COMP 3331/9331: Computer Networks and Applications

Week 9

Network Security

Reading Guide: Chapter 8: 8.1 – 8.2

Network Security: Overview

Our goals:

- understand principles of network security:
  - cryptography and its *many* uses beyond "confidentiality"
  - authentication
  - message integrity

Network Security: roadmap

8.1 What is network security?

8.2 Principles of cryptography

8.3 Message integrity

8.4 Authentication

8.5 Securing email

8.6 - 8.9 SSL, IPSec, Firewall/IDS not covered.

There are several security electives offered

What is network security?

confidentiality: only sender, intended receiver should "understand" message contents

- sender encrypts message
- receiver decrypts message

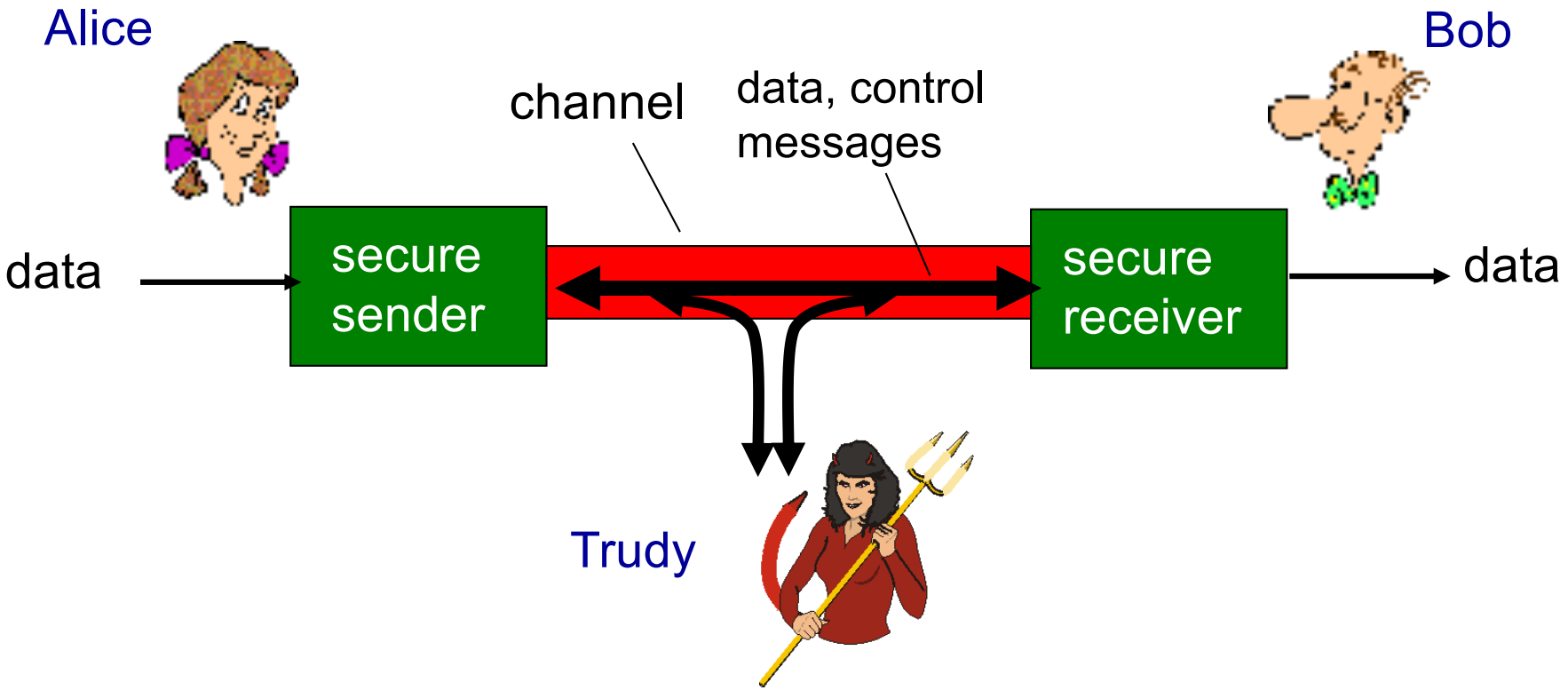
authentication: sender, receiver want to confirm identity of each other

message integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

access and availability: services must be accessible and available to users

Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob, Alice (lovers!) want to communicate "securely"
- Trudy (intruder) may intercept, delete, add messages



Who might Bob, Alice be?

- … well, *real-life* Bobs and Alices!
- Web browser/server for electronic transactions (e.g., on-line purchases)
- on-line banking client/server
- DNS servers
- routers exchanging routing table updates
- etc.

There are bad guys (and girls) out there!

Q: What can a "bad guy" do?

A: A lot!

