Computer Communication Networks

Security

IECE / ICSI 416– Spring 2020

Prof. Dola Saha

Properties and Threat Models

- Secrecy/Confidentiality
  - Can secret data be leaked to an attacker?
- Integrity
  - Can the system be modified by the attacker?
- Authenticity
  - Who is the system communicating/interacting with?
- Availability
  - Is the system always able to perform its function?
- Need to think about Threat (attacker) Models

What is network security?

- ***confidentiality:*** only sender, intended receiver should "understand" message contents
  - Method – encrypt at sender, decrypt at receiver
  - A protocol that prevents an adversary from understanding the message contents is said to provide *confidentiality*.
  - Concealing the quantity or destination of communication is called *traffic confidentiality*.
- ***message integrity:*** sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
  - A protocol that detects message tampering provides *data integrity*.
  - The adversary could alternatively transmit an extra copy of your message in a *replay attack*.
  - A protocol that detects message tampering provides *originality*.
  - A protocol that detects delaying tactics provides *timeliness*.

What is network security?

- ***authentication:*** sender, receiver want to confirm identity of each other
  - A protocol that ensures that you really are talking to whom you think you're talking is said to provide *authentication*.
  - Example: DNS Attack [correct URL gets converted to malicious IP]
- ***access and availability***: services must be accessible and available to users
  - A protocol that ensures a degree of access is called *availability*.
  - Denial of Service (DoS) Attack
  - Example: SYN Flood attack (Client not transmitting 3rd message in TCP 3-way handshake, thus consuming server's resource)
  - Example: Ping Flood (attacker transmits ICMP Echo Request packets)

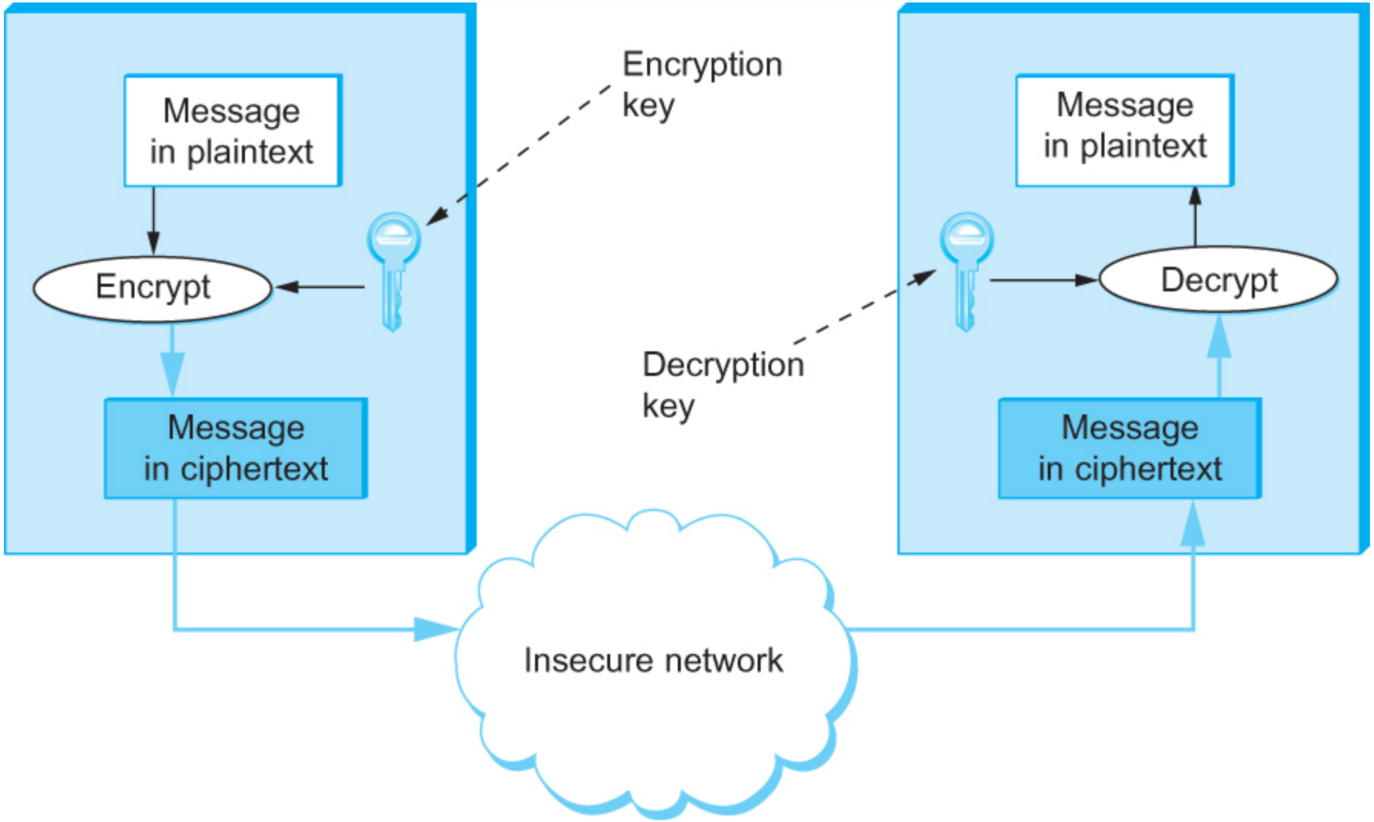
There are bad guys (and girls) out there!

