Introduction to Network Security

Gerald A. Marin

These slides are provided solely for the use of FIT students taking this course in Network Security. No further copies are permitted. Some materials are permitted for classroom use by other authors, who retain all copyrights. In particular, a number of slides in this section represent copyright protected material that accompanies the text Computer Networking by J. F. Kurose and K. W. Ross; they are used with the authors' permission.

Organization

- What is network security?
- Principles of cryptography
- Security Requirements: Confidentiality, authentication, …
- Key Distribution and certification
- Access control: firewalls
- Attacks and counter measures

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

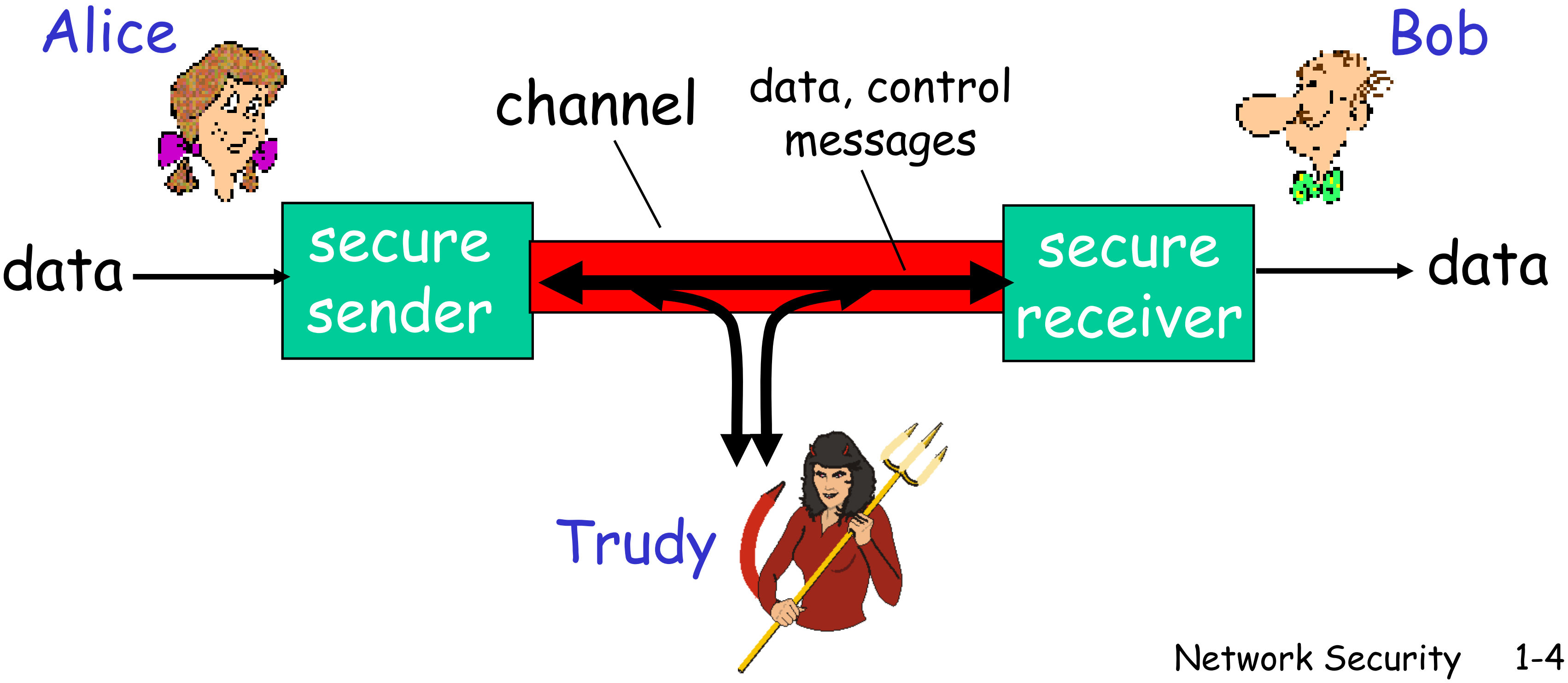
Message Integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

Access Control and Availability: services must be accessible and available to intended users

Non-repudiation: sender should not be able to disavow later.

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
- other examples?

There are bad guys (and girls) out there!

Q: What can a "bad guy" do?

A: a lot!

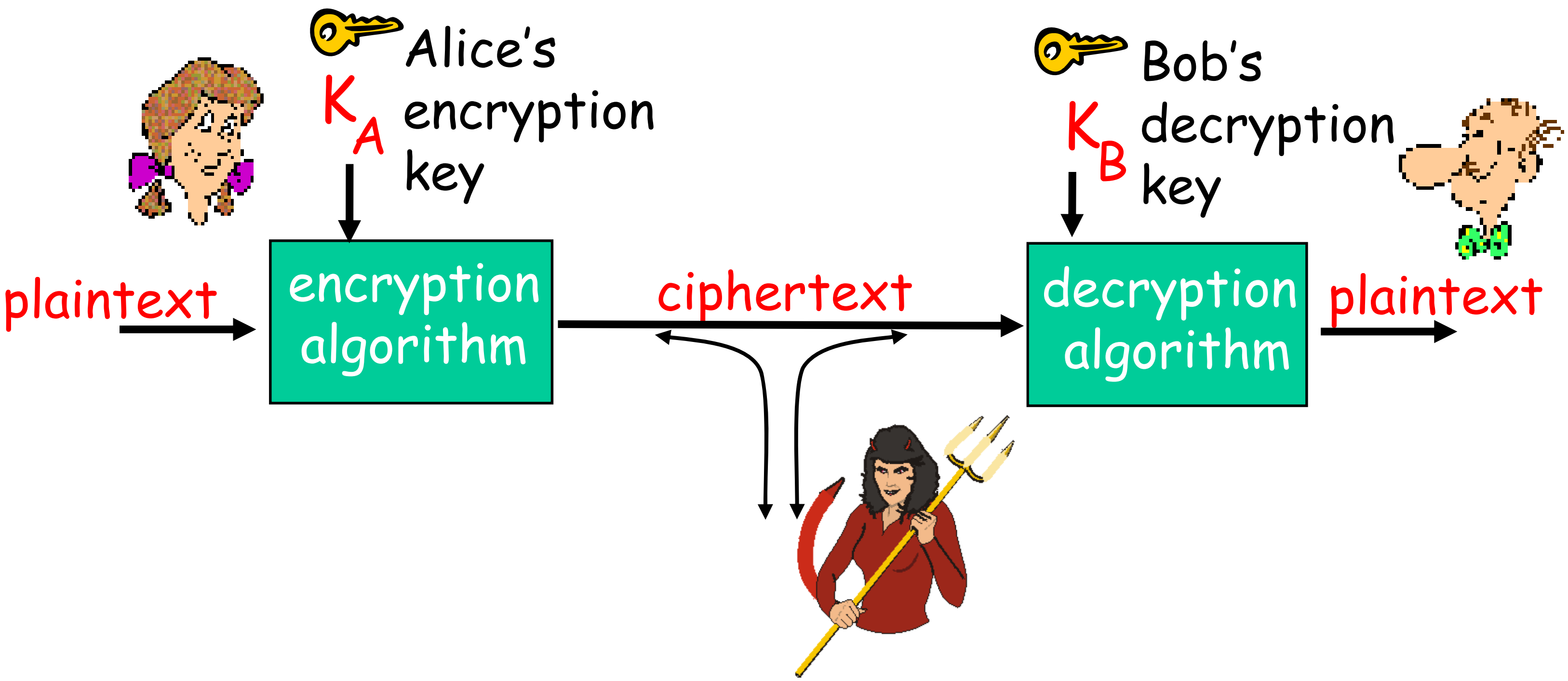
- *eavesdrop:* intercept messages
- actively *insert* messages into connection
- *impersonation:* can fake (spoof) source address in packet (or any field in packet)
- *hijacking:* "take over" ongoing connection by removing sender or receiver, inserting himself in place
- *denial of service*: prevent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

more on this later ……

Organization

- What is network security?
- Principles of cryptography
- Security Requirements
- Key Distribution and certification
- Access control: firewalls
- Attacks and counter measures
- Security in many layers

The language of cryptography



Symmetric key crypto: sender, receiver keys *identical*

Asymmetric key crypto: keys NOT identical.

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

Symmetric key cryptography

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
 plaintext:  abcdefghijklmnopqrstuvwxyz

