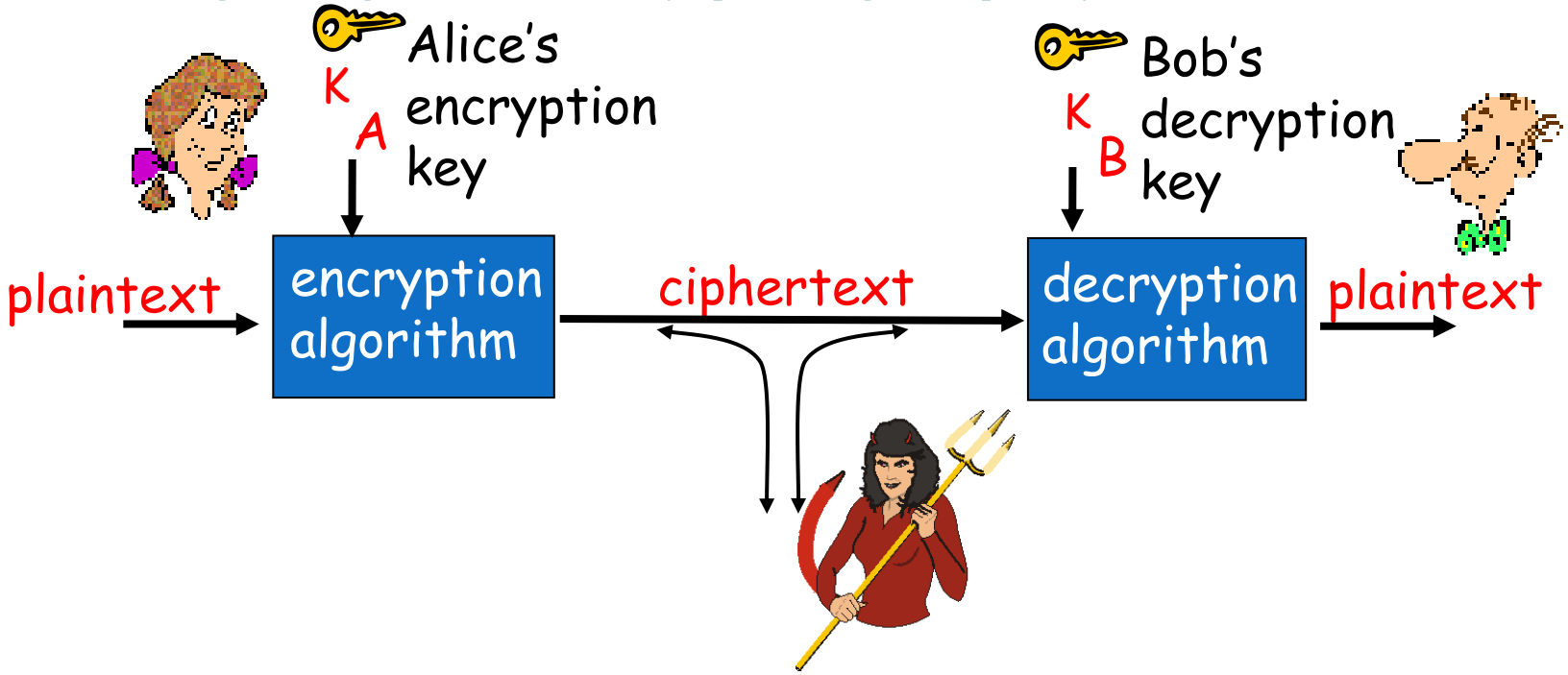
Computer Network Security

Dr. Nadine ZBIB
Assistant Professor
College of Science and Information Systems

Chapter II: Symmetric encryption and message confidentiality

1. Symmetric Encryption Principles
2. Symmetric Block Encryption Algorithms
3. Stream Ciphers and RC4
4. Cipher Block Modes of Operation
5. Location of Encryption Devices
6. Key Distribution

The language of cryptography



symmetric key crypto: sender, receiver keys *identical*

public-key crypto: encryption key *public*, decryption key *secret* (private)

The language of cryptography

Plaintext: This is the original message or data that is fed into the algorithm as input

Encryption algorithm: this algorithm performs various substitutions and transformations on the plaintext.

Secret key: is also input to the algorithm. The exact substitutions and transformations performed by the algorithm depend on the key.

ciphertext: This is the scrambled message produced as output. T depends on the plaintext and the secret key. For given 2 different keys will produce 2 different ciphertexts.

Decryption algorithm: this is essentially the encryption algorithm run in reverse. It takes the ciphertext and the same secret key and produces the original plaintext.

Confidentiality using Symmetric Encryption

- have two major placement alternatives
- **link encryption**
  - encryption occurs independently on every link
  - implies must decrypt traffic between links
  - encrypts all the data along a specific communication path,
- **end-to-end encryption**
  - Sender and receiver must have obtained copies of the secret key
  - encryption occurs between original source and final destination

Advantages	Disadvantages
end-to-end encryption	
•It provides more flexibility to the user in choosing what gets encrypted and how. • Each hop computer on the network does not need to have a key to decrypt each packet.	•Headers, addresses, and routing information are not encrypted, and therefore not protected.
link encryption	
•All data are encrypted, including headers, addresses, and routing information. • Users do not need to do anything to initiate it. It works at a lower layer in the OSI model.	•Key distribution and management are more complex because each hop device must receive a key, and when the keys change, each must be updated. • Packets are decrypted at each hop; thus, more points of vulnerability exist.

Traffic Analysis

- when using end-to-end encryption must leave headers in clear
  - so network can correctly route information
- hence although contents protected, traffic pattern flows are not
- ideally want both at once
  - end-to-end protects data contents over entire path and provides authentication
  - link protects traffic flows from monitoring

Placement of Encryption

- can place encryption function at various layers in OSI Reference Model
  - link encryption occurs at layers 1 or 2
  - end-to-end can occur at layers 3, 4, 6, 7
  - as move higher less information is encrypted but it is more secure though more complex with more entities and keys

Traffic Analysis

- is monitoring of communications flows between parties
  - useful both in military & commercial spheres
  - can also be used to create a covert channel (canal cache)
- link encryption obscures header details
  - but overall traffic volumes in networks and at end-points is still visible
- traffic padding can further obscure flows
  - but at cost of continuous traffic

