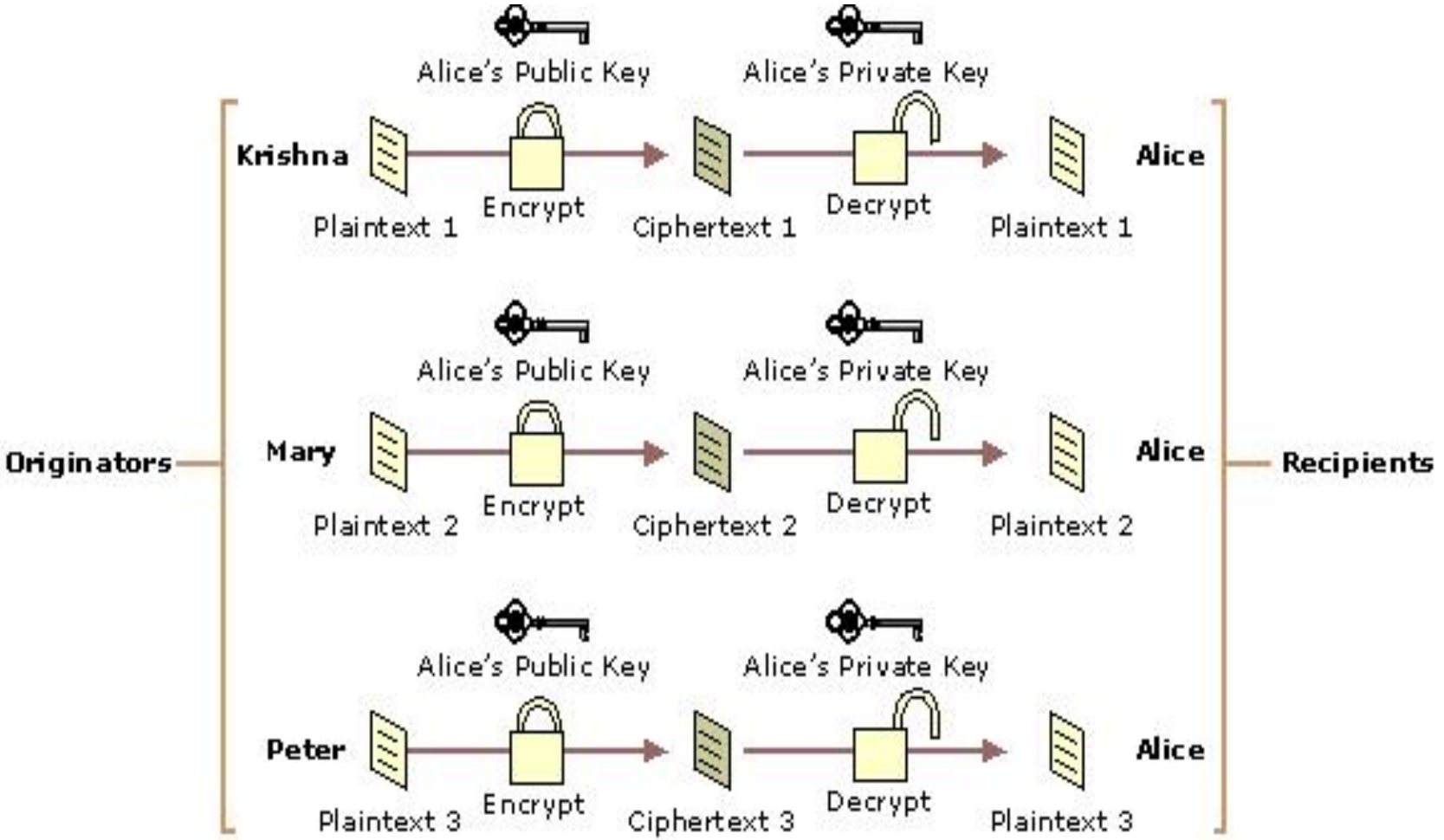
Locking and unlocking with the same key



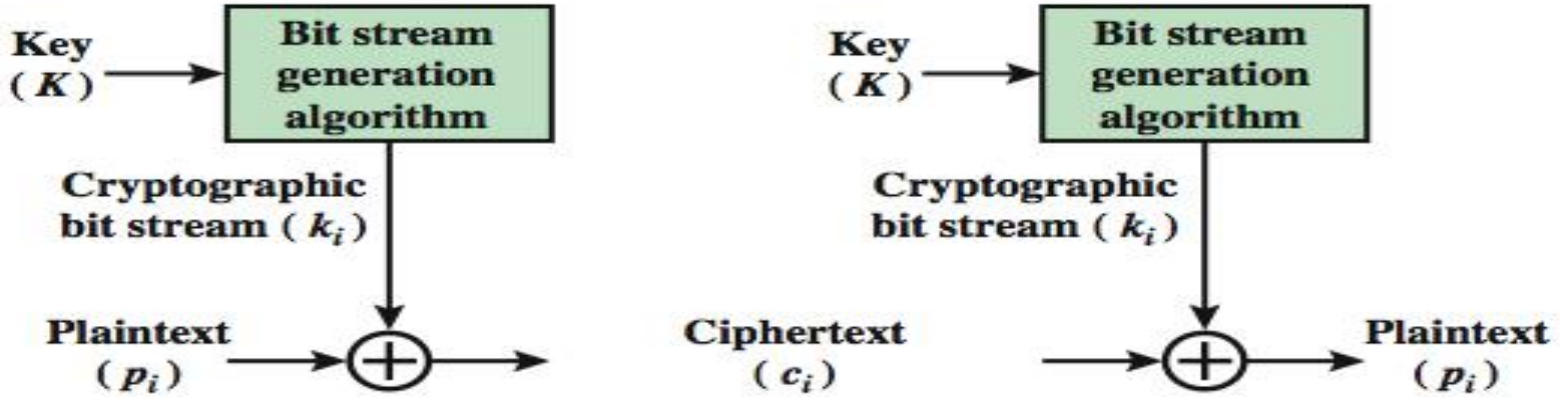
Asymmetric-key Cipher



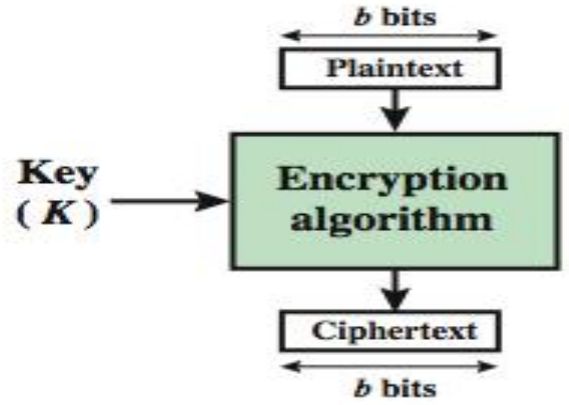
Asymmetric-key Cipher



Block Vs Stream Ciphers



(a) Stream Cipher Using Algorithmic Bit Stream Generator



(b) Block Cipher

Block vs Stream Ciphers

- block ciphers process messages in blocks, each of which is then en/decrypted
  - like a substitution on very big characters
  - 64-bits or more
- stream ciphers process messages a bit or byte at a time when en/decrypting
- many current ciphers are block ciphers
  - broader range of applications

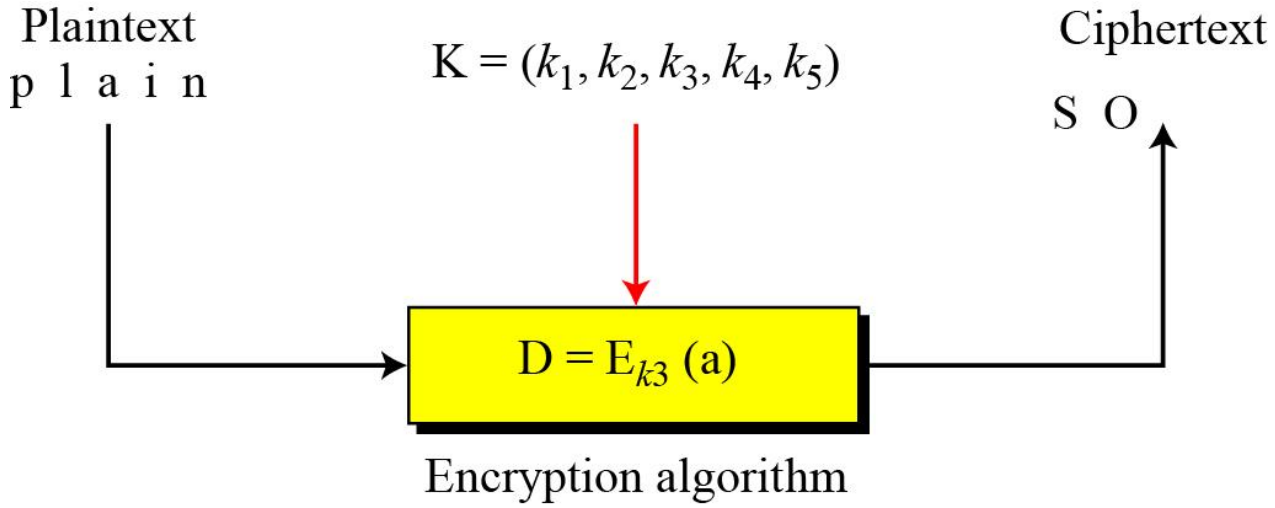
Stream Ciphers

Call the plaintext stream P, the ciphertext stream C, and the key stream K.

$$P = P_1P_2P_3, \ldots \qquad C = C_1C_2C_3, \ldots \qquad K = (k_1, k_2, k_3, \ldots)$$

$$C_1 = E_{k1}(P_1) \qquad C_2 = E_{k2}(P_2) \qquad C_3 = E_{k3}(P_3) \ldots$$

Stream cipher



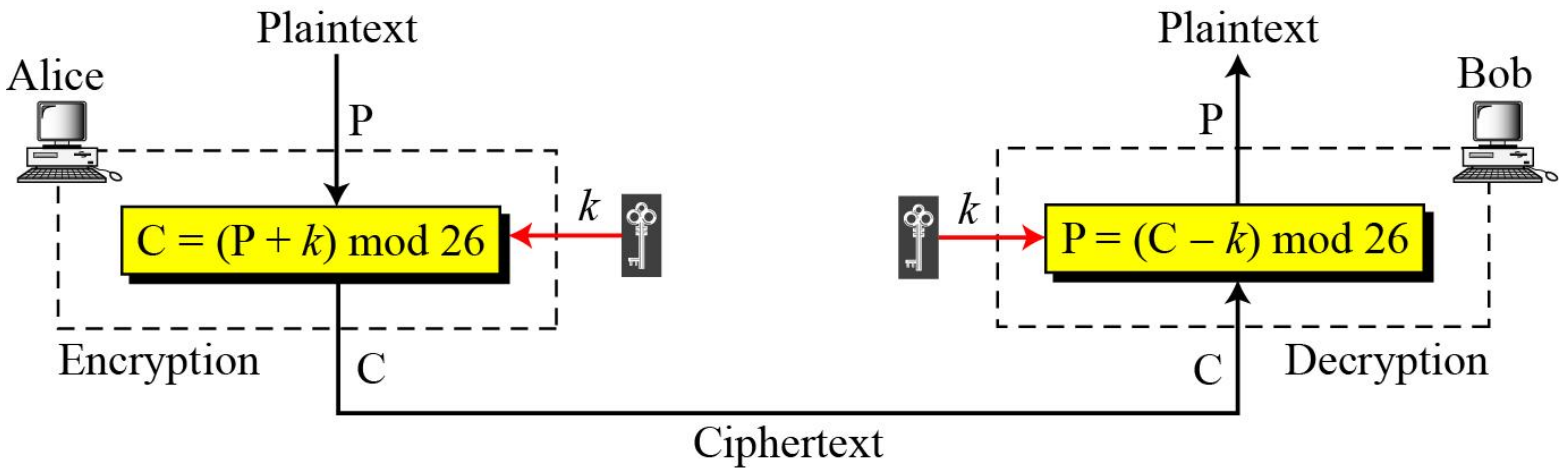
- Additive ciphers can be categorized as stream ciphers in which the key stream is the repeated value of the key. In other words, the key stream is considered as a predetermined stream of keys or K = (k, k, …, k).

- The monoalphabetic substitution ciphers are also stream ciphers.

- However, each value of the key stream in this case is the mapping of the current plaintext character to the corresponding ciphertext character in the mapping table.

Continued

Additive cipher



The simplest monoalphabetic cipher is the additive cipher. This cipher is sometimes called a shift cipher and sometimes a Caesar cipher, but the term additive cipher better reveals its mathematical nature.

When the cipher is additive, the plaintext, ciphertext, and key are integers in Z_{26}.

Continued

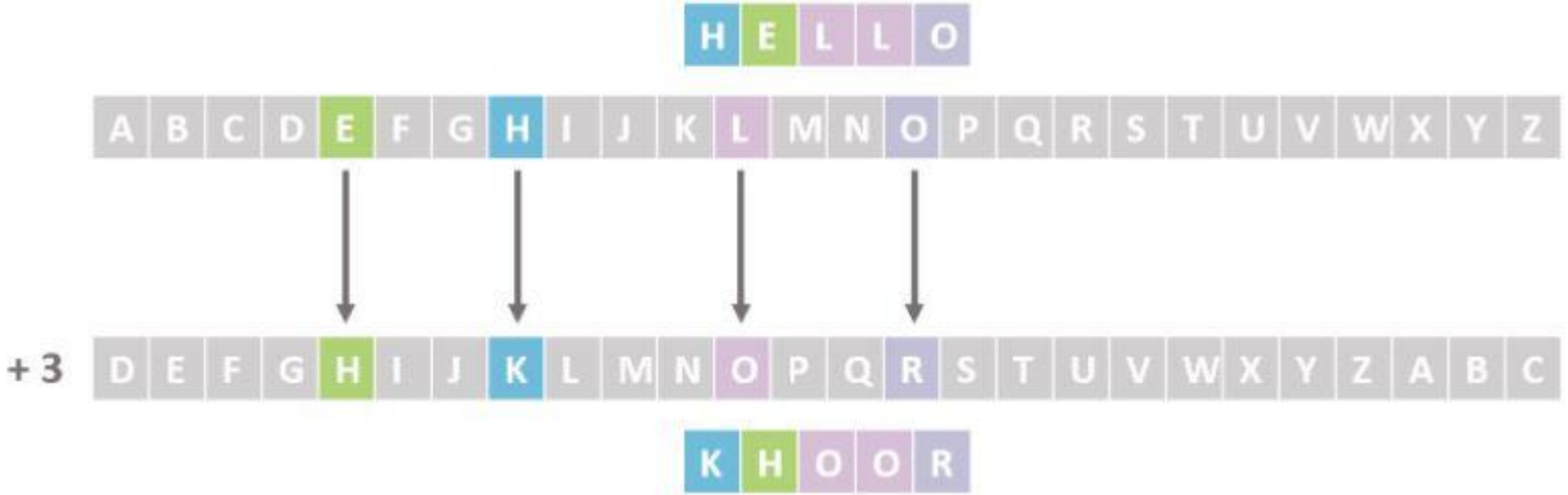
Example

Use the additive cipher with key = 15 to encrypt the message “hello”.