ciphertext:  mnbvcxzasdfghjklpoiuytrewq
```

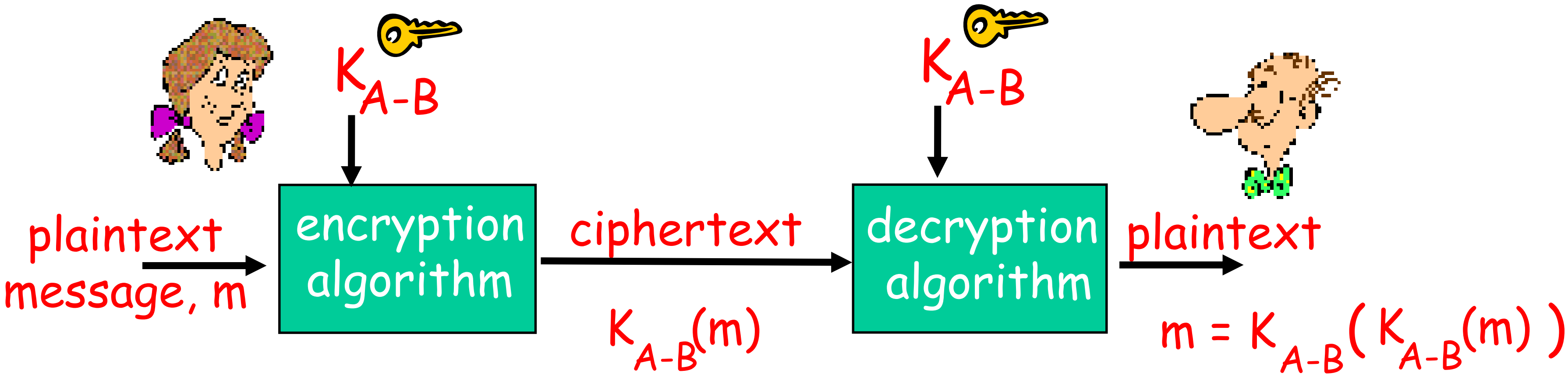
E.g.:

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

Q: How hard to break this simple cipher?:

- ❑ brute force (how hard?)
- ❑ other?

Symmetric key cryptography



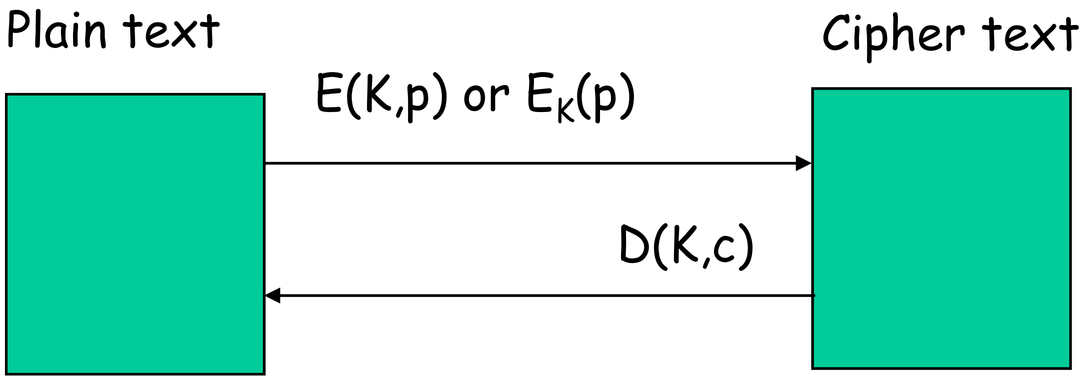
symmetric key crypto: Bob and Alice share know same (symmetric) key: K_{A-B}

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

Block Cipher

The idea of a block cipher is fundamental to the study of cyptography.

A **block cipher** is a function that encrypts fix-sized blocks. Today block ciphers commonly encrypt blocks of 128 bits. They are said to take 128 bits of “plain text” and produce 128 bits of “cipher text.” This is done using a **secret** key and a **public** algorithm.



K is said to be a “symmetric” secret key.

3-bit Example

With 3 bits only 2^3 messages are possible. Notice that there are $2^3!$ (or 8!) possible permutations of 2^3 messages. Each permutation can be thought of as a "lookup table" that represents an encryption of the possible messages through a reordering. For example:

000		101
001		110
010		111
011		000
100	becomes	001
101		010
110		011
111		100

"Lookup table" represents one encryption (possible reordering) of all possible messages. A particular lookup table corresponds to one particular secret key. Thus, we need 16 bits to represent all 8!=40,320 possible keys.

An “Ideal” Cipher

- Two commanders have eight possible messages to send. Message 000 is “do not attack” while 001 is “attack at 1am,” message 002 is “attack at 2am,”…etc.
- Each has a code book that lists 8! = 40,320 “lookup tables” generated randomly. All officers have a copy of this book. (Represents the algorithm.)
- Just before they go to their respective commands they are directed to use table number 12,123 by the crypto staff. (Represents the secret key.)
- **The ideal cipher represents the best that can be done.** Namely, we choose one lookup table randomly from all possible lookup tables.

Huge Lookup Tables

- If we think of a block cipher as a lookup table (corresponding to a key), the size of real tables is a problem.
- For 32-bit blocks a table would be 16 gigabytes.
- For For 64-bit blocks a table would be 150 million terabytes.
- For 128-bit blocks a table would be 5×10^{39} bytes.
- Thus, **real codes use encryption algorithms plus a key** to generate ciphertext from plaintext directly.
  - Note that this approach may NOT produce an ideal cypher.

Brute Force Example

- I created a message using the following letters: acelps.
- I have used a substitution code to encrypt it (again based only on the 6 letters above).
- Any spaces have been ignored, that is, no space becomes nospace.
- The encrypted text is lcapclscaec.
- Find the key and break the code! Due 1/20/09 and each student will present his/her solution and turn in a report with code.
    - How many possible messages are there?
    - How many possible keys are there?

Kerckhoffs' Principle

- Security depends only on the secrecy of the Key and not on the secrecy of algorithms.
  - Algorithms hard to change and built into system hardware/software
  - Algorithms don't change for long periods
  - Someone may obtain physical access to a laptop that contains the algorithms
  - Algorithms SHOULD be published so that other experts can check them for vulnerabilities.

Cipher Attacks

- Ciphertext-only attack: trying to decrypt a message when all you know is the ciphertext. This is the most difficult case.
- Known plaintext attack: trying to decrypt a message when you know both the plaintext and the ciphertext (by prior example or autoreply).
- Chosen plaintext attack: now you get to specify specially prepared plaintexts for which you will then see the ciphertexts.
  - Offline: prepare plaintexts all ahead of time
  - Online: prepare next plaintext after receiving ciphertexts from previous submissions.
- Chosen ciphertext attack: Receive the ciphertext corresponding to your chosen plaintext AND receive the plaintext corresponding to your chosen ciphertext.

Birthday Attack (ciphertext attack example)

- Named after the "birthday paradox." If you have 23 people in a room, then the probability that two of them have the same birthday is greater than 0.5.
  - Useful approximation: for large n the probability of generating a duplicate (also called the probability of a collision) is close to 0.5 after approximately $\sqrt{n}$ attempts.
- Birthday attack: If keys are being generated randomly, then a key collision will occur relatively soon.
  - Determine this has happened (see a ciphertext of header twice implies same key).
  - Insert previous message ciphertext into current message and it will be accepted because the keys match.

Meet-in-Middle Attack (Known plaintext example)

- Suppose we know a header or any other part of a message always sent from Alice to Bob. Suppose further they use a 64-bit key. (Brute force attack requires evaluating $2^{64} = 1.845 \times 10^{19}$ keys.)
- Generate 2^{32} keys randomly and encode the header with each of the keys (produce table).
- Watch for the encoded header in each message. It will likely occur during during first $2^{32} = 4.295 \times 10^{9}$ transactions. Usually applied to all authentication messages sent by Alice.
- When encoded header occurs we look to see which key we used to generate it.

"Distinguishing" Attack

- Many types of attacks:
  - …decrypt only a specific message
  - …reveal partial information about a message
  - …other vulnerabilities?
- Given many kinds of attacks crypto analysts generally defend against a "distinguishing attack."
- A distinguishing attack is an attack that detects a non-trivial difference between the ideal cipher and the actual cipher.
  - Encryption and decryption available for comparisons between ideal and actual.
  - Free to choose any key.
  - More about "non-trivial" later.

Conventional Encryption Principles

- An encryption scheme has five ingredients:
  - Plaintext
  - Encryption algorithm
  - Secret Key
  - Ciphertext
  - Decryption algorithm (perhaps different key)
- Security depends on the secrecy of the key, not the secrecy of the algorithm
- In modern encryption encryp/decrypt is done with a block cipher – an encryption function for fix-sized blocks.

Cryptography

- Classified along three independent dimensions:
    - The type of operations used for transforming plaintext to ciphertext
    - The number of keys used
        - symmetric (single key)
        - asymmetric (two-keys, or public-key encryption)
    - The way in which the plaintext is processed

Average time required for exhaustive key search

Key Size (bits)	Number of Alternative Keys	Time required at 10^6 Decryption/µs
32	$2^{32} = 4.3 \times 10^9$	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	10 hours
128	$2^{128} = 3.4 \times 10^{38}$	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	5.9×10^{30} years

Feistel Cipher Structure

- Virtually all conventional block encryption algorithms, including DES have a structure first described by Horst Feistel of IBM in 1973.
- Feistel ciphers are a special class of iterated block ciphers where the ciphertext is calculated from the plaintext by repeated application of the same transformation or “round” function.
- In a Feistel cipher, the text being encrypted is split into two halves. The round function f is applied to one half using a subkey and the output of f is exclusive-ored with the other half. The two halves are then swapped. Each round follows the same pattern except for the last round where there is no swap.

Feistel Structure (Continued)

- A nice feature of a Feistel cipher is that encryption and decryption are structurally identical, though the subkeys used during encryption at each round are taken in reverse order during decryption.

Feistel Cipher Structure

- The realization of a Feistel Network depends on the choice of the following parameters and design features:
  - **Block size:** larger block sizes mean greater security
  - **Key Size:** larger key size means greater security
  - **Number of rounds:** multiple rounds offer increasing security
  - **Subkey generation algorithm:** greater complexity will lead to greater difficulty of cryptanalysis.
  - **Fast software encryption/decryption:** the speed of execution of the algorithm becomes a concern

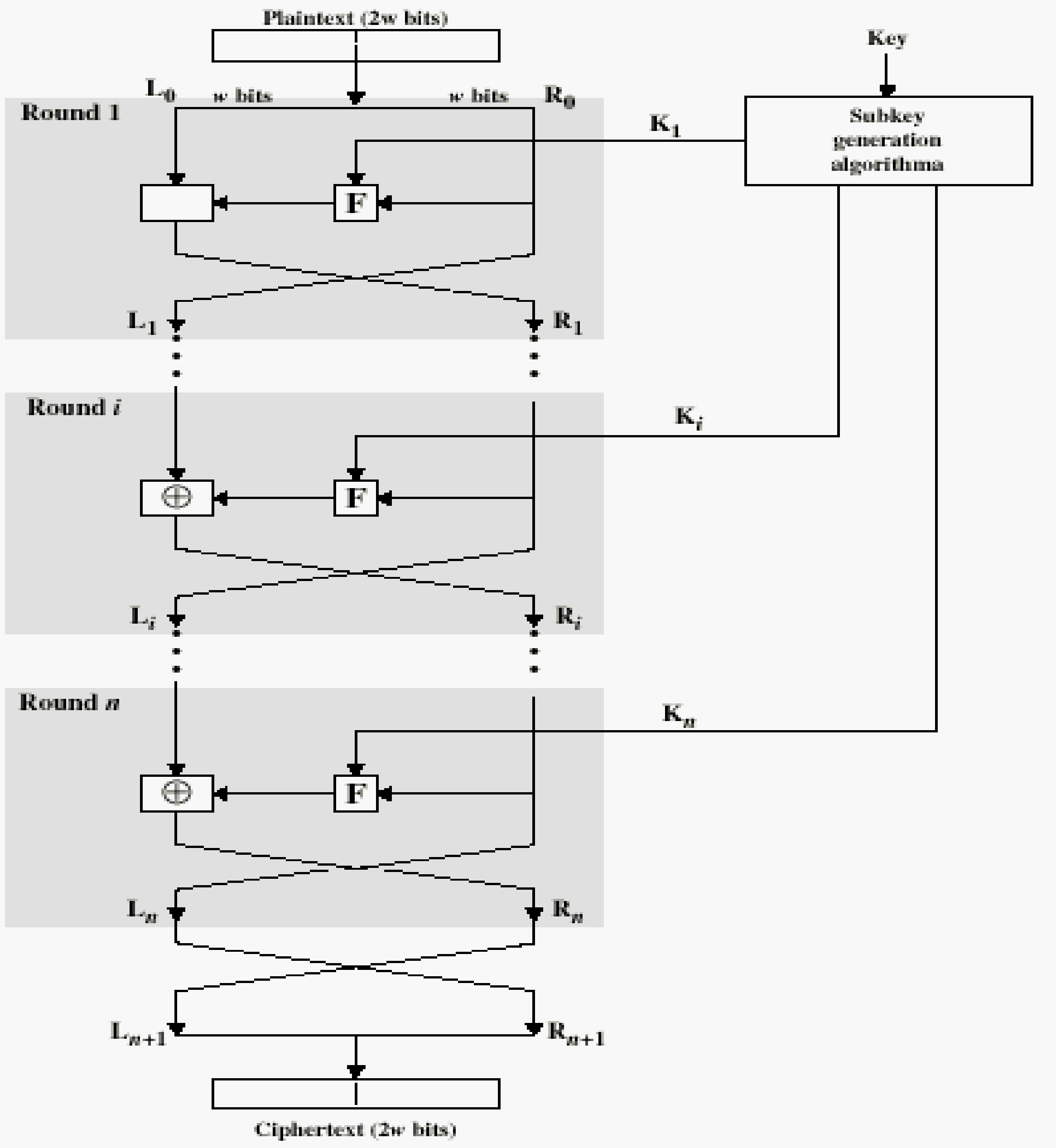


Figure 2.2 Classical Feistel Network

Conventional Encryption Algorithms

- Data Encryption Standard (DES)
  - WAS the most widely used encryption scheme (now vulnerable).
  - DES is a block cipher.
  - The plaintext is processed in 64-bit blocks.
  - The key is 56-bits in length
- Making DES more secure:
  - use three keys sequentially (3-DES) on each datum
  - use cipher-block chaining
  - The algorithm is referred to as the Data Encryption Algorithm (DEA).

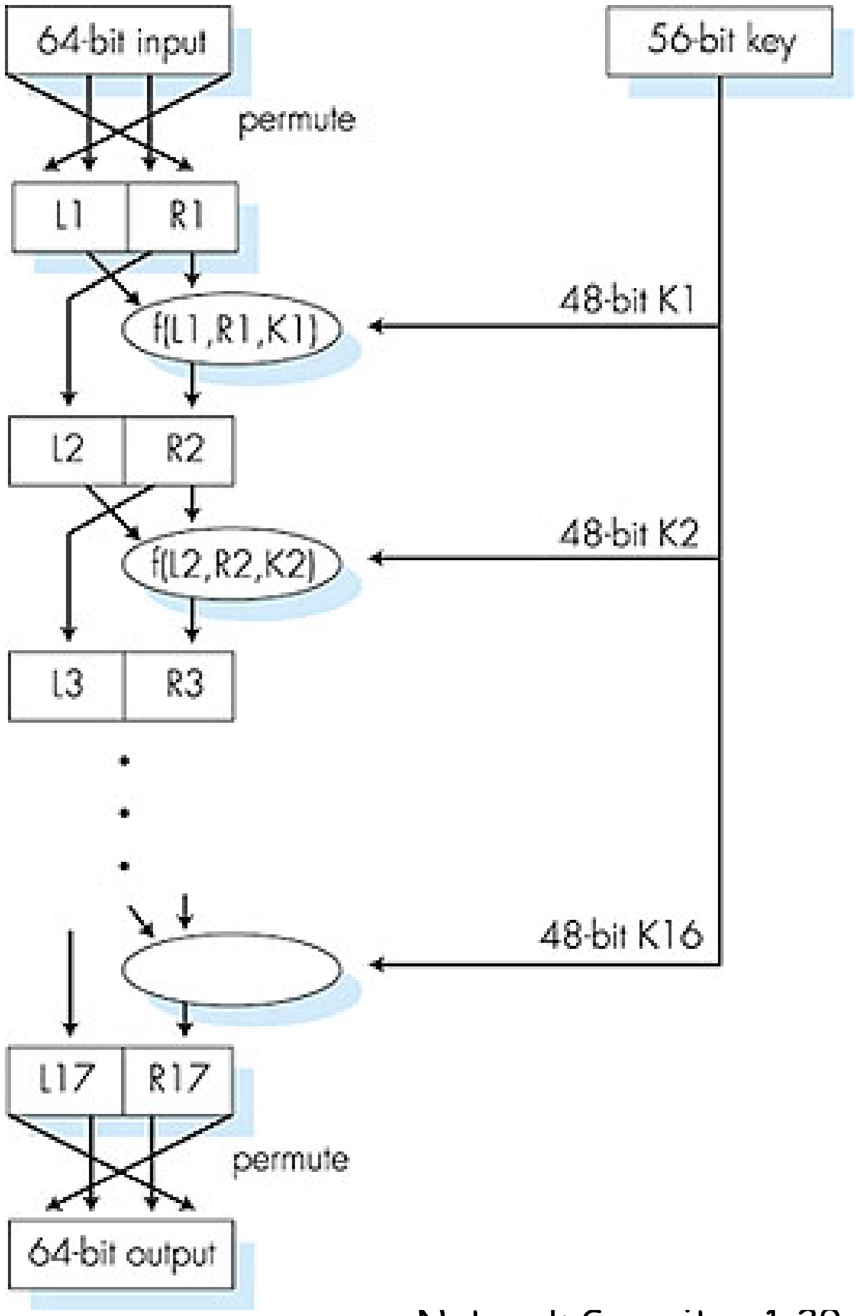
Symmetric key crypto: DES

DES operation

initial permutation

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

final permutation



Initial Permutation

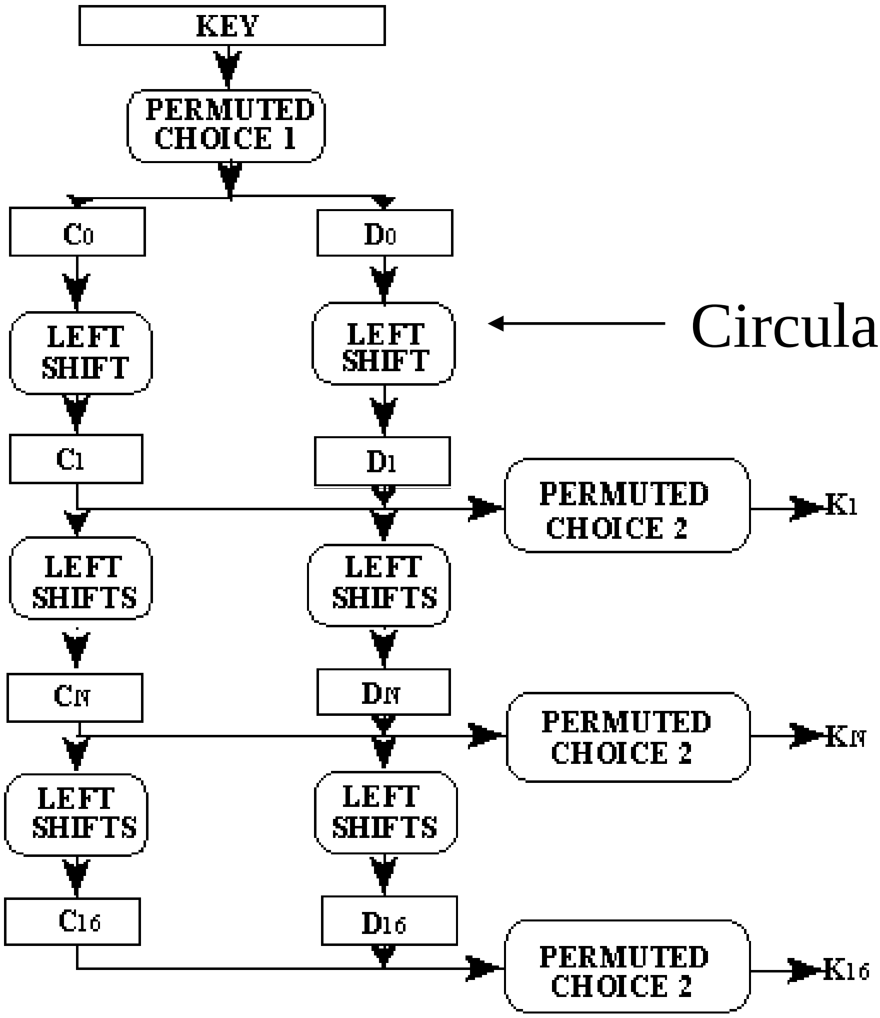