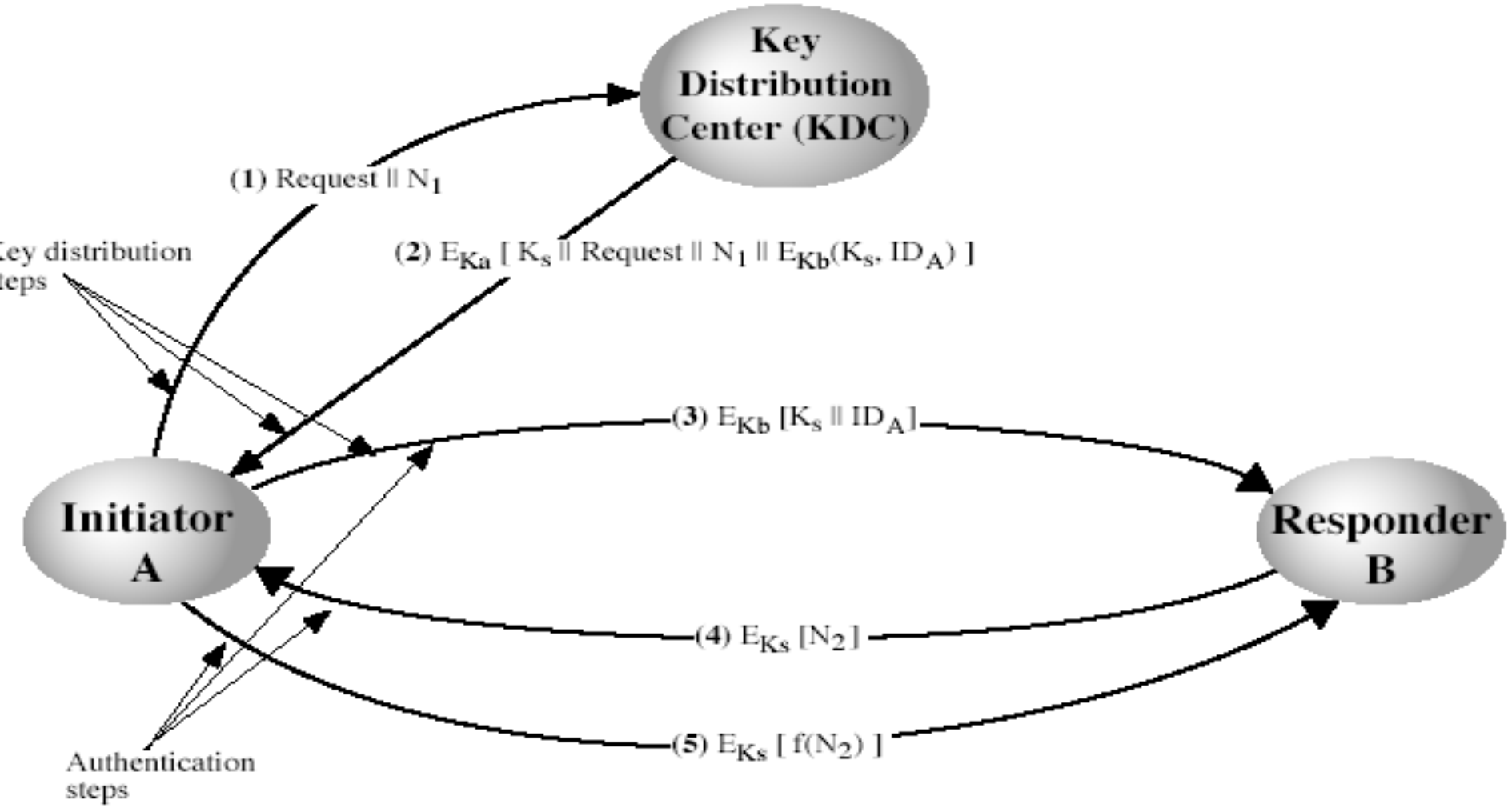
Key Distribution

- symmetric schemes require both parties to share a common secret key
- issue is how to securely distribute this key
- often secure system failure due to a break in the key distribution scheme

Key Distribution

- given parties A and B have various **key distribution** alternatives:
  1. A can select key and physically deliver to B
  2. third party can select & deliver key to A & B
  3. if A & B have communicated previously can use previous key to encrypt a new key
  4. if A & B have secure communications with a third party C, C can relay (transmettre) key between A & B

Key Distribution Scenario



Key Distribution Issues

- Session Key: when 2 end systems wish to communicate, they establish a logical connection. For the duration of that logical connection, all user data are encrypted with a one-time session key.

- Permanent Key: is a key used between entities for the purpose of distributing session keys.

- Key distribution center: determines which systems are allowed to communicate with each other. When permission is obtained for 2 systems to establish a connection, the KDC provides a one-time session key for that connection.

Key Distribution Issues

- hierarchies of KDC's required for large networks, but must trust each other
- session key lifetimes should be limited for greater security
- use of automatic key distribution on behalf of users, but must trust system
- use of decentralized key distribution
- controlling purposes keys are used for

Summary

- have considered:
  - use of symmetric encryption to protect confidentiality
  - need for good key distribution
  - use of trusted third party KDC's

Symmetric key cryptography

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
plaintext:   abcdefghijklmnopqrstuvwxyz
                  |                   |
                  v                   v
ciphertext:  mnbvcxzasdfghjklpoiuytrewq
```

E.g.:

```
Plaintext: bob. i love you. alice

ciphertext: nkn. s gktc wky. mgsbc
```

Q: How hard to break this simple cipher?:

- brute force (how hard?)
- other?

Symmetric Encryption

Tools

- Substitution,
- Transposition,
- exclusive Or,
- logical Shift
- Combination of the above functions.

A. Substitution Cypher

Each letter that you want to encipher is substituted by another letter or symbol, but the order in which these appear is kept the same. Another way to say this is that the message you want to keep secret (called the plaintext) is transformed into the enciphered message (called the ciphertext) by using a different alphabet.

Each character or group of characters of the plaintext message is substituted by another character or group of characters. This process should increase the confusion so difficulties in linking the plaintext message. The inverse substitution gives the plaintext.

If the cipher operates on single letters, it is termed a **simple substitution** cipher; a cipher that operates on larger groups of letters is termed **polygraphic** (each group of letters is substitued by another group of letters).

A **monoalphabetic** cipher uses fixed substitution over the entire message, whereas a **polyalphabetic** cipher uses a number of substitutions at different times in the message, where a unit from the plaintext is mapped to one of several possibilities in the ciphertext and vice-versa.

Substitution cipher
simple monoalphabetic substitution cipher

- A monoalphabetic substitution is a cipher in which each occurrence of a plaintext symbol is replaced by a corresponding ciphertext symbol to generate ciphertext. The key for such a cipher is a table of the correspondence or a function from which the correspondence is computed.

Example: If the keyword is `ANDREW DICKSON WHITE`, then the cipher alphabet is given by

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

Substitution Cipher (1/2)

- Substitute cipher is used to encrypt ordinary English text.
- The encryption and decryption rules are all permutations of alphabetic characters.
- Example: if the encryption rule is

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