Q: What can a "bad guy" do?

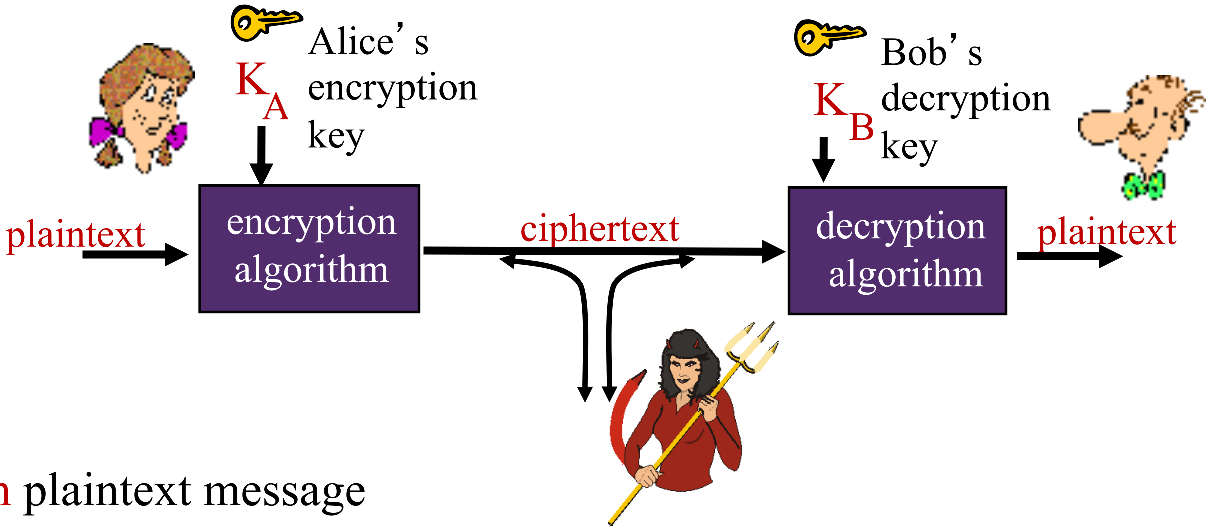
A: A lot!