Solution

We apply the encryption algorithm to the plaintext, character by character:

Plaintext: h → 07	Encryption: (07 + 15) mod 26	Ciphertext: 22 → W
Plaintext: e → 04	Encryption: (04 + 15) mod 26	Ciphertext: 19 → T
Plaintext: l → 11	Encryption: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: l → 11	Encryption: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: o → 14	Encryption: (14 + 15) mod 26	Ciphertext: 03 → D


H E L L O
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
+3
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
K H O O R

- mathematically give each letter a number

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

- then have Caesar cipher as:

$$c = E(p) = (p + k) \bmod (26)$$

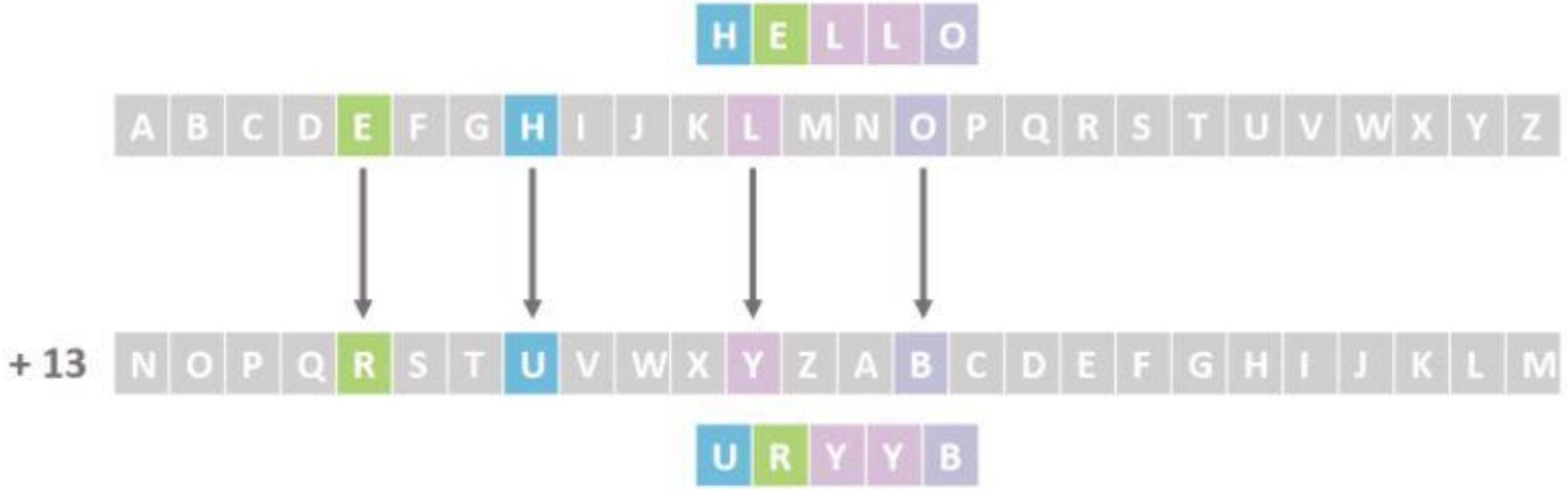
$$p = D(c) = (c - k) \bmod (26)$$

Plaintext : HELLO Key =13 Ciphertext : ?

Note: mod operation → x mod n then if x > n and n can divide x so output of mod is ZERO

If x < n then output will be x itself

If x is negative then output will be x+n


H E L L O
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
+ 13
N O P Q R S T U V W X Y Z A B C D E F G H I J K L M
U R Y Y B

Continued

Monoalphabetic Substitution Cipher

- Because additive, multiplicative, and affine ciphers have small key domains, they are very vulnerable to brute-force attack.
- A better solution is to create a mapping between each plaintext character and the corresponding ciphertext character. Alice and Bob can agree on a table showing the mapping for each character.

An example key for monoalphabetic substitution cipher

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	N	O	A	T	R	B	E	C	F	U	X	D	Q	G	Y	L	K	H	V	I	J	M	P	Z	S	W

Continued

Example
encrypt the message

this message is easy to encrypt but hard to find the key

The ciphertext is

ICFVQRVVNEFVRNVSIYRGAHSLIOJICNHTIYBFGTICRXRS

Vigenere ciphers are also stream ciphers according to the definition.

In this case, the key stream is a repetition of m values, where m is the size of the keyword.

In other words,

$$K = (k_1, k_2, \ldots k_m, k_1, k_2, \ldots k_m, \ldots)$$

Polyalphabetic Ciphers

In polyalphabetic substitution, each occurrence of a character may have a different substitute. The relationship between a character in the plaintext to a character in the ciphertext is one-to-many.

Autokey Cipher

$P = P_1P_2P_3 \ldots$ $C = C_1C_2C_3 \ldots$ $k = (k_1, P_1, P_2, \ldots)$

Encryption: $C_i = (P_i + k_i) \bmod 26$ Decryption: $P_i = (C_i - k_i) \bmod 26$

Continued

Vigenere Cipher

$P = P_1P_2P_3 \ldots$ $C = C_1C_2C_3 \ldots$ $K = [(k_1, k_2, \ldots, k_m), (k_1, k_2, \ldots, k_m), \ldots]$

Encryption: $C_i = P_i + k_i$ Decryption: $P_i = C_i - k_i$

Example

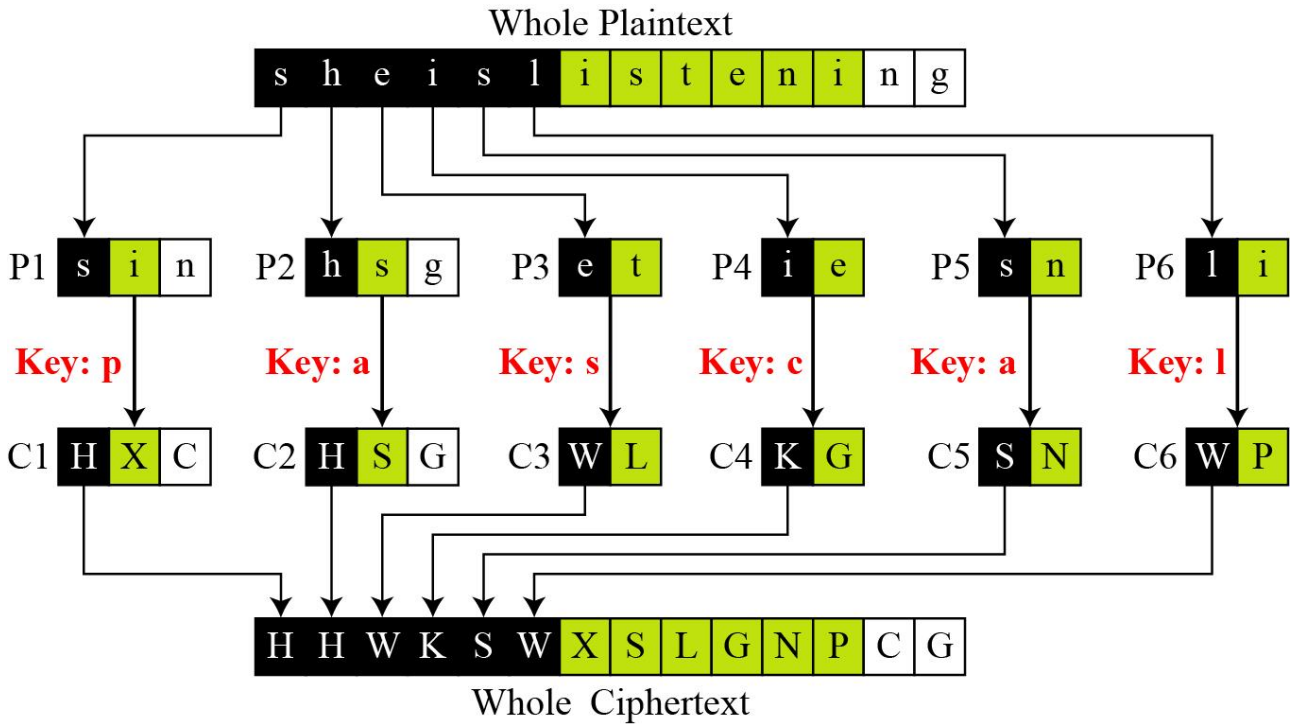
We can encrypt the message “She is listening” using the 6-character keyword “PASCAL”.

Continued

Example

Vigenere cipher can be seen as combinations of m additive ciphers.

A Vigenere cipher as a combination of m additive ciphers



Continued

Example

Let us see how we can encrypt the message “She is listening” using the 6-character keyword “PASCAL”. The initial key stream is (15, 0, 18, 2, 0, 11). The key stream is the repetition of this initial key stream (as many times as needed).

Plaintext:	s	h	e	i	s	l	i	s	t	e	n	i	n	g
P’s values:	18	07	04	08	18	11	08	18	19	04	13	08	13	06
Key stream:	***15***	***00***	***18***	***02***	***00***	***11***	***15***	***00***	***18***	***02***	***00***	***11***	***15***	***00***
C’s values:	**07**	**07**	**22**	**10**	**18**	**22**	**23**	**18**	**11**	**6**	**13**	**19**	**02**	**06**
Ciphertext:	**H**	**H**	**W**	**K**	**S**	**W**	**X**	**S**	**L**	**G**	**N**	**T**	**C**	**G**

Ciphertext : DATG

Key : 19

Decryption : ??

Ciphertext : DATG

Key : 19

DECRYPTION

	D	A	T	G	
	3	0	19	6	
-	19	19	19	19	
(	-16	-19	0	-13	) mod 26
	10	7	0	13	
	K	H	A	N	

Continued

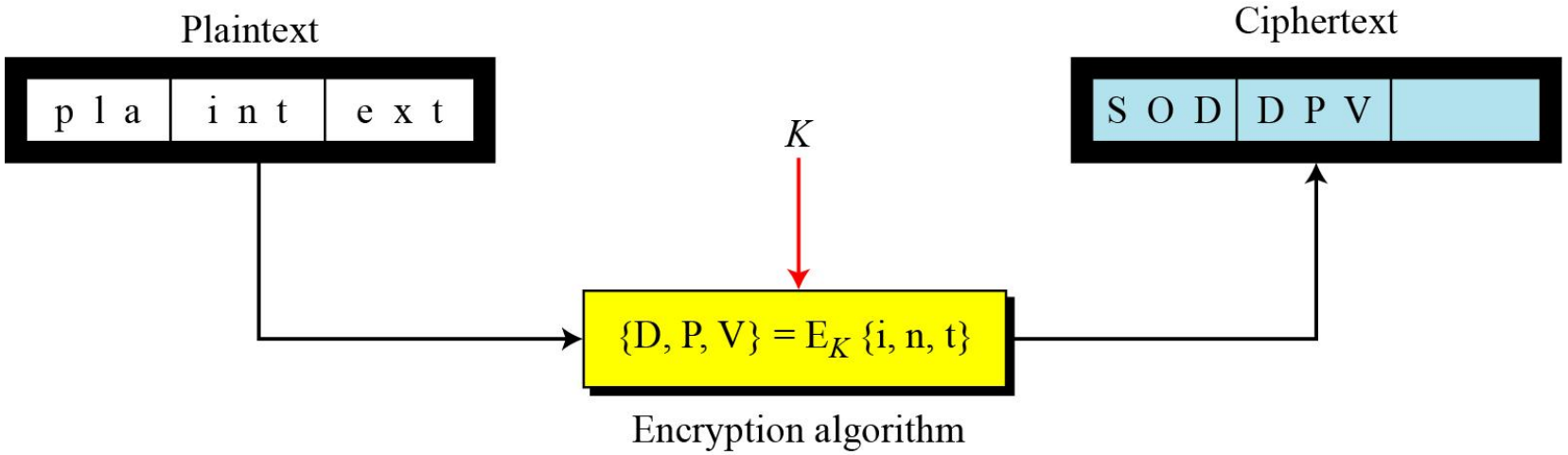
A Vigenere Table

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	v	v	w	x	y	z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Block Ciphers

•In a block cipher, a group of plaintext symbols of size m (m > 1) are encrypted together creating a group of ciphertext of the same size.

•A single key is used to encrypt the whole block even if the key is made of multiple values.



Block Ciphers

•Playfair ciphers are block ciphers. The size of the block is m = 2. Two characters are encrypted together.

•Hill ciphers are block ciphers. A block of plaintext, of size 2 or more is encrypted together using a single key (a matrix).

•Although the key is made of m × m values, it is considered as a single key.

Playfair Cipher

- not even the large number of keys in a monoalphabetic cipher provides security
- one approach to improve security is to encrypt multiple letters
- the **Playfair Cipher** is an example
- invented by Charles Wheatstone in 1854, but named after his friend Baron Playfair

Playfair Key Matrix

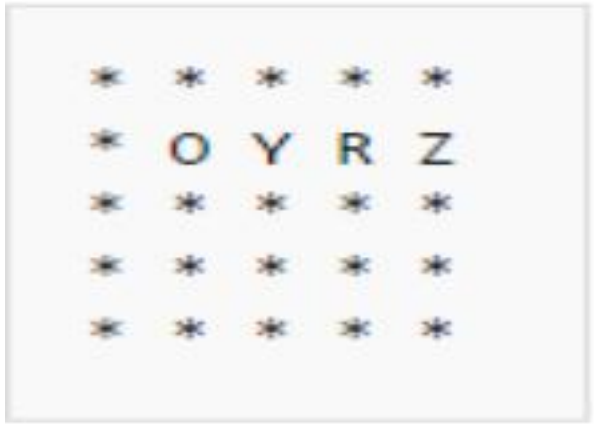
- a 5X5 matrix of letters based on a keyword
- fill in letters of keyword (without duplicates)
- fill rest of matrix with other letters
- eg. using the keyword MONARCHY

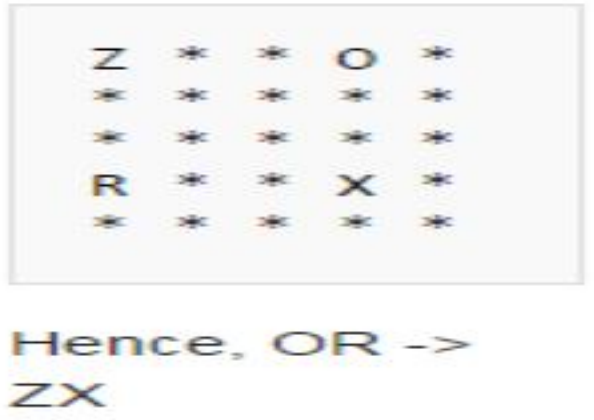
M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