The decryption rule is

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*
d	*l*	*r*	*y*	*v*	*o*	*h*	*e*	*z*	*x*	*w*	*p*	*t*	*b*	*g*	*f*	*j*	*q*	*n*	*m*	*u*	*s*	*k*	*a*	*c*	*i*

Plaintext: have a nice weekend

Ciphertext: GXEHXSZYHKHWHSZ

Plaintext	V	O	Y	A	G	E	R
Key	+3	+3	+3	+3	+3	+3	+3
Ciphertext	Y	R	B	D	J	H	U
Plaintext	V	O	Y	A	G	E	R
Ciphertext	J	H	K	T	X	N	M

A more complex substitution cipher would be created if, instead of incrementing each character by three, we used a more complex key. This table shows a simple substitution cipher with a key of "123".

Plaintext	V	O	Y	A	G	E	R
Key	+1	+2	+3	+1	+2	+3	+1
Ciphertext	W	Q	B	B	I	H	S

An even more complex substitution cipher can be made by having each character of the alphabet correspond to a different letter of the alphabet, without a set pattern.

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

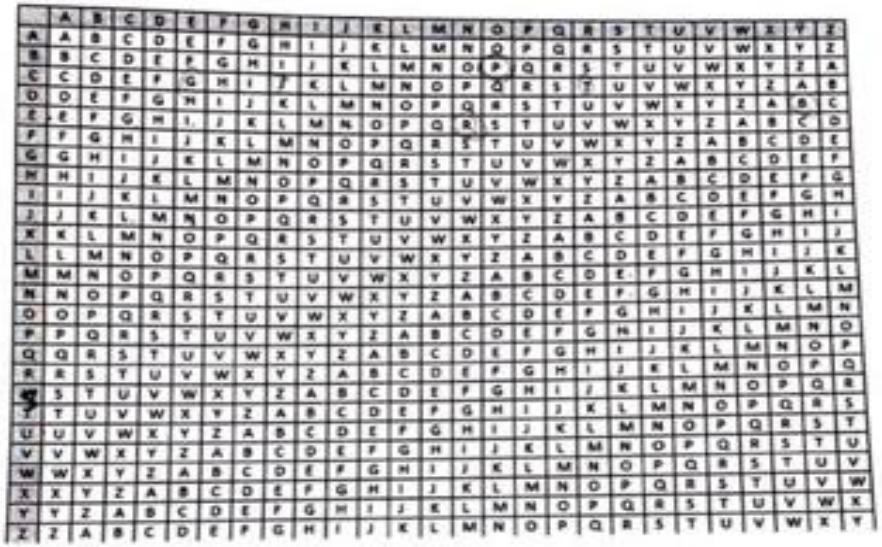
Using this substitution cipher to encrypt VOYAGER would give us these results:

Plaintext	V	O	Y	A	G	E	R
Ciphertext	J	H	K	T	X	N	M

Cipher text: WKLV PHVVDJH ZDV HQFUBSWHG XVLQJ D FDHVDU VKLIW FLSKHU

A.2 Simple polyalphabetic substitution

a) Vigenère cipher: The Vigenère cipher is a method of encrypting alphabetic text by using a series of different Caesar ciphers based on the letters of a keyword. It is a simple form of polyalphabetic substitution.



Vigenere square

Example:

Plain text: HELLO EVERYONE

Key: CDBE

The key is composed of four letters, then we subdivide the plain text into blocks composed of 4 letters: M1M2M3M4 with M1=HELL M2=OEVE M3=RYON M4=E

In each block, the first letter is substitued according to the key K1=C i.e. 2, the second letter according to K2=D i.e. 3, the third letter according to K3=B i.e. 1 and the last one according to k4=E i.e. 4.

Accordind to the Vigenere square table, we obtain:

Cipher text: JHMPQ HWITBPRG

b) One time Pad : the one-time pad (OTP) is a type of encryption which has been proven to be impossible to crack if used correctly. Each bit or character from the plaintext is encrypted by a modular addition with a bit or character from a secret random key (or pad) of the same length as the plaintext, resulting in a ciphertext. If the key is truly random, as large as or greater than the plaintext, never reused in whole or part, and kept secret, the ciphertext will be impossible to decrypt or break without knowing the key. However, practical problems have prevented one-time pads from being widely used.