- *eavesdrop:* intercept messages
- actively *insert* messages into connection
- *impersonation:* can fake (spoof) source address in packet (or any field in packet)
- *hijacking:* "take over" ongoing connection by removing sender or receiver, inserting himself in place
- *denial of service:* prevent service from being used by others (e.g., by overloading resources)

Network Security: roadmap

8.1 What is network security?

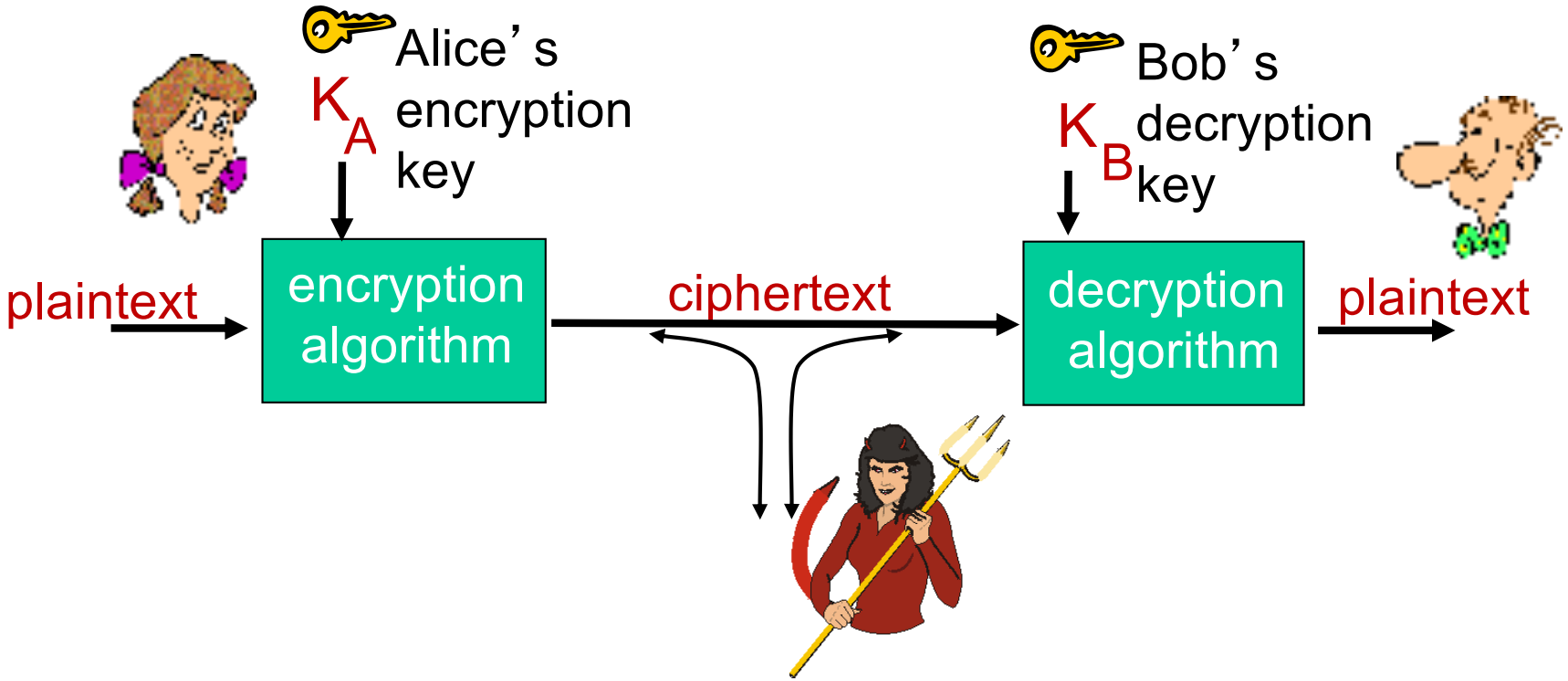
8.2 Principles of cryptography

8.3 Message integrity

8.4 Authentication

8.5 Securing email

The language of cryptography

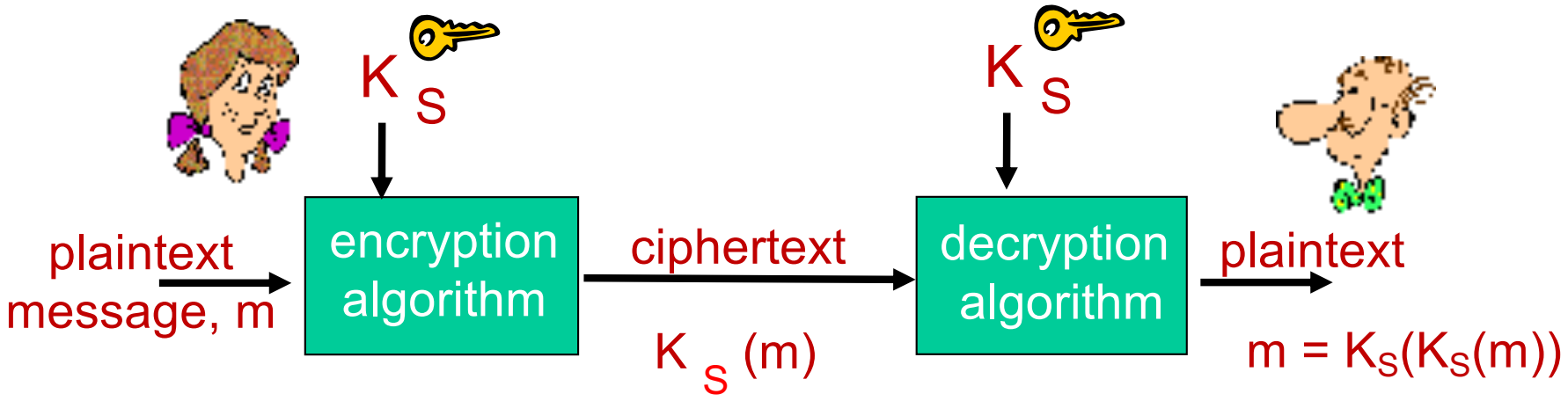


m plaintext message

$K_A(m)$ ciphertext, encrypted with key K_A

$m = K_B(K_A(m))$

Symmetric key cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: K_S

Q: how do Bob and Alice agree on key value?