The 64 bits of the input block to be enciphered are first subjected to the following permutation, called the initial permutation **IP**:

L1	58	50	42	34	26	18	10	2
	60	52	44	36	28	20	12	4
	62	54	46	38	30	22	14	6
	64	56	48	40	32	24	16	8
R1	57	49	41	33	25	17	9	1
	59	51	43	35	27	19	11	3
	61	53	45	37	29	21	13	5
	63	55	47	39	31	23	15	7

DES

- **The overall processing at each iteration:**
  - $L_i = R_{i-1}$
  - $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$ (XOR addition)
  - Function F to be described.
  - K_i is a "subkey" formed as a permuted subset of the original 64-bit key.
- Left and Right are then swapped for next iteration.
- It remains only to understand key generation and the function F ("mangler") .

Key Generation Steps



Circular left shift.

Permuted choice 1 is determined by the following table:

C_0

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

D_0

63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Note that bits 8, 16, …64 are not used; they are reserved for use as parity bits. The key is actually 56-bits. BUT Permuted Choice 2 from previous slide actually selects 48 of these bits at each step.

Shift Schedule

Iteration Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of Left Shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Permuted Choice 2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

→ 48 bit subkey.

This completes the subkey generation description.

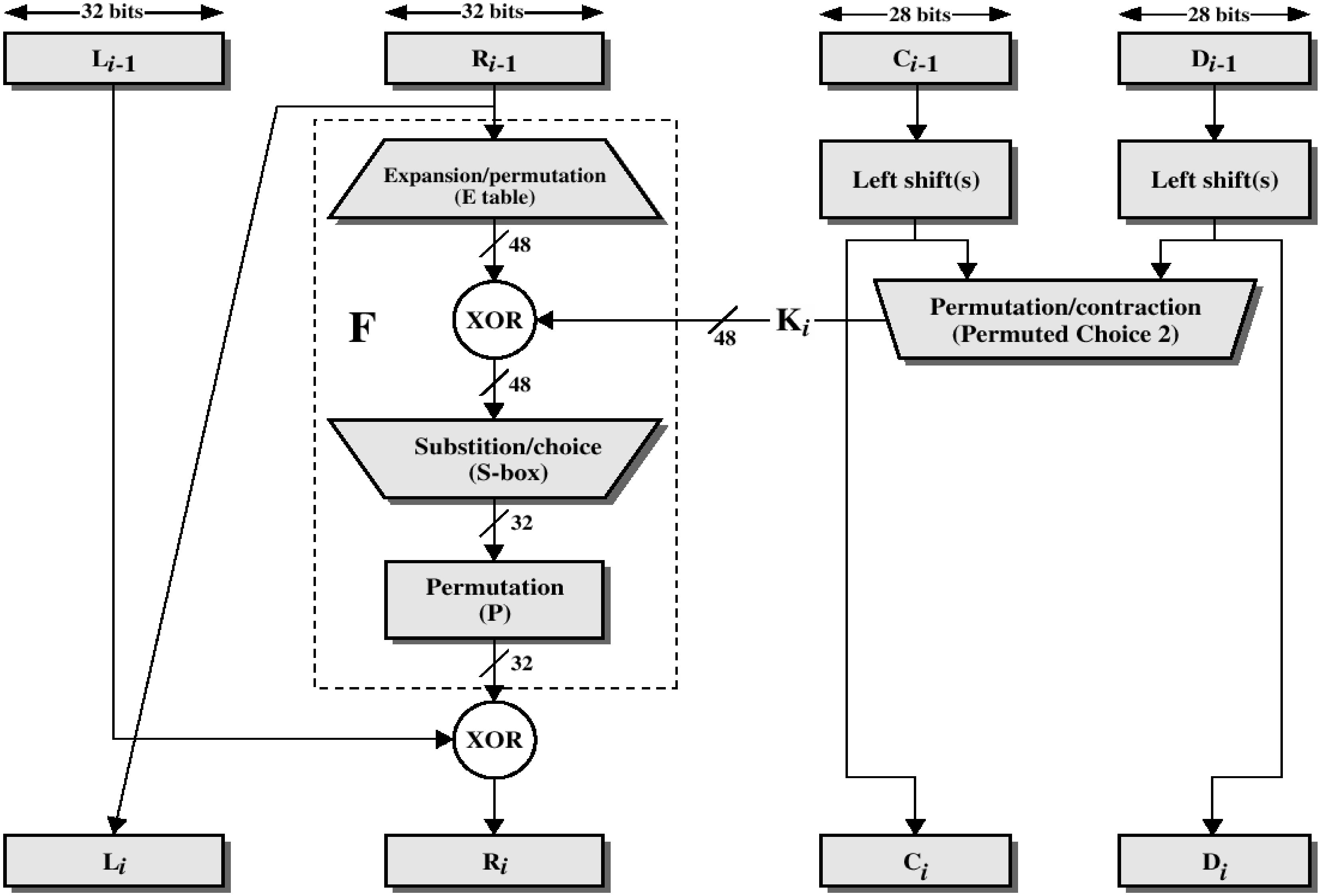


Figure 2.4 Single Round of DES Algorithm

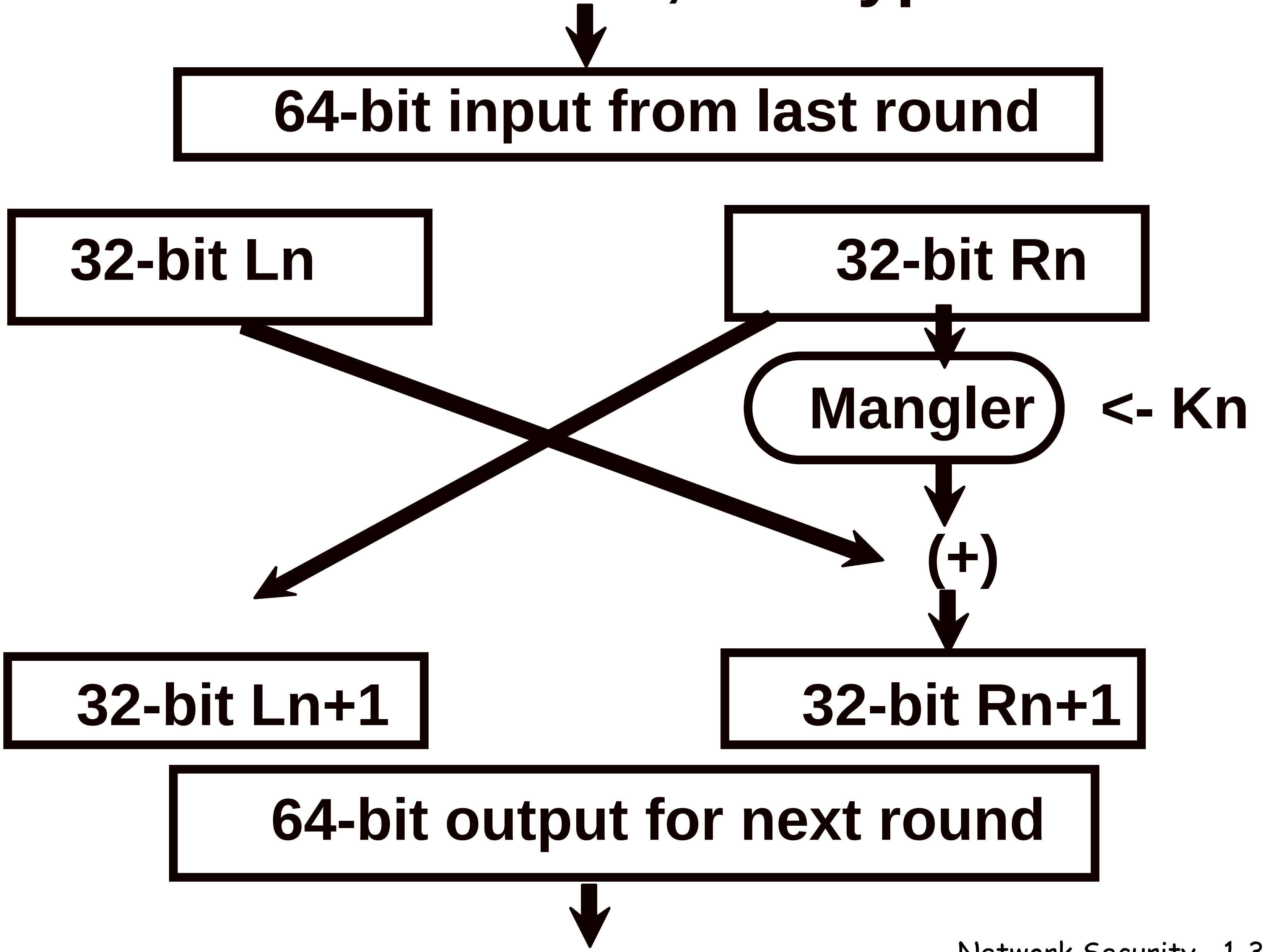
E-Table (Expands 32 bits to 48)

32 bit (right)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

48 bits out

DES Round n, Encryption



64-bit input from last round

32-bit Ln

32-bit Rn

Mangler

<- Kn

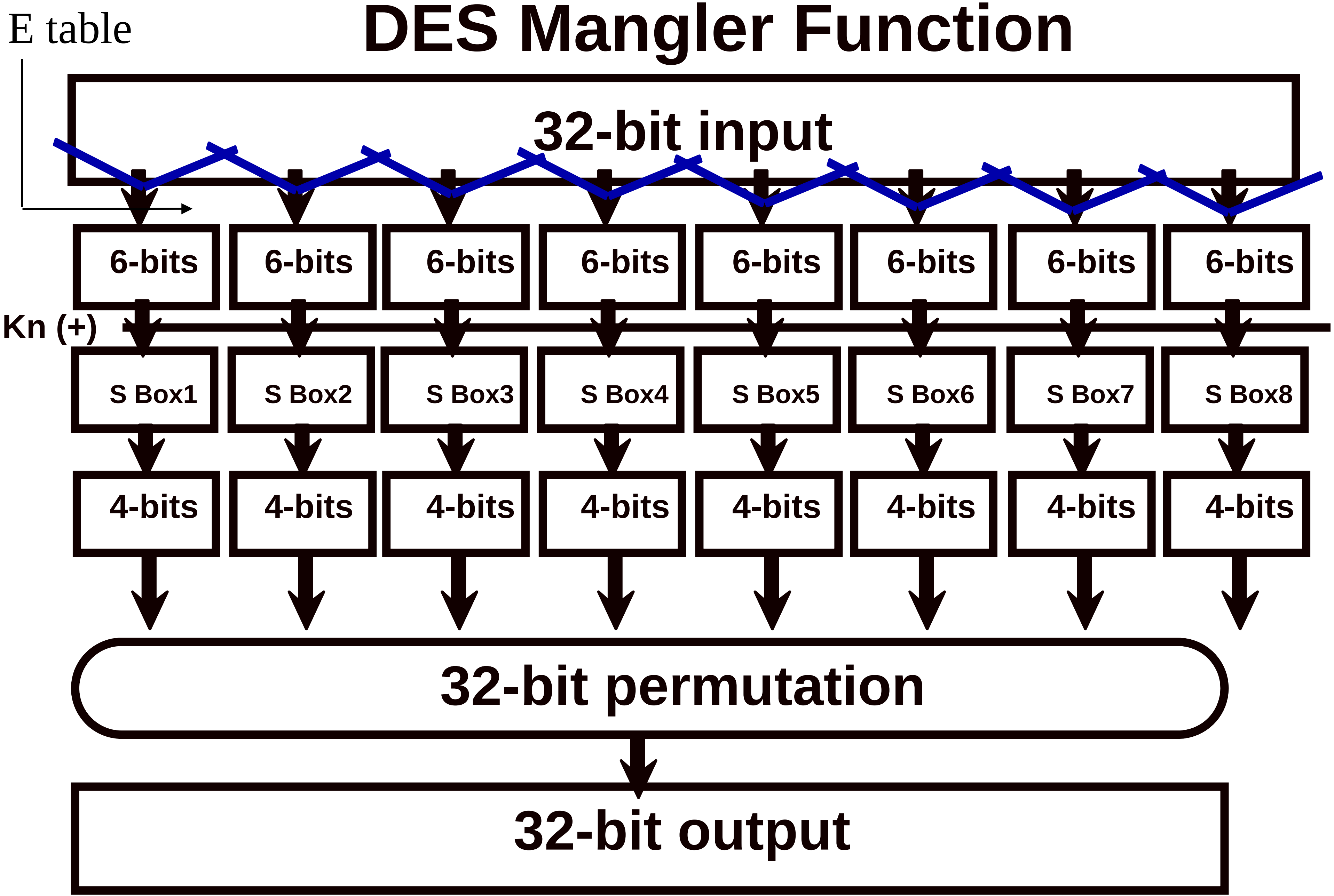
(+)

32-bit Ln+1

32-bit Rn+1

64-bit output for next round

DES Mangler Function



First Four S Boxes:

S_1

14 4 13 1 2 15 11 8 3 10 6 12 5 9 0 7
O 15 7 4 14 2 13 1 10 6 12 11 9 5 3 8
4 1 14 8 13 6 2 11 15 12 9 7 3 10 5 0
15 12 8 2 4 9 1 7 5 11 3 14 10 O 6 13

S_2

15 1 8 14 6 11 3 4 9 7 2 13 12 O 5 10
3 13 4 7 15 2 8 14 12 0 1 10 6 9 11 5
0 14 7 11 10 4 13 1 5 8 12 6 9 3 2 15
13 8 10 1 3 15 4 2 11 6 7 12 0 5 14 9

S_3

10 0 9 14 6 3 15 5 1 13 12 7 11 4 2 8
13 7 O 9 3 4 6 10 2 8 5 14 12 11 15 1
13 6 4 9 8 15 3 0 11 1 2 12 5 10 14 7
1 10 13 0 6 9 8 7 4 15 14 3 11 5 2 12

S_4

7 13 14 3 0 6 9 10 1 2 8 5 11 12 4 15
13 8 11 5 6 15 O 3 4 7 2 12 1 10 14 9
10 6 9 0 12 11 7 13 15 1 3 14 5 2 8 4
3 15 O 6 10 1 13 8 9 4 5 11 12 7 2 14

S_5

2 12 4 1 7 10 11 6 8 5 3 15 13 O 14 9

14 11 2 12 4 7 13 1 5 0 15 10 3 9 8 6

4 2 1 11 10 13 7 8 15 9 12 5 6 3 O 14

11 8 12 7 1 14 2 13 6 15 O 9 10 4 5 3

S_6

12 1 10 15 9 2 6 8 O 13 3 4 14 7 5 11

10 15 4 2 7 12 9 5 6 1 13 14 O 11 3 8

9 14 15 5 2 8 12 3 7 0 4 10 1 13 11 6

4 3 2 12 9 5 15 10 11 14 1 7 6 0 8 13

S_7

4 11 2 14 15 0 8 13 3 12 9 7 5 10 6 1

13 0 11 7 4 9 1 10 14 3 5 12 2 15 8 6

1 4 11 13 12 3 7 14 10 15 6 8 0 5 9 2

6 11 13 8 1 4 10 7 9 5 0 15 14 2 3 12

S_8

13 2 8 4 6 15 11 1 10 9 3 14 5 0 12 7

1 15 13 8 10 3 7 4 12 5 6 11 0 14 9 2

7 11 4 1 9 12 14 2 0 6 10 13 15 3 5 8

2 1 14 7 4 10 8 13 15 12 9 0 3 5 6 11

S-box Result is 4 bits (from 6)

$$b_1b_2b_3b_4b_5b_6 \rightarrow \text{S-box} \rightarrow s_1s_2s_3s_4$$

where $s_1s_2s_3s_4$ is the S-box entry at row b_1b_6 and column $b_2b_3b_4b_5$.

Rows are numbered 0 through 3 and columns are 0 through 15.

Example: S_1

$011101 \rightarrow$

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
O	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	O	6	13

$\rightarrow 0011$

32-Bit Permutation

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

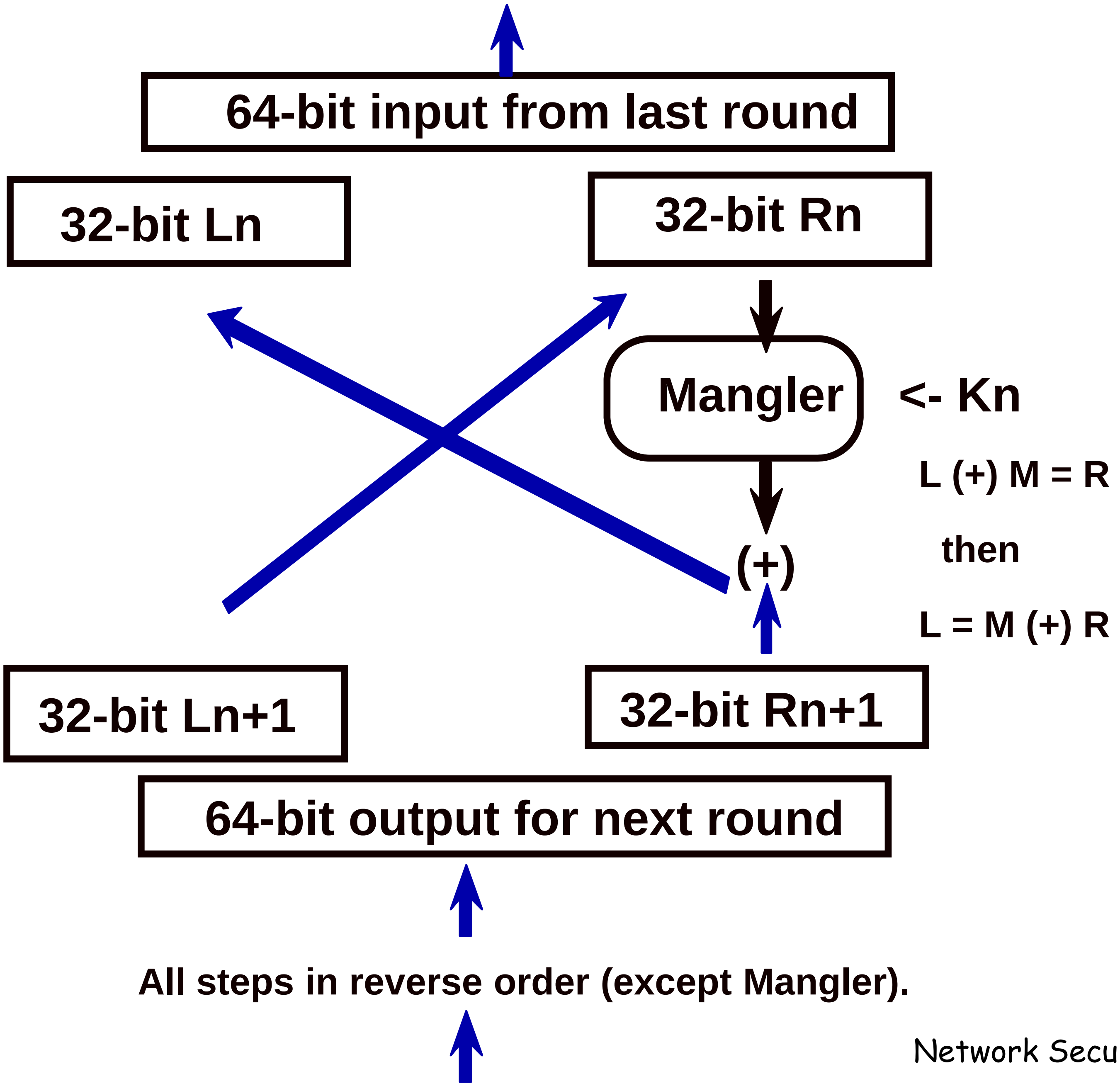
Inverse of Initial Permutation

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

End of DES Encryption

Note that the DES Standard is available on our web site.
Assignment: Read and Study.

DES Round n, Decryption



64-bit input from last round

32-bit Ln

32-bit Rn

Mangler

<- Kn

L (+) M = R

then

L = M (+) R

(+)

32-bit Ln+1

32-bit Rn+1

64-bit output for next round

All steps in reverse order (except Mangler).

Concerns about DES

A “DES Cracker” was designed for less than $250,000 that will try 1E12 56-bit keys per second (1000 per nanosecond). This will find the right key in about 3 days (if the plaintext is recognized as such when it appears).

The answer is to use longer keys. 128-bit keys are in fashion.

Triple-DES effectively uses a 112-bit key.

Triple DEA

- Use three keys and three executions of the DES algorithm (encrypt-decrypt-encrypt)

$$C = E_{K3}[D_{K2}[E_{K1}[P]]]$$

  - C = ciphertext
  - P = Plaintext
  - EK[X] = encryption of X using key K
  - DK[Y] = decryption of Y using key K

- Effective key length of 168 bits

Triple DEA

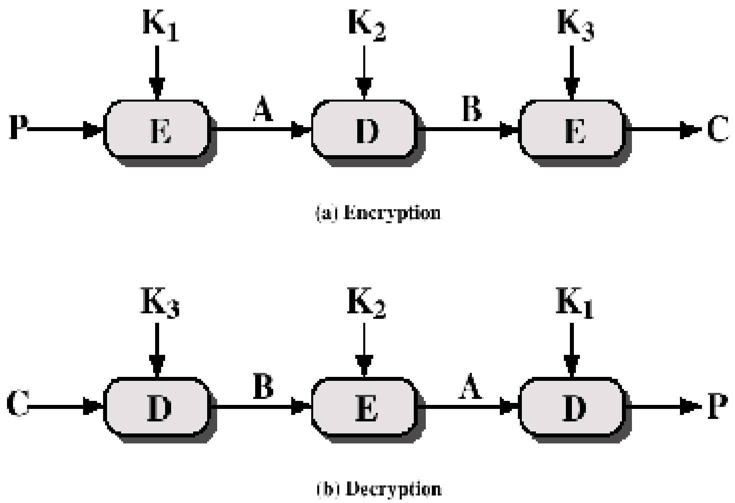


Figure 2.6 Triple DEA

Other Symmetric Block Ciphers

- International Data Encryption Algorithm (IDEA)
  - 128-bit key
  - Used in PGP
- Blowfish
  - Easy to implement
  - High execution speed
  - Run in less than 5K of memory

Other Symmetric Block Ciphers

- **RC5**
  - Suitable for hardware and software
  - Fast, simple
  - Adaptable to processors of different word lengths
  - Variable number of rounds
  - Variable-length key
  - Low memory requirement
  - High security
  - Data-dependent rotations
- **Cast-128**
  - Key size from 40 to 128 bits
  - The round function differs from round to round

Suppose plaintext is less than block length (or last segment of text is less than block length)?

Padding is required. Schneier* suggests two methods:

Let P be the plaintext and let $\ell(P)$ be the length of P in bytes. Let b be the block size of the block ciper in bytes. Then do either of the following:

1. Append a single byte with value 128 and as many zero bytes as required to make the overall length a multiple of b. The number of zero bytes added is in the range 0, 1, ..., $b-1$.
2. Determine the number of padding bytes required. This is a number, n (between 1 and b), and $n+\ell(P)$ is a multiple of b. Pad the plaintext by appending n bytes, each with value n.

*Practical Cryptography, Schneier and Ferguson

Electronic CodeBook Mode (ECB)

- This is the simplest way of encrypting plaintext into a sequence of adjacent blocks. Thus,

$$C_i = E(K, P_i), \text{ for } i = 1, 2, \ldots, \max \text{ blocks.}$$

- Warning: if two plaintext blocks are the same, then their ciphertext will also be the same. Thus, this mode should never be used.

Cipher Block Modes of Operation

- Cipher Block Chaining Mode (CBC)
  - The input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block.
  - Repeating pattern of 64-bits are not exposed
  - Must initialize with an “initial vector” or IV.

$$C_i = E_k[C_{i-1} \oplus P_i]$$

$$D_K[C_i] = D_K[E_K(C_{i-1} \oplus P_i)]$$

$$D_K[C_i] = (C_{i-1} \oplus P_i)$$

$$C_{i-1} \oplus D_K[C_i] = C_{i-1} \oplus C_{i-1} \oplus P_i = P_i$$

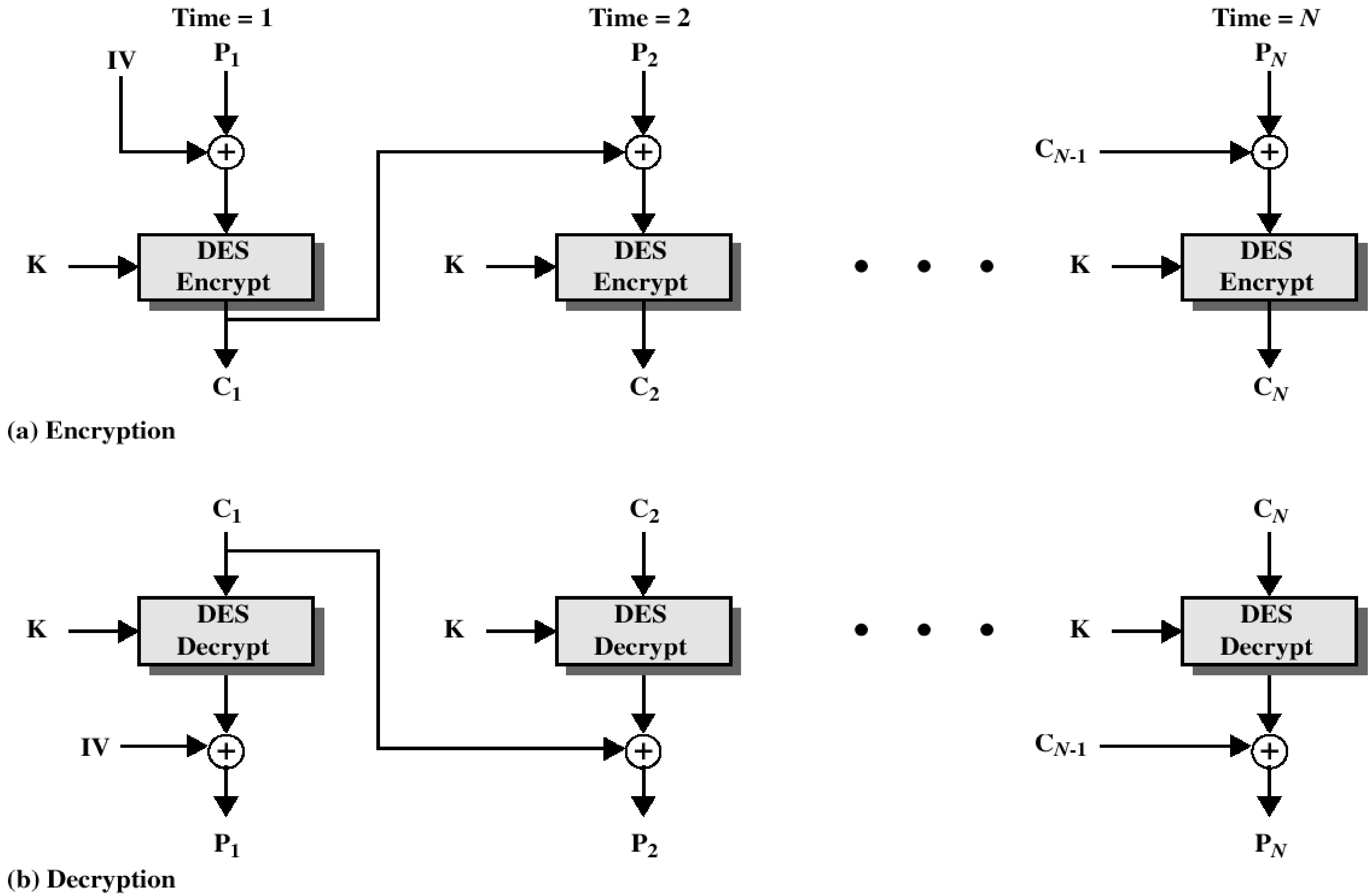


Figure 2.7 Cipher Block Chaining (CBC) Mode

Random IV

Would like to choose a random IV for reasonable protection of the first block, but receiver must know the IV (and it must be generated).

Idea: Create C_0 as the IV and send as the zeroth block. Then:

$$\mathrm{C}_0 = randomblock$$

$$C_i = E(K, P_i \oplus C_{i-1}).$$

For decryption: $P_i = D(K, C_i) \oplus C_{i-1} = P_i \oplus C_{i-1} \oplus C_{i-1} = P_i.$

Disadvantages:

1. Must implement a random number generator.
2. The ciphertext is one block longer than the plaintext. This is especially troubling for short messages.

AES: Advanced Encryption Standard

- new (Nov. 2001) symmetric-key NIST standard, replacing DES
- 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

Possible Projects

- Implement DES (or triple DES or AES…) plus block chaining mode using a random IV.
  - Prompt for plaintext or ciphertext input or file name.
  - Use ASCII character set.
  - Create key from input HEX.
  - Encrypt or Decrypt as specified by user.

More About AES

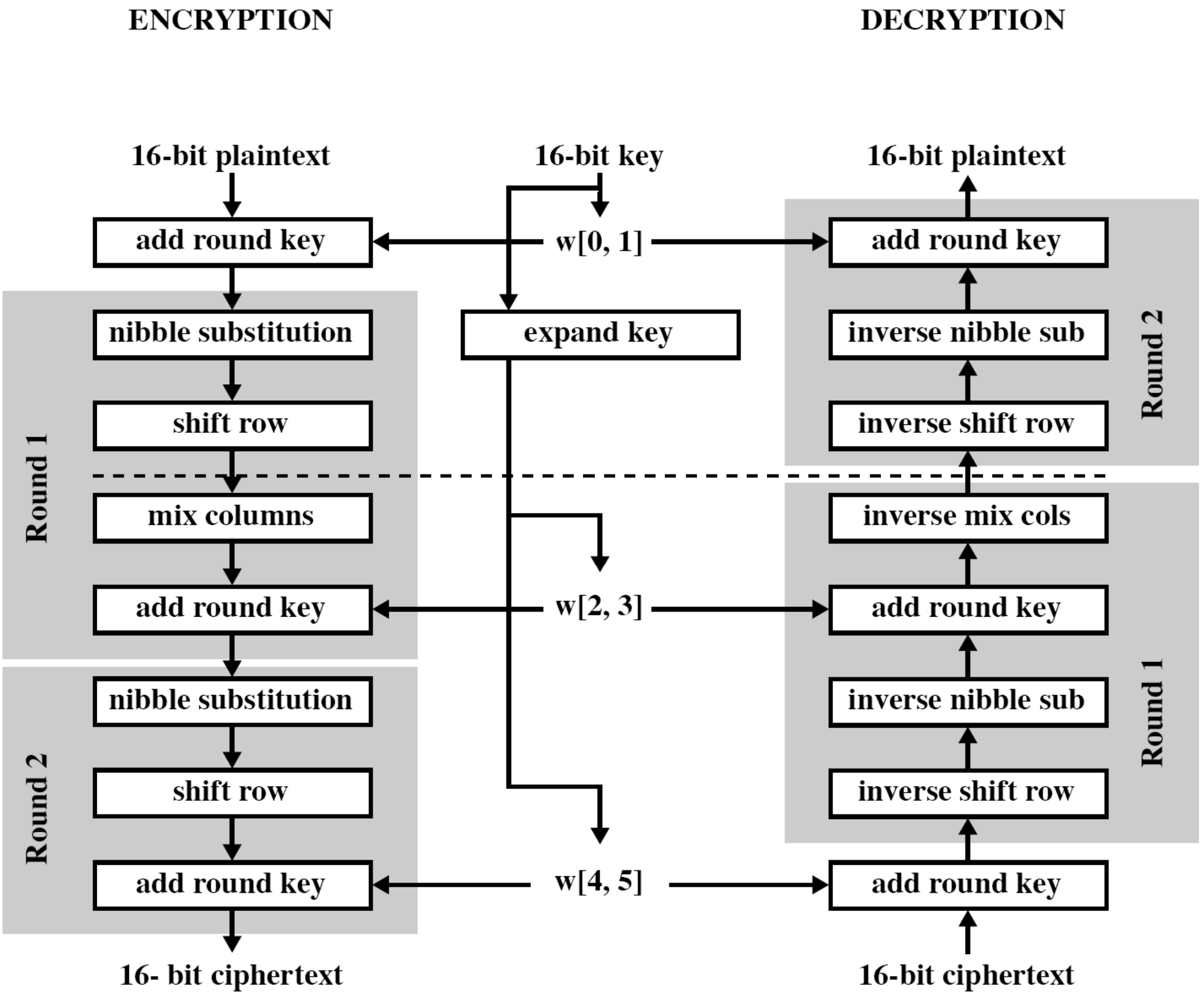
- The AES standard was published in November 2001 (FIPS 197).
- Chosen from 5 finalists.
- Winner was Rijndael by Vincent Rijmen and Joan Daemen.
- NOT a Feistel Cypher
- 10 full rounds include byte substitution, permutation, arithmetic operations on a finite field, and XOR with key.

Illustration with Simplified AES*

- S-AES created by Professor Edward Schaefer and students (Santa Clara University).
- Similar structure with fewer rounds and smaller parameters.
  - For tutorial purposes only – not for encryption

*William Stallings, Cryptography and Network Security, Principles and Practices, 4th edition, Pearson Education, Inc., New Jersey, 2006

S-AES Encryption/Decryption



Algorithm can be expressed using function composition:

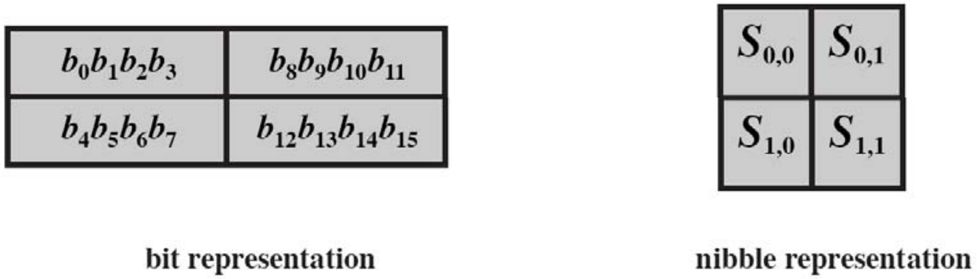
$$AK_2 \circ SR \circ NS \circ AK_1 \circ MC \circ SR \circ NS \circ AK_0 \circ \; plaintext$$

Functions are applied right to left as usual:

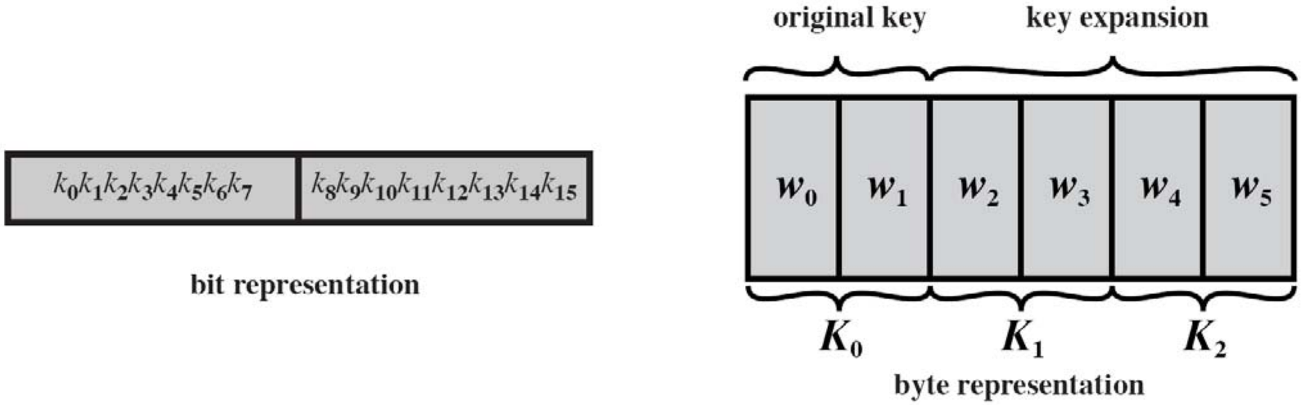
$$f \circ g(x) = f\left[g(x)\right].$$

Example: $f(x) = x^2$ and $g(x) = \sin x,$ then $f \circ g(x) = \left(\sin x\right)^2$.