- *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)

Cryptography in Insecure Network



The language of cryptography



m plaintext message

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

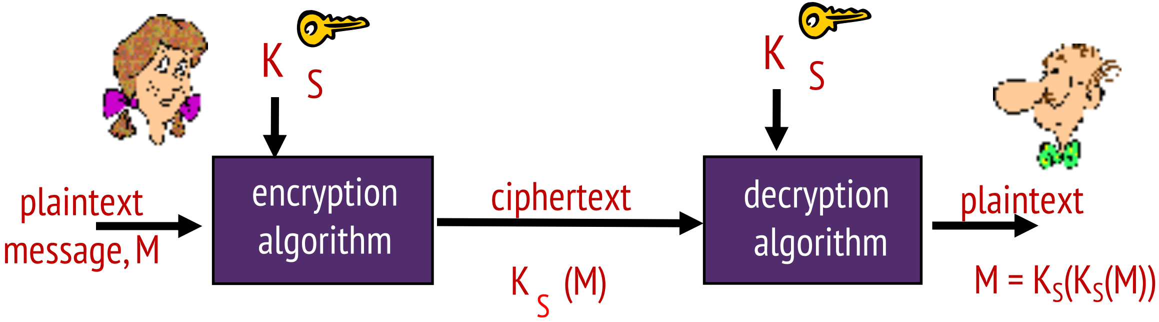
$m = K_B(K_A(m))$

Kerckhoff's Principle

- A cryptographic algorithm should be secure even if everything about the system, except the key, is public knowledge.
- Even if adversary knows the algorithm, he should be unable to recover the plaintext as long as he does not know the key.

Symmetric key cryptography

n-bit plaintext message, $M = m_1 m_2 m_3 \ldots m_n \in \{0, 1\}^n$



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

Two properties:

- Bob should be able to easily recover M from C
- Any adversary who does not know K should not, by observing C, be able to gain any more information about M

One-time Pad

Alice and Bob share an n-bit secret key $K = k_1 k_2 k_3 \ldots k_n \in \{0, 1\}^n$, where the n bits are chosen independently at random. K is known as the one-time pad.

$$C = M \oplus K. \qquad \text{Bit-wise XOR}$$

To decode C,

$$C \oplus K = (M \oplus K) \oplus K = M \oplus (K \oplus K) = M \oplus 0 = M.$$

This uses the facts that exclusive OR ($\oplus$) is associative and commutative, that $B \oplus B = 0$ for any B, and that $B \oplus 0 = B$ for any B.

How is One-Time Pad Secure?

- Assumptions:
  - Eve observes C.
  - Fixed plaintext message M (Eve does not know).

- Every unique ciphertext $C \in \{0, 1\}^n$ can be obtained from M with a corresponding unique choice of key K
  - Set $K = C \oplus M$ where C is the desired ciphertext
  - $C = M \oplus K = M \oplus (C \oplus M) = C \oplus (M \oplus M) = C$

- A uniformly random bit-string $K \in \{0, 1\}^n$ generates a uniformly random ciphertext $C \in \{0, 1\}^n$.
- Thus, with known C, Eve can do no better than guessing at the value of K uniformly at random.

Use the key more than once?

- Eve has access to two ciphertexts
  - $C_1 = M_1 \oplus K$ and $C_2 = M_2 \oplus K$
- Eve computes $C_1 \oplus C_2$
  - $C_1 \oplus C_2 = (M_1 \oplus K) \oplus (M_2 \oplus K) = (M_1 \oplus M_2)$
- Eve has partial knowledge of M
- If Eve knows one of the messages
  - It can decode other M
  - It can decode Key K

Simple encryption scheme

substitution cipher: substituting one thing for another

- *monoalphabetic* cipher: substitute one letter for another

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

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 plaintext corresponding to ciphertext [when an intruder knows some of the (plain, cipher) pairings]
  - e.g., in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,
- chosen-plaintext attack: Trudy can get ciphertext for chosen plaintext
  - If Trudy could get Alice to send encrypted message, "The quick brown fox jumps over the lazy dog", then the encryption is broken.