Simple encryption scheme

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another
- Ceaser Cipher: replace each letter of the alphabet with the letter standing three places further down the alphabet.

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: 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.g.: **Plaintext: meet me after the party**

ciphertext: phhw ph diwhu wkh sduwb

Encryption key: c = (p+3) mod 26

Each plaintext letter p substituted by the ciphertext letter c
In general, we have c = (p+k) mode 26
where k is in range 1 to 25

Simple encryption scheme

- With only 25 possible keys, the Caeser cipher is vulnerable to brute force cryptanalysis
- Cipher can be any permutation of the 26 alphabet characters

```
plaintext:  abcdefghijklmnopqrstuvwxyz
ciphertext: mnbvcxzasdfghjklpoiuytrewq
```

e.g.:

```
Plaintext: bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc
```

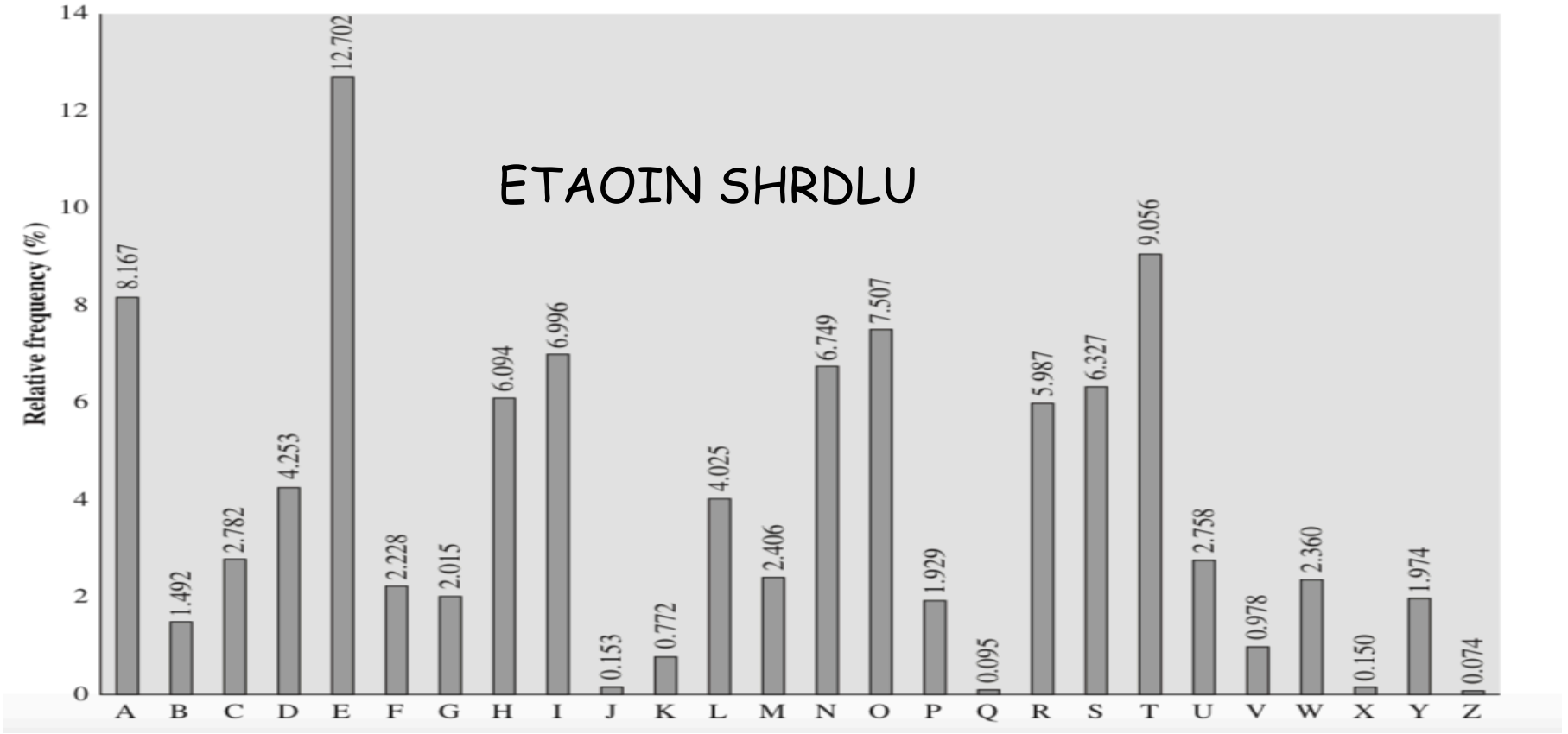
Encryption key: mapping from set of 26 letters to set of 26 letters

We have 26! (> 4 x10^{26}) possible keys

Breaking an encryption scheme

- cipher-text only attack:
  Trudy has ciphertext she can analyze
- two approaches:
  - brute force: search through all keys
  - statistical analysis
- known-plaintext attack:
  Trudy has (part of) plaintext corresponding to ciphertext
  - e.g., in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,b
- chosen-plaintext attack:
  Trudy can get ciphertext for chosen plaintext

Breaking an encryption scheme



Frequency Histogram Analysis for letters in English language

Monoalphabetic ciphers are easy to break because they reflect the frequency data of the original alphabet