Encrypting and Decrypting

plaintext is encrypted two letters at a time

- if a pair is a repeated letter, insert filler like 'X
- ABC AB CX



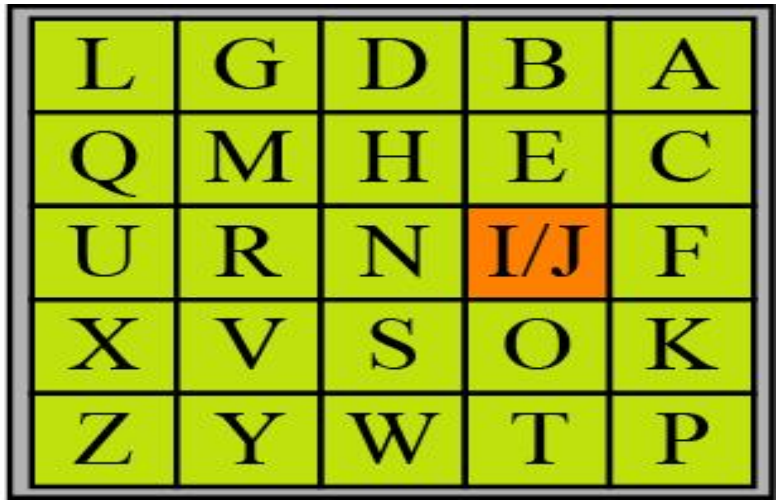


Playfair Cipher

Example

An example of a secret key in the Playfair cipher

Secret Key =



L	G	D	B	A
Q	M	H	E	C
U	R	N	I/J	F
X	V	S	O	K
Z	Y	W	T	P

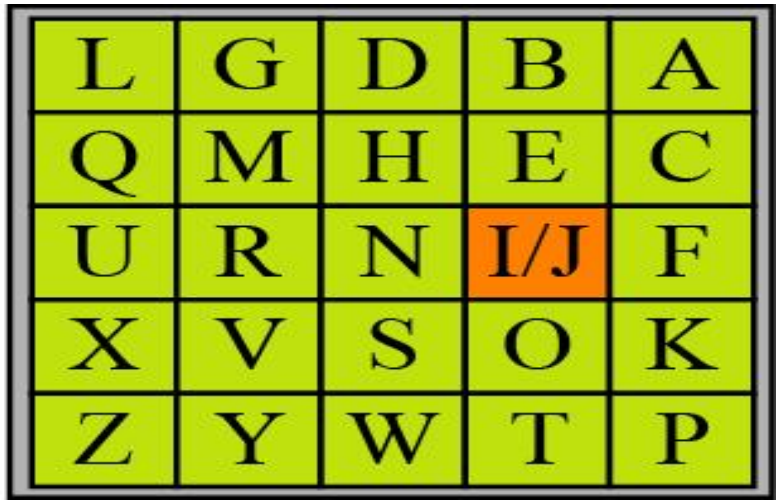
Encrypt the Plaintext “HELLO” ????

Playfair Cipher

Example

An example of a secret key in the Playfair cipher

Secret Key =



L	G	D	B	A
Q	M	H	E	C
U	R	N	I/J	F
X	V	S	O	K
Z	Y	W	T	P

encrypt the plaintext “hello”

he → EC lx → QZ lo → BX

Plaintext: hello Ciphertext: ECQZBX

Example

Key: playfair example

Message "Hide the gold in the tree stump“

(note the null "X" used to separate the repeated "E"s)

Example

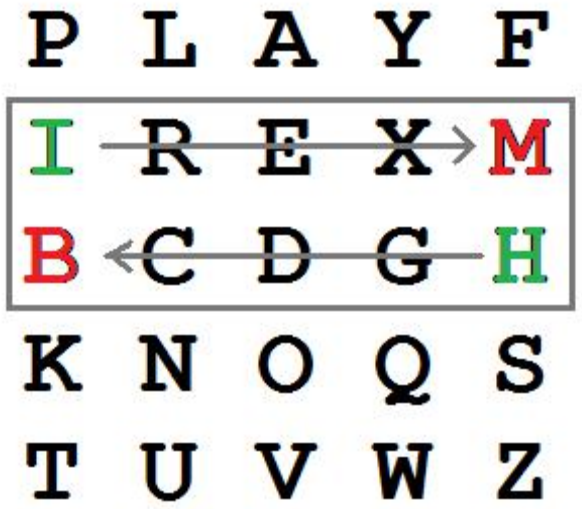
Key: playfair example

P	L	A	Y	F
I	R	E	X	M
B	C	D	G	H
K	N	O	Q	S
T	U	V	W	Z

Message "Hide the gold in the tree stump“

(note the null "X" used to separate the repeated "E"s)

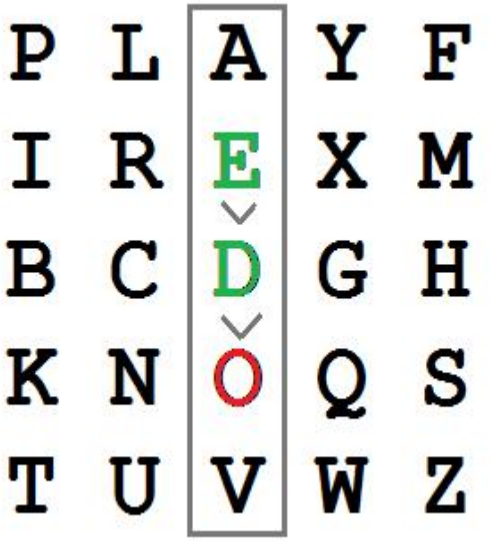
```
HI DE TH EG OL DI NT HE TR EX ES TU MP
                              ^
```



HI

Shape: Rectangle
Rule: Pick Same Rows, Opposite Corners

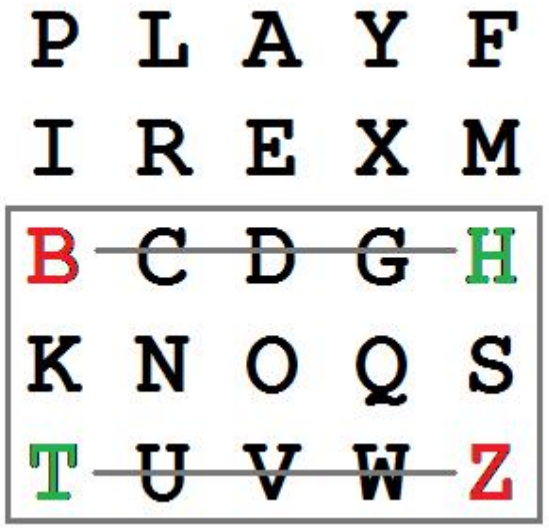
BM



DE

Shape: Column
Rule: Pick Items Below Each Letter, Wrap to Top if Needed

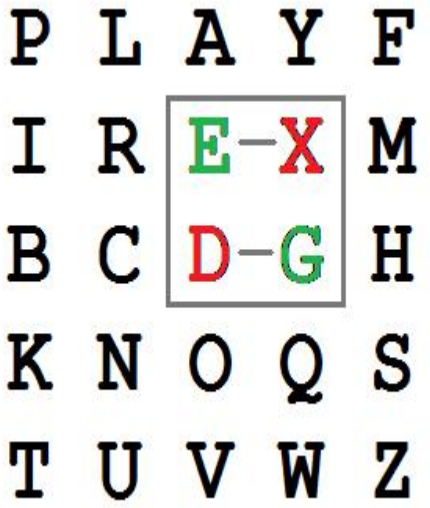
OD



TH

Shape: Rectangle
Rule: Pick Same Rows, Opposite Corners

ZB

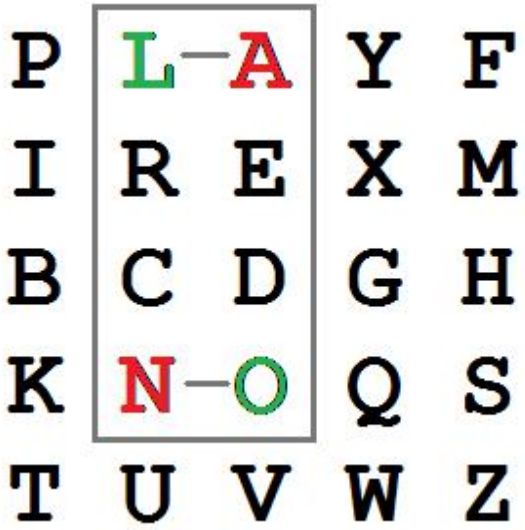


EG

Shape: Rectangle
Rule: Pick Same Rows, Opposite Corners

XD

6. The pair DI forms a rectangle, replace it with BE
7. The pair NT forms a rectangle, replace it with KU
8. The pair HE forms a rectangle, replace it with DM
9. The pair TR forms a rectangle, replace it with UI



OL

Shape: Rectangle
Rule: Pick Same Rows, Opposite Corners

NA

11. The pair ES forms a rectangle, replace it with MO
12. The pair TU is in a row, replace it with UV
13. The pair MP forms a rectangle, replace it with IF

Ciphertext: BM OD ZB XD NA BE KU DM UI XM MO UV IF

Hill Cipher

Key in the Hill cipher

$$\mathrm{K} = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1m} \\ k_{21} & k_{22} & \cdots & k_{2m} \\ \vdots & \vdots & & \vdots \\ k_{\mathrm{m1}} & k_{\mathrm{m2}} & \cdots & k_{mm} \end{bmatrix}$$

$$\begin{aligned} \mathrm{C}_1 &= \mathrm{P}_1 k_{11} + \mathrm{P}_2 k_{21} + \cdots + \mathrm{P}_m k_{m1} \\ \mathrm{C}_2 &= \mathrm{P}_1 k_{12} + \mathrm{P}_2 k_{22} + \cdots + \mathrm{P}_m k_{m2} \\ &\cdots \\ \mathrm{C}_{\mathrm{m}} &= \mathrm{P}_1 k_{1m} + \mathrm{P}_2 k_{2m} + \cdots + \mathrm{P}_m k_{mm} \end{aligned}$$

The key matrix in the Hill cipher needs to have a multiplicative inverse.

Continued

Example

For example, the plaintext “code is ready” can make a 3 × 4 matrix when adding extra padding character “z” to the last block and removing the spaces. The ciphertext is “OHKNIHGKLISS”.

$$\underset{C}{\begin{bmatrix} 14 & 07 & 10 & 13 \\ 08 & 07 & 06 & 11 \\ 11 & 08 & 18 & 18 \end{bmatrix}} = \underset{P}{\begin{bmatrix} 02 & 14 & 03 & 04 \\ 08 & 18 & 17 & 04 \\ 00 & 03 & 24 & 25 \end{bmatrix}} \overset{K}{\begin{bmatrix} 09 & 07 & 11 & 13 \\ 04 & 07 & 05 & 06 \\ 02 & 21 & 14 & 09 \\ 03 & 23 & 21 & 08 \end{bmatrix}}$$

a. Encryption

$$\underset{P}{\begin{bmatrix} 02 & 14 & 03 & 04 \\ 08 & 18 & 17 & 04 \\ 00 & 03 & 24 & 25 \end{bmatrix}} = \underset{C}{\begin{bmatrix} 14 & 07 & 10 & 13 \\ 08 & 07 & 06 & 11 \\ 11 & 08 & 18 & 18 \end{bmatrix}} \overset{K^{-1}}{\begin{bmatrix} 02 & 15 & 22 & 03 \\ 15 & 00 & 19 & 03 \\ 09 & 09 & 03 & 11 \\ 17 & 00 & 04 & 07 \end{bmatrix}}$$

b. Decryption

Example:

Message: “ pay more money”

Key : 17 17 5
21 18 21
2 2 19

$$P.T = \begin{bmatrix} p & m & e & n \\ a & o & m & e \\ y & r & o & y \end{bmatrix} = \begin{bmatrix} 15 & 12 & 4 & 13 \\ 0 & 14 & 12 & 4 \\ 24 & 17 & 14 & 24 \end{bmatrix}$$

$$P.T_1 = \begin{bmatrix} p \\ a \\ y \end{bmatrix} = \begin{bmatrix} 15 \\ 0 \\ 24 \end{bmatrix}$$

$$C.T_1 = Key\ x\ P.T_1\ mod\ 26 = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \begin{bmatrix} 15 \\ 0 \\ 24 \end{bmatrix}\ mod\ 26 = \begin{bmatrix} 11 \\ 13 \\ 18 \end{bmatrix} = \begin{bmatrix} L \\ N \\ S \end{bmatrix}$$

$$C.T_2 = Key\ x\ P.T_2\ mod\ 26 = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \begin{bmatrix} 12 \\ 14 \\ 17 \end{bmatrix}\ mod\ 26 = \begin{bmatrix} 7 \\ 3 \\ 11 \end{bmatrix} = \begin{bmatrix} H \\ D \\ L \end{bmatrix}$$

- And so on… then the C.T = **LNS HDL……..**

Example

Ciphertext : WIIT

Key: 3 7

1 5

Message: ?

Example

Ciphertext : WIIT

Key: 3 7

1 5

Message: ?

HELP

Inverse : 2 x 2 Matrix

$$\begin{bmatrix} 4 & 7 \\ 2 & 6 \end{bmatrix}^{-1} = \frac{1}{4\times6-7\times2}\begin{bmatrix} 6 & -7 \\ -2 & 4 \end{bmatrix}$$

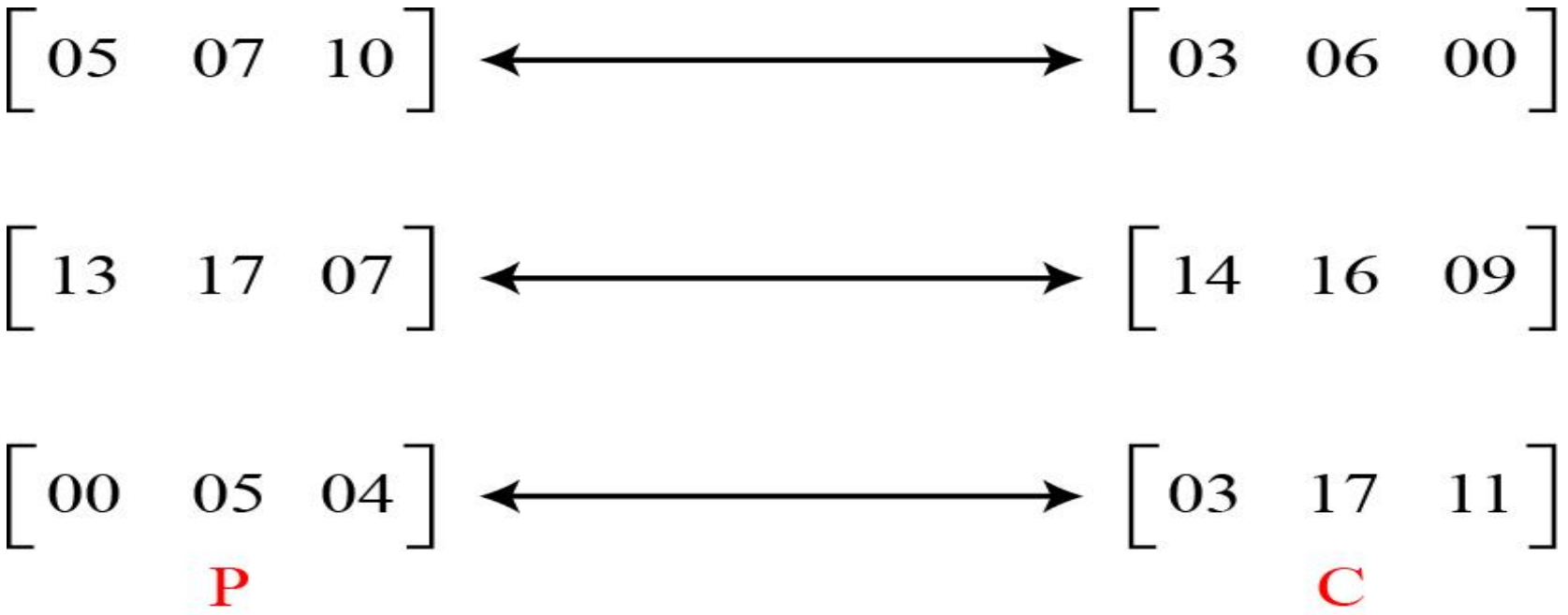
$$= \frac{1}{10}\begin{bmatrix} 6 & -7 \\ -2 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} 0.6 & -0.7 \\ -0.2 & 0.4 \end{bmatrix}$$