A chosen-plaintext attack is more powerful than known-plaintext attack

Polyalphabetic Cipher

Plaintext letter:	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
$C_1(k = 5)$:	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
$C_2(k = 19)$:	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

- n substitution ciphers, $C_1, C_2, \ldots, C_n$
- cycling pattern:
  - e.g., n=4 [C_1-C_4], k=key length=5: C_1,C_3,C_4,C_3,C_2; C_1,C_3,C_4,C_3,C_2; ..
- for each new plaintext symbol, use subsequent substitution pattern in cyclic pattern
  - dog: d from C_1, o from C_3, g from C_4

  Encryption key: n substitution ciphers, and cyclic pattern
  - key need not be just n-bit pattern

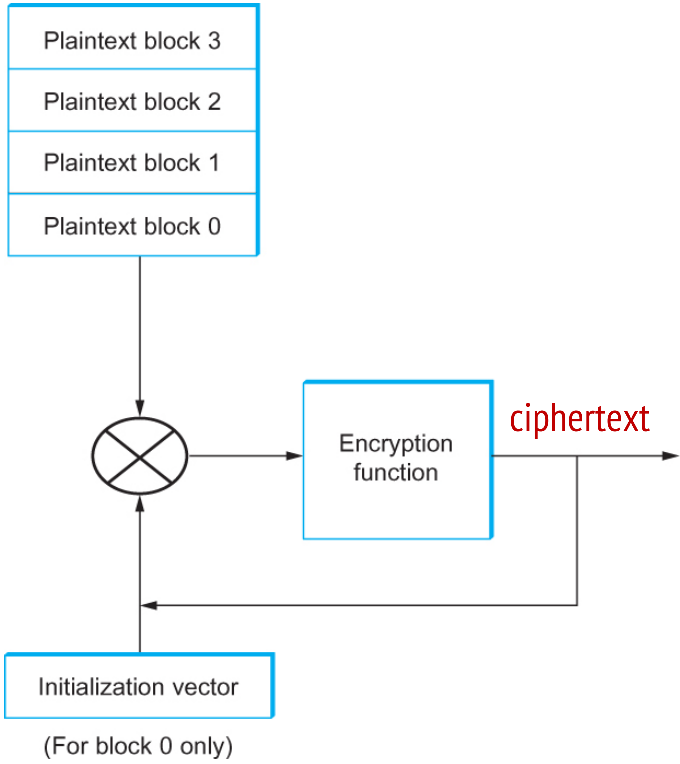
Block vs Stream Cipher

- Block ciphers process messages into blocks, each of which is then en/decrypted
  - 64-bits or more
  - Example: DES, AES
- Stream ciphers process messages a bit or byte at a time when en/decrypting
  - Example: WEP (used in 802.11)

- Brute Force attack is possible if few number of bits are chosen

Cipher Block Chaining

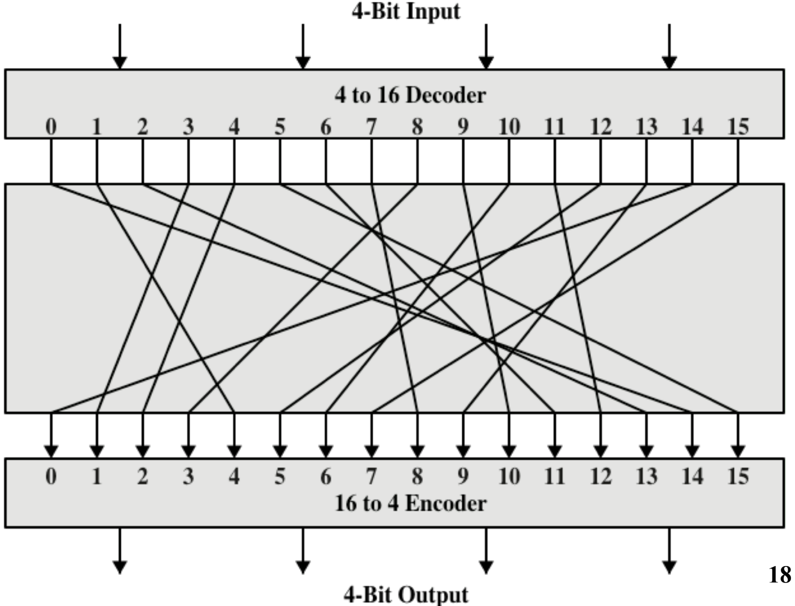
- Plaintext block is XORed with the previous block's ciphertext before being encrypted.
  - Each block's ciphertext depends on the preceding blocks
  - First plaintext block is XORed with a random number.
    - ✓ That random number, called an *initialization vector (IV), is included with the series of ciphertext blocks so that the first ciphertext* block can be decrypted.
- Provides better efficiency for brute force attack