A more sophisticated encryption approach

- Polyalphabet ciphers
- n substitution ciphers, $M_1,M_2,\ldots,M_n$
- cycling pattern:
  - e.g., n=4: and key is M_1,M_3,M_4,M_3,M_2; M_1,M_3,M_4,M_3,M_2; ..
- for each new plaintext symbol, use subsequent subsitution pattern in cyclic pattern
  - dog: d from M_1, o from M_3, g from M_4

Encryption key: n substitution ciphers, and cyclic pattern

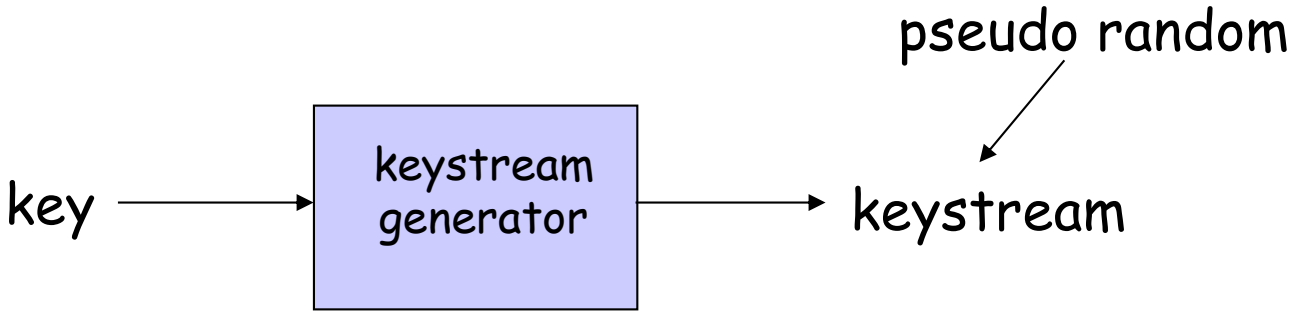
A more sophisticated encryption approach

- Vigen`ere Cipher
- 26 substitution ciphers with shifts of 0 through 25
- cycling pattern:
  - Assume Key consist of m sequence of letters
  - $(p_0 + k_0)$ mode 26, $(p_1 + k_1)$mode 26, …, $(p_{m-1} + k_{m-1})$ mode 26, $(p_m + k_0)$mode 26, $(p_{m+1} + k_1)$mode 26, …, $(p_{2m-1} + k_{m-1})$ mode 26, …
  - $(p_i + k_{i \bmod m})$ mod 26
- There are multiple ciphertext letters for each plaintext letter, one for each unique letter of the keyword

Two types of symmetric ciphers

- **Stream ciphers**
  - encrypt one bit at time
- **Block ciphers**
  - Break plaintext message in equal-size blocks
  - Encrypt each block as a unit

Stream Ciphers



- Combine each bit of keystream with bit of plaintext to get bit of ciphertext
- m(i) = ith bit of message
- ks(i) = ith bit of keystream
- c(i) = ith bit of ciphertext
- $c(i) = ks(i) \oplus m(i)$ ($\oplus$ = exclusive or)
- $m(i) = ks(i) \oplus c(i)$

RC4 Stream Cipher

- RC4 is a popular stream cipher
  - Extensively analyzed and considered good
  - Key can be from 1 to 256 bytes
  - Used in WEP for 802.11