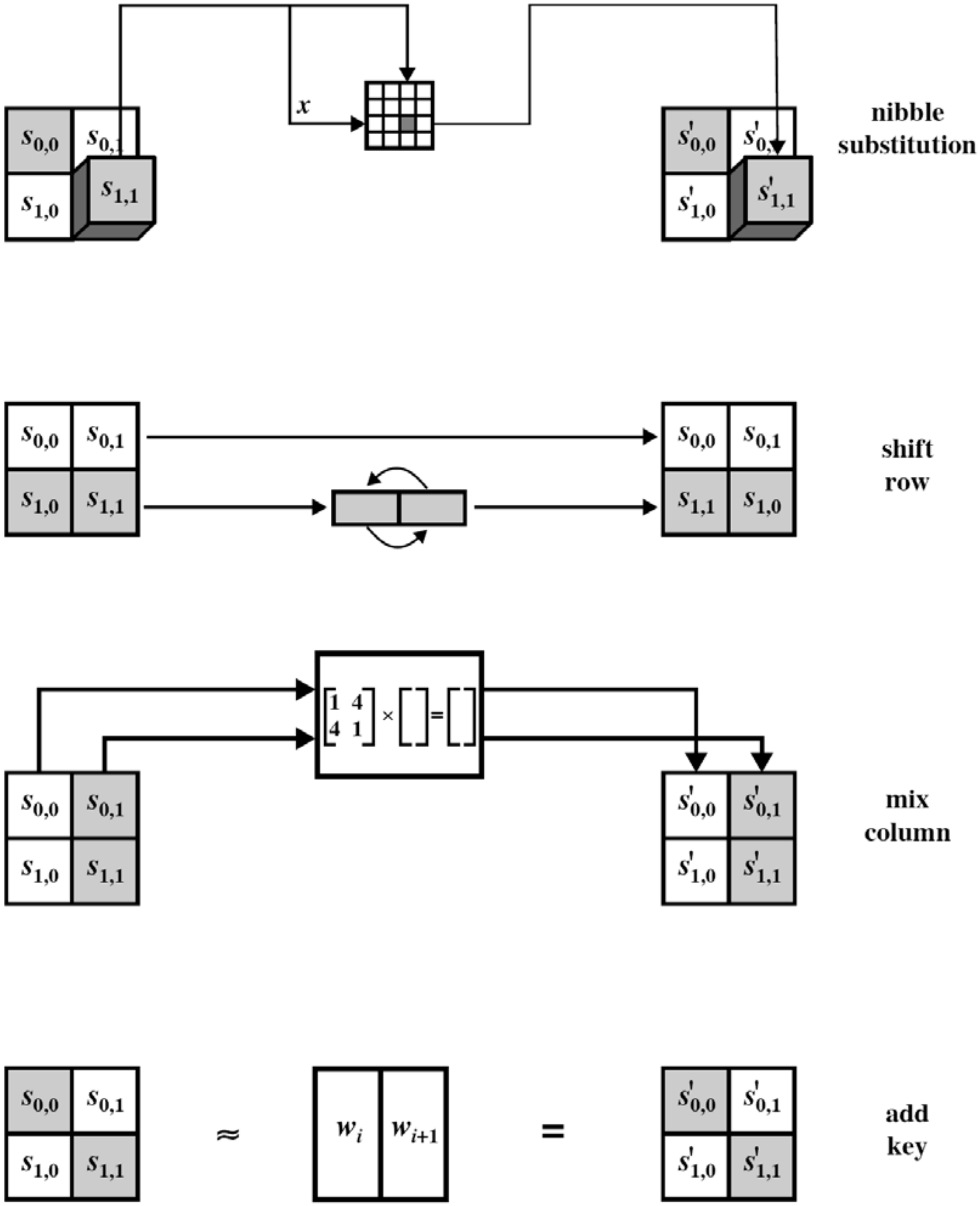
S-AES Data Structures



(a) State matrix



(b) Key


$s_{0,0}$ $s_{0,1}$ $s_{1,0}$ $s_{1,1}$
x
$s'_{0,0}$ $s'_{0,1}$ $s'_{1,0}$ $s'_{1,1}$
nibble substitution
$s_{0,0}$ $s_{0,1}$ $s_{1,0}$ $s_{1,1}$
$s_{0,0}$ $s_{0,1}$ $s_{1,1}$ $s_{1,0}$
shift row
$s_{0,0}$ $s_{0,1}$ $s_{1,0}$ $s_{1,1}$
$\begin{bmatrix}1 & 4\\4 & 1\end{bmatrix} \times \begin{bmatrix}\,\end{bmatrix} = \begin{bmatrix}\,\end{bmatrix}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{1,0}$ $s'_{1,1}$
mix column
$s_{0,0}$ $s_{0,1}$ $s_{1,0}$ $s_{1,1}$
$\approx$
w_i w_{i+1}
=
$s'_{0,0}$ $s'_{0,1}$ $s'_{1,0}$ $s'_{1,1}$
add key

Transformations

ADD Key Example:

Plain Text: $A749$

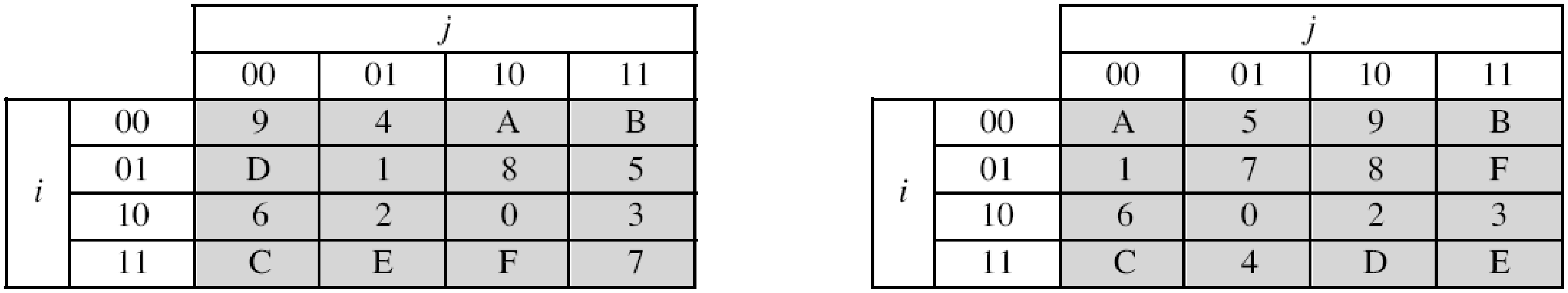
Key_0: $2D55$

$$\begin{bmatrix} A & 4 \\ 7 & 9 \end{bmatrix} \oplus \begin{bmatrix} 2 & 5 \\ D & 5 \end{bmatrix} = \begin{bmatrix} 8 & 1 \\ A & C \end{bmatrix}$$

In particular $A \oplus 2 \Rightarrow 1010 \oplus 0010 = 1000 = 8.$

Nibble Substitution

Table 5.5 S-AES S-Boxes



		j			
		00	01	10	11
i	00	9	4	A	B
	01	D	1	8	5
	10	6	2	0	3
	11	C	E	F	7

(a) S-Box

		j			
		00	01	10	11
i	00	A	5	9	B
	01	1	7	8	F
	10	6	0	2	3
	11	C	4	D	E

(b) Inverse S-Box

Note: hexadecimal numbers in shaded boxes; binary numbers in unshaded boxes.

S-table lookup left(2 bits) are row; right(2 bits) column.

Example:

$$\begin{bmatrix} 8 & 1 \\ A & C \end{bmatrix} \rightarrow \begin{bmatrix} 6 & 4 \\ 0 & C \end{bmatrix}$$

Shift Row

Shift Row operates only on the second row of the input state matrix. It performs a one-nibble circular shift of the second row.

Example:

$$\begin{bmatrix} 6 & 4 \\ 0 & C \end{bmatrix} \rightarrow \begin{bmatrix} 6 & 4 \\ C & 0 \end{bmatrix}$$

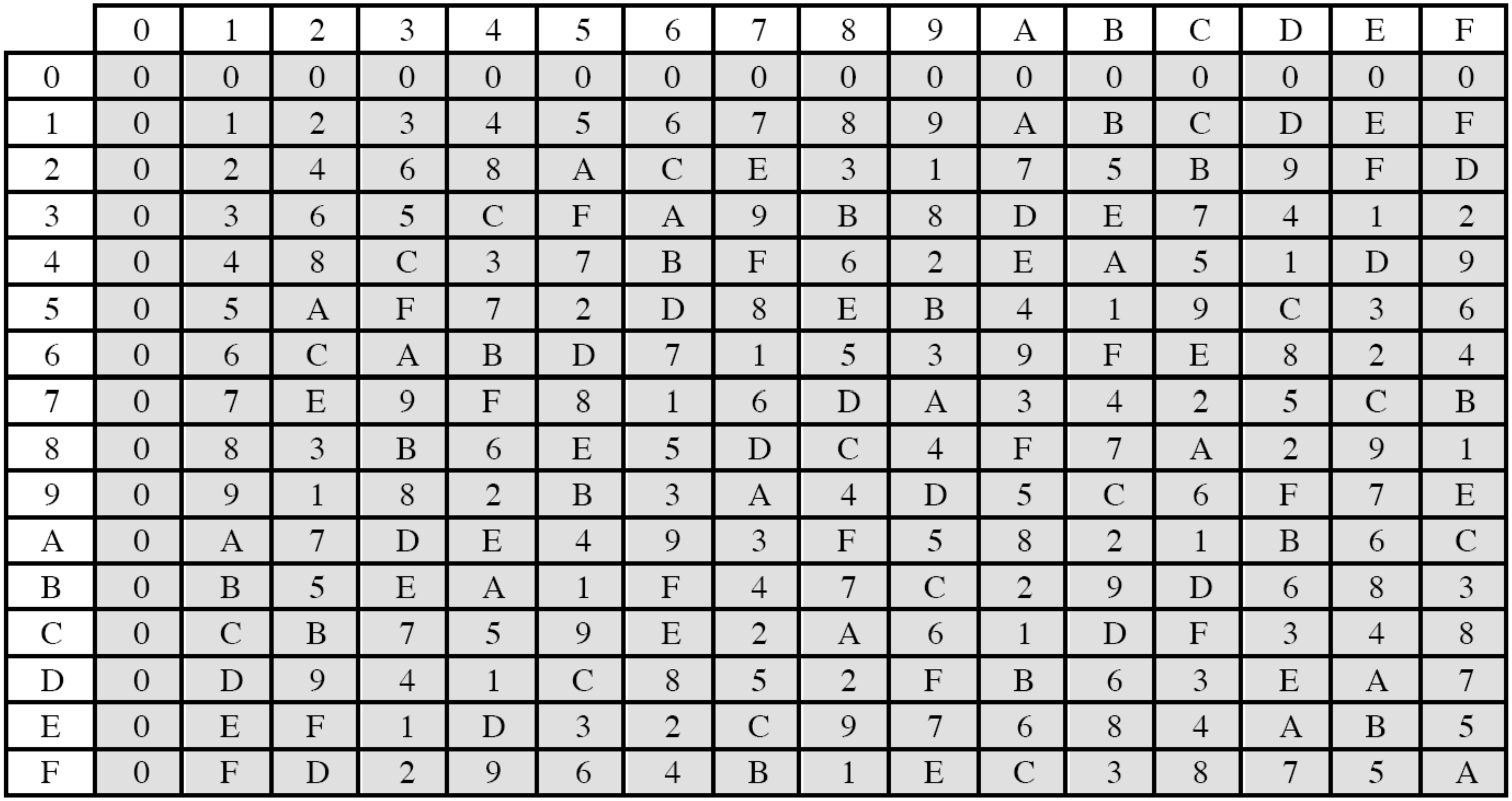
Mix Column

The mix column transformation is defined by a type of matrix multiplication on the state matrix:

$$\begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \bullet \begin{bmatrix} s_{00} & s_{01} \\ s_{10} & s_{11} \end{bmatrix} = \begin{bmatrix} s'_{00} & s'_{01} \\ s'_{10} & s'_{11} \end{bmatrix}.$$

- Addition is defined as XOR
- Multiplication is defined as the multiplication operation in– $GF\left(2^4\right)$ “Galois Field” aka finite field.

Multiplication in $GF\left(2^4\right)$



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	3	1	7	5	B	9	F	D
3	0	3	6	5	C	F	A	9	B	8	D	E	7	4	1	2
4	0	4	8	C	3	7	B	F	6	2	E	A	5	1	D	9
5	0	5	A	F	7	2	D	8	E	B	4	1	9	C	3	6
6	0	6	C	A	B	D	7	1	5	3	9	F	E	8	2	4
7	0	7	E	9	F	8	1	6	D	A	3	4	2	5	C	B
8	0	8	3	B	6	E	5	D	C	4	F	7	A	2	9	1
9	0	9	1	8	2	B	3	A	4	D	5	C	6	F	7	E
A	0	A	7	D	E	4	9	3	F	5	8	2	1	B	6	C
B	0	B	5	E	A	1	F	4	7	C	2	9	D	6	8	3
C	0	C	B	7	5	9	E	2	A	6	1	D	F	3	4	8
D	0	D	9	4	1	C	8	5	2	F	B	6	3	E	A	7
E	0	E	F	1	D	3	2	C	9	7	6	8	4	A	B	5
F	0	F	D	2	9	6	4	B	1	E	C	3	8	7	5	A

Mix Column Example

$$\begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \bullet \begin{bmatrix} 6 & 4 \\ C & 0 \end{bmatrix} = \begin{bmatrix} 6 \oplus 4 \bullet C & 4 \oplus 4 \bullet 0 \\ 4 \bullet 6 \oplus C & 4 \bullet 4 \oplus 0 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 7 & 3 \end{bmatrix}.$$

Key Expansion

Original 16-bit key is represented as shown on slide 63 $w_0 : w_1$ using two 8-bit "words."

The four expansion words are found as follows:

$$w_2 = w_0 \oplus RCON(1) \oplus SubNib(RotNib(w_1))$$

$$w_3 = w_1 \oplus w_2$$

$$w_4 = w_2 \oplus RCON(2) \oplus SubNib(RotNib(w_3))$$

$$w_5 = w_3 \oplus w_4$$

where $RCON(1) = 10000000$ and $RCON(2) = 00110000.$

Key Expansion (continued)

- The function ROTNIB does a one nibble (4bits) circular rotation on the input 8-bit word which has the effect of swapping the first 4 bits with the second 4 bits.
- The function SUBNIB performs nibble substitution on the input 8-bit word using the S-box.

Key Expansion Example

From:

$$w_2 = w_0 \oplus RCON(1) \oplus SubNib\left(RotNib\left(w_1\right)\right)$$

$$w_3 = w_1 \oplus w_2$$

$$w_4 = w_2 \oplus RCON(2) \oplus SubNib\left(RotNib\left(w_3\right)\right)$$

$$w_5 = w_3 \oplus w_4$$

with initial key $2D55 = 0010\ 1101\ 0101\ 0101 = w_0 w_1$ we have:

$$w_2 = 00101101 \oplus 10000000 \oplus SubNib(01010101)$$

$$= 00101101 \oplus 10000000 \oplus 00010001 = 10111100$$

$$w_3 = 10111100 \oplus 01010101 = 11101001$$

$$w_4 = 10111100 \oplus 00110000 \oplus SubNib(10011110)$$

$$= 10111100 \oplus 00110000 \oplus 00101111 = 10100011$$

$$w_5 = 10100011 \oplus 11101001 = 01001010.$$

Homework 2

- Use S-AES to encrypt the message FACE using the key D24E.
- Work on your own (follow the slides).
- Can be done without programming.

Location of Encryption Device

- **Link encryption:**
  - A lot of encryption devices
  - High level of security
  - Decrypt each packet at every switch
- **End-to-end encryption**
  - The source encrypts and the receiver decrypts
  - Payload encrypted
  - Header in the clear
- **High Security:** Both link and end-to-end encryption are needed (see Figure 2.9)

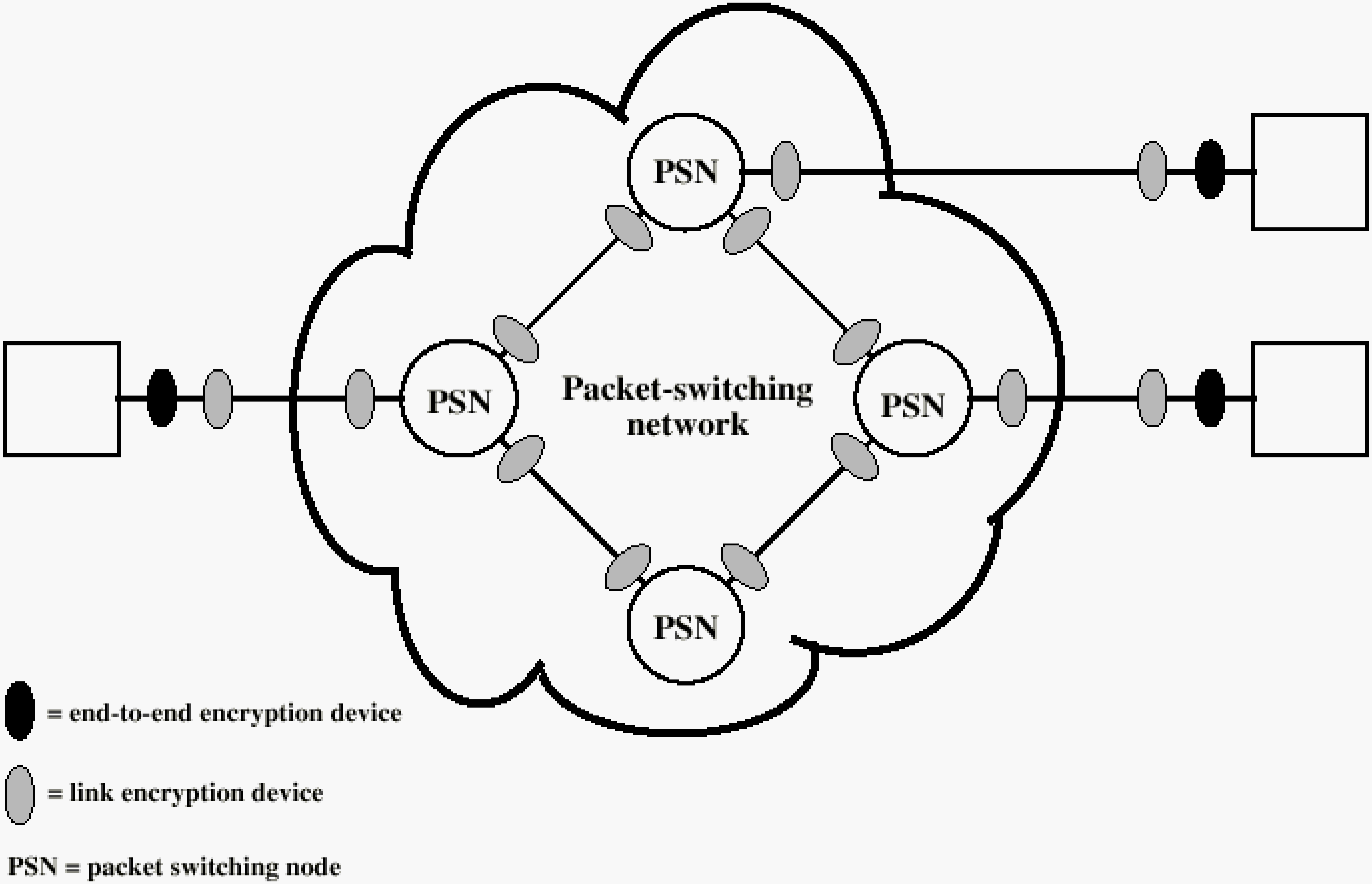

PSN
PSN
Packet-switching network
PSN
PSN
● = end-to-end encryption device
● = link encryption device
PSN = packet switching node

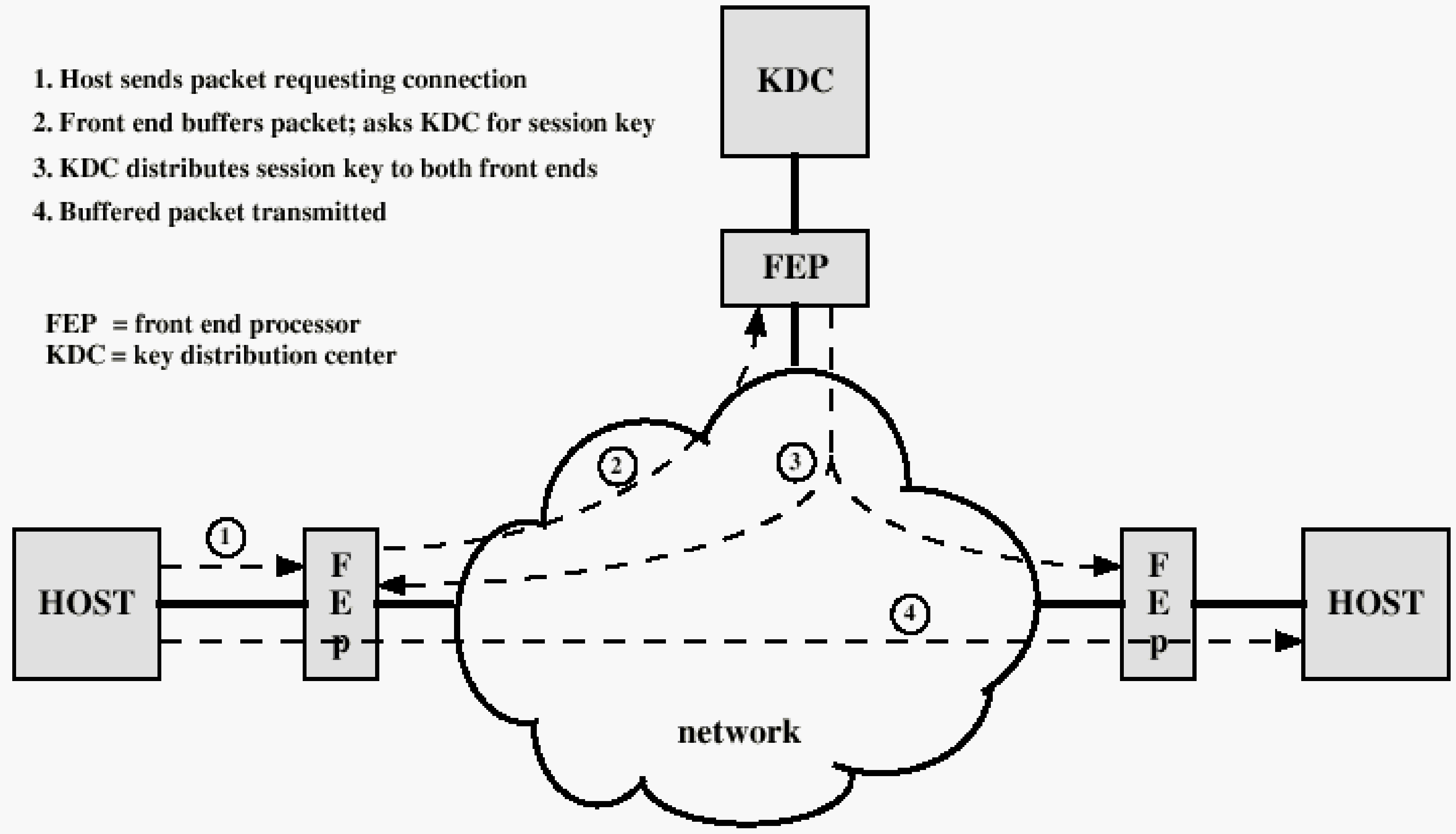
Figure 2.9 Encryption Across a Packet-Switching Network

Key Distribution

1. A key could be selected by A and physically delivered to B.
2. A third party could select the key and physically deliver it to A and B.
3. If A and B have previously used a key, one party could transmit the new key to the other, encrypted using the old key.
4. If A and B each have an encrypted connection to a third party C, C could deliver a key on the encrypted links to A and B.

Key Distribution (See Figure 2.10)

- **Session key:**
  - Data encrypted with a one-time session key.At the conclusion of the session the key is destroyed
- **Permanent key:**
  - Used between entities for the purpose of distributing session keys



1. Host sends packet requesting connection
2. Front end buffers packet; asks KDC for session key
3. KDC distributes session key to both front ends
4. Buffered packet transmitted

FEP = front end processor
KDC = key distribution center

Figure 2.10 Automatic Key Distribution for Connection-Oriented Protocol

Recommended Reading

- Stallings, W. *Cryptography and Network Security: Principles and Practice, 2nd edition*. Prentice Hall, 1999
- Scneier, B. *Applied Cryptography*, New York: Wiley, 1996
- Mel, H.X. Baker, D. *Cryptography Decrypted*. Addison Wesley, 2001
- Ferguson, Niels and Schneier, Bruce *Practical Cryptography*, Wiley, 2003

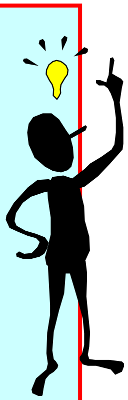
Public Key Cryptography

symmetric key crypto

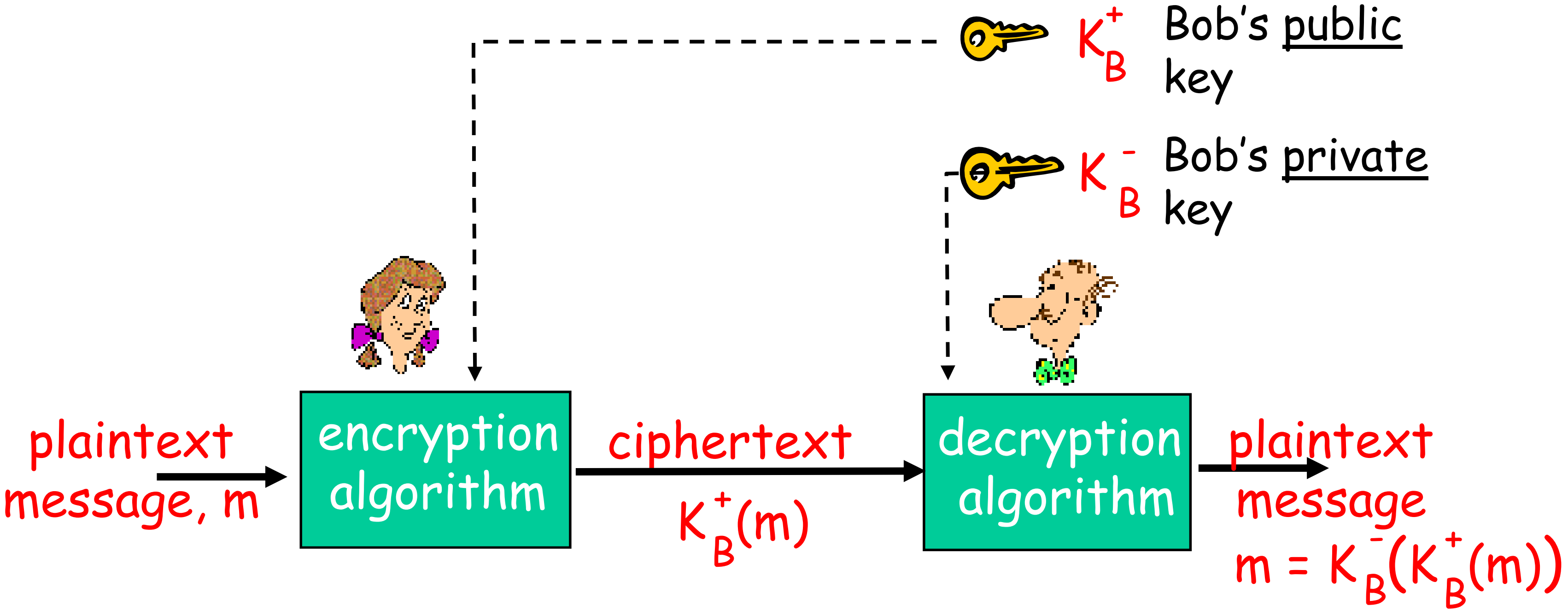
- ❐ requires sender, receiver know shared secret key
- ❐ Q: how to agree on key in first place (particularly if never "met")?

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 $K_B^+(\cdot)$ 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

Math Foundations for Public Key Cryptography

This material motivated by Ferguson/Schneier
Practical Cryptography

- Divisibility and Primes
- Generating Primes
- Computations Modulo a Prime
- Large Primes

Divisibility and Primes (Review)

1. We write $a \mid b$ and say "a divides b" if we can divide b by a with no remainder.
2. We say that a number is "prime" if its only divisors are 1 and itself.
3. Any number larger than 1 that is NOT a prime is a "composite" number.
4. Divisibility is "transitive" that is: If $a \mid b$ and $b \mid c$, then $a \mid c$. (Prove this.)
5. Lemma 1: Let n be a positive number greater than 1. Let d be the smallest divisor of n that is greater than 1. Then d is prime.

   We will consider the proof of this next. Lemma 1 is used to prove Theorem 1.
6. Theorem 1. (Due to Euclid). There are an infinite number of primes.

Proof of Lemma

First students should prove that d is well-defined. That is, show that every positive integer n has a smallest divisor. Next suppose this number d is NOT prime. By defn of prime then d is divisible by a number e that satisfies: $1 < e < d$. Any such e would be the smallest divisor of n because divisibility is transitive. This is a contradiction. □

Proof of Theorem 1

Assume, on the contrary, that the number of primes is finite and that $p_1, p_2, ..., p_k$ is the list of all primes. Let $n = p_1 p_2 \cdots p_k + 1$. Let d be the smallest divisor of n that is greater than 1. We know from Lemma 1 that d is prime. We also know that $d \mid n$. None of the primes $p_1 \cdots p_k$ divides n. Thus, d is a prime that is not in our list of all primes. This is a contradiction so the number of primes must be infinite. □

Fundamental Theorem of Arithmetic

Any integer greater than 1 can be written as the product of primes and this representation is unique except for the order of the primes.

Example: $147 = 7^2 \times 3$

$$700 = 7 \times 2^2 \times 5^2.$$

Generating Small Primes (The Sieve of Eratosthenes)

- Generate all primes less than a positive integer n greater than 2 but less than a max determined by available memory.
- Example: n := 200000

- $\text{init} := 1, 2..\, n$
  $b_{\text{init}} := 0$

Must initialize array b to pass to function f as a parameter.

- Define function f(n,b), then call.