Continued

Example

Assume that Eve knows that m = 3. She has intercepted three plaintext/ciphertext pair blocks (not necessarily from the same message)



Example

She makes matrices P and C from these pairs. Because P is invertible, she inverts the P matrix and multiplies it by C to get the K matrix

$$\begin{bmatrix} 02 & 03 & 07 \\ 05 & 07 & 09 \\ 01 & 02 & 11 \end{bmatrix} = \begin{bmatrix} 21 & 14 & 01 \\ 00 & 08 & 25 \\ 13 & 03 & 08 \end{bmatrix} \begin{bmatrix} 03 & 06 & 00 \\ 14 & 16 & 09 \\ 03 & 17 & 11 \end{bmatrix}$$

$$K \qquad P^{-1} \qquad C$$

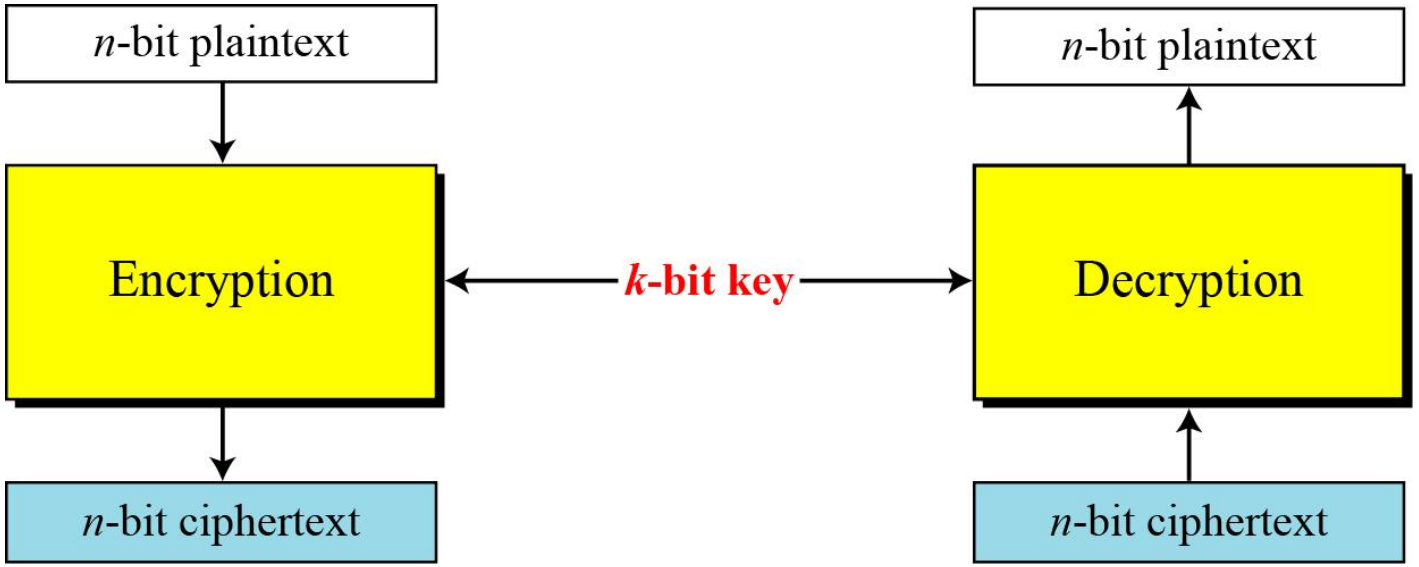
Now she has the key and can break any ciphertext encrypted with that key.

Block Ciphers

A symmetric-key modern block cipher encrypts an n-bit block of plaintext or decrypts an n-bit block of ciphertext.

The encryption or decryption algorithm uses a k-bit key.

A modern block cipher

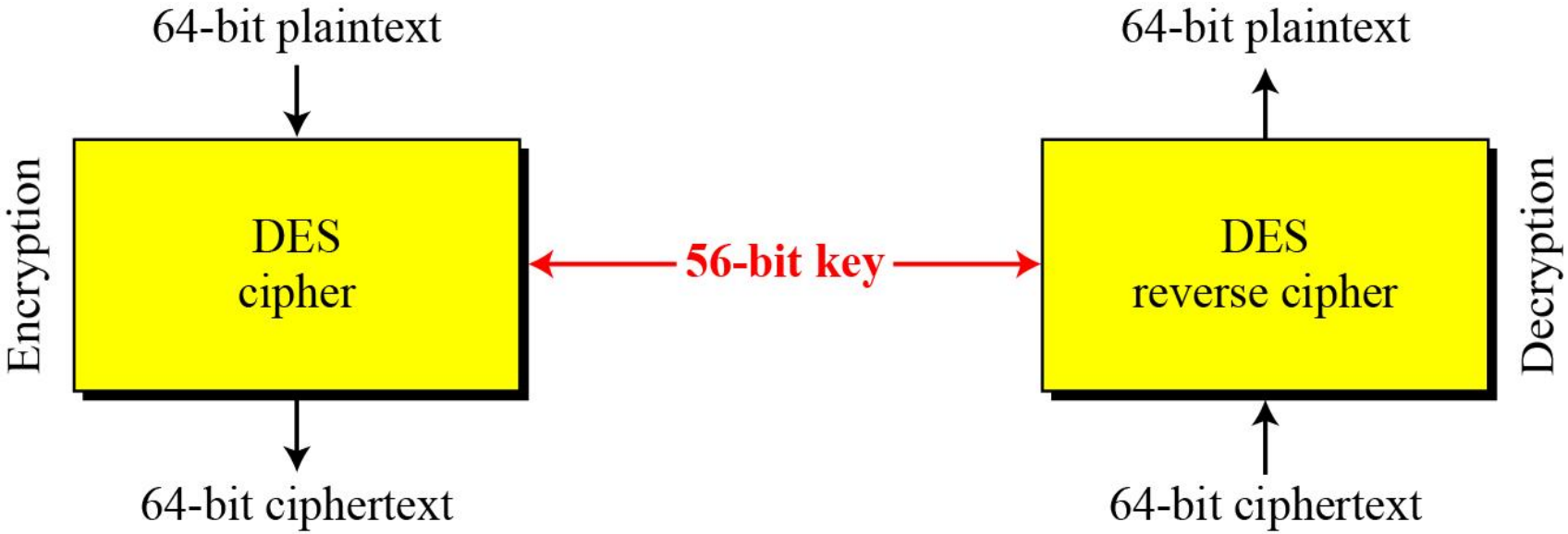


Substitution or Transposition

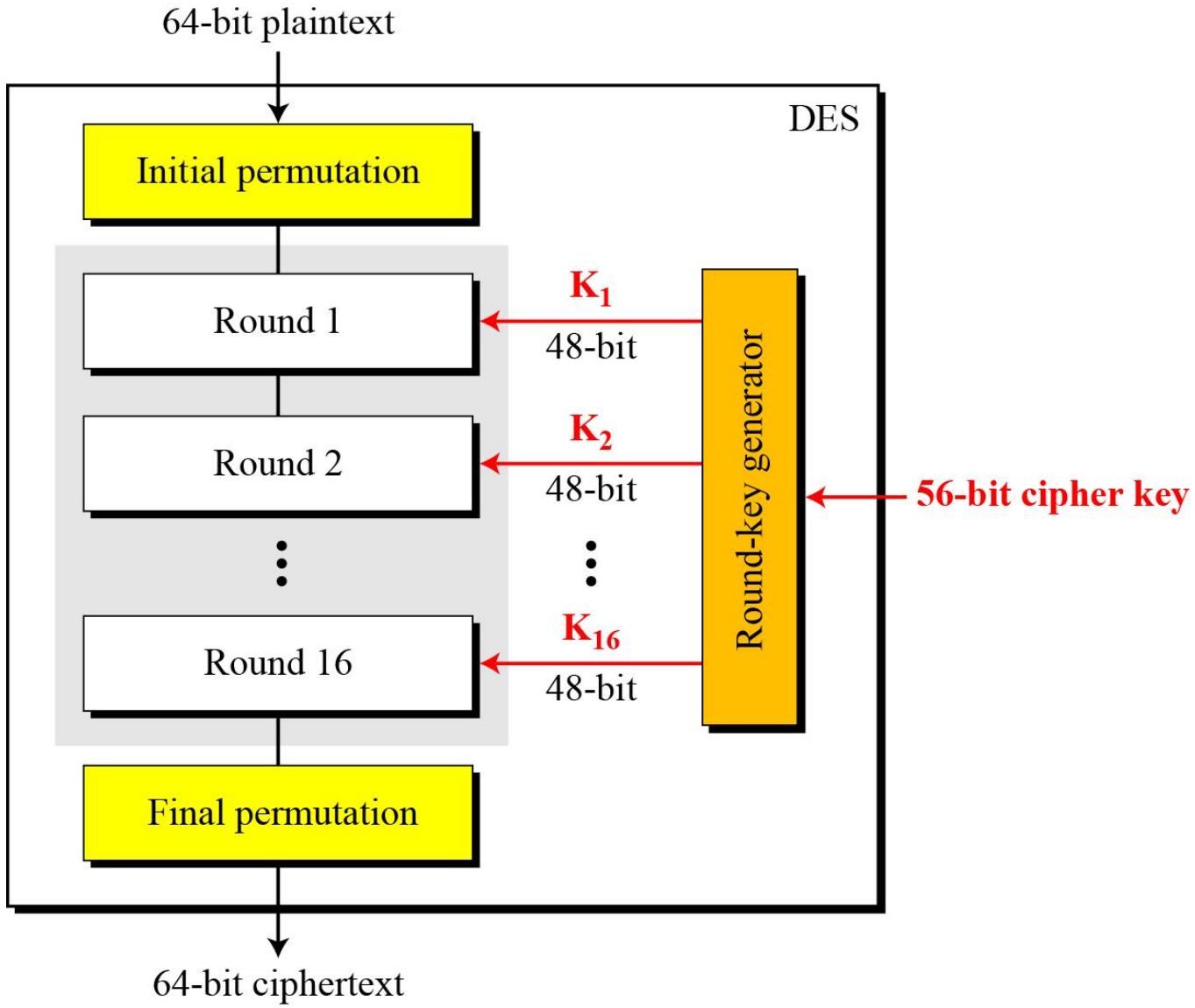
A modern block cipher can be designed to act as a substitution cipher or a transposition cipher.

To be resistant to exhaustive-search attack, a modern block cipher needs to be designed as combination of both substitution and transposition .

Encryption and decryption with DES



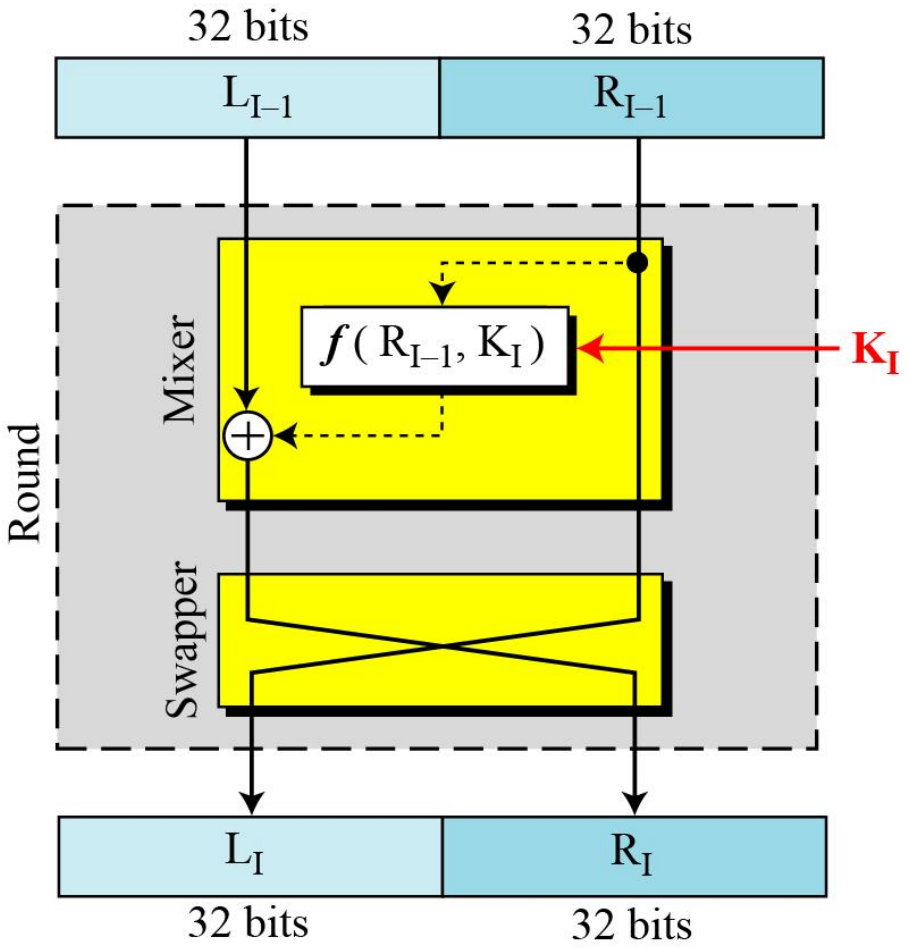
DES STRUCTURE



Continued

The initial and final permutations are straight P-boxes that are inverses of each other.

DES uses 16 rounds. Each round of DES is a Feistel cipher.