Block Cipher (Basics)

- Operates on a plaintext block of n bits to produce a ciphertext block of n bits.
- There are 2^n possible different plaintext blocks
- For the encryption to be reversible, each must produce a unique ciphertext block.
- Such a transformation is called reversible, or nonsingular.

A 4-bit input produces one of 16 possible input states, which is mapped by the substitution cipher into a unique one of 16 possible output states, each of which is represented by 4 ciphertext bits.



Ideal Block Cipher

- Feistel refers to this as the *ideal block cipher*
  - it allows for the maximum number of possible encryption mappings from the plaintext block
- Practical Problem
  - Small block size degenerates to substitution cipher
  - Note: not a problem of block cipher, but choice of n

Key length (Ideal Block Cipher)

- Mapping is the key
  - the key that determines the specific mapping from among all possible mappings
- the required key length is (4 bits) x (16 rows) = 64 bits
- The length of the key is n x 2^n bits
- For a 64-bit block the required key length is 64 x $2^{64} \sim 10^{21}$ bits

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Feistel Cipher

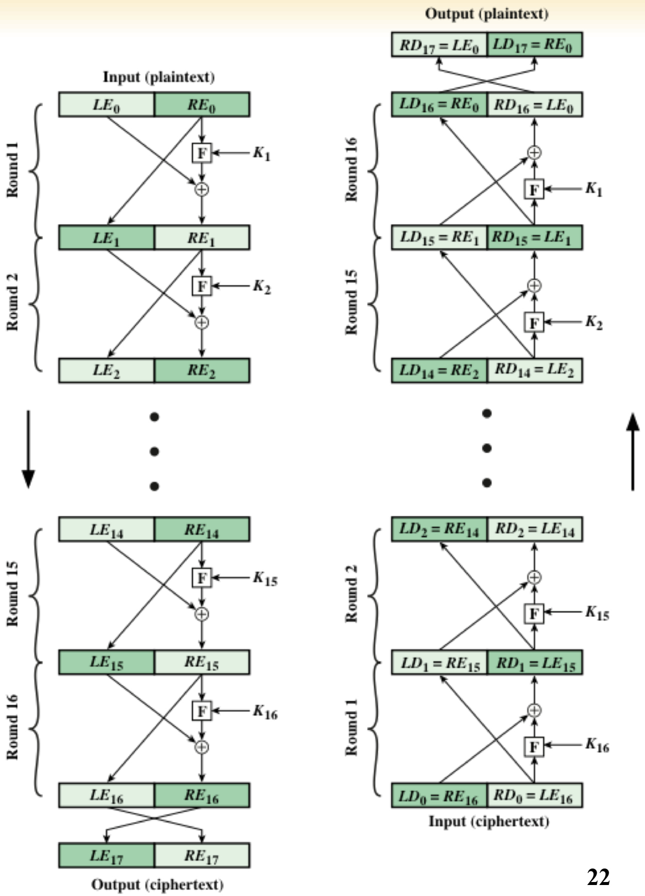
- Feistel proposed the use of a cipher that alternates substitutions and permutations