```
$f(n, b) :=$
    $i \leftarrow 2$
    for $m \in 1, 2 .. n$
        $b_m \leftarrow 1$
    while $i^2 \le n$
        $top \leftarrow \mathrm{floor}\left(\frac{n}{i}\right)$
        for $j \in 2, 3 .. top$
            $k \leftarrow j \cdot i$
            $b_k \leftarrow 0$
        $i \leftarrow i + 1$
        while $b_i = 0$
            $i \leftarrow i + 1$
    $b$
```

First prime is 2.

Sets a flag indicating all numbers less than n MAY be prime.

This i is a prime set all multiples of it (less than n) to composite.

A sample from the returned array of 200,000 elements.

$f(n, b) =$

	0
187905	0
187906	0
187907	1
187908	0
187909	1
187910	0
187911	0
187912	0
187913	0
187914	0
187915	0
187916	0
187917	0
187918	0
187919	0
187920	0

Computation Modulo a Prime

Let p be a prime (for much of this p could also be a composite number) and recall that for any integer r the number $r \pmod{p}$ is the remainder obtained after dividing r by p. Thus, $100 \pmod{13} = 9$. Note that the only distinct values (mod p) are $0,1,2,\ldots,p-1$. (Negative results can also be converted into this range.)

Recall also that the greatest common divisor or "gcd" of positive integers a and b is the largest integer $k > 0$ such that $k \mid a$ and $k \mid b$.
Example: $\gcd(12,16) = 4$.
Also recall that the least common multiple or "lcm" of a and b is the smallest integer $k > 0$ such that $a \mid k$ and $b \mid k$.
Example: $\mathrm{lcm}(12,16) = 48$.

It is well known that $\gcd(a,b) \times \mathrm{lcm}(a,b) = ab$.

Euclid's gcd Algorithm

Given two non-negative integers, the notation (a,b) implies that $0 \leq a < b.$