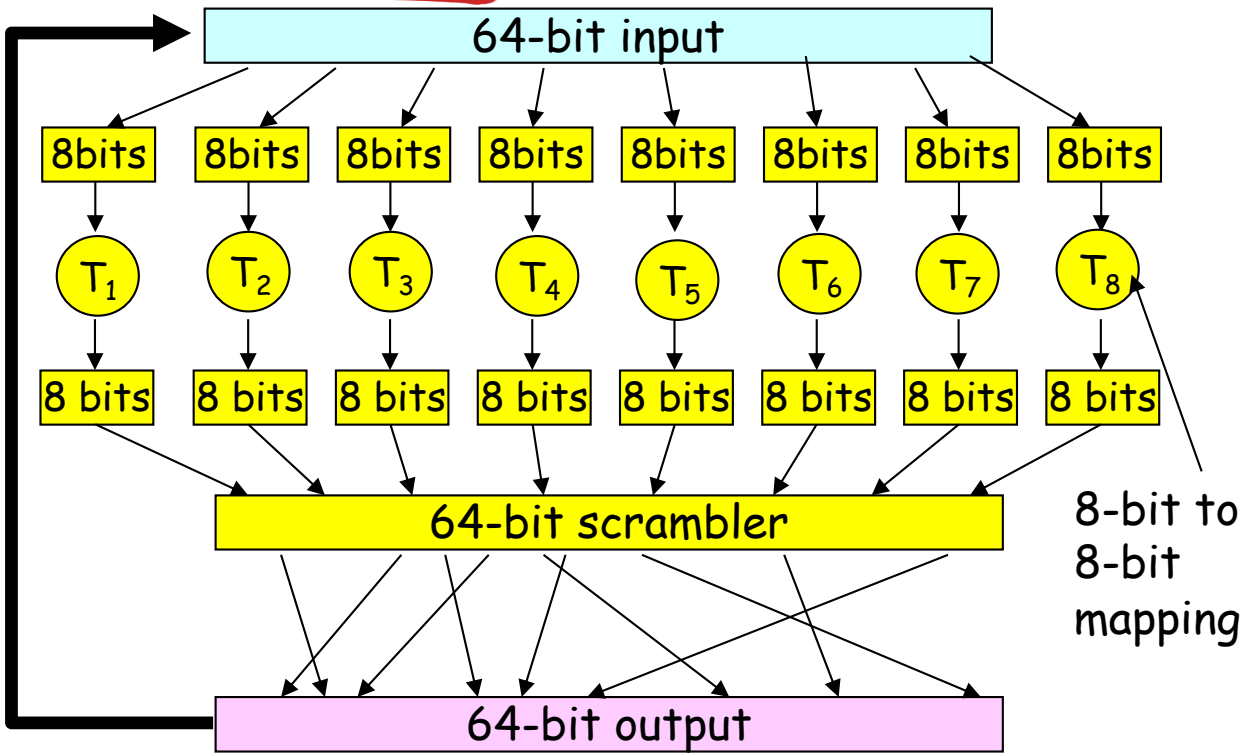
Block Cipher

- Ciphertext processed as *k* bit blocks
- 1-to-1 mapping is used to map k-bit block of plaintext to k-bit block of ciphertext
- E.g: k=3 (see table)
  - 010110001111 => 101000111001
- Possible permutations = 8! (40,320)
- To prevent brute force attacks
  - Choose large K (64, 128, etc)
- Full-table block ciphers not scalable
  - E.g., for k = 64, a table with 2^{64} entries required
  - instead use function that simulates a randomly permuted table

Input	Output
000	110
111	001
001	111
010	101
011	100
100	011
101	010
110	000

From Kaufman et al

Block Cipher (contd.)



- If only a single round, then one bit of input affects at most 8 bits of output
- In 2nd round, the 8 affected bits get scattered and inputted into multiple substitution boxes
- How many rounds?
  - How many times do you need to shuffle cards
  - Becomes less efficient as n increases

Symmetric key crypto: DES

DES: Data Encryption Standard

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- block cipher with cipher block chaining
- how secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day using distributed computing
  - no known good analytic attack
- making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys

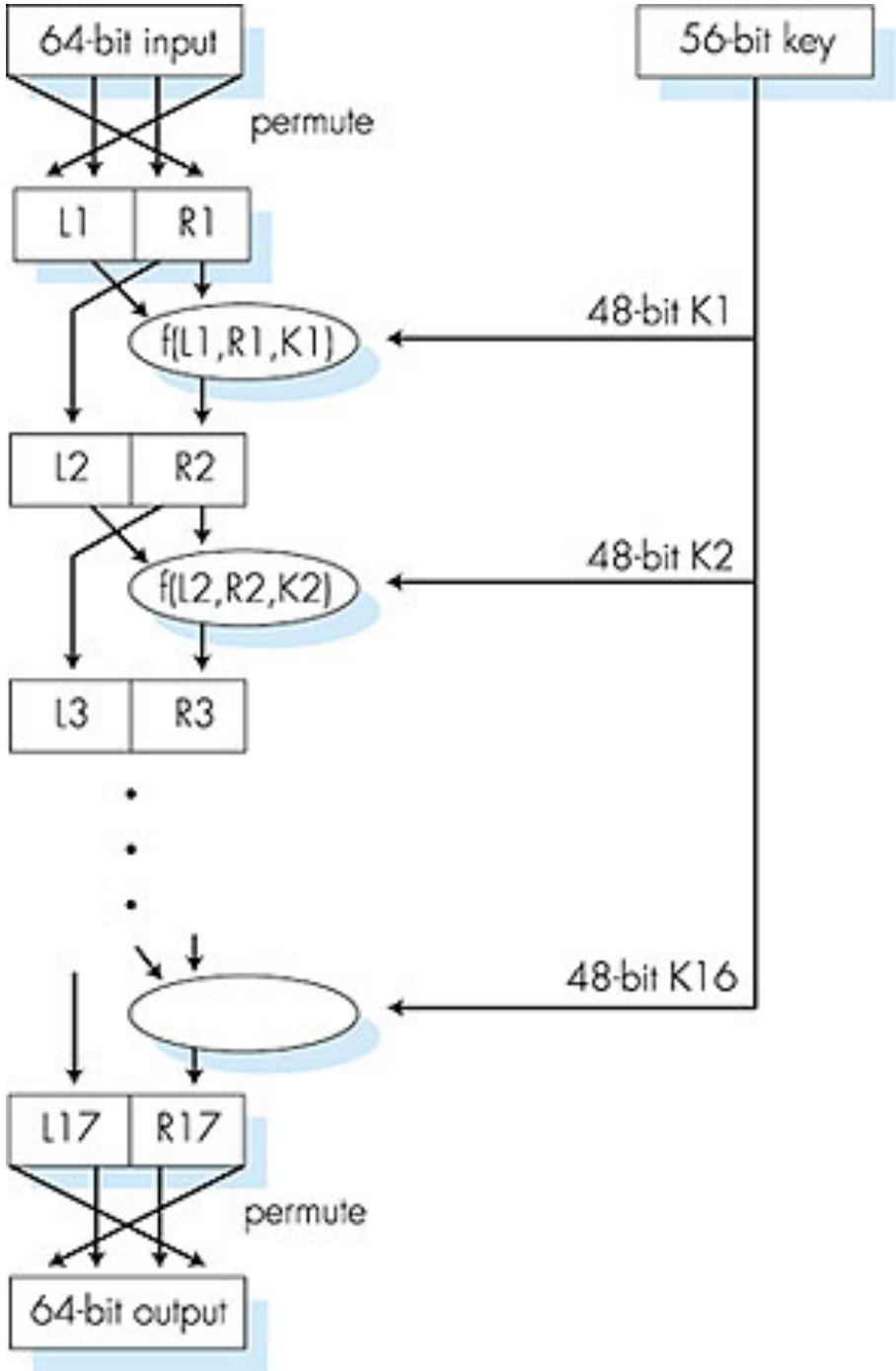
Symmetric key crypto: DES

DES operation

initial permutation

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

final permutation

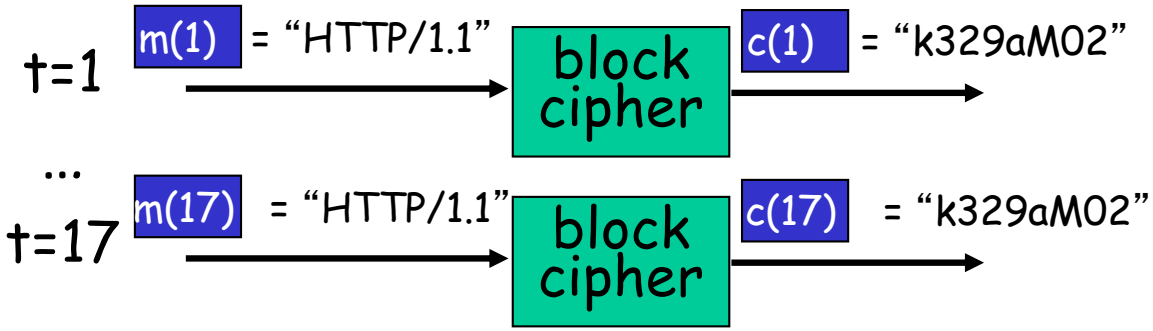


AES: Advanced Encryption Standard

- symmetric-key NIST standard, replaced DES (Nov 2001)
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

Cipher Block Chaining

- cipher block: if input block repeated, will produce same cipher text:



- *Use random numbers:* XOR ith input block, m(i) and random number r(i) and apply block-cipher encryption algorithm
  - $C(i) = Ks(m(i) \oplus r(i))$
  - Send across c(i) and r(i)

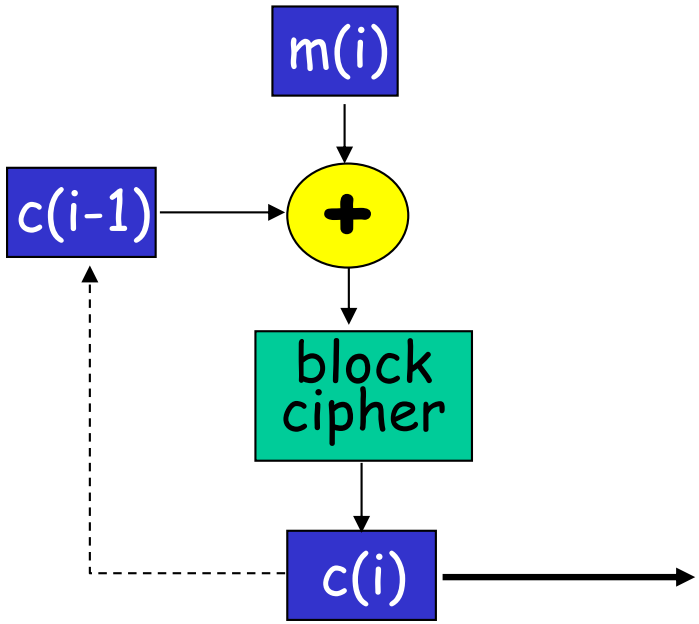
CBC Example

- Plaintext: 010 010 010
- If no CBC, sent txt : 101 101 101
  - 1-to-1 mapping table used
- Lets use the following random bits
  - r1: 001, r2: 111, r3: 100
  - XoR the plaintext with these random bits
  - 010 XoR 001 = 011
  - Now do table lookup for 011 -> 100
- We get c(1)=100, c(2)=010 and c(3)=000, although plaintext is the same (010)
- Need to transmit twice as many bits (c(i) as well as r(i))