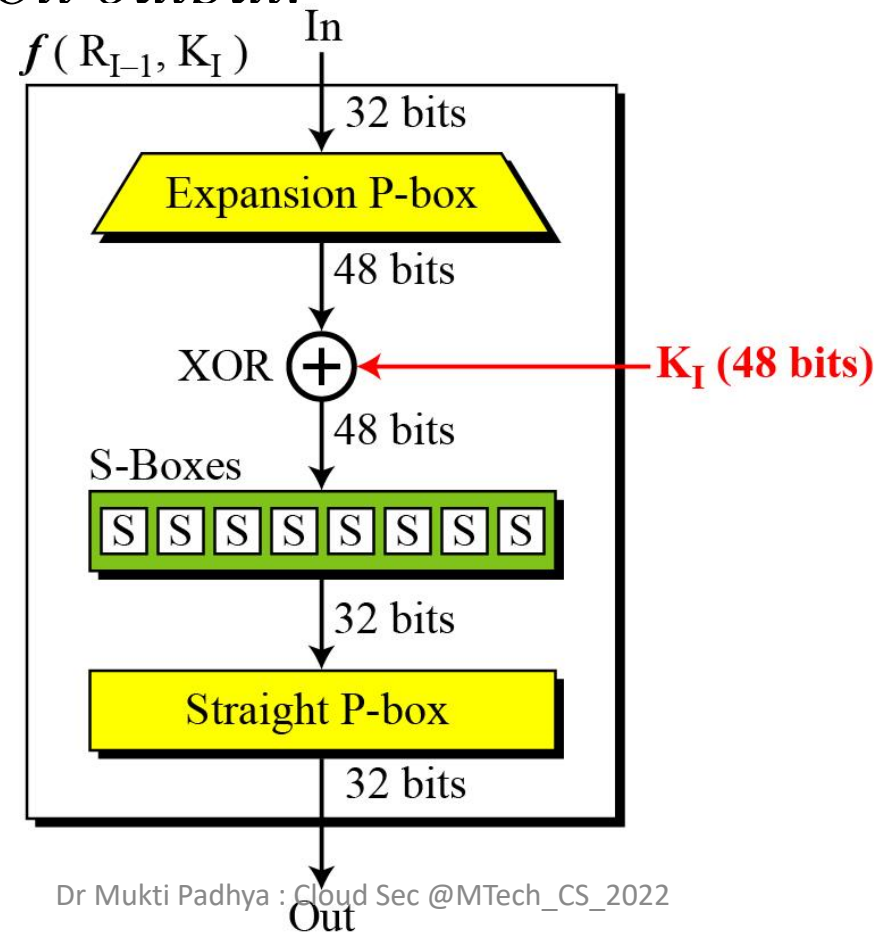


A round in DES (encryption site)

DES Function

The heart of DES is the DES function. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.

DES function



Asymmetric-Key Cryptography

Note

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

Symmetric-Key Cryptography

- traditional **private/secret/single key** cryptography uses **one** key
- shared by both sender and receiver
- if this key is disclosed communications are compromised
- also is **symmetric**, parties are equal hence does not protect sender from receiver forging a message & claiming is sent by sender

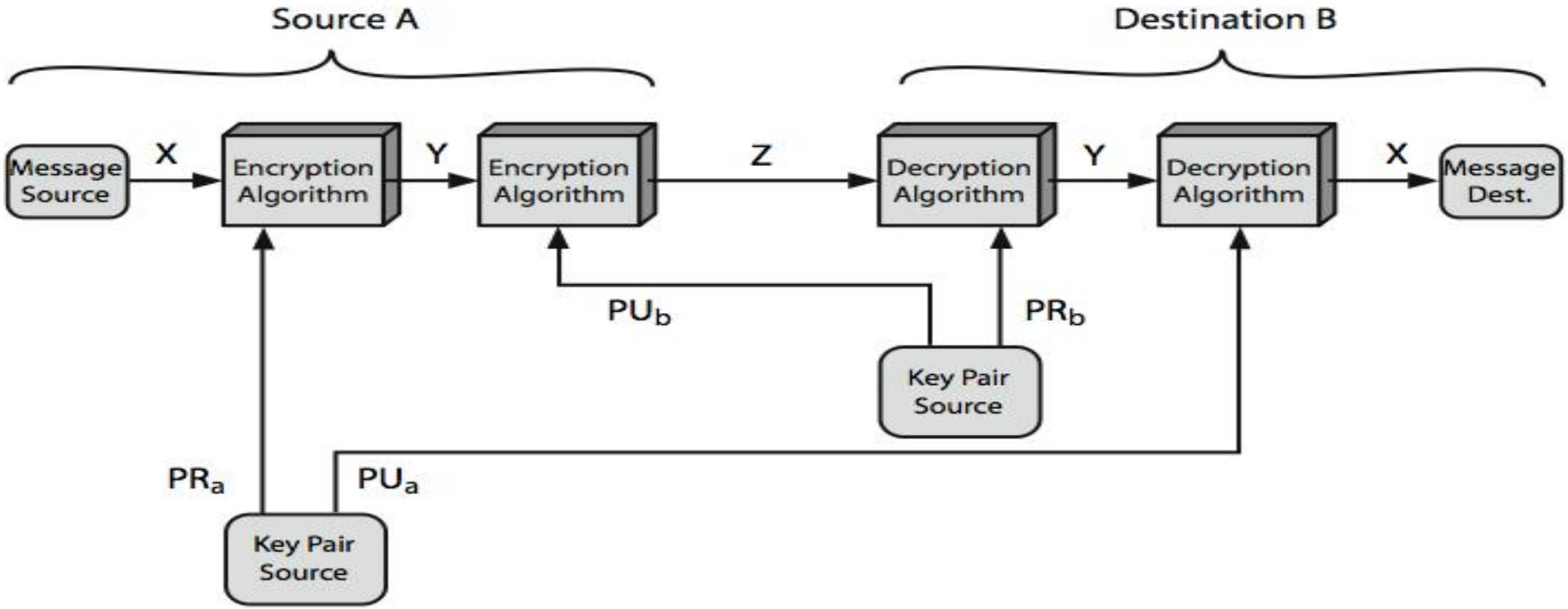
Asymmetric-Key Cryptography

- probably most significant advance in the 3000 year history of cryptography
- uses **two** keys – a public & a private key
- **asymmetric** since parties are **not** equal
- uses clever application of number theoretic concepts to function
- complements **rather than** replaces private key crypto

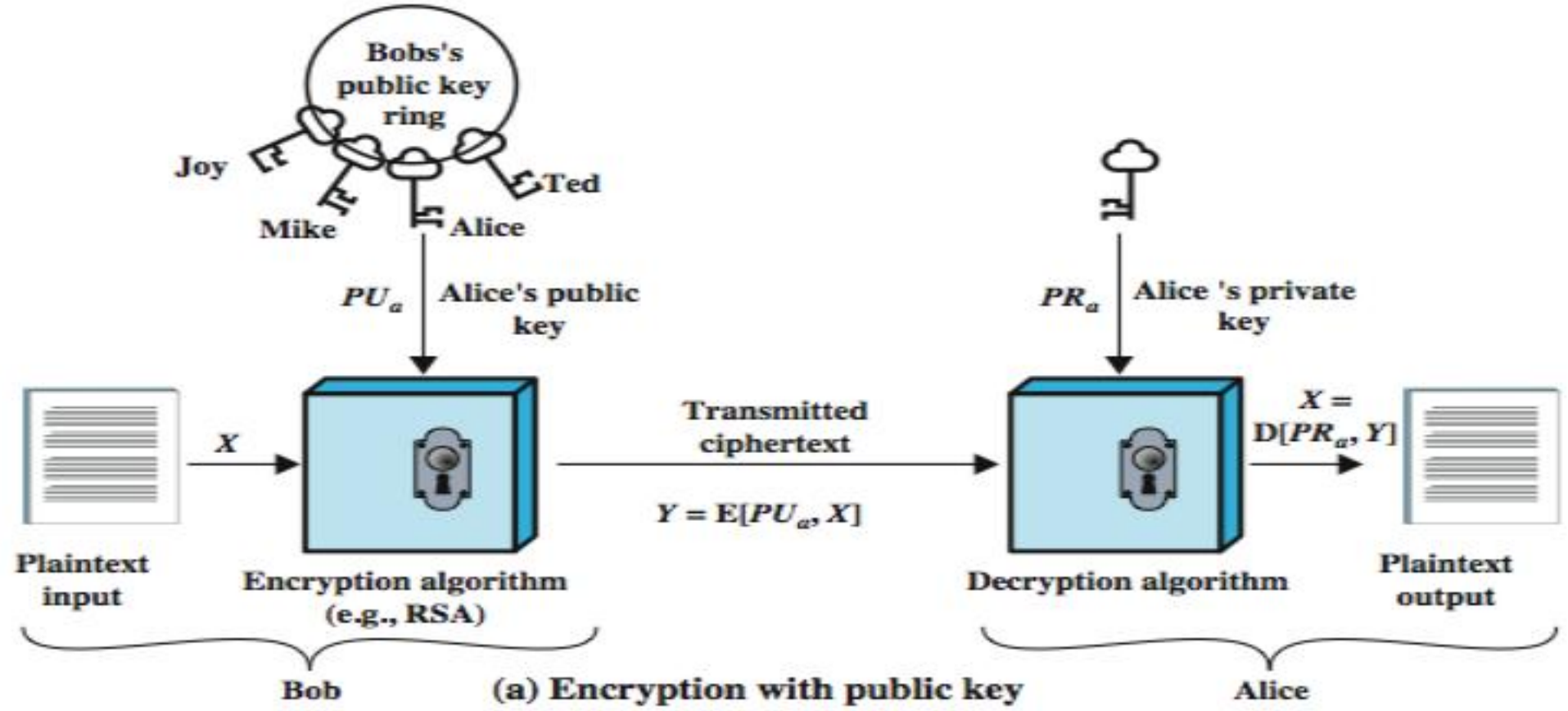
Public-Key Cryptography

- **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 related **private-key**, known only to the recipient, used to **decrypt messages**, and **sign** (create) **signatures**
- **infeasible to determine private key from public**
- is **asymmetric** because
  - those who encrypt messages or verify signatures **cannot** decrypt messages or create signatures

Public-Key Cryptosystems



Asymmetric/Public-Key Cryptography Many to one communication



Public-Key Requirements

- Public-Key algorithms rely on two keys where:
  - it is computationally infeasible to find decryption key knowing only algorithm & encryption key
  - it is 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 (for some algorithms)
- these are formidable requirements which only a few algorithms have satisfied

Symmetric vs Public-Key

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 no other information is available. 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 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 no other information is available. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

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 **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

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
- public invention due to Whitfield Diffie & Martin Hellman at Stanford University in 1976

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

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

Symmetric-key Distribution

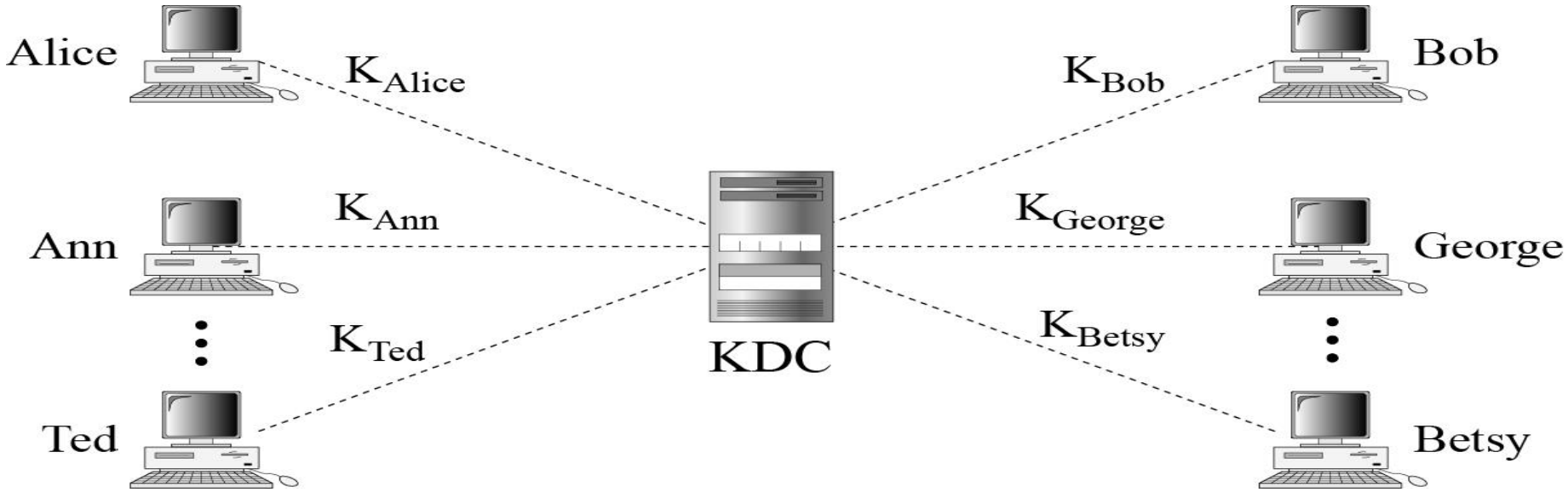
Symmetric-key cryptography is more efficient than asymmetric-key cryptography for enciphering large messages. Symmetric-key cryptography, however, needs a shared secret key between two parties. The distribution of keys is another problem.

Two Possible Ways:

1. Key-Distribution Center: KDC
2. Session Keys

Key-Distribution Center: KDC

Key-distribution center (KDC)



Symmetric-key Agreement

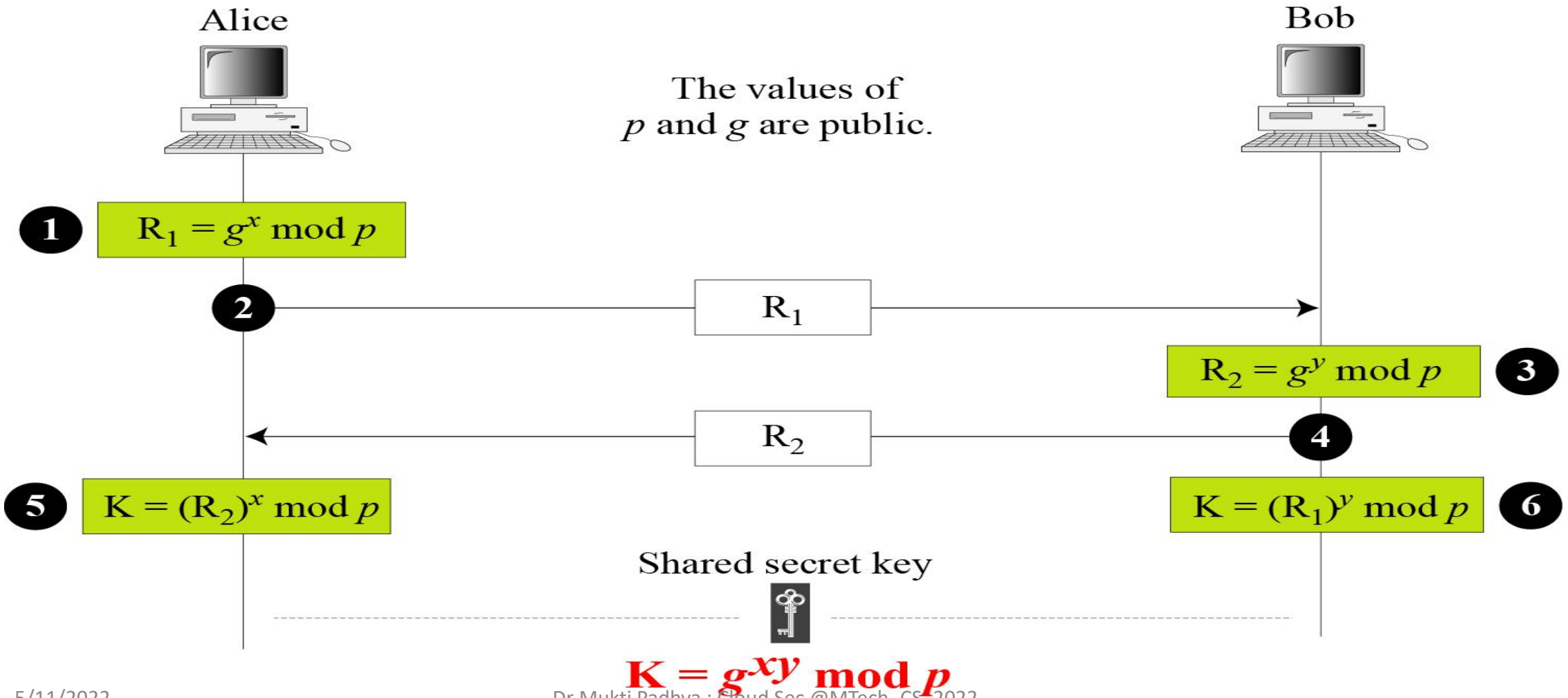
Alice and Bob can create a session key between themselves without using a KDC. This method of session-key creation is referred to as the symmetric-key agreement.