```
$(x, y) \leftarrow (a,b)$
While $x > 0$
        $(x, y) \leftarrow (y \bmod x, x)$
$\gcd(a,b) \leftarrow y.$
```

Example: $(21,30) \rightarrow (9,21) \rightarrow (3,9) \rightarrow (0,3) \Rightarrow \gcd(21,30) = 3.$

Definition: If $\gcd(a,b) = 1,$ then a and b are said to be relatively prime.

Extended Euclid Algorithm

It is well known that there exist integers x and y such that $\gcd(a,b) = ax + by$. In the previous example, we want $3 = \gcd(21,30) = 21x + 30y$. If we write down our intermediate steps,

1. $30 = 1 \times 21 + 9$
2. $21 = 2 \times 9 + 3$
3. $9 = 3 \times 3 + 0$, which implies $\gcd(21,30) = 3$.

From step 2 we have $3 = 21 - 2 \times 9$. We use step 1 to substitute for the 9. This yields $3 = 21 - 2 \times (30 - 1 \times 21) = 3 \times 21 - 2 \times 30$.

Thus, $3 = 3 \times 21 - 2 \times 30$. The extended-gcd algorithm produces the two sought integers x and y. This, in turn, will enable division modulo a specified value, which is a critical step in the RSA algorithm.

RSA: Choosing keys

1. Choose two large prime numbers *p, q.*
   (e.g., 2000+ bits for their product)
   I used the small primes approach to find
   p = 39607 and q = 78517.

2. Compute $n = pq$, $z = (p-1)(q-1)$

   n = 3109822819

   Z = 3109704696

RSA Keys Continued

3. Choose *e* (with *e<n)* that has no common factors with z. (*e, z* are "relatively prime").

These factors are easy to guess. My first guess was 47785.

It happens that $\gcd(47785,3109704696) = 1$.

4. Choose *d* such that *ed-1* is exactly divisible by *z*. (in other words: *ed* mod *z* = *1*).

This is the difficult step and requires the extended Euclidean algorithm.

Finding d Using Ext Euclid Alg

Must choose d so that ed = 1 (mod z):

Find gcd(47785,z) where 47785 is a guess.

$3109704696 = 65077 \times 47785 + 251$

$\Rightarrow 47785 = 190 \times 251 + 95$

$\Rightarrow 251 = 2 \times 95 + 61$

$\Rightarrow 95 = 1 \times 61 + 34$

$\Rightarrow 61 = 1 \times 34 + 27$

$\Rightarrow 34 = 1 \times 27 + 7$

$\Rightarrow 27 = 3 \times 7 + 6$

$\Rightarrow 7 = 1 \times 6 + 1.$

Relatively Prime

Reversing Steps to Find 1=ax+by

sub 1: 1 = 7 - 1x6

sub 6: = 7 - 1x(27-3x7) = 4x7 - 1x27

sub 7: = 4x(34 - 1x27) -1x27 = 4x34 - 5x27

sub 27: = 4x34 - 5x(61 - 1x34) = 9x34 - 5x61

sub 34: = 9x(95 - 1x61) - 5x61 = 9x95 - 14x61

sub 61: = 9x95 - 14x(251 - 2x95) = 37x95 - 14x251

sub 95: = 37x(47785 - 190x251) - 14x251 = 37x47785 - 7044x251

sub 251: = 37x47785 - 7044(3109704696 - 65077x47785)
= 458402425x47785 - 7044x3109704696

Because second term on right-hand side is zero mod z, it follows that 458402425x47785=1 mod z. Let d = 458402425.

RSA Keys Continued

5. *Public* key is $\underbrace{(n,e)}_{K_B^+}$. *Private* key is $\underbrace{(n,d)}_{K_B^-}$.

n = 3109822819

e = 47785

d = 458402425

Trudy knows that Bob's public key is (*n*,*e*) and would like to find *d*. She knows *ed* mod *z* =1 but does not know *z*. To find *z* she needs *p* and *q* and so must factor *n*.

Security depends on difficulty of factoring *n* (which has 2000+ digits).

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

	letter	m	m^e	$c = m^e \bmod n$
encrypt:	l	12	1524832	17

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

Why does RSA work?

RSA depends on a well-known result from number theory:

If p and q are prime and $n = pq$, then

$x^y \bmod n = x^{y \bmod (p-1)(q-1)} \bmod n$ for any positive integers x and y.

Example:

Let $n = 3 \times 5 = 15$ be the product of two primes. Choose $x = 2$ and $y = 12$.

Then result says that $2^{12} \bmod 15 = 2^{12 \bmod 8} \bmod 15$. Correct?

LHS $= 2^{12} \bmod 15 = 4096 \bmod 15 = 1$.

RHS $= 2^{12 \bmod 8} \bmod 15 = 2^4 \bmod 15 = 16 \bmod 15 = 1$.

RSA: Why does $m = (m^e \bmod n)^d \bmod n$?

Recall 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:

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

Result is the same!

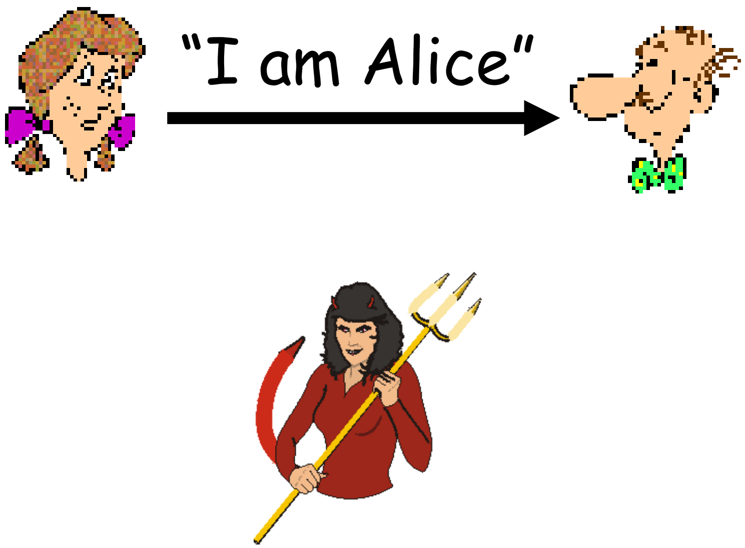
Security Overview

- What is network security?
- Principles of cryptography
- Security Requirements
- Key Distribution and certification
- Access control: firewalls
- Attacks and counter measures

Authentication

Goal: Bob wants Alice to "prove" her identity to him

Protocol ap1.0: Alice says "I am Alice"

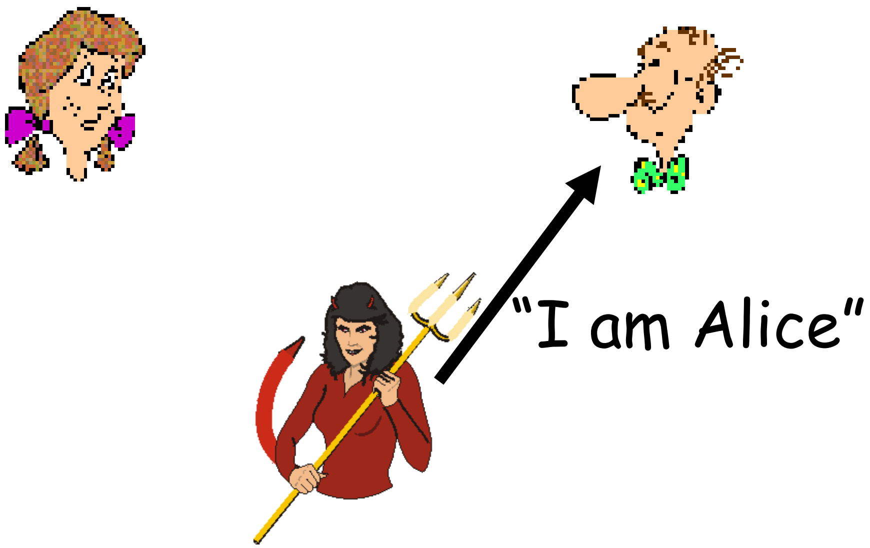


Failure scenario??

Authentication

Goal: Bob wants Alice to "prove" her identity to him

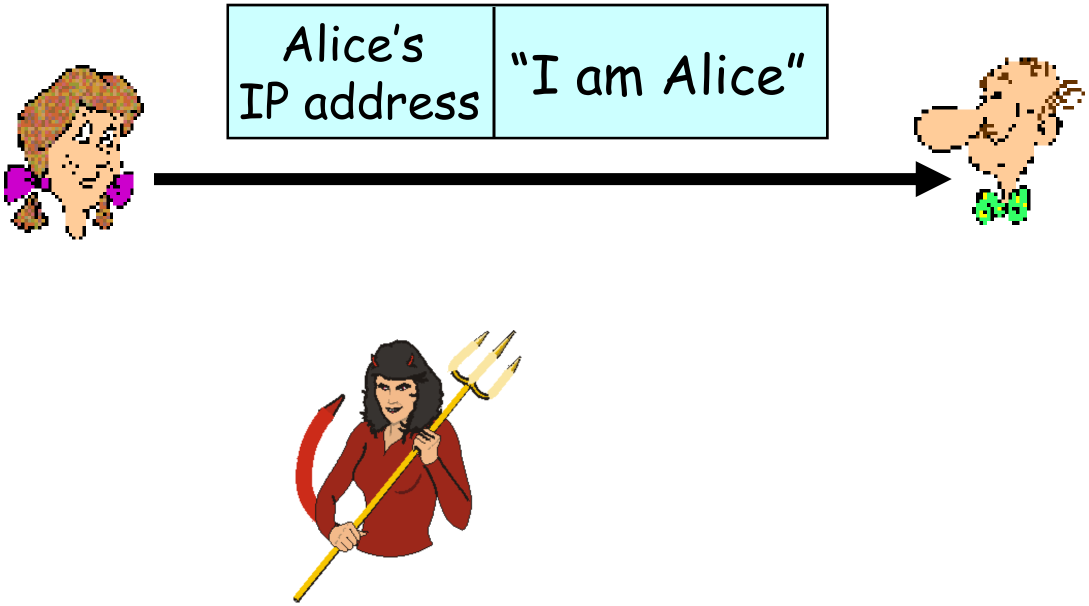
Protocol ap1.0: Alice says "I am Alice"



in a network,
Bob can not "see"
Alice, so Trudy simply
declares
herself to be Alice

Authentication: another try

Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address

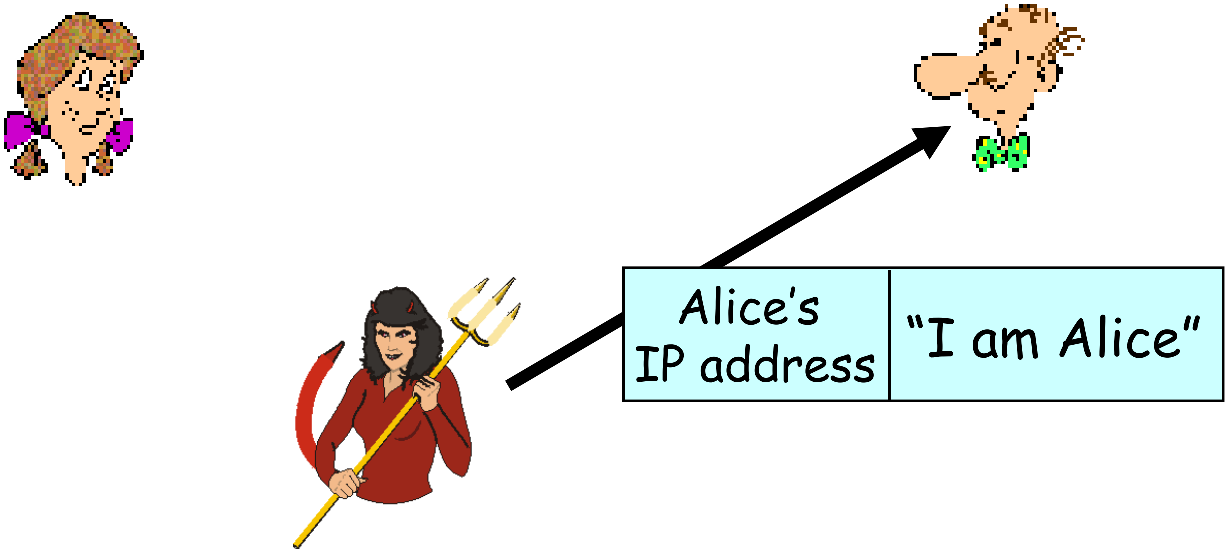


Alice's IP address	"I am Alice"

Failure scenario??

Authentication: another try

Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address



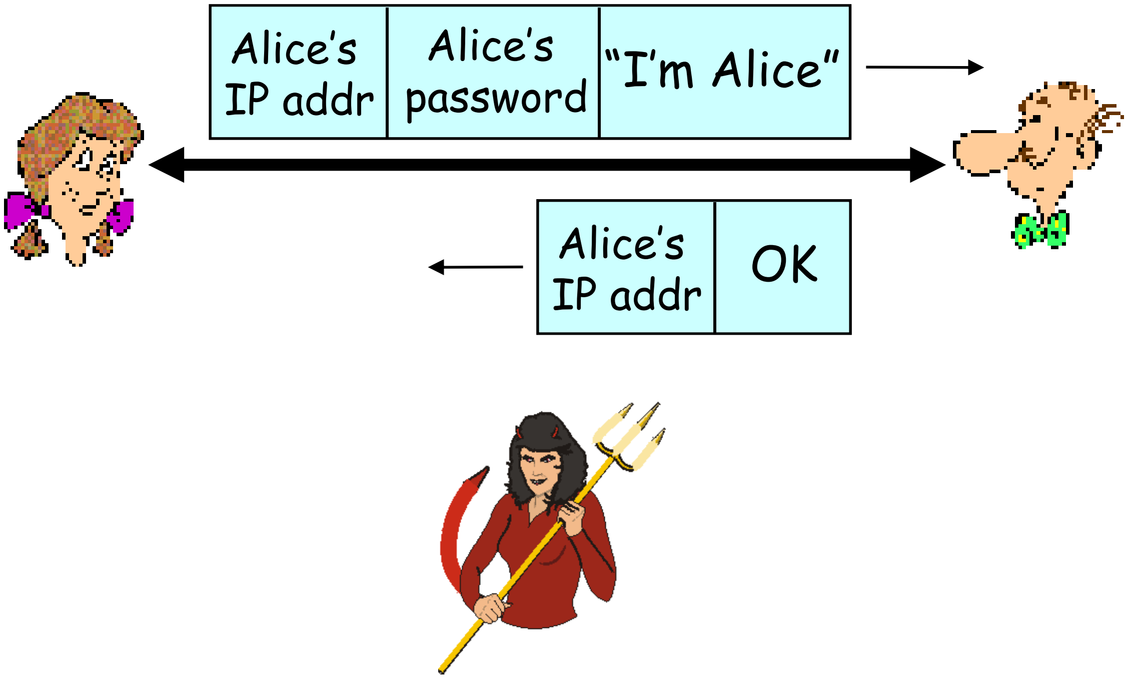
Alice's IP address

"I am Alice"

Trudy can create a packet "spoofing" Alice's address

Authentication: another try

Protocol ap3.0: Alice says "I am Alice" and sends her unencrypted secret password to "prove" it.



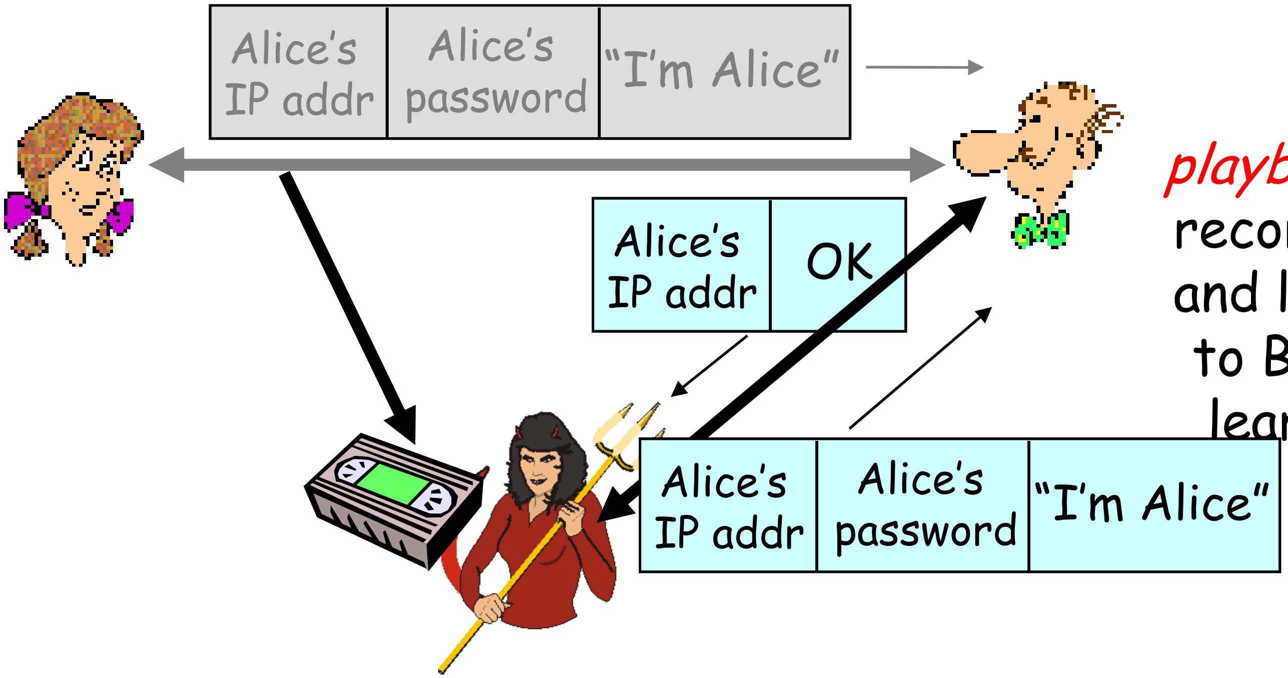
Alice's IP addr | Alice's password | "I'm Alice"

Alice's IP addr | OK

Failure scenario??

Authentication: another try

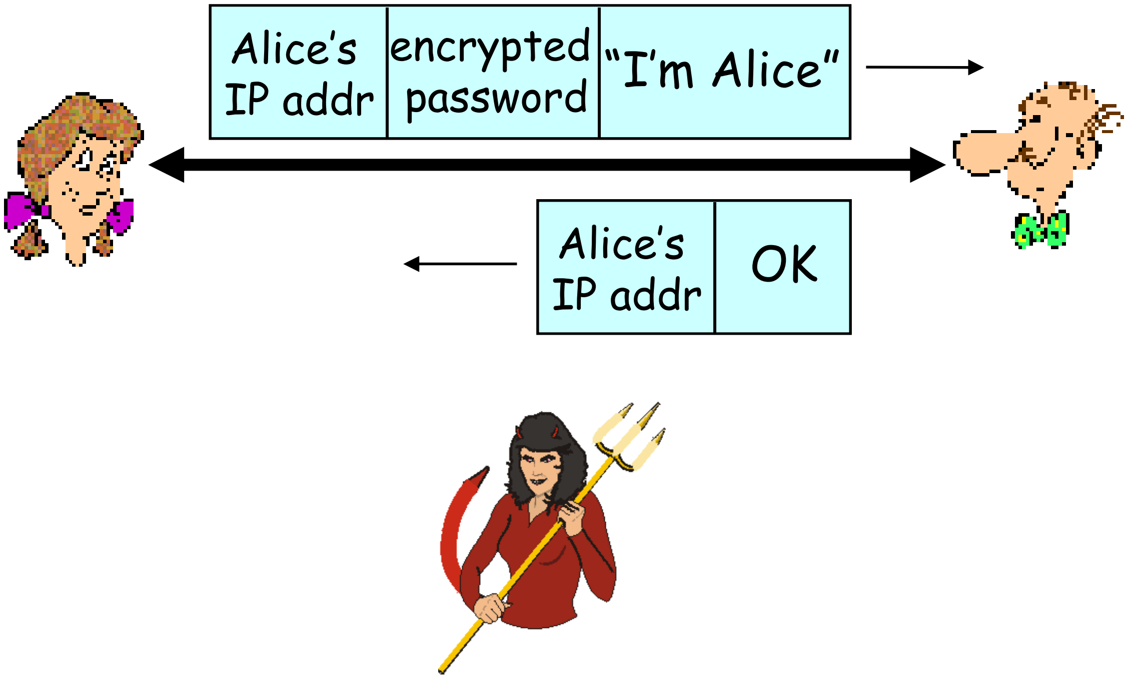
Protocol ap3.0: Alice says "I am Alice" and sends her unencrypted secret password to "prove" it.



playback attack: Trudy records Alice's packet and later plays it back to Bob. She can also learn the password.

Authentication: yet another try

Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



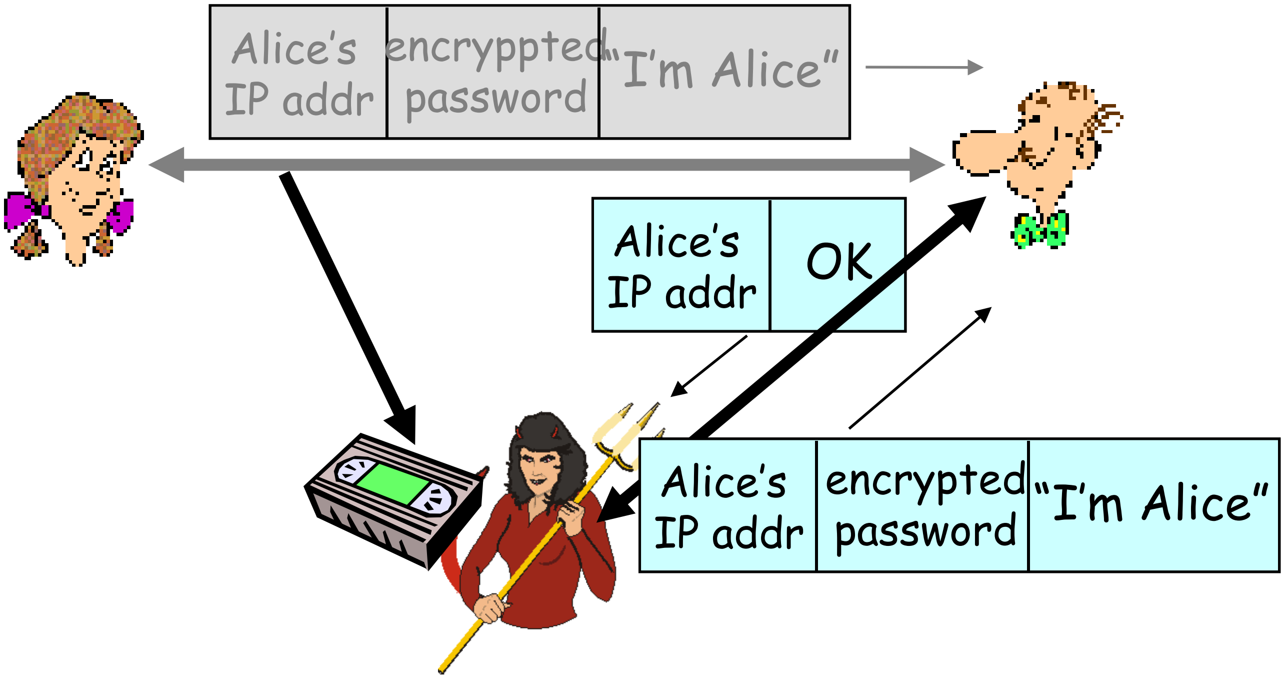
Alice's IP addr | encrypted password | "I'm Alice"

Alice's IP addr | OK

Failure scenario??

Authentication: another try

Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



Alice's IP addr | encryppted password | "I'm Alice"

Alice's IP addr | OK

Alice's IP addr | encrypted password | "I'm Alice"

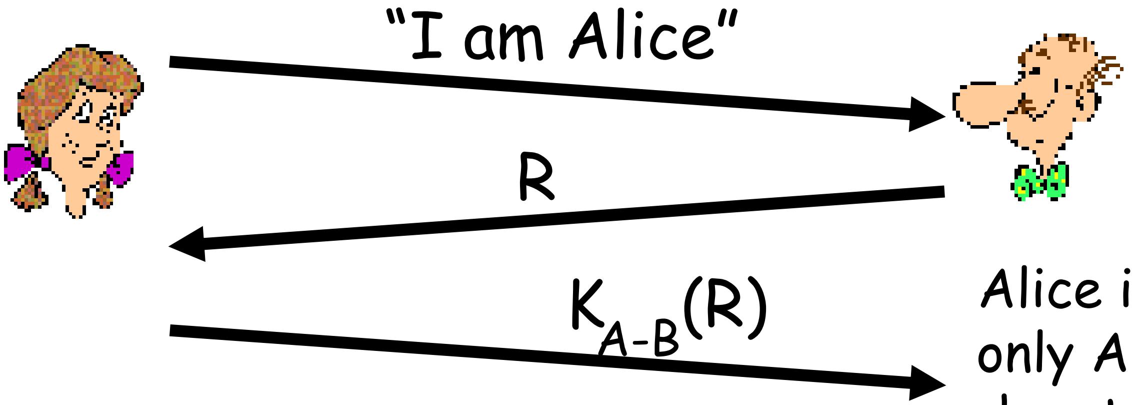
record
and
playback
still works!

Authentication: yet another try

Goal: avoid playback attack

Nonce: number (R) used only *once –in-a-lifetime*

ap4.0: to prove Alice "live", Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key



Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

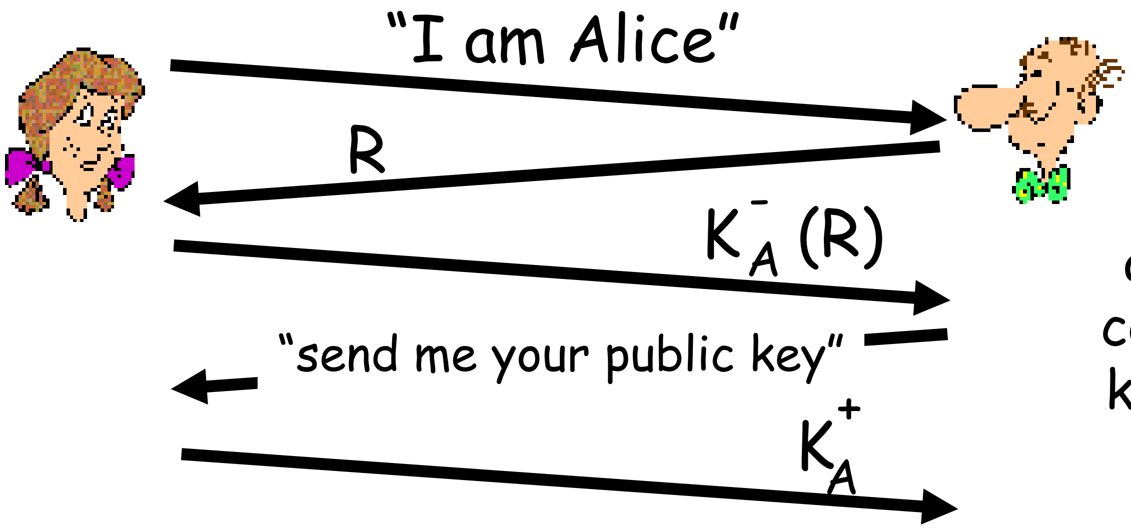
Failures, drawbacks?

Authentication: ap5.0

ap4.0 requires shared symmetric key

- can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography



"I am Alice"

R

$K_A^-(R)$

"send me your public key"

K_A^+

Bob computes

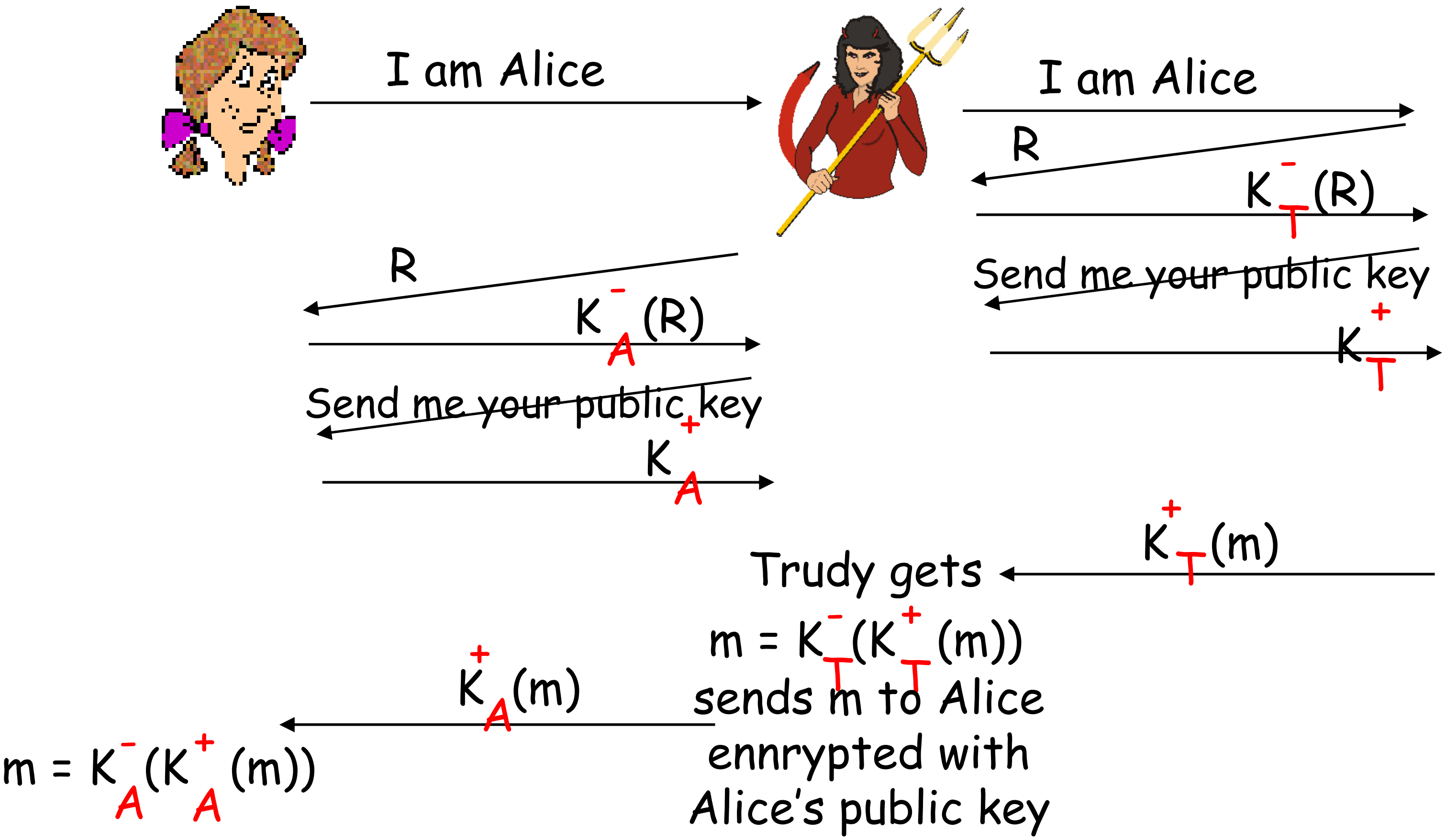
$K_A^+(K_A^-(R)) = R$

and knows only Alice could have the private key, that encrypted R such that

$K_A^+(K_A^-(R)) = R$

ap5.0: security hole

Man (woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



I am Alice

I am Alice

R

$K_T^-(R)$

Send me your public key

K_T^+

R

$K_A^-(R)$

Send me your public key

K_A^+

$K_T^+(m)$

Trudy gets

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

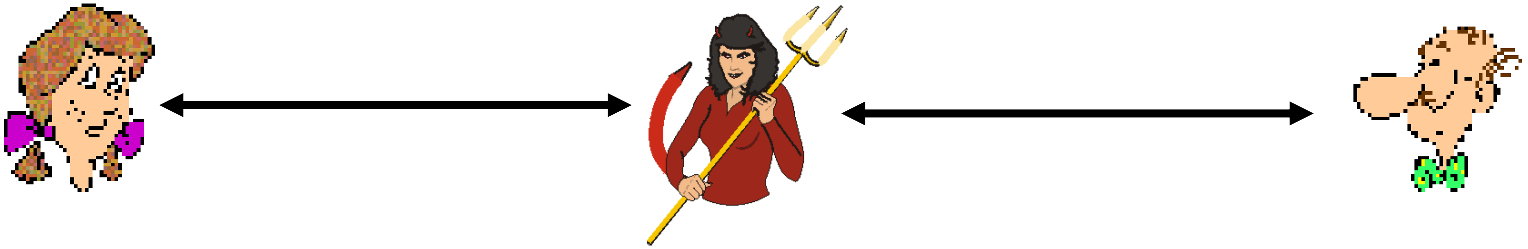
sends m to Alice ennrypted with Alice's public key

$K_A^+(m)$

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

ap5.0: security hole

Man (woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



Difficult to detect:

- Bob receives everything that Alice sends, and vice versa. (e.g., so Bob, Alice can meet one week later and recall conversation)
- problem is that Trudy receives all messages as well!

Integrity: Digital Signatures

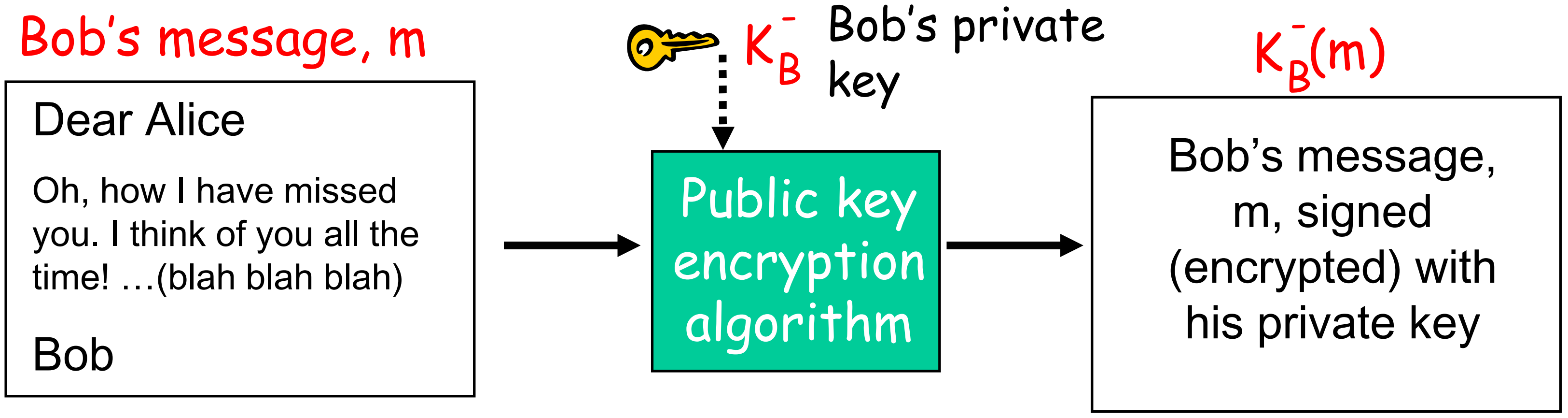
Cryptographic technique analogous to hand-written signatures.

- sender (Bob) digitally signs document, establishing he is document owner/creator.
- verifiable, nonforgeable: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed the document