Example:

Plaintext: HELLO

Key: XMCKL

Ciphertext: EQNVZ

B. Transposition cipher:

In a transposition cipher, the units (letters, bits, numbers, etc) of the plaintext are rearranged in a different and usually quite complex order, but the units themselves are left unchanged.

B.1 Simple transposition

Example:

Plain text: HELLO EVERYONE

key:CBVA

The size of the key is 4, this means that we subdivise the plain text into blocks of size 4. Alphabetic order of the key: ABCV, this means that, for each block, **the first letter in the plaintext will be transposed to the letter number 4, the second letter will remain at the same position, the third letter will be transposed to the first one and the last one will be in the position number 3**: ABCV→CBVA. Then the cipher text matching our example will be:

1 2 3 4 3 2 4 1

Cipher text: **LELH VEEO OYNRE**

.2 Columnar transposition

In a columnar transposition, the message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order. Both the width of the rows and the permutation of the columns are usually defined by a keyword. For example, the word ZEBRAS is of length 6 (so the rows are of length 6), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be "632415".

ZEBRAS → ABERSZ

632415 → 632415

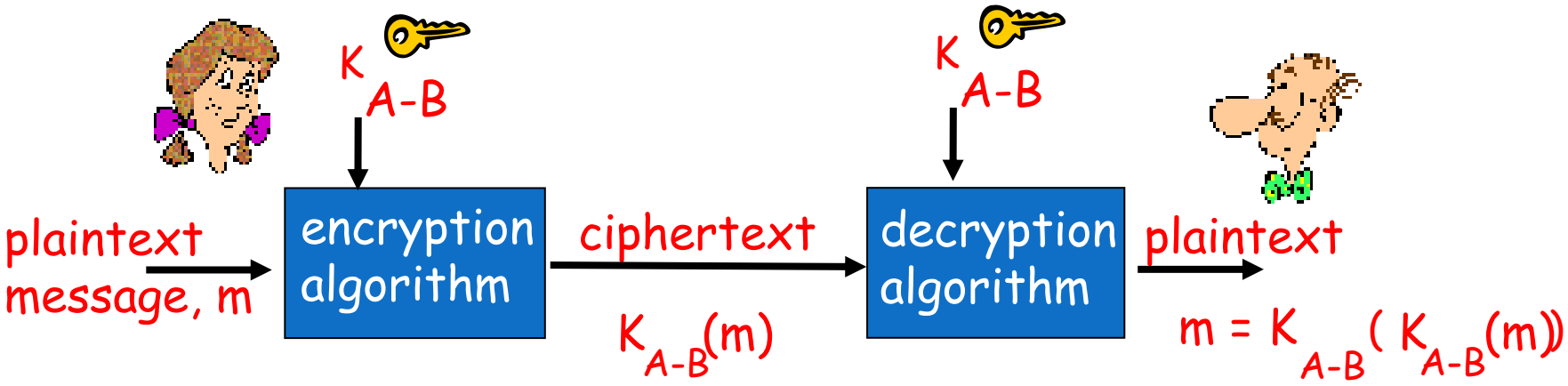
Example:

Plaintext: WE ARE DISCOVERED. FLEE AT ONCE

keyword: ZEBRAS

Ciphertext: EVLNACDTESEAROFODEECWIREE

Symmetric key cryptography



symmetric key crypto: Bob and Alice share know same (symmetric) key: K_{A-B}

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher
- Q: how do Bob and Alice agree on key value?

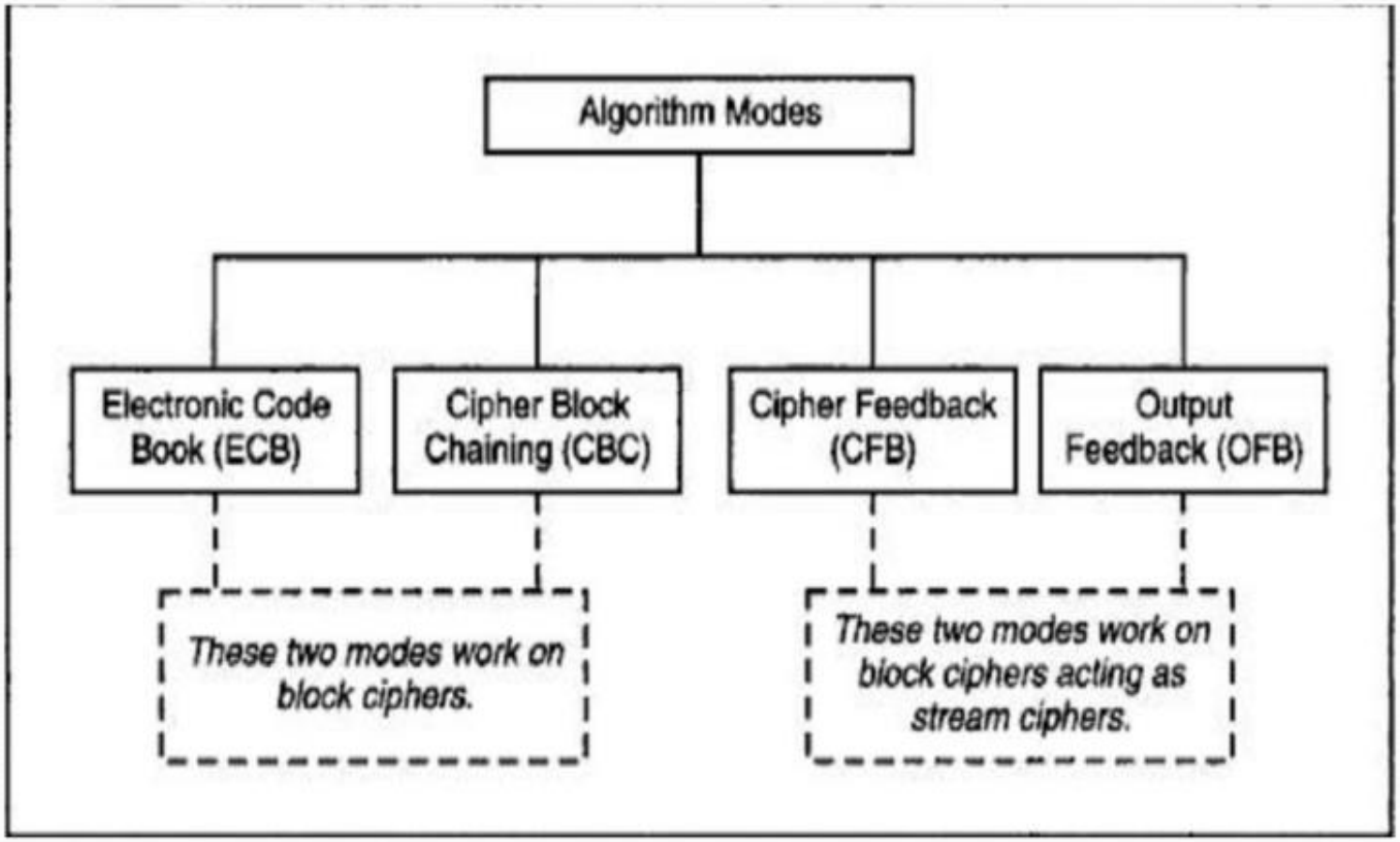
Symmetric key crypto: DES

DES: Data Encryption Standard

- Use encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- How secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase ("Strong cryptography makes the world a safer place") decrypted (brute force) in 4 months