Input	Output
000	110
111	001
001	111
010	101
011	100
100	011
101	010
110	000

Cipher Block Chaining

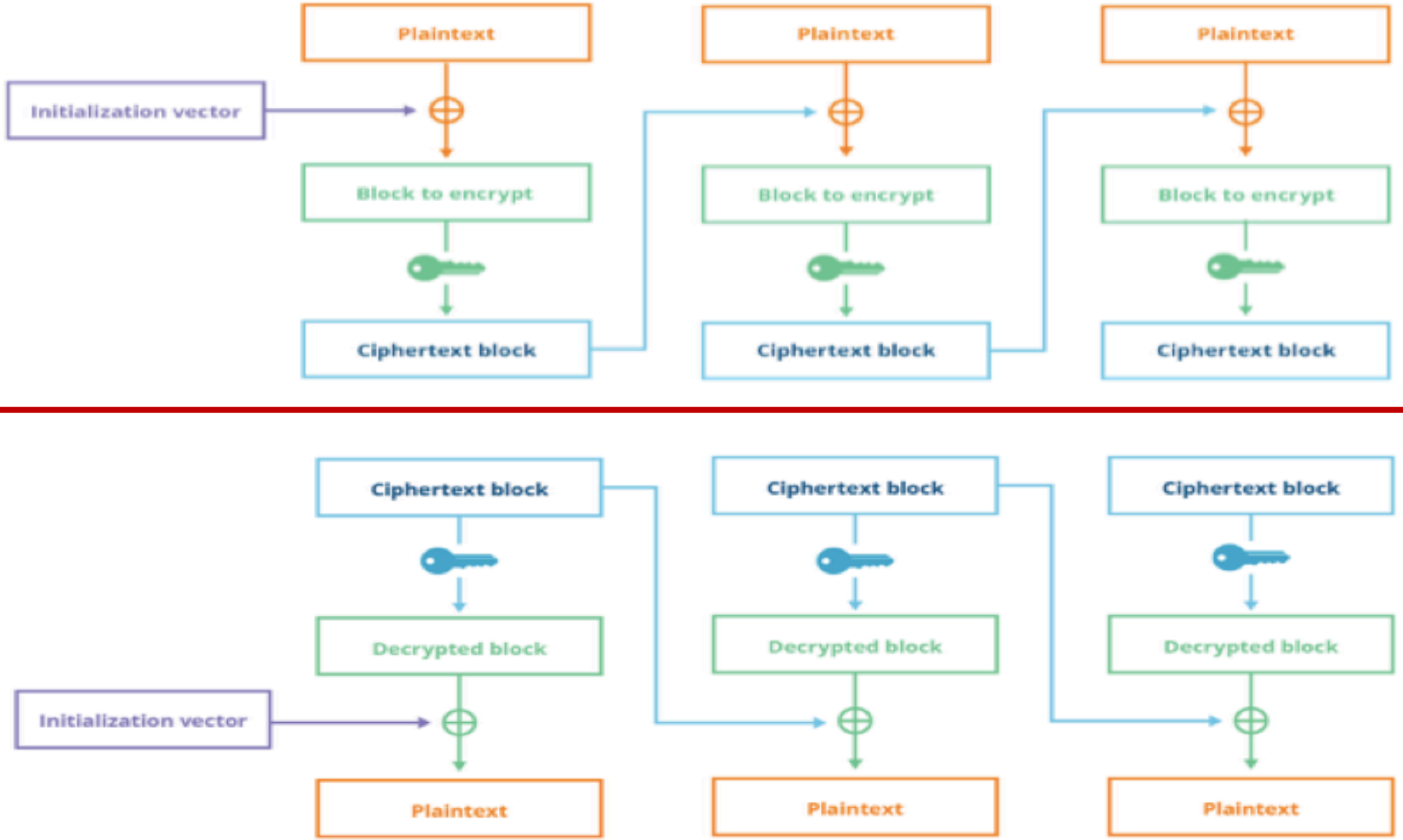
- *cipher block chaining:* send only one random value alongwith the very first message block, and then have the sender and receiver use the computed cipher block in place of the subsequent random number
- XOR ith input block, m(i), with previous block of cipher text, c(i-1)
  - c(0) is an initialisation vector (random) transmitted to receiver in clear



Cipher Block Chaining (CBC)

- CBC generates its own random numbers
  - Have encryption of current block depend on result of previous block
  - $c(i) = K_S(m(i) \oplus c(i-1))$
  - $m(i) = K_S(c(i)) \oplus c(i-1)$
- How do we encrypt first block?
  - Initialization vector (IV): random block = c(0)
  - IV does not have to be secret
- Change IV for each message (or session)
  - Guarantees that even if the same message is sent repeatedly, the ciphertext will be completely different each time

Cipher Block Chaining (CBC)