Digital Signatures

Simple digital signature for message m:

- Bob signs m by encrypting with his private key K_B^-, creating "signed" message, $K_B^-(m)$



Bob's message, m

Dear Alice

Oh, how I have missed you. I think of you all the time! …(blah blah blah)

Bob

K_B^- Bob's private key

Public key encryption algorithm

$K_B^-(m)$

Bob's message, m, signed (encrypted) with his private key

Digital Signatures (more)

- ❐ Suppose Alice receives msg m, digital signature $K_B^-(m)$
- ❐ Alice verifies m signed by Bob by applying Bob's public key K_B^+ to $K_B^-(m)$ then checks $K_B^+(K_B^-(m)) = m$.
- ❐ If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key.

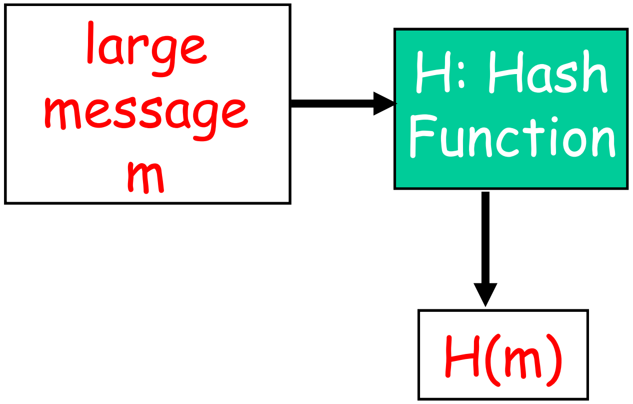
Alice thus verifies that:

- ✓ Bob signed m.
- ✓ No one else signed m.
- ✓ Bob signed m and not m'.

Non-repudiation:

- ✓ Alice can take m, and signature $K_B^-(m)$ to court and prove that Bob signed m.

Message Digests



Computationally expensive to public-key-encrypt long messages

Goal: fixed-length, easy-to-compute digital "fingerprint"

- apply hash function H to *m*, get fixed size message digest, *H(m)*.

Hash function properties:

- many-to-1
- produces fixed-size msg digest (fingerprint)
- given message digest x, computationally infeasible to find m such that x = H(m)

Internet checksum: poor crypto hash function

Internet checksum has some properties of hash function:

- ✓ produces fixed length digest (16-bit sum) of message
- ✓ is many-to-one

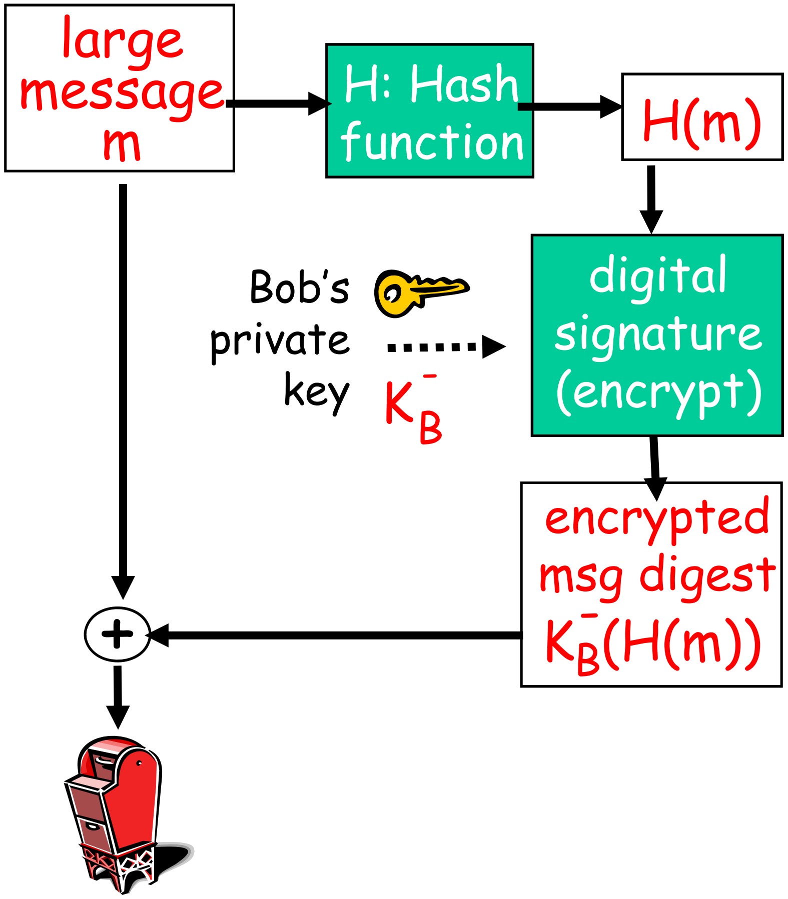
But given message with given hash value, it is easy to find another message with same hash value:

message	ASCII format	message	ASCII format
I O U 1	49 4F 55 31	I O U <u>9</u>	49 4F 55 <u>39</u>
0 0 . 9	30 30 2E 39	0 0 . <u>1</u>	30 30 2E <u>31</u>
9 B O B	39 42 D2 42	9 B O B	39 42 D2 42
	B2 C1 D2 AC		B2 C1 D2 AC

different messages but identical checksums!

Digital signature = signed message digest

Bob sends digitally signed message:



large message m

H: Hash function

H(m)

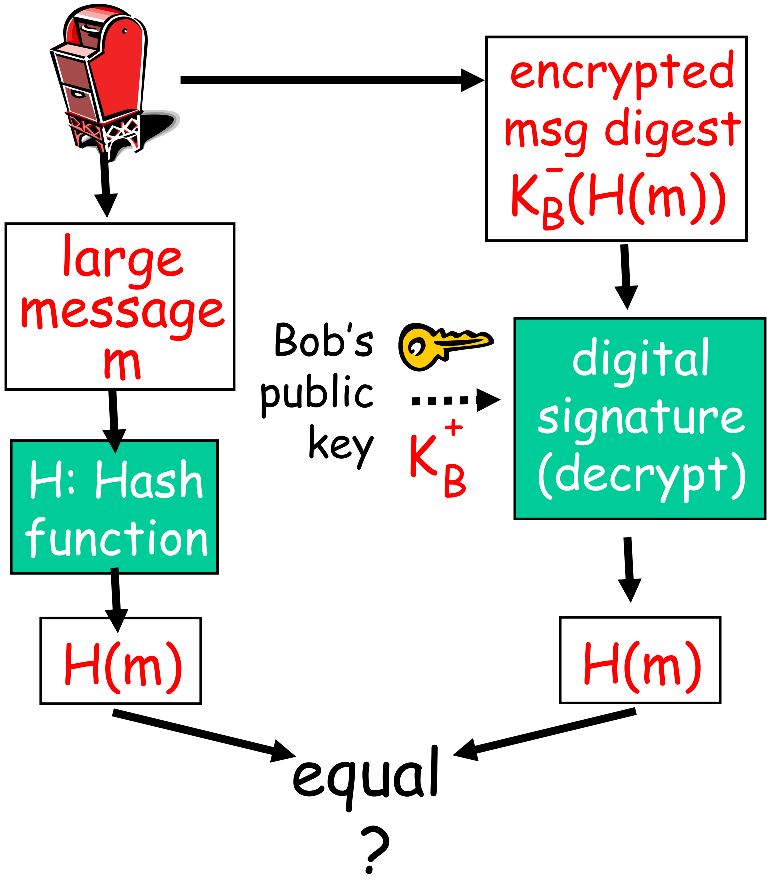
Bob's private key K_B^-

digital signature (encrypt)

encrypted msg digest $K_B^-(H(m))$

+

Alice verifies signature and integrity of digitally signed message:



encrypted msg digest $K_B^-(H(m))$

large message m

H: Hash function

H(m)

Bob's public key K_B^+

digital signature (decrypt)

H(m)

equal ?

Hash Function Algorithms

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process.
  - arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x.
- SHA-1 is also used.
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest

Chapter 7 roadmap

- What is network security?
- Principles of cryptography
- Security Requirements
- Key distribution and certification
- Access control: firewalls
- Attacks and counter measures
- Security in many layers

Trusted Intermediaries

Symmetric key problem:

- How do two entities establish shared secret key over network?

Solution:

- trusted key distribution center (KDC) acting as intermediary between entities

Public key problem:

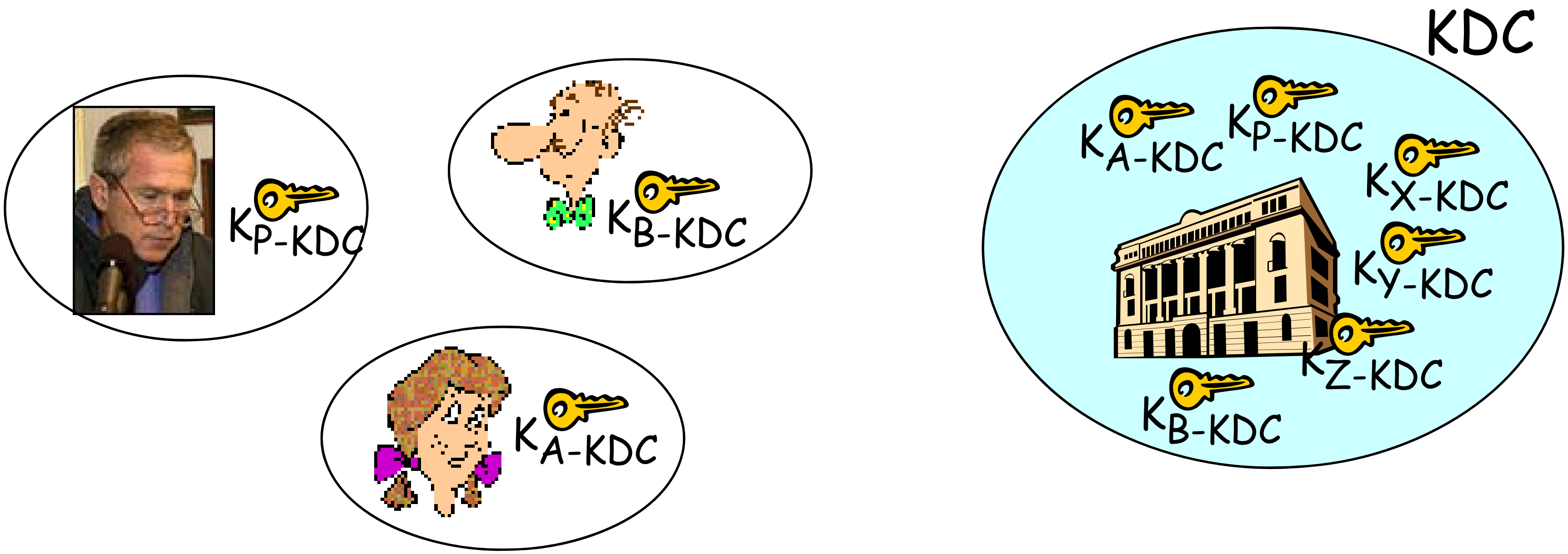
- When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Trudy's?

Solution:

- trusted certification authority (CA)

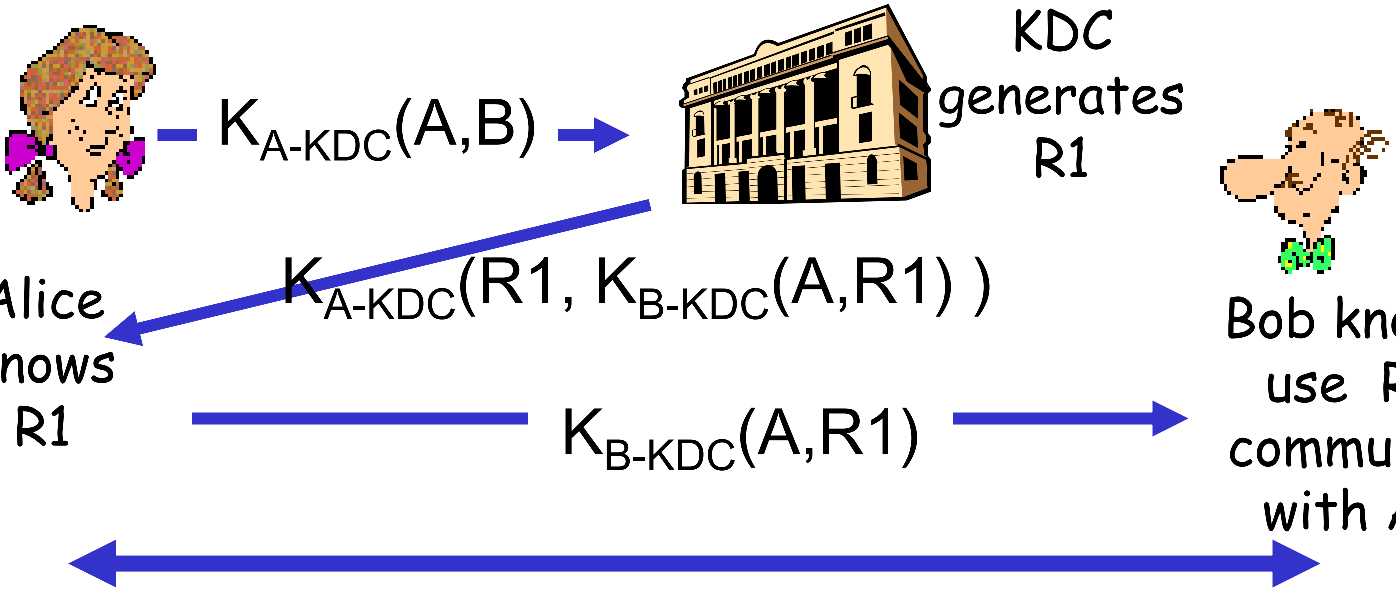
Key Distribution Center (KDC)

- Alice, Bob need shared symmetric key.
- KDC: server shares different secret key with *each* registered user (many users)
- Alice, Bob know own symmetric keys, $K_{A\text{-}KDC}$ $K_{B\text{-}KDC}$, for communicating with KDC.



Key Distribution Center (KDC)

Q: How does KDC allow Bob, Alice to determine shared symmetric secret key to communicate with each other?



$K_{A-KDC}(A,B)$

KDC generates R1

$K_{A-KDC}(R1, K_{B-KDC}(A,R1))$

Alice knows R1

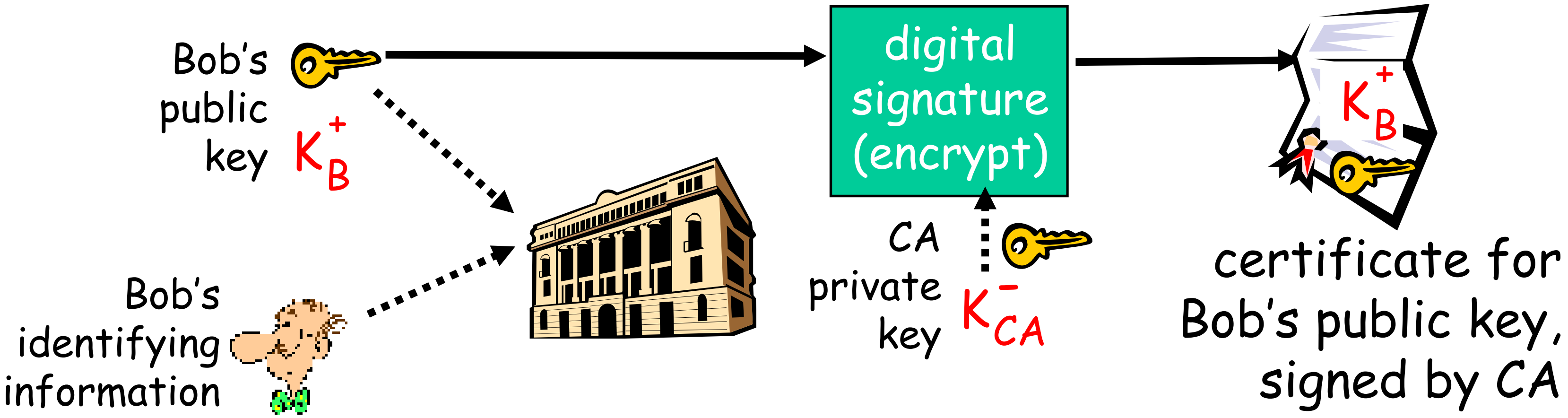
$K_{B-KDC}(A,R1)$

Bob knows to use R1 to communicate with Alice

Alice and Bob communicate: using R1 as *session key* for shared symmetric encryption

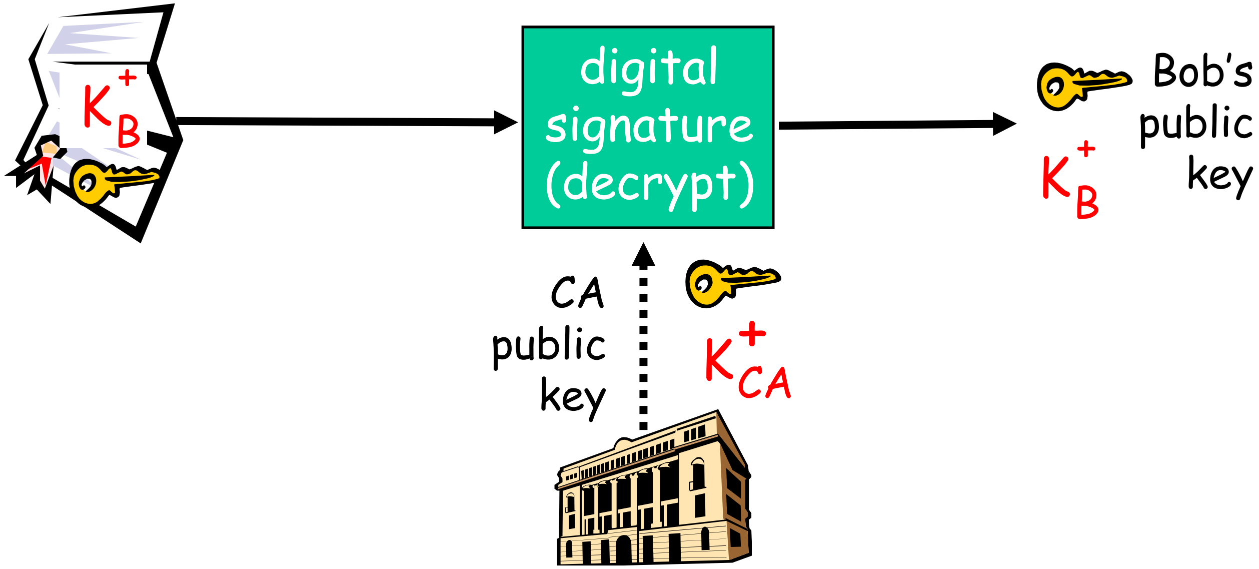
Certification Authorities

- Certification authority (CA): binds public key to particular entity, E.
- E (person, router) registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - certificate containing E's public key digitally signed by CA – CA says "this is E's public key"


Bob's public key
K_B^+
Bob's identifying information
digital signature (encrypt)
CA private key
K_{CA}^-
K_B^+
certificate for Bob's public key, signed by CA

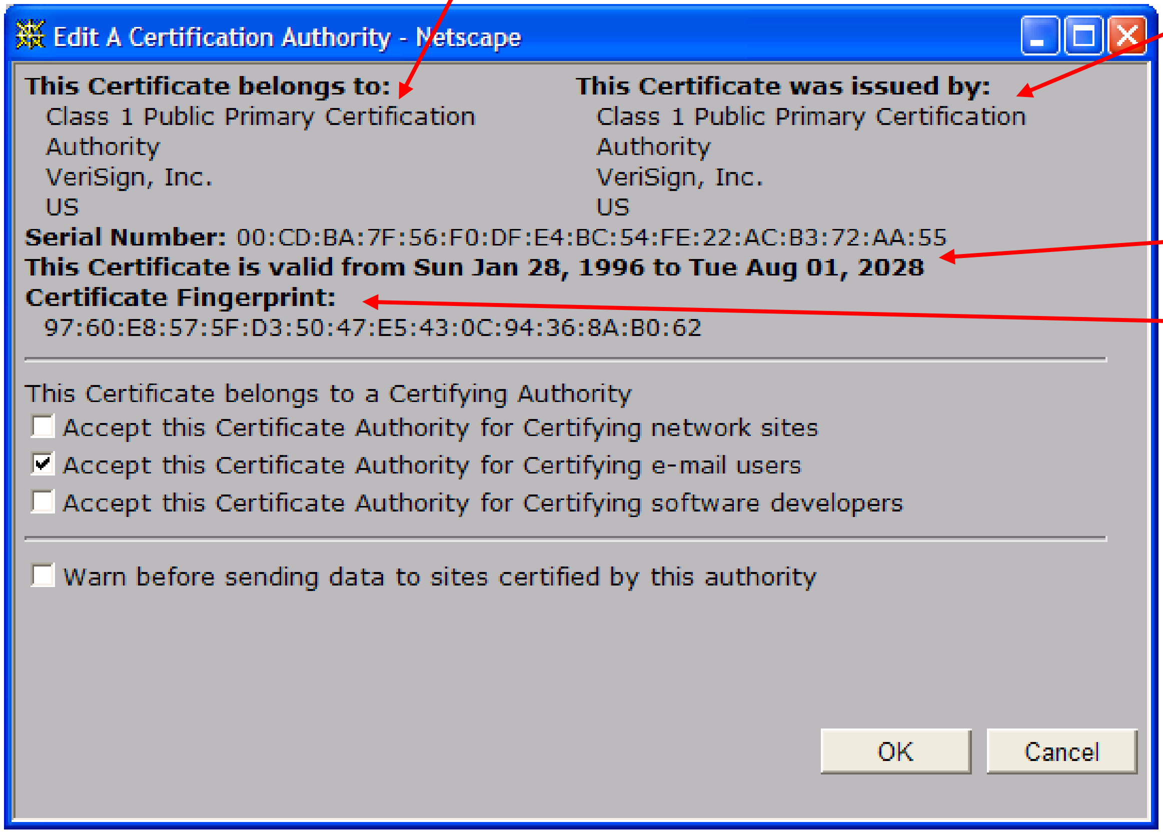
Certification Authorities

- When Alice wants Bob's public key:
  - gets Bob's certificate (Bob or elsewhere).
  - apply CA's public key to Bob's certificate, get Bob's public key



A certificate contains:

- Serial number (unique to issuer)
- info about certificate owner, including algorithm and key value itself (not shown)



This Certificate belongs to:
Class 1 Public Primary Certification
Authority
VeriSign, Inc.
US

This Certificate was issued by:
Class 1 Public Primary Certification
Authority
VeriSign, Inc.
US

Serial Number: 00:CD:BA:7F:56:F0:DF:E4:BC:54:FE:22:AC:B3:72:AA:55
This Certificate is valid from Sun Jan 28, 1996 to Tue Aug 01, 2028
Certificate Fingerprint:
97:60:E8:57:5F:D3:50:47:E5:43:0C:94:36:8A:B0:62

This Certificate belongs to a Certifying Authority
□ Accept this Certificate Authority for Certifying network sites
☑ Accept this Certificate Authority for Certifying e-mail users
□ Accept this Certificate Authority for Certifying software developers

□ Warn before sending data to sites certified by this authority

- info about certificate issuer
- valid dates
- digital signature by issuer

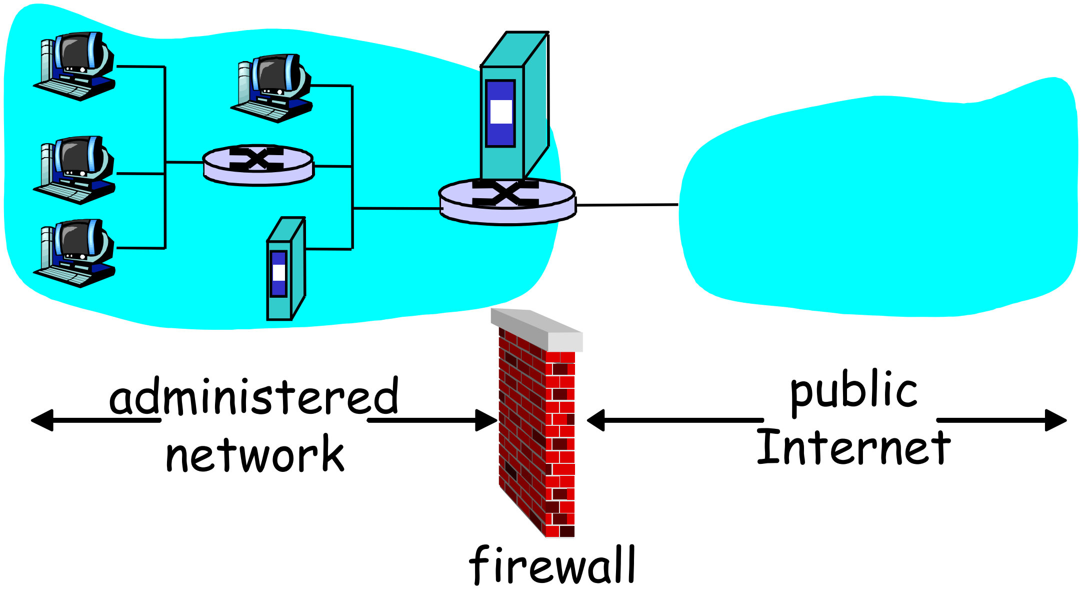
Security Overview

- What is network security?
- Principles of cryptography
- Security Requirements
- Key Distribution and certification
- Access control: firewalls
- Attacks and counter measures
- Security in many layers

Firewalls

firewall

isolates organization's internal net from larger Internet, allowing some packets to pass, blocking others.



Firewalls: Why

prevent denial of service attacks:

- SYN flooding: attacker establishes many bogus TCP connections, no resources left for "real" connections.

prevent illegal modification/access of internal data.

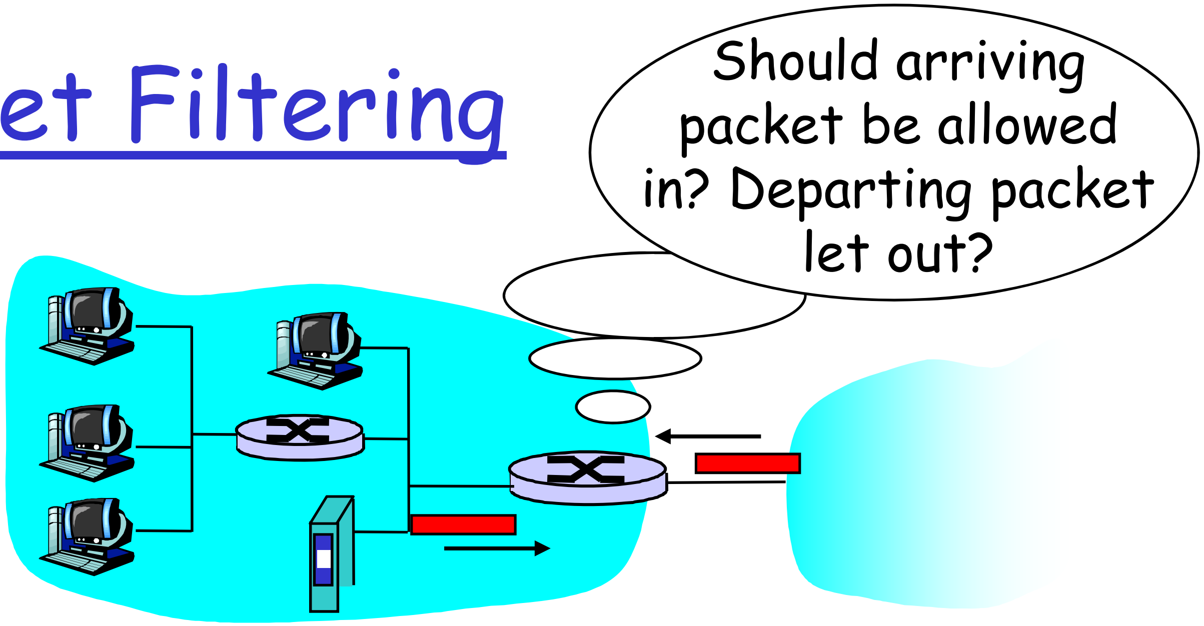
- e.g., attacker replaces CIA's homepage with something else

allow only authorized access to inside network (set of authenticated users/hosts)

two types of firewalls:

- application-level
- packet-filtering

Packet Filtering


Should arriving packet be allowed in? Departing packet let out?

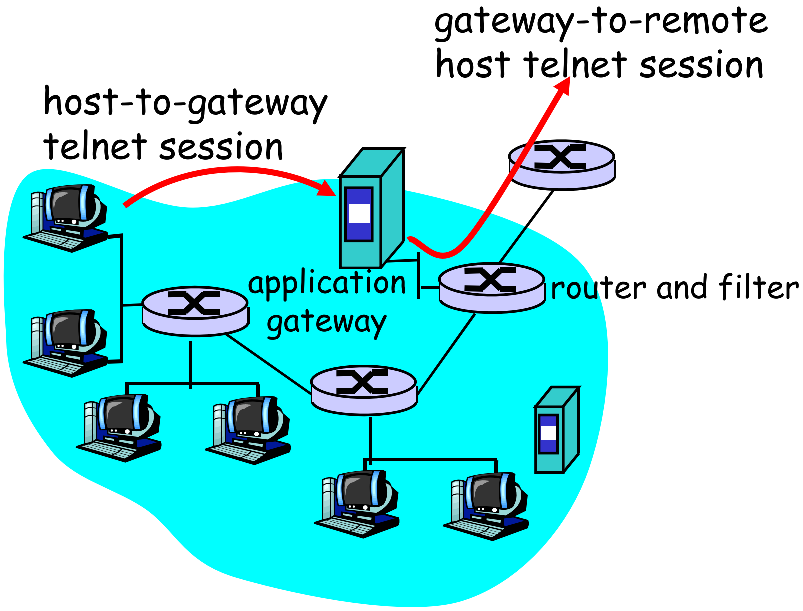
- internal network connected to Internet via router firewall
- router filters packet-by-packet, decision to forward/drop packet based on:
    - source IP address, destination IP address
    - TCP/UDP source and destination port numbers
    - ICMP message type
    - TCP SYN and ACK bits

Packet Filtering

- Example 1: block incoming and outgoing datagrams with IP protocol field = 17 (UDP) and with either source or dest port = 23 (telnet).
  - All incoming and outgoing UDP flows and telnet connections are blocked.
  - Protocols: www.iana.org/assignments/protocol-numbers
  - Protocols: www.iana.org/assignments/port-numbers.
- Example 2: Block inbound TCP segments with ACK=0.
  - Prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside.

Application gateways

- Filters packets on application data as well as on IP/TCP/UDP fields.
- Example: allow select internal users to telnet outside.


gateway-to-remote host telnet session
host-to-gateway telnet session
application gateway
router and filter

1. Require all telnet users to telnet through gateway.
2. For authorized users, gateway sets up telnet connection to dest host. Gateway relays data between 2 connections
3. Router filter blocks all telnet connections not originating from gateway.