- making DES more secure:
  - use three keys sequentially (3-DES) on each datum
  - use cipher-block chaining

[NIST: National Institute of Standards and Technology]

Algorithm Modes



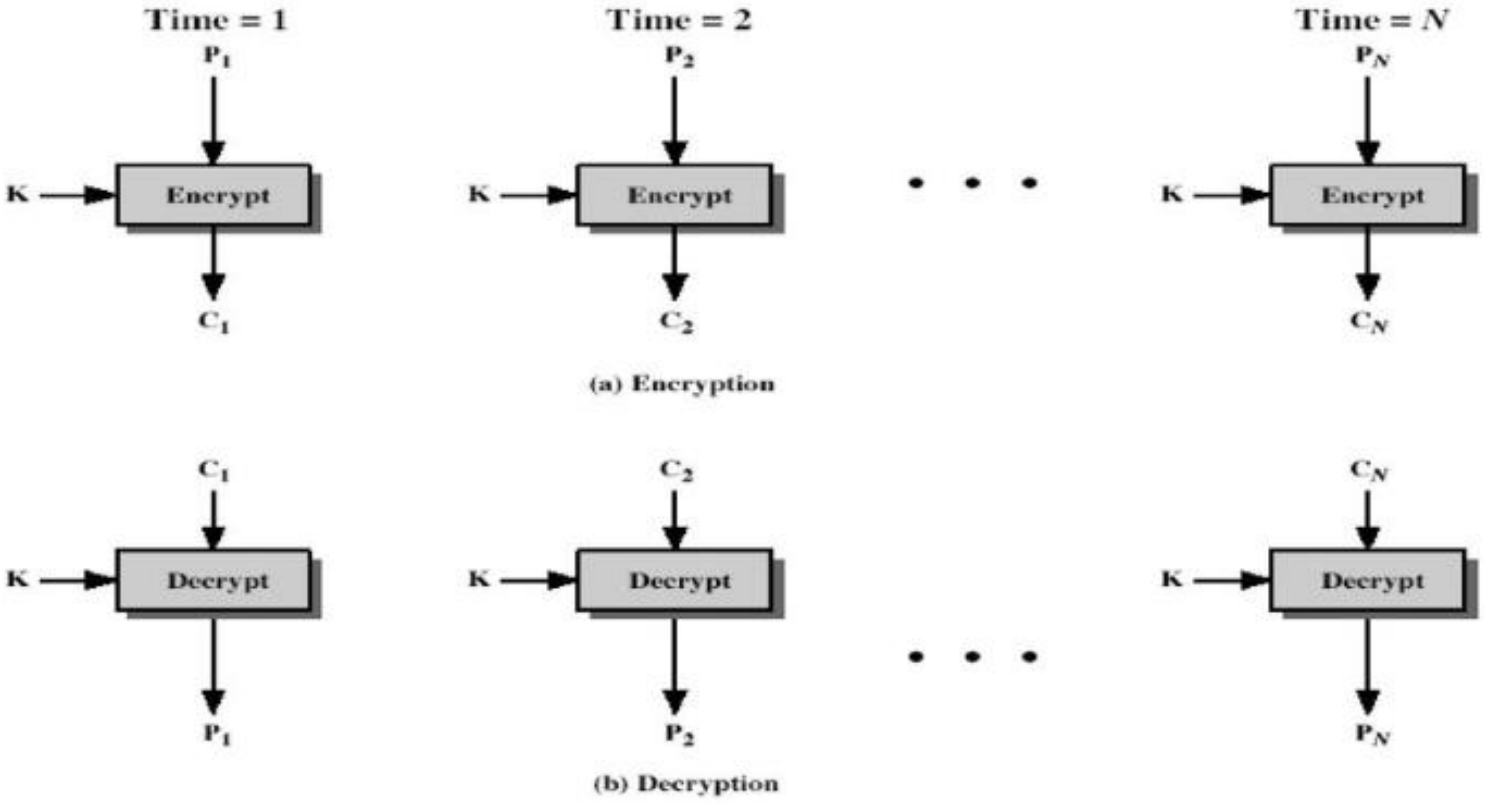
Electronic Codebook Book (ECB)

- Message is broken into independent blocks which are encrypted
- *Each block is a value which is substituted, like a codebook, hence name*
- Each block is encoded independently of the other blocks

$$C_i = {}_{K1}(P_i)$$

- Uses: transmission of single values.(i.e;-PW or key for E/D) in secure fashion.
  - E=Encryption , D= Decryption

Electronic Codebook Book (ECB)



Cipher Block Chaining (CBC)

- Message is broken into blocks
- But these are linked together in the encryption operation
- *Each previous cipher blocks is chained with curren plaintext block, hence name*
- Use Initial Vector (IV) to start process (Block 1)

$$C_{-1} = IV$$

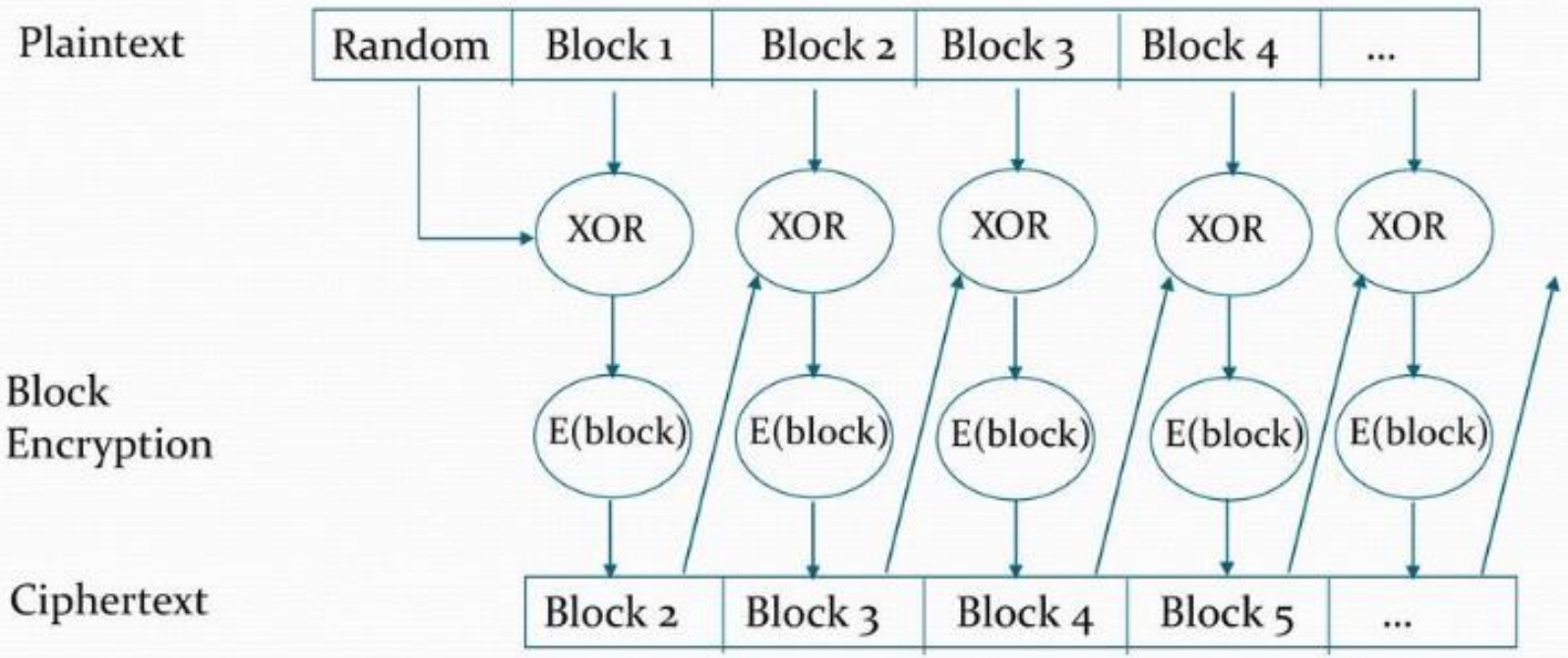
$$C_i = {}_{K1}(P_i \text{ XOR } IV)$$

- From block 2

$$C_i = {}_{K1}(P_i \text{ XOR } C_{i-1})$$

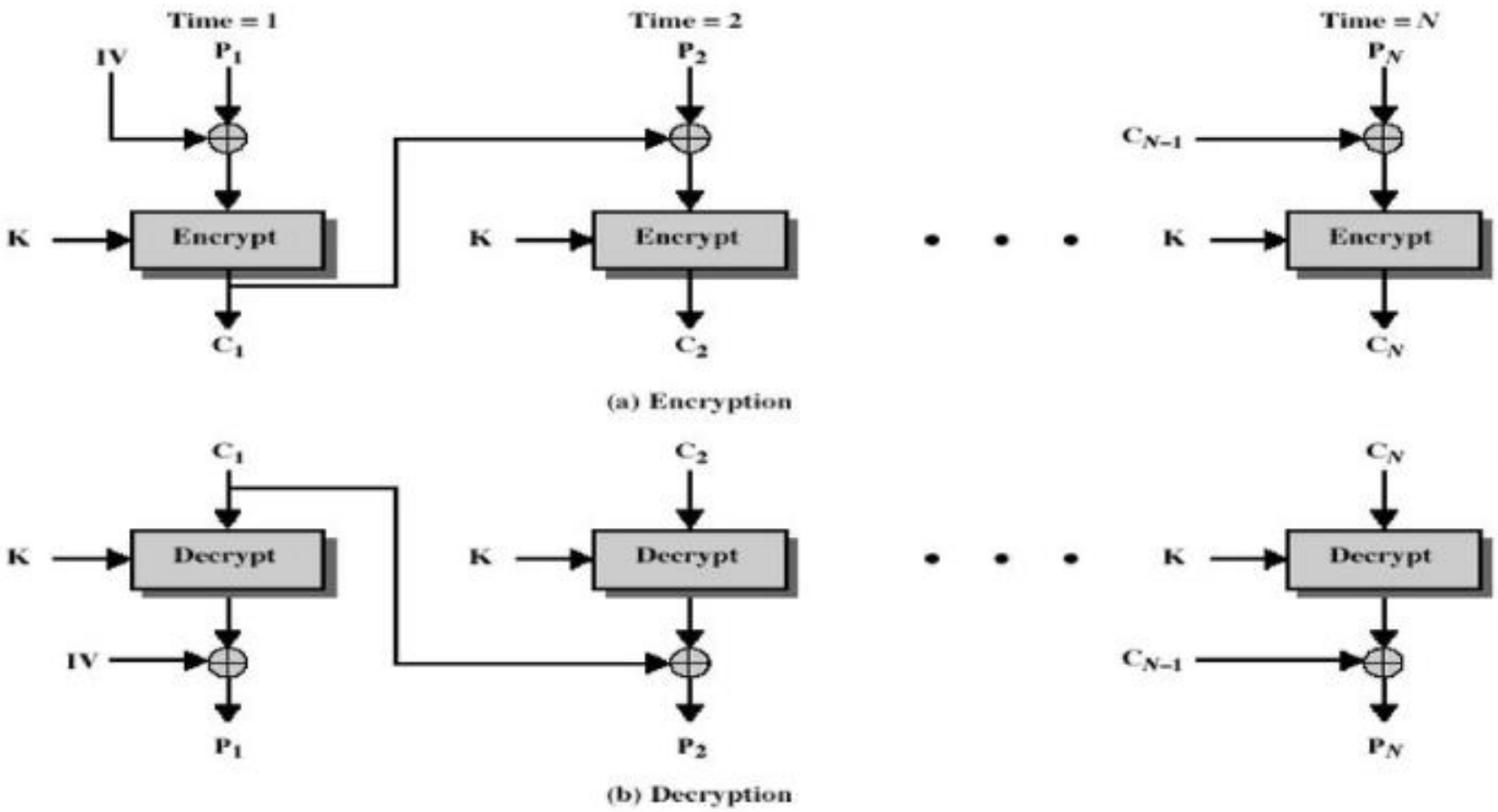
- Uses: bulk data encryption, authentication

Cipher Block Chaining (CBC) Mode



- Pad last block, if necessary
-

Cipher Block Chaining (CBC)



Cipher Feed Back (CFB)

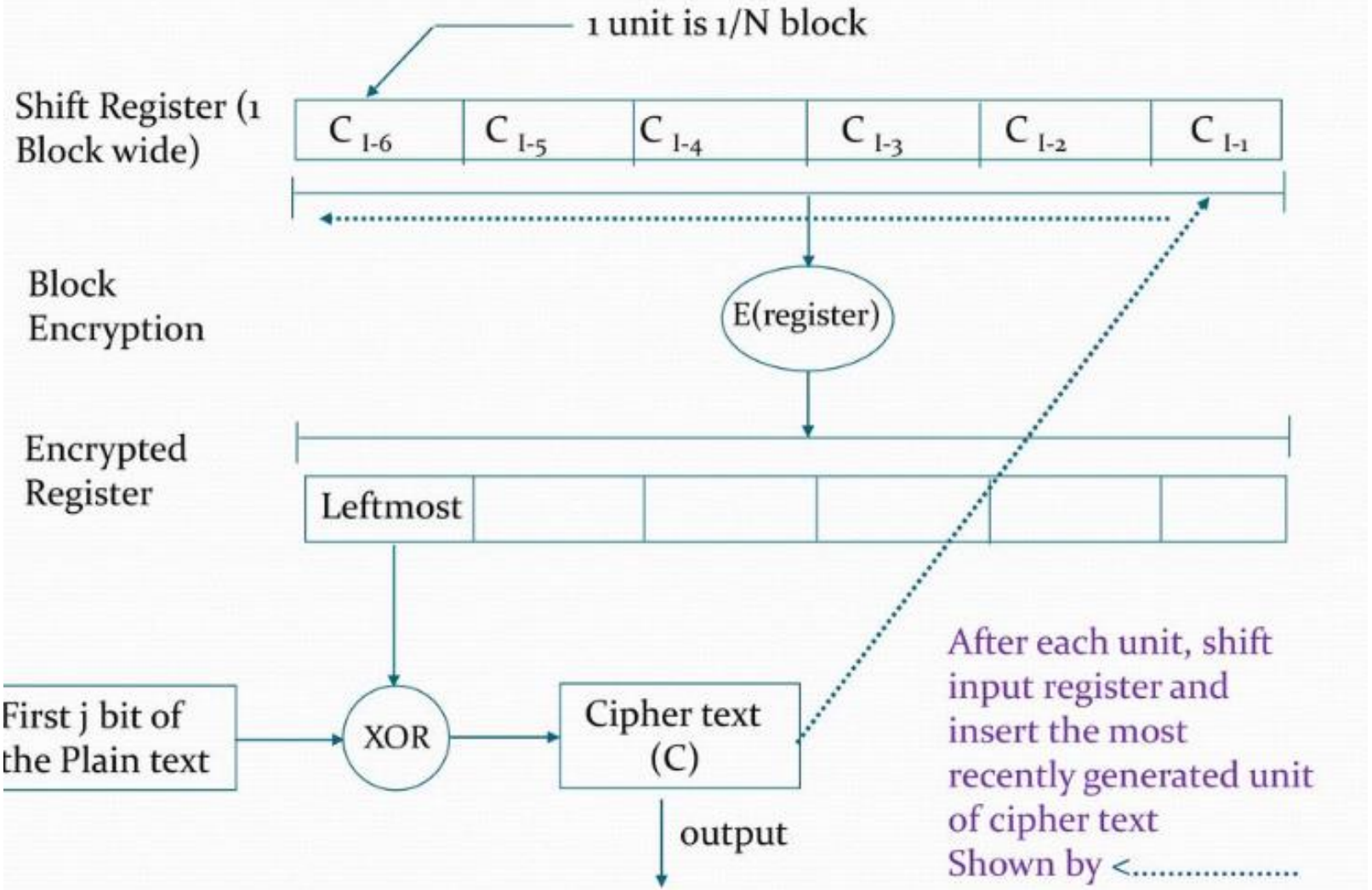
- Message is treated as a stream of bits
- Added to the output of the block cipher
- Result is feed back for next stage (hence name)
- Standard allows any number of bit (1,8 or 64 or whatever) to be feed back
  - denoted CFB-1, CFB-8, CFB-64 etc
- Is most efficient to use all 64 bits (CFB-64)

$$C_i = P_i \text{ XOR } _{K1}(C_{i-1})$$

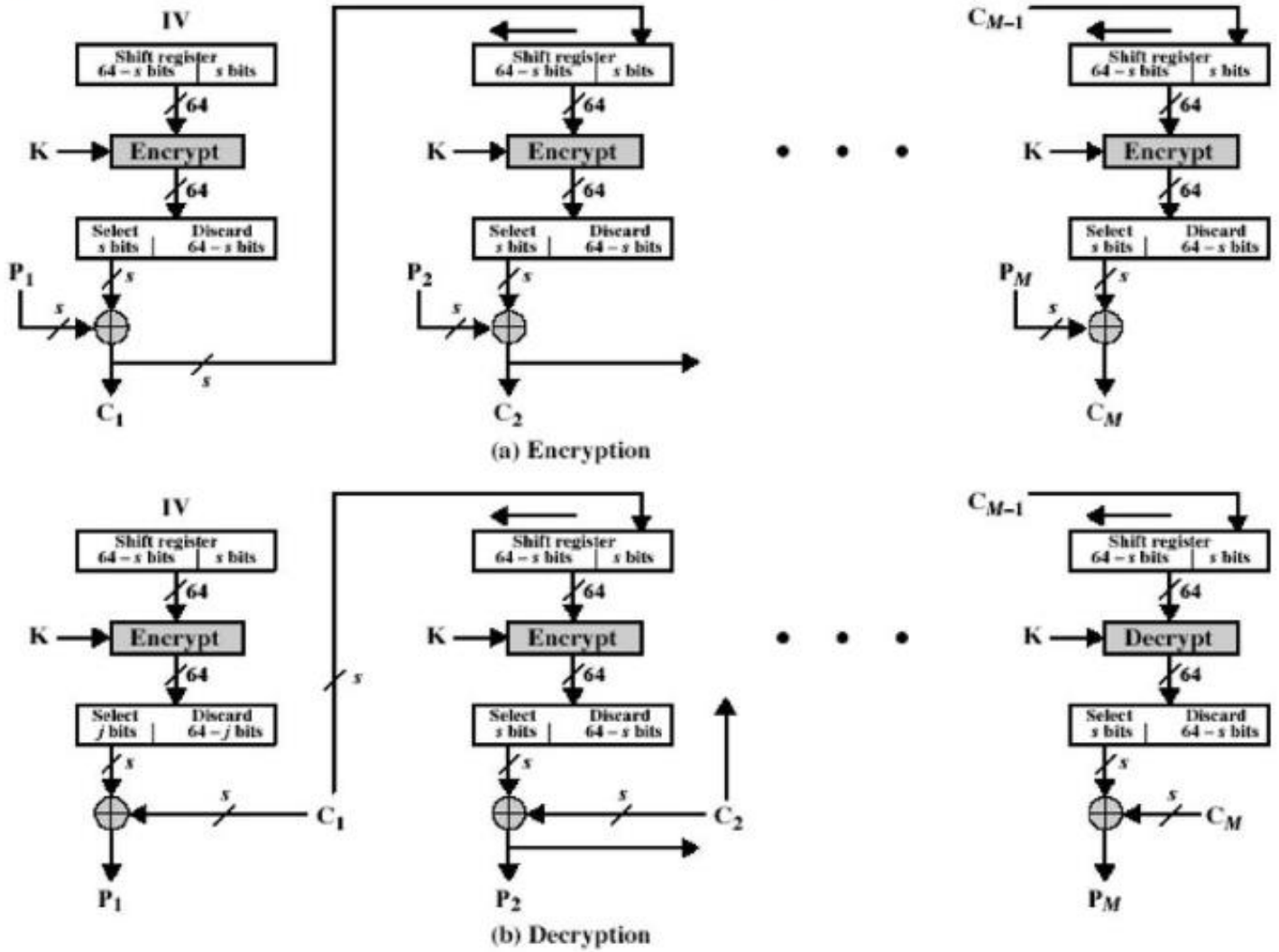
$$C_{-1} = IV$$

- Uses: stream data encryption, authentication

Cipher Feedback Mode (CFB)



Cipher FeedBack (CFB)



Find the limitations of each Algo?

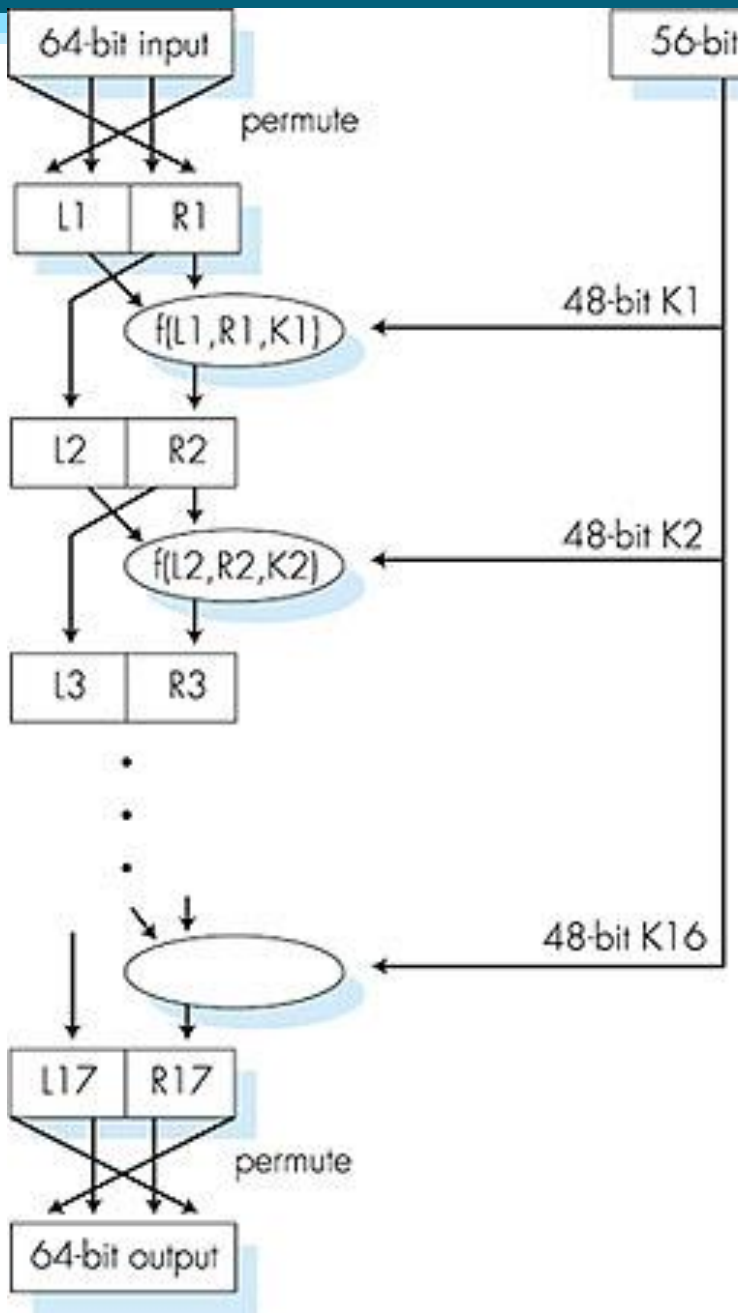
Symmetric key crypto: DES

DES operation

initial permutation

16 identical "rounds" of function application, each using different 48 bits of key

final permutation



Public key cryptography

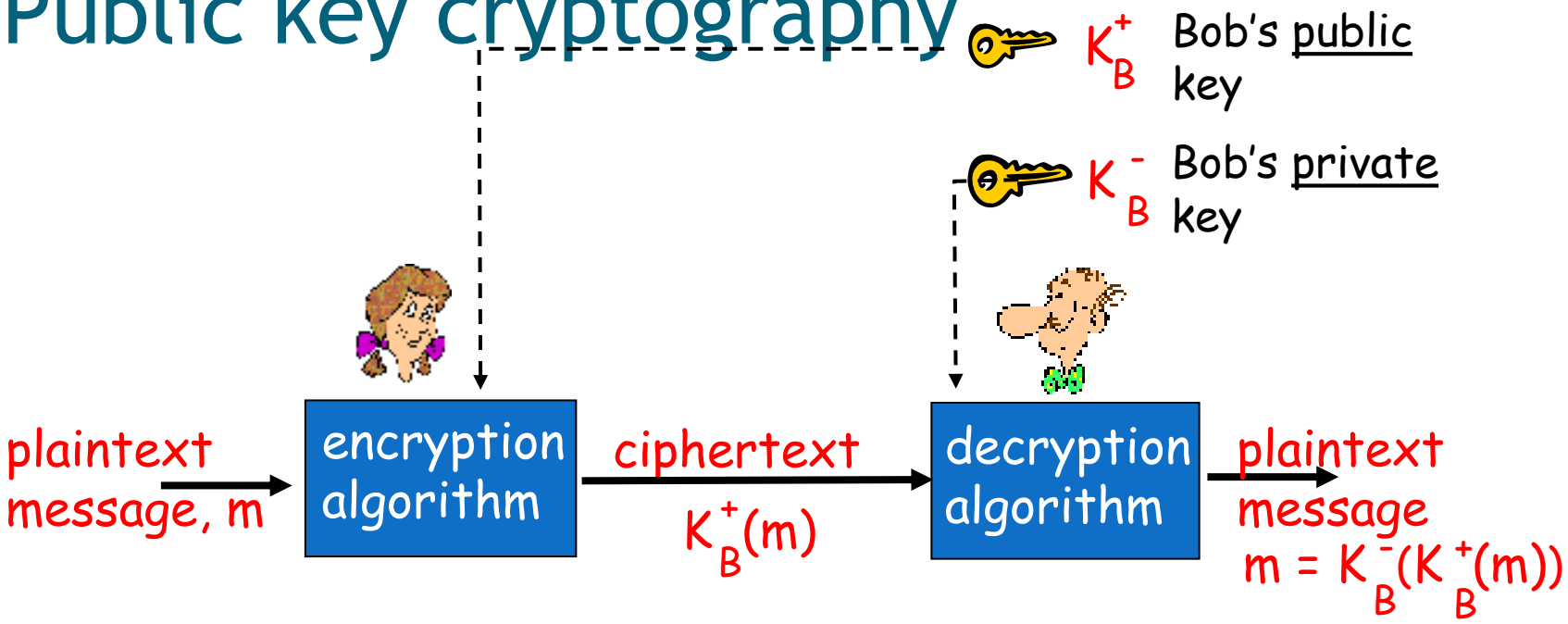
symmetric key crypto

- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?

public key cryptography

- radically different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver

Public key cryptography



Public key encryption algorithms

Requirements:

1. need $K_B^+(\bullet)$ and $K_B^-(\bullet)$ such that

$$K_B^-(K_B^+(m)) = m$$

2. given public key K_B^+, it should be impossible to compute private key K_B^-

RSA: Rivest, Shamir, Adleman algorithm

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)

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)

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 $\phi(n)=(p-1)(q-1)$
- selecting at random the encryption key e
  - where $1<e<\phi(n)$, $\gcd(e,\phi(n))=1$
- solve following equation to find decryption key d
  - $e.d=1 \bmod \phi(n)$ and $0 \le d \le n$
- publish their public encryption key: PU={e,n}
- keep secret private decryption key: PR={d,n}

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:

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

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

- a public-key distribution scheme
  - cannot be used to exchange an arbitrary message
  - rather it can establish a common key
  - known only to the two participants
- value of key depends on the participants (and their private and public key information)
- based on exponentiation in a finite (Galois) field (modulo a prime or a polynomial) - easy
- security relies on the difficulty of computing discrete logarithms (similar to

Diffie-Hellman Setup

- all users agree on global parameters:
  - large prime integer or polynomial q
  - a being a primitive root mod q
- each user (eg. A) generates their key
  - chooses a secret key (number): $x_A < q$
  - compute their **public key**: $y_A = a^{x_A} \bmod q$
- each user makes public that key y_A

Diffie-Hellman Key Exchange

- shared session key for users A & B is K_{AB}:

$$K_{AB} = a^{x_A . x_B} \bmod q$$
$$= y_A^{x_B} \bmod q \quad \text{(which } \mathbf{B} \text{ can compute)}$$
$$= y_B^{x_A} \bmod q \quad \text{(which } \mathbf{A} \text{ can compute)}$$

- K_{AB} is used as session key in private-key encryption scheme between Alice and Bob