Public Key Cryptography

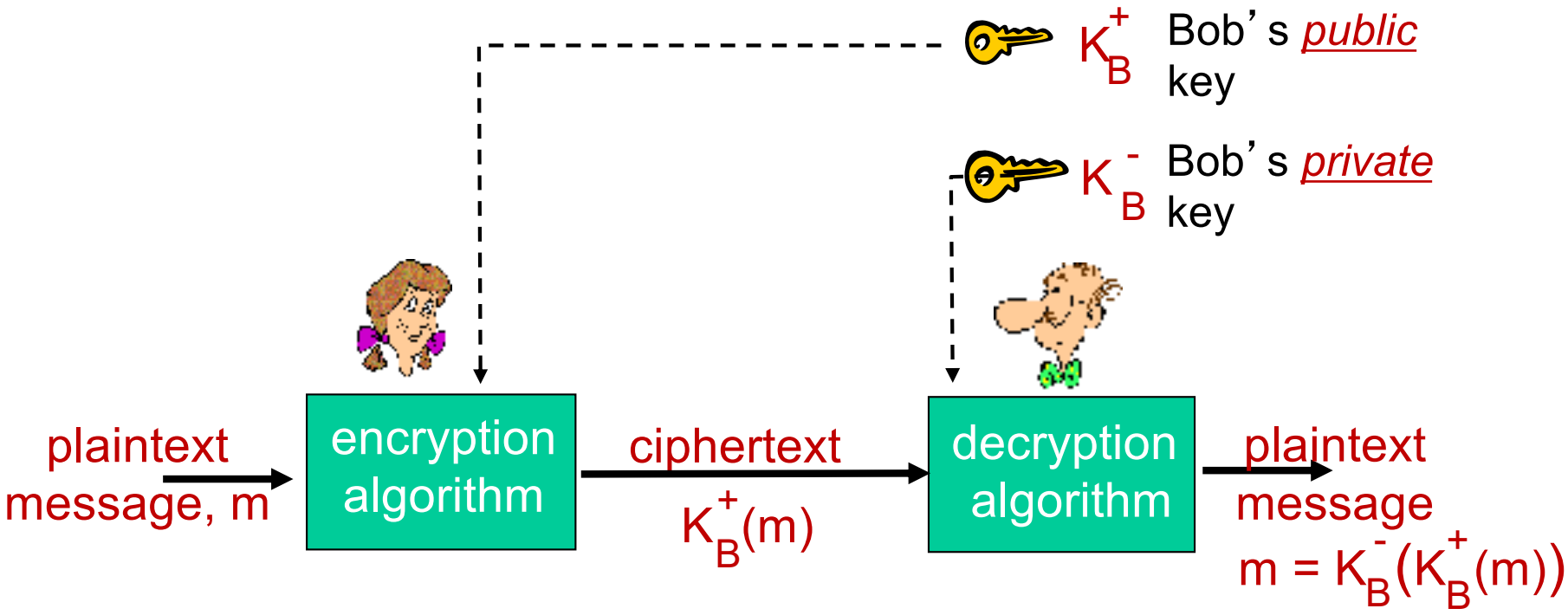
symmetric key crypto

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

public key crypto

- 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 $K_B^+(\cdot)$ and $K_B^-(\cdot)$ 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, Adelson algorithm

Prerequisite: modular arithmetic

- x mod n = remainder of x when divide by n
- facts:

  [(a mod n) + (b mod n)] mod n = (a+b) mod n

  [(a mod n) - (b mod n)] mod n = (a-b) mod n

  [(a mod n) * (b mod n)] mod n = (a*b) mod n
- thus

  $(a \bmod n)^d \bmod n = a^d \bmod n$
- example: x=14, n=10, d=2:

  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$

  $x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$

RSA: getting ready

- message: just a bit pattern
- bit pattern can be uniquely represented by an integer number
- thus, encrypting a message is equivalent to encrypting a number.

example:

- m= 10010001 . This message is uniquely represented by the decimal number 145.
- to encrypt m, we encrypt the corresponding number, which gives a new number (the ciphertext).

RSA: Creating public/private key pair

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* mod *z* = *1*).
5. *public* key is *(n,e)*. *private* key is *(n,d)*.

K_B^+ K_B^-

RSA: encryption, decryption

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

1. to encrypt message m ($<n$), compute

$$c = m^e \bmod n$$

2. to decrypt received bit pattern, c, compute

$$m = c^d \bmod n$$

magic happens!

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

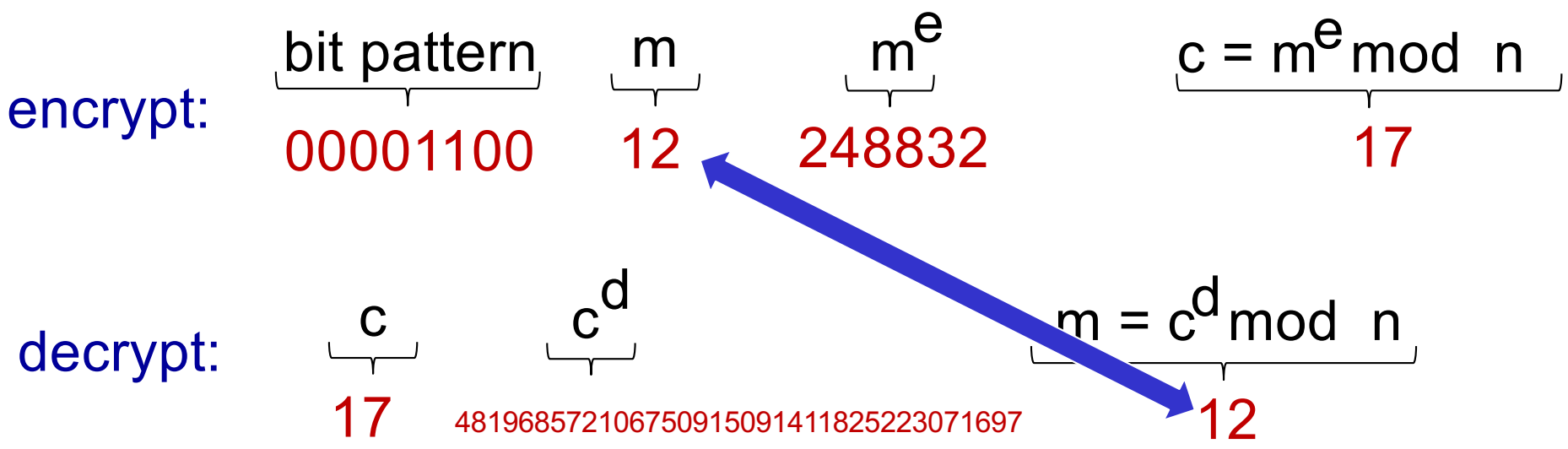
RSA example:

Bob chooses *p=5, q=7*. Then *n=35, z=24*.

e=5 (so *e, z* relatively prime).

d=29 (so *ed-1* exactly divisible by z).

encrypting 8-bit messages.



RSA: another important property

The following property will be *very* useful later:

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

use public key first, followed by private key

use private key first, followed by public key

result is the same!

Why is RSA secure?

- suppose you know Bob's public key (n,e). How hard is it to determine d?
- essentially need to find factors of n without knowing the two factors p and q
  - fact: factoring a big number is hard

RSA in practice: session keys

- exponentiation in RSA is computationally intensive
- DES is at least 100 times faster than RSA
- use public key crypto to establish secure connection, then establish second key – symmetric session key – for encrypting data

session key, K_S

- Bob and Alice use RSA to exchange a symmetric key K_S
- once both have K_S, they use symmetric key cryptography