Techniques for Key Agreement:

1 Diffie-Hellman Key Agreement
2 Station-to-Station Key Agreement

Diffie-Hellman Key Agreement

Diffie-Hellman method



Note

The symmetric (shared) key in the Diffie-Hellman method is
$K = g^{xy} \bmod p$.

Example

Let us give a trivial example to make the procedure clear. Our example uses small numbers, but note that in a real situation, the numbers are very large. Assume that $g = 7$ and $p = 23$. The steps are as follows:

1. Alice chooses $x = 3$ and calculates $R_1 = 7\text{^}3 \bmod 23 = 21$.
2. Bob chooses $y = 6$ and calculates $R_2 = 7\text{^}6 \bmod 23 = 4$.
3. Alice sends the number 21 to Bob.
4. Bob sends the number 4 to Alice.
5. Alice calculates the symmetric key $K = 4\text{^}3 \bmod 23 = 18$.
6. Bob calculates the symmetric key $K = 21\text{^}6 \bmod 23 = 18$.
7. The value of K is the same for both Alice and Bob; $g^{xy} \bmod p = 7\text{^}18 \bmod 35 = 18$.

Example

Let us give a more realistic example. We used a program to create a random integer of 512 bits (the ideal is 1024 bits). The integer p is a 159-digit number. We also choose g, x, and y as shown below:

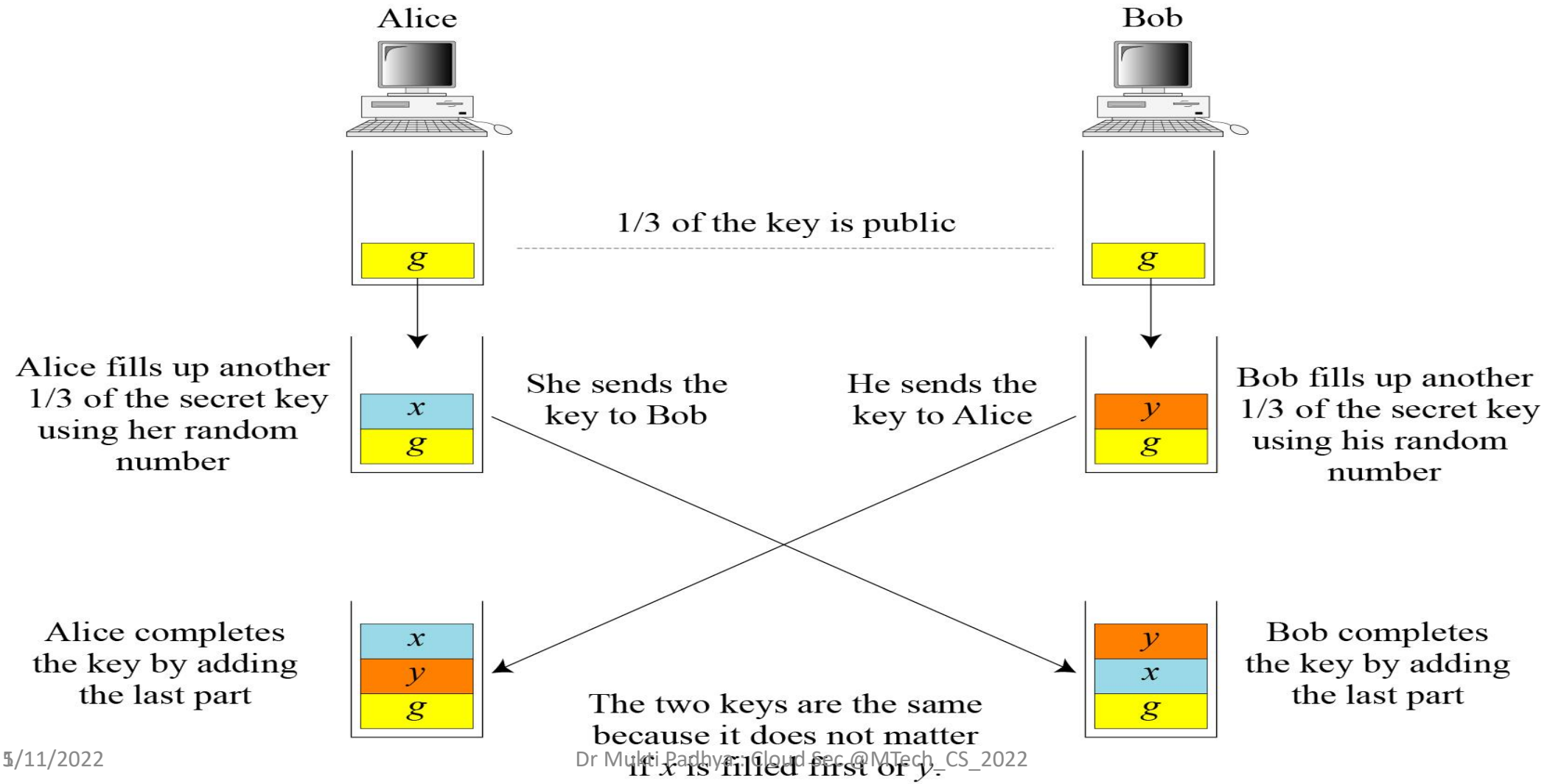
p	7646242985634935721824937659550305074763380967269497489235737728609252356666607554236374233096611800333381061947301309504147387009991780436548785807987581
g	2
x	557
y	273

Continued

The following shows the values of R_1, R_2, and K.

R_1	844920284205665505216172947491035094143433698520012660862863631067673619959280828586700802131859290945140217500319973312945836083821943065966020157955354
R_2	435262838709200379470747114895581627636389116262115557975123379218566310011435718208390040181876486841753831165342691630263421106721508589625520128859414 3
K	155638000664522290596225827523270765273218046944423678520320400146406500887936651204257426776608327911017153038674561252213151610976584200120408643361774 0

Diffie-Hellman idea



Example:

Step 1: Alice and Bob get public numbers P = 23, G = 9

Step 2: Alice selected a private key x = 4 and Bob selected a private key y = 3

Step 3: Alice and Bob compute public values

Alice: R1= ?

Bob: R2 = ?

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key R2 and Bob receives public key R1

Step 6: Alice and Bob compute symmetric keys

Alice: ka = ?

Bob: kb = ?

Step 7: What is the value of shared secret key k??.

Example:

Step 1: Alice and Bob get public numbers P = 23, G = 9

Step 2: Alice selected a private key x = 4 and Bob selected a private key y = 3

Step 3: Alice and Bob compute public values

Alice: R1 =(9^4 mod 23) = (6561 mod 23) = 6

Bob: R2 = (9^3 mod 23) = (729 mod 23) = 16

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key R2 =16 and Bob receives public key R1 = 6

Step 6: Alice and Bob compute symmetric keys

Alice: ka = R2^a mod p = 65536 mod 23 = 9

Bob: kb = R1^b mod p = 216 mod 23 = 9

Step 7: K=9 is the shared secret.

RSA Cryptosystem

The most common public-key algorithm is the RSA cryptosystem, named for its inventors (Rivest, Shamir, and Adleman).

RSA

- by Rivest, Shamir & Adleman of MIT in 1977
- best known & widely used public-key scheme
- based on exponentiation in a finite (Galois) field over integers modulo a prime
  - nb. exponentiation takes $O((\log n)^3)$ operations (easy)
- uses large integers (eg. 1024 bits)
- security due to cost of factoring large numbers
  - nb. factorization takes $O(e^{\log n \log \log n})$ operations (hard)

Multiplicative Inverses(cont.)

- Find the multiplicative inverse of 11 in Z_{26}.

Solution

q	r_1	r_2	r	t_1	t_2	t
2	26	11	4	0	1	–2
2	11	4	3	1	–2	5
1	4	3	1	–2	5	–7
3	3	1	0	5	–7	26
	1	0		–7	26	

The gcd (26, 11) is 1; the inverse of 11 is -7 or 19.

Multiplicative Inverses(cont.)

- Find the inverse of 12 in Z_{26}.

Solution

q	r_1	r_2	r	t_1	t_2	t
2	26	12	2	0	1	–2
6	12	2	0	1	–2	13
	2	0		–2	13	

The gcd (26, 12) is 2; the inverse does not exist.

Multiplicative Inverses(cont.)

- Find the multiplicative inverse of 23 in Z_{100}.

Multiplicative Inverses(cont.)

- Find the multiplicative inverse of 23 in Z_{100}.

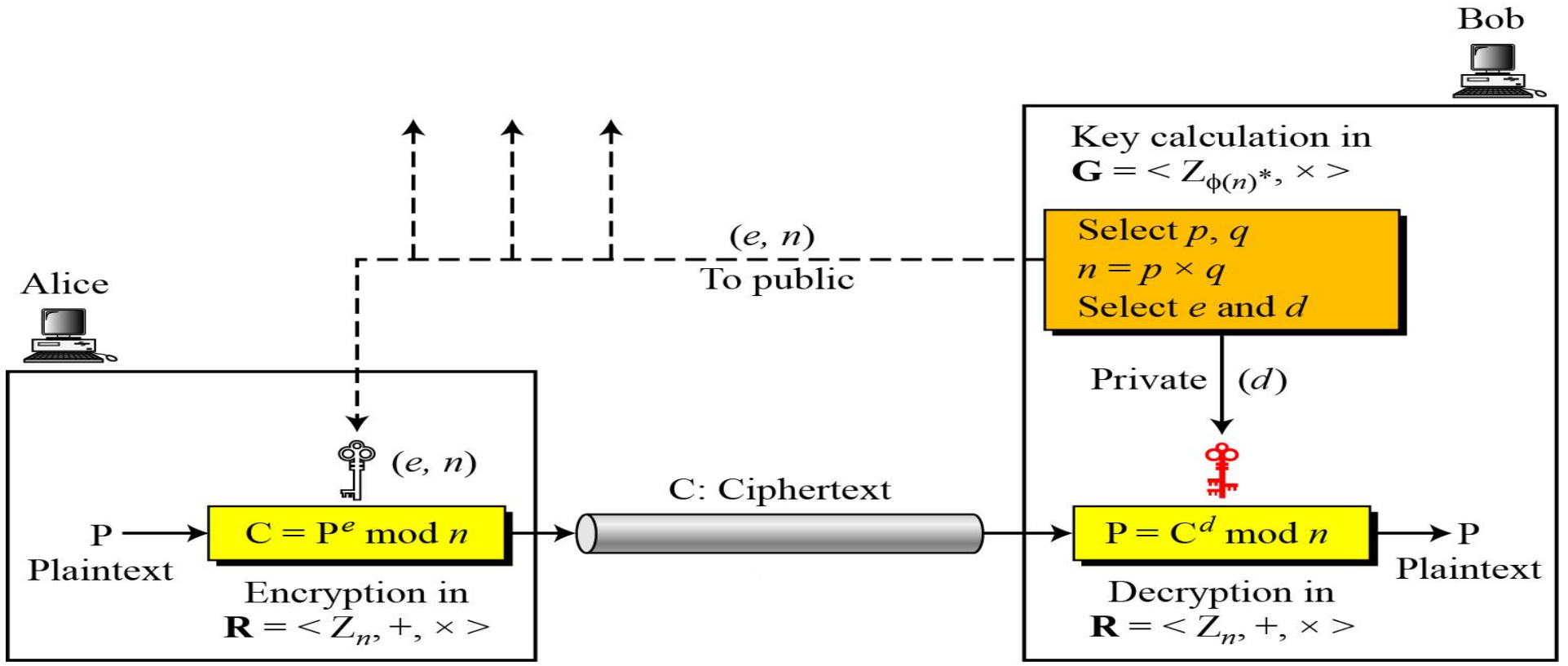
Solution

q	r_1	r_2	r	t_1	t_2	t
4	100	23	8	0	1	−4
2	23	8	7	1	−4	19
1	8	7	1	−4	9	−13
7	7	1	0	9	−13	100
	1	0		−13	100	

The gcd (100, 23) is 1; the inverse of 23 is -13 or 87.

Procedure

Encryption, decryption, and key generation in RSA



RSA Key Generation

- users of RSA must:
  - determine two primes at random - `p, q`
  - select either `e` or `d` and compute the other
- primes `p,q` must not be easily derived from modulus `n=p.q`
  - means must be sufficiently large
  - typically guess and use probabilistic test
- exponents `e, d` are inverses, so use Inverse algorithm to compute the other

RSA Key Setup

- each user generates a public/private key pair by:
- selecting two large primes at random: p, q
- computing their system modulus $n=p.q$
  - note $\varnothing(n)=(p-1)(q-1)$
  - Gcd(3,7) = ?
- selecting at random the encryption key e
  - where $1<e<\varnothing(n), \gcd(e,\varnothing(n))=1$
- solve following equation to find decryption key d
  - $e.d=1 \mod \varnothing(n)$ and $0 \leq d \leq n$
- publish their public encryption key: PU={e,n}
- keep secret private decryption key: PR={d,n}

Algorithm 10.2 *RSA Key Generation*

RSA_Key_Generation

{

Select two large primes p and q such that $p \neq q$.