Substitutions	• Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements
Permutation	• No elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed

- Is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates confusion and diffusion functions
- Is the structure used by many significant symmetric block ciphers currently in use

Feistel Cipher

- ➢ Block size and Key Size
  - Larger block/key sizes → greater security
  - Larger block/key sizes → reduced encryption/decryption speed
- ➢ Number of rounds
  - a single round offers inadequate security but that multiple rounds offer increasing security
- ➢ Subkey generation algorithm
  - Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis



Symmetric key crypto: DES

DES: Data Encryption Standard

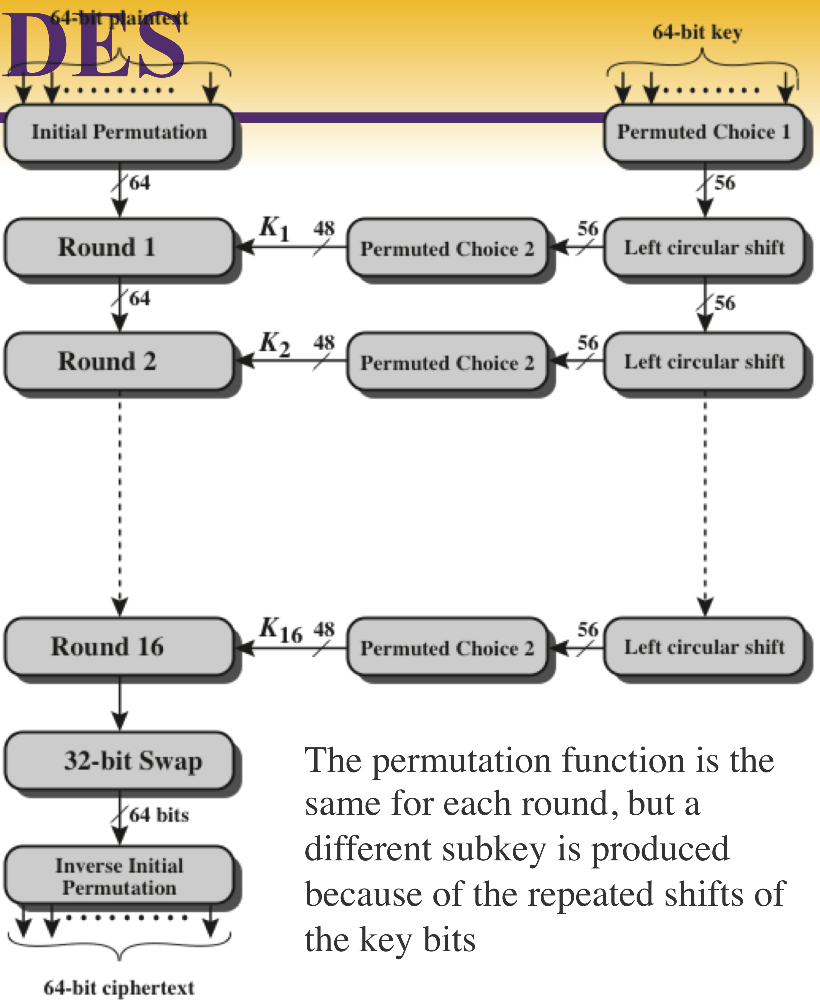
- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- block cipher with cipher block chaining
- how secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase, decrypted (brute force) in less than a day
  - no known good analytic attack
- making DES more secure:
  -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

Symmetric key crypto: DES

- initial permutation (on 64 bits)
- 16 identical "rounds" of function application
  - each using different 48 bits of key
  - a subkey (K_i) is produced by the combination of a left circular shift and a permutation
  - rightmost 32 bits are moved to leftmost 32 bits
- final permutation (on 64 bits)