- if Alice and Bob subsequently communicate, they will have the **same** key as before, unless they choose new public-keys
- attacker needs an x, must solve discrete log

Diffie-Hellman Example

- users Alice & Bob who wish to swap keys:
- agree on prime $q=353$ and $a=3$
- select random secret keys:
  - A chooses $x_A=97$, B chooses $x_B=233$
- compute respective public keys:
  - $y_A=3^{97} \mod 353 = 40$ (Alice)
  - $y_B=3^{233} \mod 353 = 248$ (Bob)
- compute shared session key as:
  - $K_{AB}= y_B^{x_A} \mod 353 = 248^{97} = 160$ (Alice)
  - $K_{AB}= y_A^{x_B} \mod 353 = 40^{233} = 160$ (Bob)

RSA: Choosing keys

1. Choose two large prime numbers p, q. (e.g., 1024 bits each)
2. Compute $n = pq$, $z = (p-1)(q-1)$
3. Choose e (with $e<n$) that has no common factors with z. (e, z are "relatively prime").
4. Choose d such that $ed-1$ is exactly divisible by z. (in other words: $ed \bmod z = 1$).
5. *Public* key is $\underbrace{(n,e)}_{K_B^+}$. *Private* key is $\underbrace{(n,d)}_{K_B^-}$.

RSA: Encryption, decryption

0. Given (n,e) and (n,d) as computed above

1. To encrypt bit pattern, m, compute

   $c = m^e \bmod n$ (i.e., remainder when m^e is divided by n)

2. To decrypt received bit pattern, c, compute

   $m = c^d \bmod n$ (i.e., remainder when c^d is divided by n)

Magic happens!

$$m = \underbrace{(m^e \bmod n)}_{c}{}^d \bmod n$$

RSA example:

Bob chooses *p=5, q=7*. Then *n=35, z=24 (euler totient)*.

e=5 (so *e, z* relatively prime).
d=29 (so *ed-1* exactly divisible by z.

(e*d)mod(z)=1

encrypt:

letter	m	m^e	$c = m^e \bmod n$
l	12	248832	17

decrypt:

c	c^d	$m = c^d \bmod n$	letter
17	48196857210675091509141182522307169 7	12	l

RSA: Why is that $m = (m^e \bmod n)^d \bmod n$

Useful number theory result: If p,q prime and $n = pq$, then:

$$x^y \bmod n = x^{y \bmod (p-1)(q-1)} \bmod n$$

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

$$= m^{ed \bmod (p-1)(q-1)} \bmod n$$

(using number theory result above)

$$= m^1 \bmod n$$

(since we chose ed to be divisible by $(p-1)(q-1)$ with remainder 1)

$$= m$$

RSA: another important property

The following property will be *very* useful later:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

Result is the same!