$n \leftarrow p \times q$

$\phi(n) \leftarrow (p-1) \times (q-1)$

Select e such that $1 < e < \phi(n)$ and e is coprime to $\phi(n)$

$d \leftarrow e^{-1} \bmod \phi(n)$ // d is inverse of e modulo $\phi(n)$

Public_key $\leftarrow$ (e, n) // To be announced publicly

Private_key $\leftarrow d$ // To be kept secret

return Public_key and Private_key

}

RSA En/decryption

- to encrypt a message M the sender:
  - obtains **public key** of recipient $PU=\{e,n\}$
  - computes: $C = M^e \bmod n$, where $0 \leq M < n$
- to decrypt the ciphertext C the owner:
  - uses their private key $PR=\{d,n\}$
  - computes: $M = C^d \bmod n$
- note that the message M must be smaller than the modulus n (block if needed)

Encryption

Algorithm 10.3 *RSA encryption*

```
RSA_Encryption (P, e, n)                    // P is the plaintext in Z_n and P < n
{
    C  ←  Fast_Exponentiation (P, e, n)     // Calculation of (P^e mod n)
    return C
}
```

In RSA, p and q must be at least 512 bits; n must be at least 1024 bits.

Decryption

Algorithm 10.4 *RSA decryption*

```
RSA_Decryption (C, d, n)                    //C is the ciphertext in Z_n
{
    P ← Fast_Exponentiation (C, d, n)       // Calculation of (C^d mod n)
    return P
}
```

Why RSA Works

- because of Euler's Theorem:
  - $a^{\phi(n)} \bmod n = 1$ where $\gcd(a,n)=1$
- in RSA have:
  - $n=p.q$
  - $\phi(n)=(p-1)(q-1)$
  - carefully chose e & d to be inverses $\bmod \phi(n)$
  - hence $e.d=1+k.\phi(n)$ for some k
- hence :

$$C^d = M^{e.d} = M^{1+k.\phi(n)} = M^1.(M^{\phi(n)})^k$$
$$= M^1.(1)^k = M^1 = M \bmod n$$

RSA Example - Key Setup

1. Select primes: $p=17$ & $q=11$
2. Calculate $n = pq = 17 \times 11 = 187$
3. Calculate $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$
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 $23 \times 7 = 161 = 10 \times 160 + 1$
6. Publish public key $PU=\{7,187\}$
7. Keep secret private key $PR=\{23,187\}$

RSA Example - En/Decryption

- sample RSA encryption/decryption is:
- given message $M = 88$ (nb. $88<187$)
- encryption:

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

- decryption:

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

Example

Bob chooses 7 and 11 as *p* and *q* and calculates *n* = 77.

The value of ϕ(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
Now imagine that Alice wants to send the plaintext 5 to Bob.

C = ?

Example

Bob chooses 7 and 11 as *p* and *q* and calculates *n* = 77. The value of ϕ(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

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

Example

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

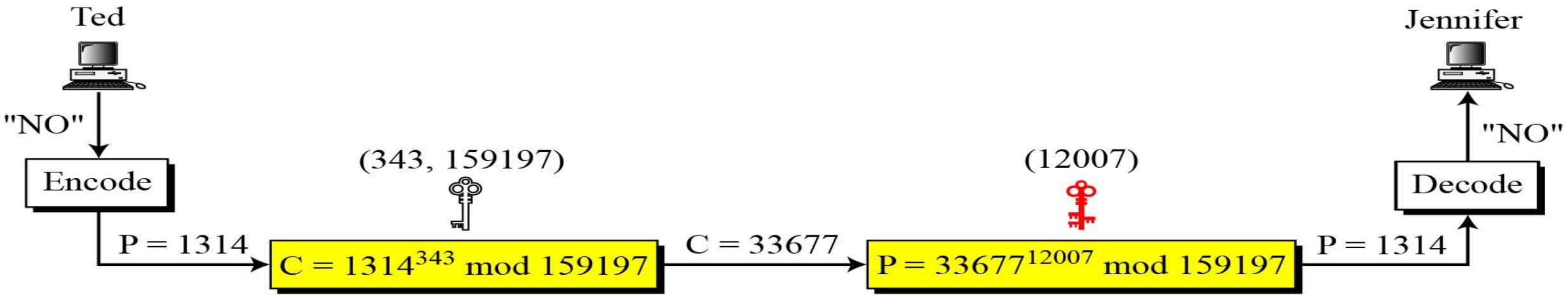
Suppose Ted wants to send the message “NO” to Jennifer.
The plaintext is ?.

Example

Jennifer creates a pair of keys for herself.
She chooses p = 397 and q = 401.
She calculates n = ?.
She then calculates ϕ(n) = ?.
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 ?.

Encryption and decryption Example



Exponentiation

- can use the Square and Multiply Algorithm
- a fast, efficient algorithm for exponentiation
- concept is based on repeatedly squaring base
- and multiplying in the ones that are needed to compute the result
- look at binary representation of exponent
- only takes $O(\log_2 n)$ multiples for number n
  - eg. $7^5 = 7^4.7^1 = 3.7 = 10 \bmod 11$
  - eg. $3^{129} = 3^{128}.3^1 = 5.3 = 4 \bmod 11$

RSA Security

- possible approaches to attacking RSA are:
  - brute force key search - infeasible given size of numbers
  - mathematical attacks - by factoring modulus n
  - timing attacks - on running of decryption
  - chosen ciphertext attacks - given properties of RSA

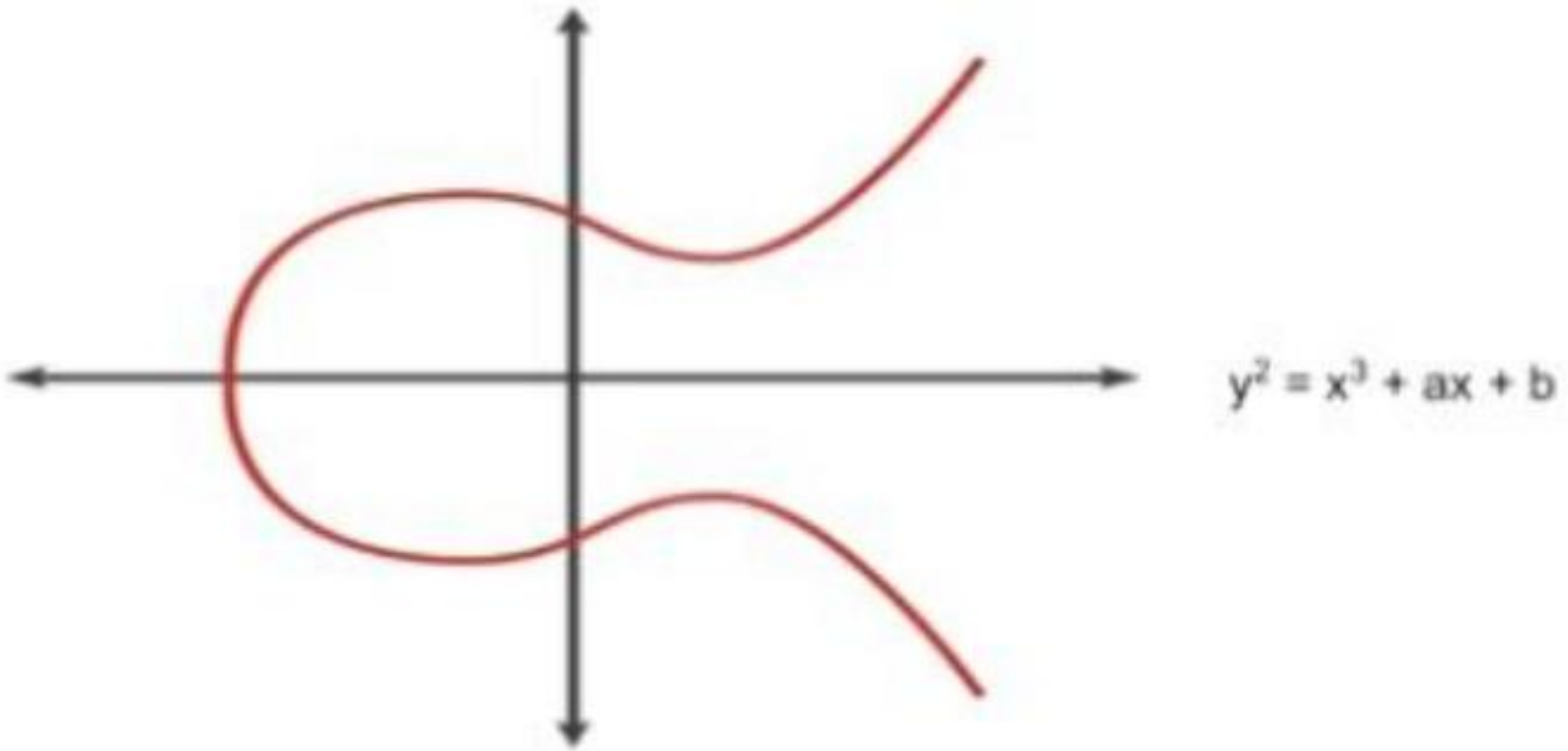
RSA Security

- possible approaches to attacking RSA are:
  - brute force key search - infeasible given size of numbers
  - mathematical attacks - based on difficulty of computing ø(n), by factoring modulus n
  - timing attacks - on running of decryption
  - chosen ciphertext attacks - given properties of RSA
  - **Solution?**
    - **We can increse key size RSA2048,...Downside?**
    - **Costly Exponentiation operation for large key**
    - **Feasible on Resource-limited device?**

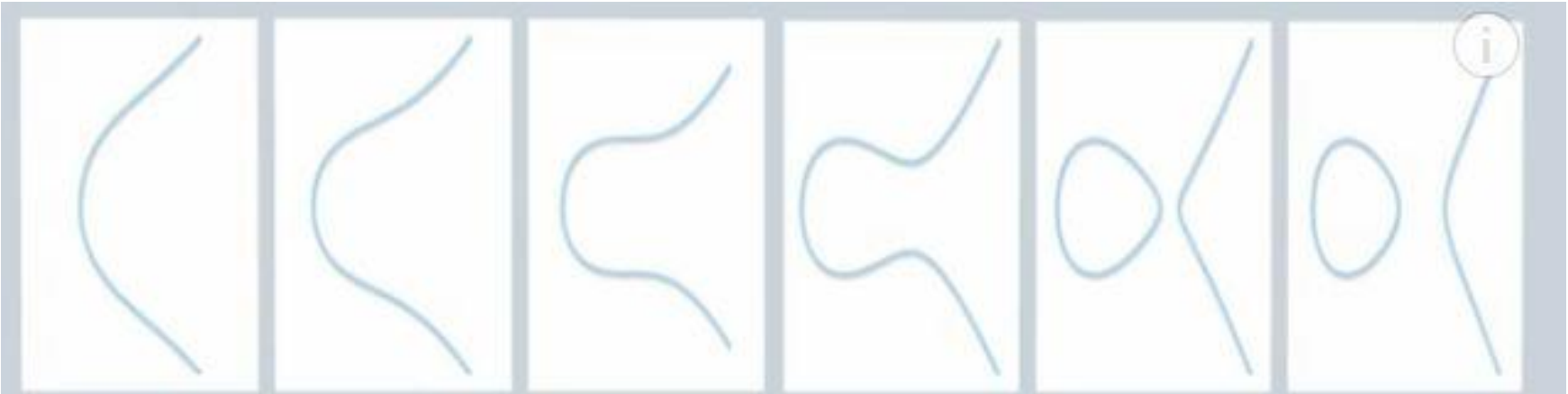
ECC : Elliptic Curve Cryptography

- Elliptic Curve Cryptography is similar to RSA
  - ECC leverages a trap door function
- However, ECC's trap door is much stronger
- Based on an esoteric branch of mathematics
  - Has proven incredibly powerful

What is Elliptic Curve ?



What is Elliptic Curve ?

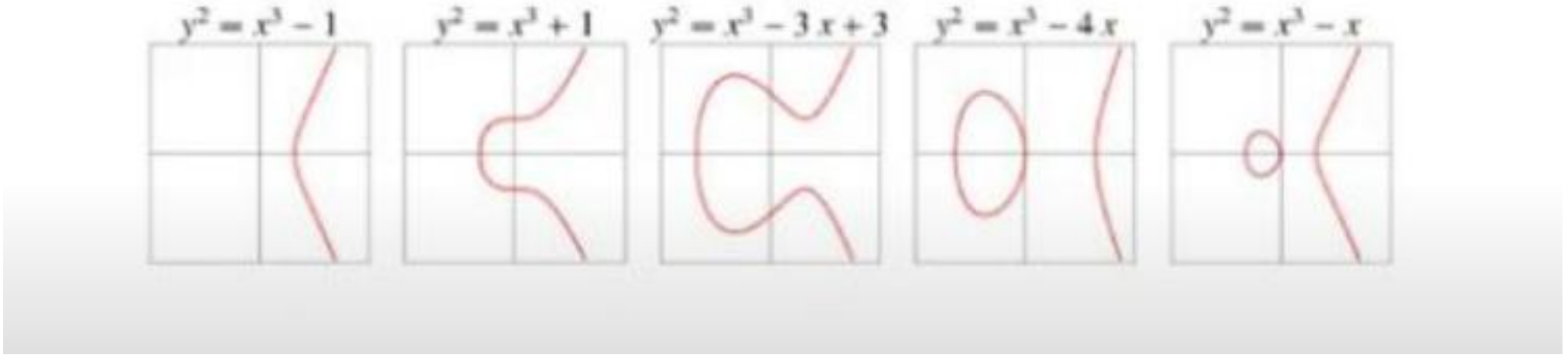


$b = 1$ and a varying from 2 to -3

$$\{(x, y) \in \mathbb{R}^2 \mid y^2 = x^3 + ax + b, 4a^3 + 27b^2 \neq 0\} \cup \{\infty\}$$

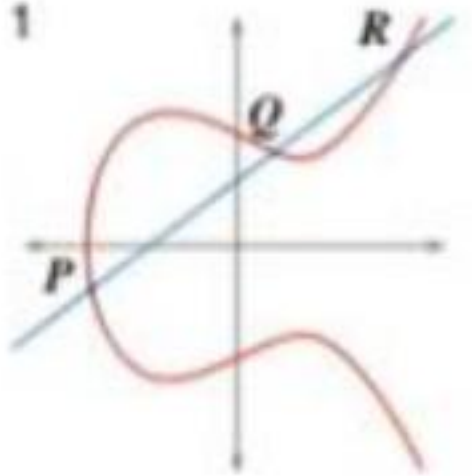
What is Elliptic Curve ?

$$y^2 = x^3 + Ax^2 + Bx + C$$



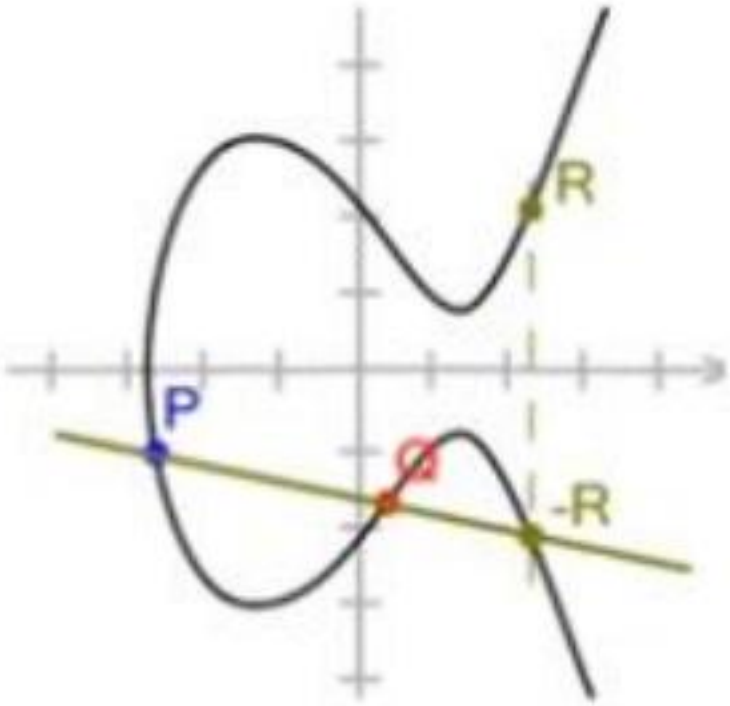
Interesting Properties

- Symmetric about the x-axis
  - Every point has an opposite
- Every two points has a third
  - Exception is vertical line