Kaufman, Schneier, 1995

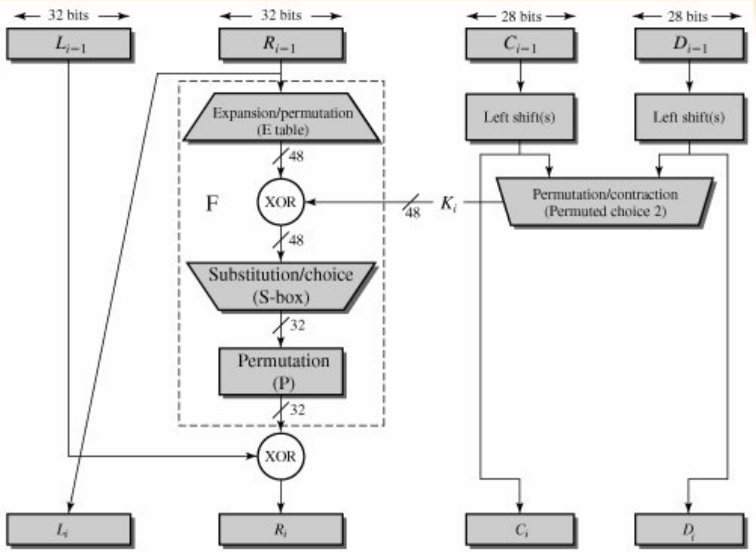
With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher



The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits

Each round of DES

- K_i is 48 bits, R input is 32 bits.
- R is first expanded to 48 bits
  - a table defines a permutation plus an expansion that involves duplication of 16 of the R bits
- Resulting 48 bits are XORed with Ki
- This 48-bit result passes through a substitution function (S box) that produces a 32-bit output
- This is permuted



$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \times F(R_{i-1}, K_i)$$

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

Public Key Cryptography

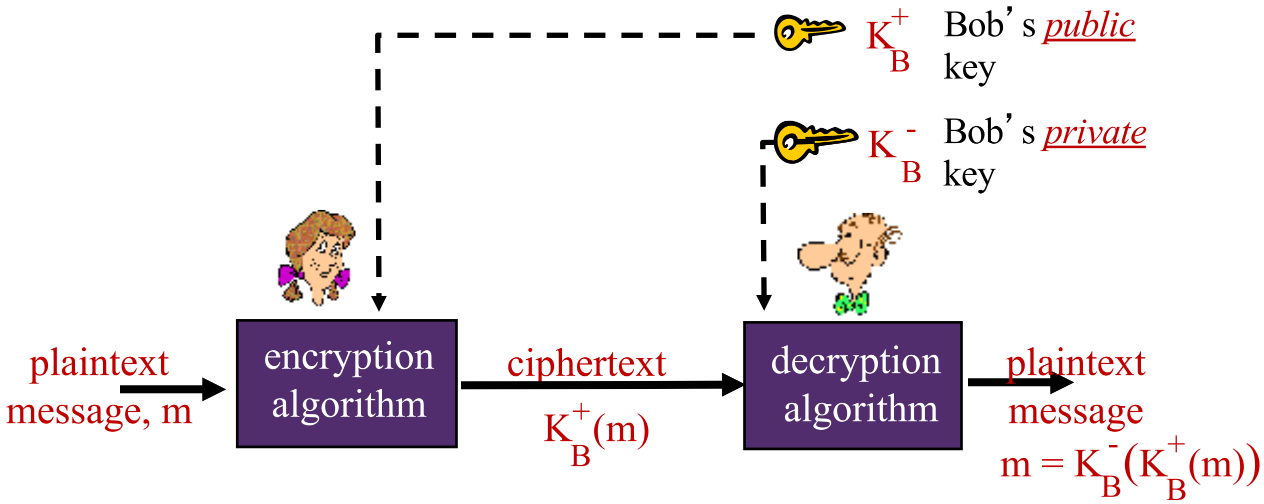
symmetric key crypto

- 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

RSA: Rivest, Shamir, Adelson algorithm [1999]

requirements:

1. need $K_B^{+}(\ )$ and $K_B^{-}(\ )$ 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's security relies on the difficulty of finding p and q knowing only n (the "factorization problem").

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: $ed \bmod z = 1$).

5. *public* key is (n,e). *private* key is (n,d).

$$\underbrace{(n,e)}_{K_B^+} \qquad \underbrace{(n,d)}_{K_B^-}$$

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

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

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

RSA example:

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

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

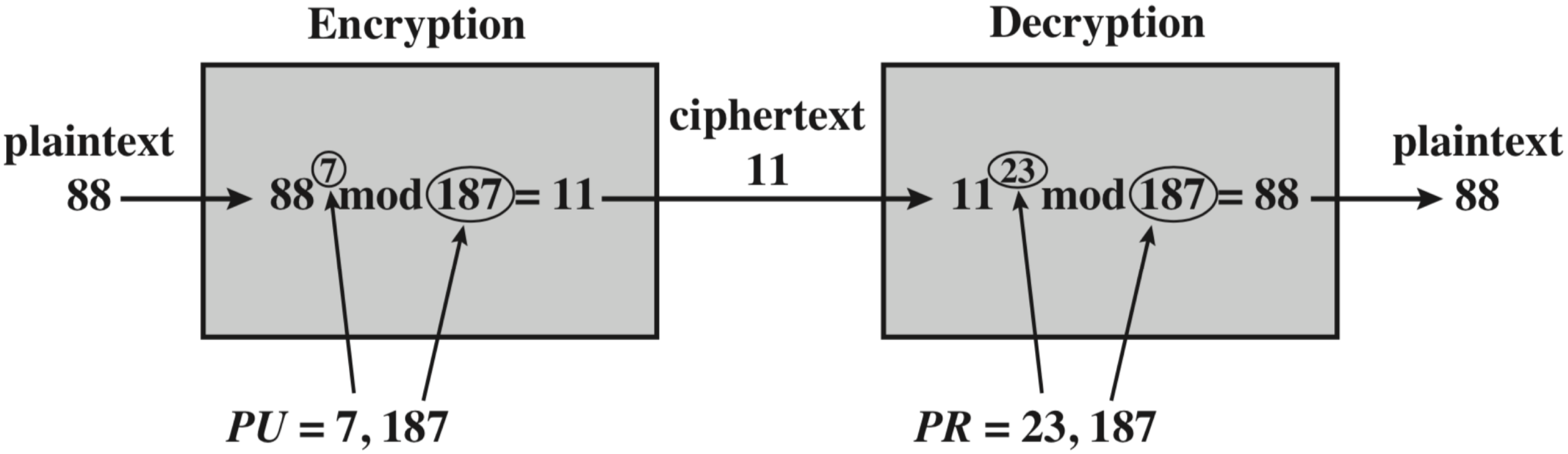
$d=29$ (so $ed-1$ exactly divisible by z).

encrypting 8-bit messages.

	bit pattern	m	m^e	$c = m^e \bmod n$
encrypt:	00001000	12	24832	17

	c	c^d	$m = c^d \bmod n$
decrypt:	17	481968572106750915091411825223071697	12

RSA Example



Encryption
Decryption
plaintext
88
88 7 mod 187 = 11
ciphertext
11
11 23 mod 187 = 88
plaintext
88
PU = 7, 187
PR = 23, 187

Why does RSA work?

- must show that $c^d \bmod n = m$
  where $c = m^e \bmod n$
- fact: for any x and y: $x^y \bmod n = x^{(y \bmod z)} \bmod n$
  - where $n= pq$ and $z = (p-1)(q-1)$
- thus,
  $c^d \bmod n = (m^e \bmod n)^d \bmod n$
  $= m^{ed} \bmod n$
  $= m^{(ed \bmod z)} \bmod n$
  $= m^1 \bmod n$
  $= m$

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!

How is it possible?

follows directly from modular arithmetic:

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

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