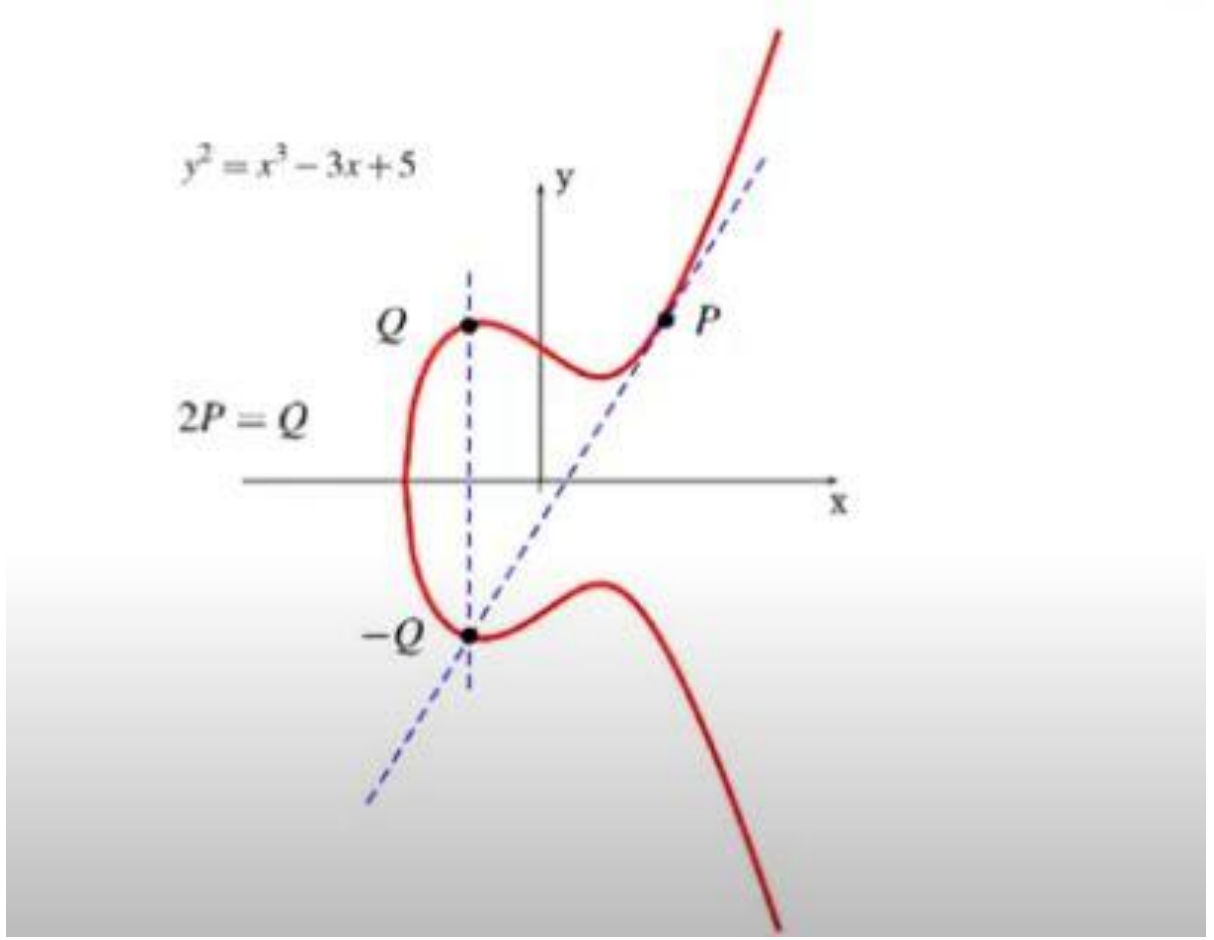
Interesting Properties

- Main machinery
  - Lets call it “dot”
- So P dot Q = R
- Not itself a trap door
  - Can be reversed



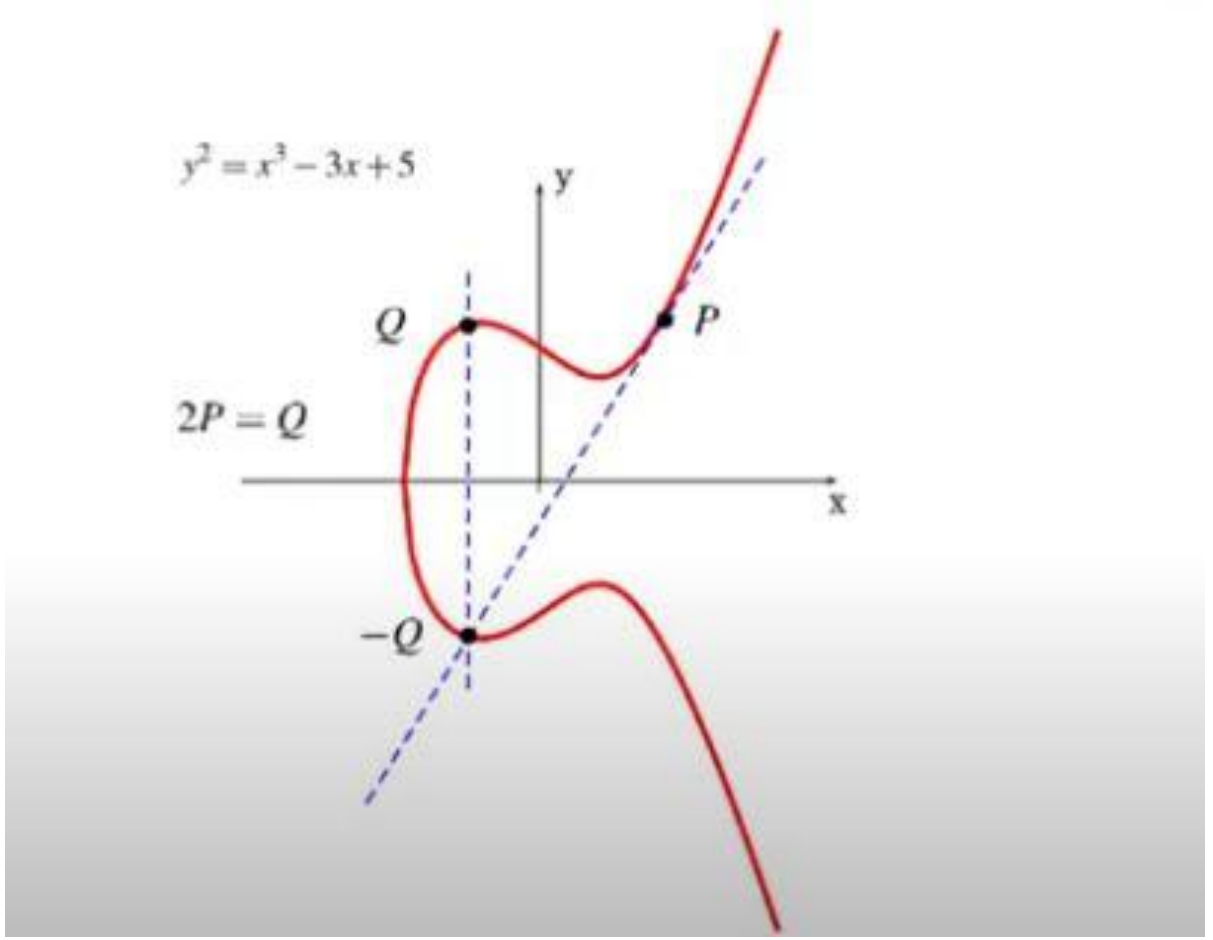
Interesting Properties

- We can also dot a point with itself
  - Use the tangent line



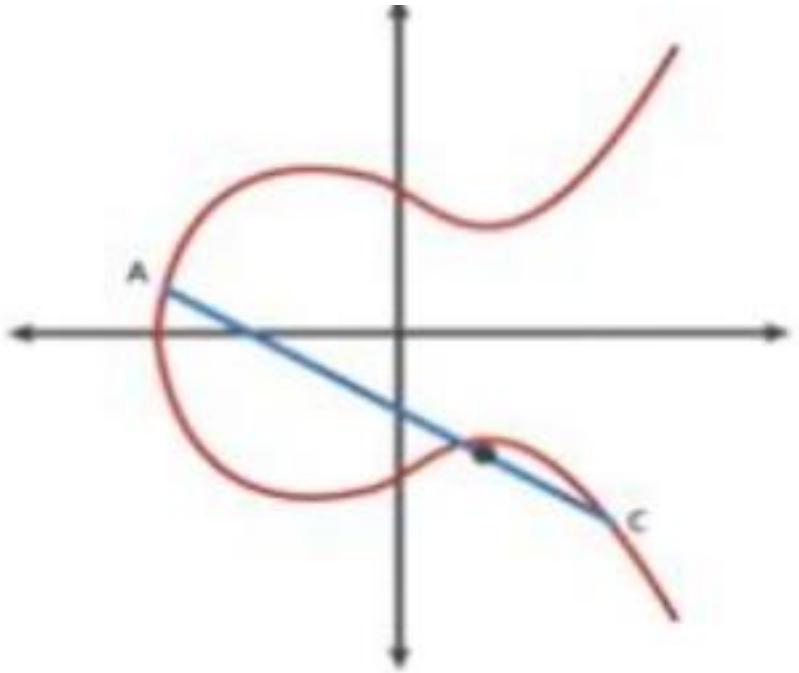
Interesting Properties

- We can also dot a point with itself
  - Use the tangent line



Interesting Properties

- We can apply this function multiple times
  - A dot B = C
  - A dot C = D
  - A dot D = E



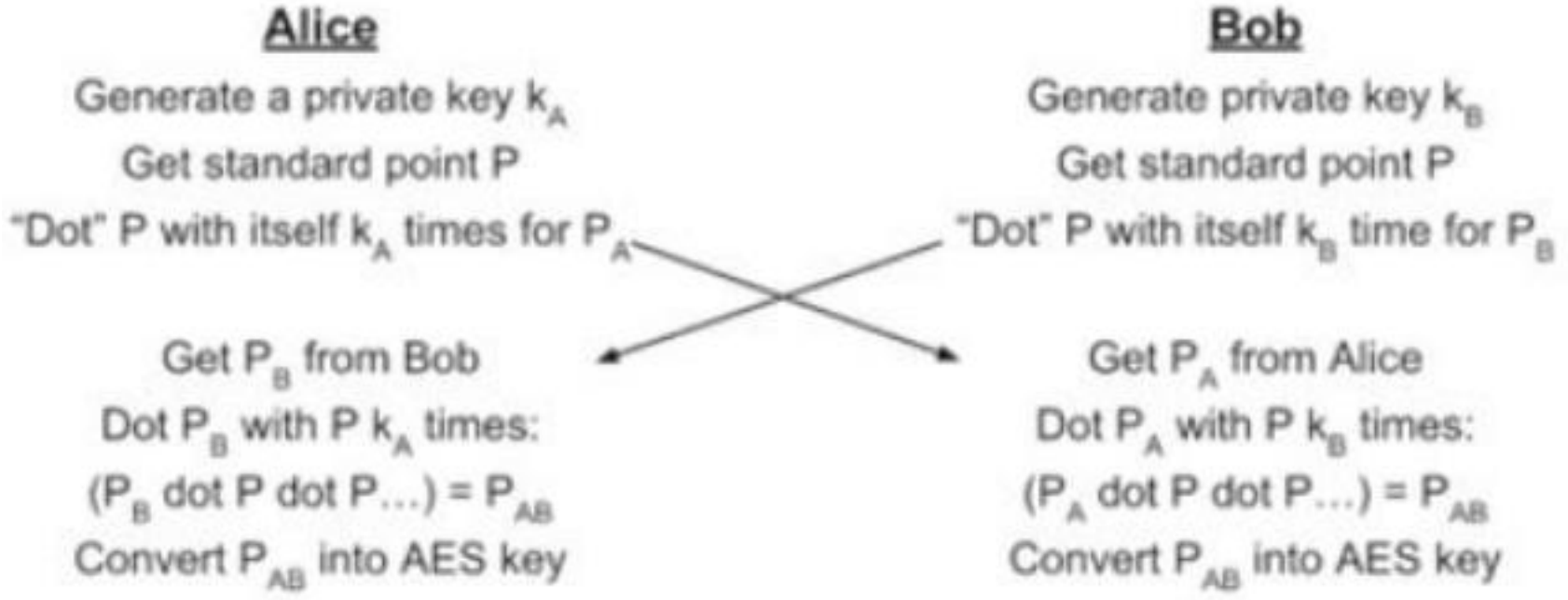
Interesting Properties

- Million Dollar Question -
  - How many time did you dot the point with itself
- Pool analogy
  - How many times was the ball hit?
- Really hard to find out
  - Need to try every possibility

ECC : Key Generation

- Standard: Curve Equation, Start point A
- Private Key - a random (should be prime) number, n
- Public key - a specified point dotted with itself n times
  - (...((A dot A) dot A) dot A … dot A) ← n times
- Our trapdoor protects the private key
- We can use these keys for everything RSA did

Diffie-Hellman 2.0 : ECC Version



Diffie-Hellman 2.0 : ECC Version

Time Complexity

$$Q = kP$$

Calculating Q: O(log n) time complexity ($nP = 0$)

Calculating k: O($2^{k/2}$) time complexity

This problem, known as the discrete log problem, is similar to what makes other public key cryptosystems secure.

ECC Security

You can compute how much energy is needed to break a cryptographic algorithm and compare that with how much water that energy could boil... By this measure, breaking a 228-bit RSA key requires less energy than it takes to boil a teaspoon of water. Comparatively, breaking a 228-bit elliptic curve key requires enough energy to boil all the water on earth. For this level of security with RSA, you'd need a key with 2,380 bits.

- Nick Sullivan, *A (Relatively Easy to Understand) Primer on Elliptic Curve Cryptography*

ECC Security

Smaller Keys = More Speed

RSA key size (bits)	Equivalent ECC key size (bits)
1024	160
2048	224
3072	256
7680	384
15360	521

Who is using ECC?

- SSL/TLS
- Bitcoin
- Tor annonymity Protection
- SSH Keys
- Apple iMessage Digital Signing
- The US Government
- All recent cryptography algo