Limitations of firewalls and gateways

- IP spoofing: router can't know if data "really" comes from claimed source
- if multiple app's. need special treatment, each has own app. gateway.
- client software must know how to contact gateway.
  - e.g., must set IP address of proxy in Web browser
- filters often use all or nothing policy for UDP.
- tradeoff: degree of communication with outside world, level of security
- many highly protected sites still suffer from attacks.

Security Overview

What is network security?

Principles of cryptography

Security Requirements

Key Distribution and certification

Access control: firewalls

Attacks and counter measures

Security in many layers

Internet security threats

Mapping:

- before attacking: “case the joint” – find out what services are implemented on network
- Use `ping` to determine what hosts have addresses on network
- Port-scanning: try to establish TCP connection to each port in sequence (see what happens)
- nmap (http://www.insecure.org/nmap/) mapper: “network exploration and security auditing”

Countermeasures?

Background

Scanners

OS Fingerprinting

Sniffers/Protocol Analyzers

Oh my!!!

Background: Scanners

- What is a scanner?
  - A scanner, in network terms, is a program that traverses through a network given a set of targets and returns information based on a set of given criteria.
- Types of scanners include
  - IP Scanners: Returns a list of active IPs
    - Superscan , NMAP
  - Port/ Service scanners: Returns what ports are open on a target and what services are being provided on it.
    - IPTools, NMAP
  - Vulnerability Scanners: Returns a list of exploits which the target might be vulnerable to.
    - Nessus, Retina
  - NAT Scanners: attempts to determine the number of systems running behind the natted firewall and their operating systems
    - firewalk

Background: OS Fingerprinting Tools

- What is OS fingerprinting?
  - A technique which queries the TCP/IP Stack of a host to determine what operating system is running on it.
- There are two different types of OS fingerprinting tools,
  - Active:
    - Generates network traffic
    - May be detected
    - Specially crafted packets
    - Catches variability's in TCP/IP stack
  - Passive:
    - No traffic is generated
    - Virtually undetectable

Background: OS Fingerprinting Tools
Early tools

- SIRC Roger Espel Llima . AKA: orabidoo (NMAP predecessors)
  - attempted to place a host in os classes; Linux, 4.4bsd, win95 or unknown
- CHECKOS, by Shok (NMAP predecessors)
  - very similar to SIRC-
  - was never suppose to go public
- SS, by SU1d (NMAP successor)
  - identified 12 different operating systems
  - the networking code was from NMAP
- Queso, Jordi Murgo, AKA:Savage, Apostols (NMAP successor)
  - first program to move the fingerprinting out of the code and into a separate file. Made adding a new operating system easier.

Background: OS Fingerprinting Tools
Today's tools

- Xprobe2, by Ofir Arkin, http://www.sys-ssecurity.com
  - Uses ICMP as the method to do fingerprinting
  - Generates fingerprints of systems scanned
- NMAP, by Fyodor, http://www.isecure.org
  - detects 100's of different OS versions and network devices
  - By far the most sophisticated fingerprinting tool on the net
    - IP/Service scanner
    - Portscanner
    - OS fingerprinting
    - Network Device fingerprinting
    - 12 different modes of scanning
    - 4 different ways to discover systems
    - 8 different packet modification options (you can select multiple options simultaneously)
    - 6 different timing options and 6 different method of detecting

Background: OS Fingerprinting Tools

Today's tools

Passive Tools

- Siphon, by Subterrain Security Group
  - http://www.blackhat.com/presentations/bh-usa-01/AbadBeddoe/1
  - Runs as a service and logs detected operating systems to a file and stdout
- P0f, by Michal Zalewski
  - http://lcamtuf.coredump.cx/p0f.shtml
  - Analyzes tcpdump formatted files
  - Excellent tool for network analysis to use with windump and tcpdump

Background: Sniffers

- What is a sniffer?
    - A program that puts the systems network interface in promiscuous mode allowing the monitoring and analysis of network traffic
- Types of sniffers;
    - Command line (CLI) sniffers:
        - Data is viewed in a pretty raw format: TCPdump. Windump, snort, sniffit
        - Sniffit was one of the first sniffers with a GUI
    - Protocol Analyzers:
        - Graphical Interfaces; Etherpeek, Iris, SniffITPro, Netasyst, Ethereal
        - In addition to what a sniffer can do, Protocol analyzers can;
            - provide detailed and formatted protocol information
            - Able to generates reports and in many cases has some intelligence
            - Assist in troubleshooting network problems
        - Both software and hardware based
    - non-switch networks: sniffit and all of the above
    - switched networks: ettercap
        - The other sniffer/protocol analyzers can sniff a network providing they are connected to what is called a mirrored port.

A Detailed view: Analyzing the Tools and the traffic they generate Part 1

Active OS Fingerprinting Tools: NMAP "*Jack of all trades*"

- IP Scanner
  - nmap -sS –P0 <ip range>
    - Action: send syn, recv syn-ack, send rst
  - nmap –sT –P0 <ip range>
    - Action: syn, recv syn-ack, send syn-ack-ack, send rst
- Port Scanner
  - nmap -sT –P0 –p<port range> <target>
    - Action: send syn, recv syn-ack, send syn-ack-ack, send rst
- Service Scanner
  - nmap <target>
- OS fingerprinting
  - nmap –O –P0 <target>

Scan Analysis ☺

Resources

Web Sites

- http://www.isecure.org
- http://www.sys-security.com
- http://securify.packetstorm.org
- http://www.protocols.com
- http://www.sans.org
- http://www.networksorcery.com/enp/default0601.htm

Books

- Network Intrusion Detection, Northcutt
- TCP/IP Illustrated vol1, Stevens

Internet security threats

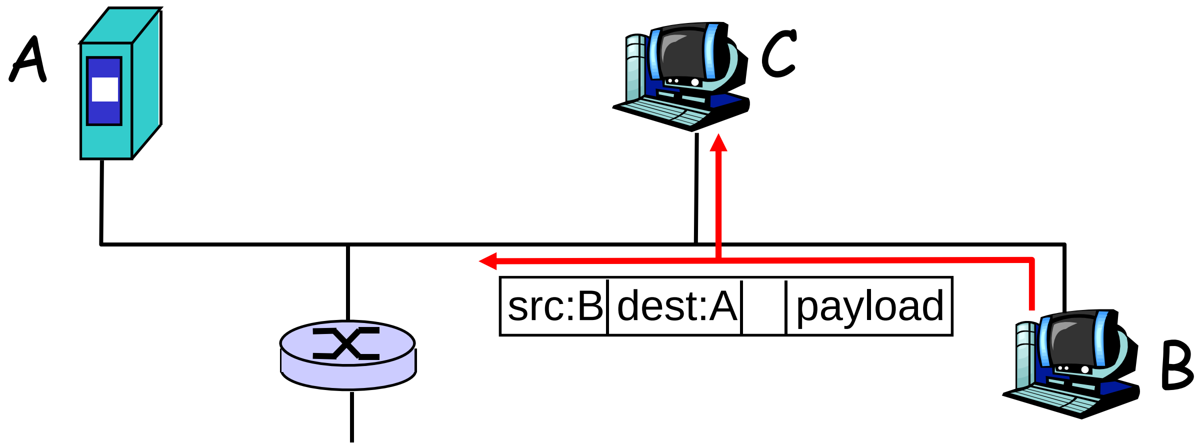
Mapping: countermeasures

- record traffic entering network
- look for suspicious activity (IP addresses, ports being scanned sequentially)

Internet security threats

Packet sniffing:

- broadcast media
- promiscuous NIC reads all packets passing by
- can read all unencrypted data (e.g. passwords)
- e.g.: C sniffs B's packets

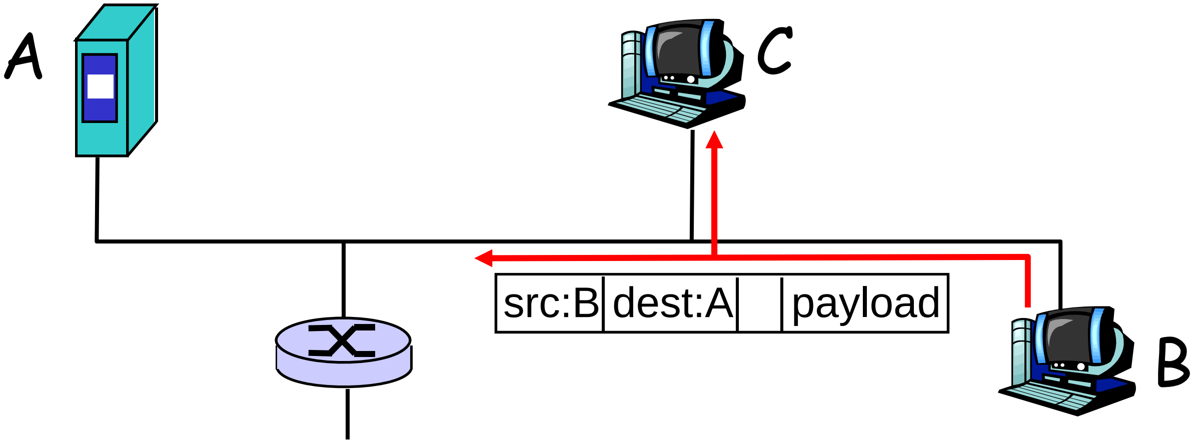


Countermeasures?

Internet security threats

Packet sniffing: countermeasures

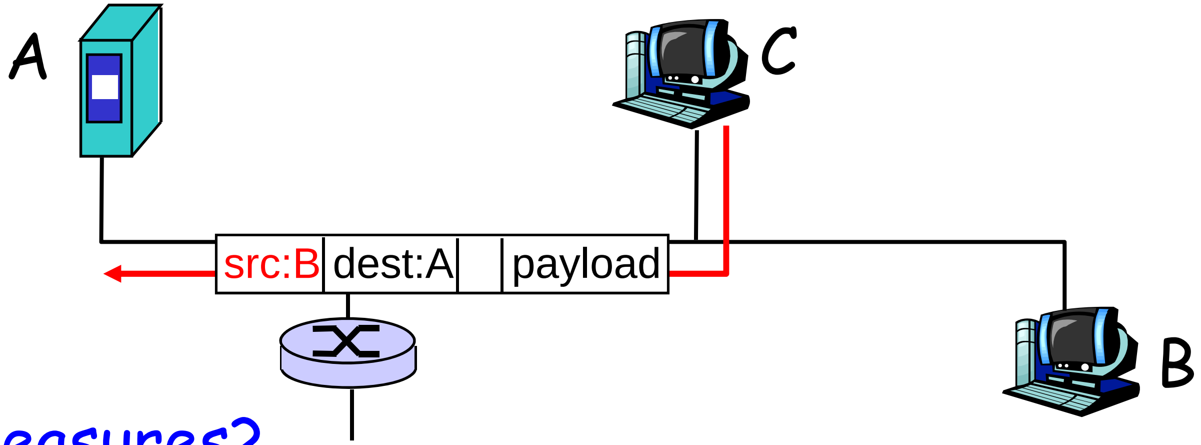
- all hosts in orgnization run software that checks periodically if host interface in promiscuous mode.
- one host per segment of broadcast media (switched Ethernet at hub)



Internet security threats

IP Spoofing:

- can generate "raw" IP packets directly from application, putting any value into IP source address field
- receiver can't tell if source is spoofed
- e.g.: C pretends to be B

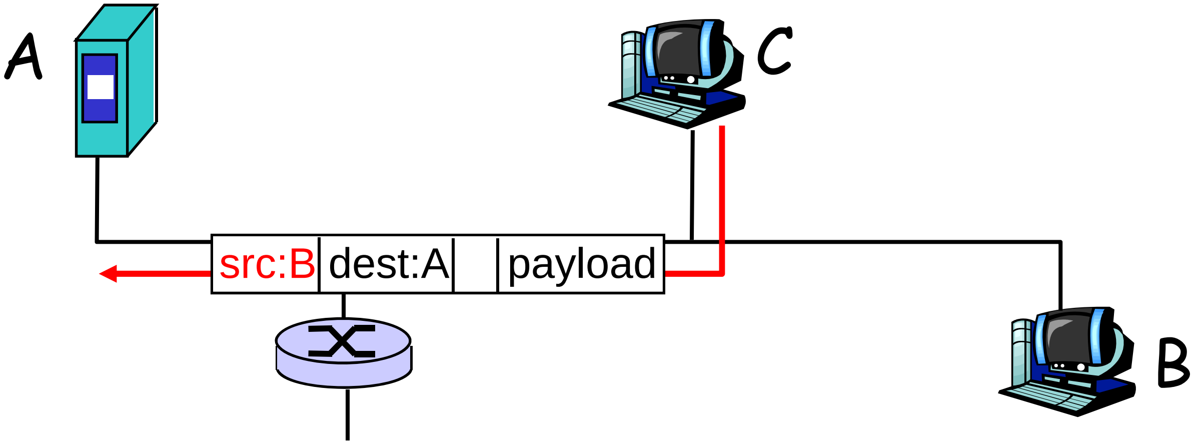


Countermeasures?

Internet security threats

IP Spoofing: ingress filtering

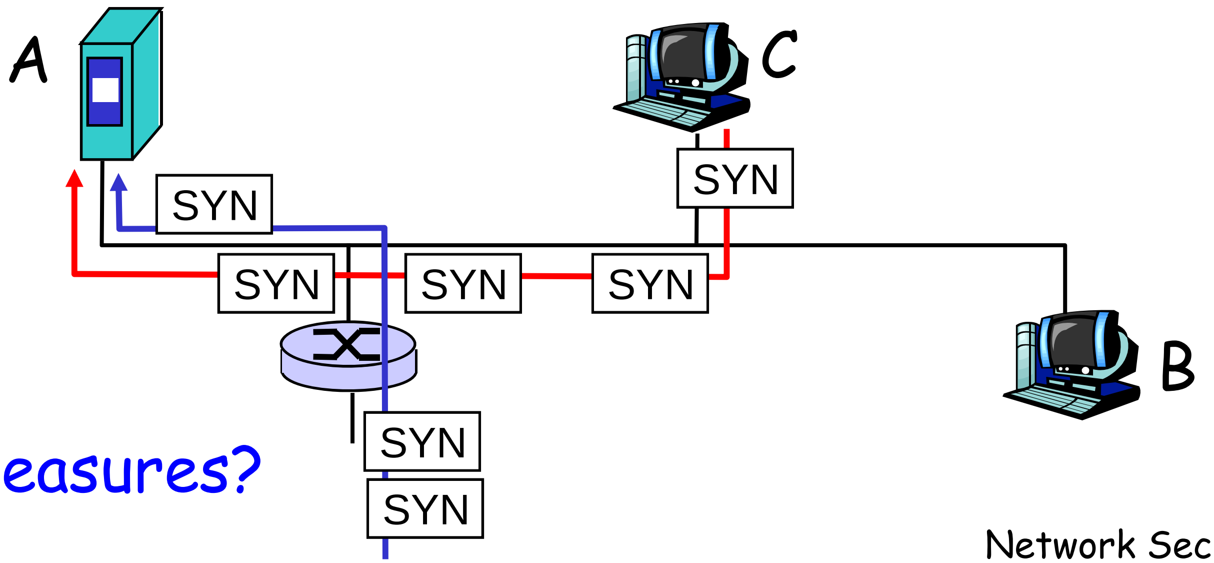
- routers should not forward outgoing packets with invalid source addresses (e.g., datagram source address not in router's network)
- great, but egress filtering can not be mandated for all networks



Internet security threats

Denial of service (DOS):

- flood of maliciously generated packets "swamp" receiver
- Distributed DOS (DDOS): multiple coordinated sources swamp receiver
- e.g., C and remote host SYN-attack A

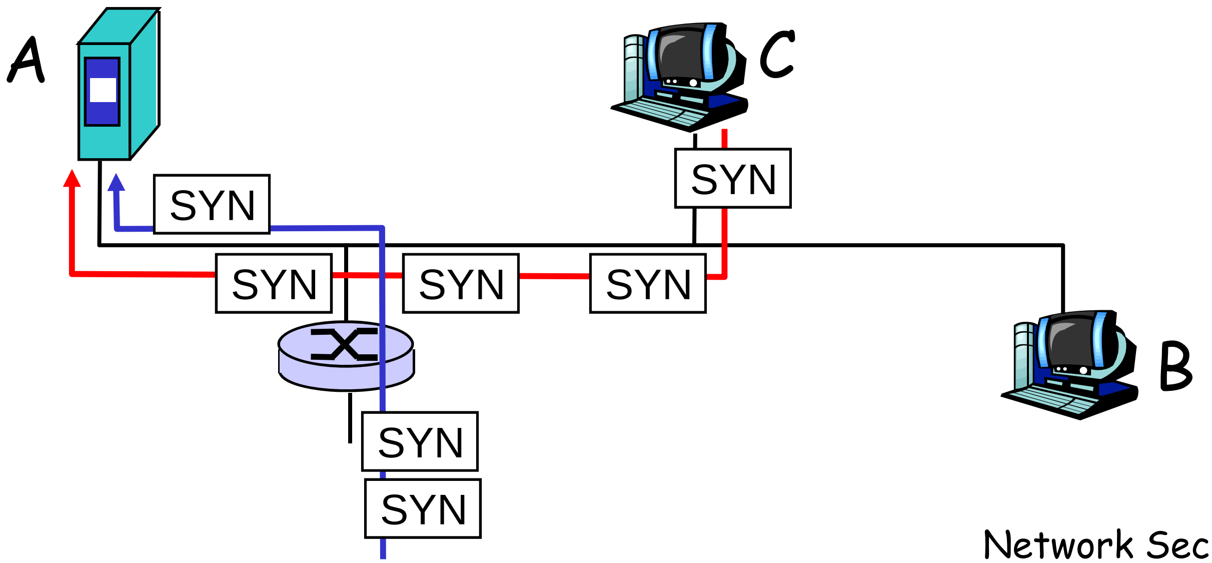


Countermeasures?

Internet security threats

Denial of service (DOS): countermeasures

- filter out flooded packets (e.g., SYN) before reaching host: throw out good with bad
- traceback to source of floods (most likely an innocent, compromised machine)

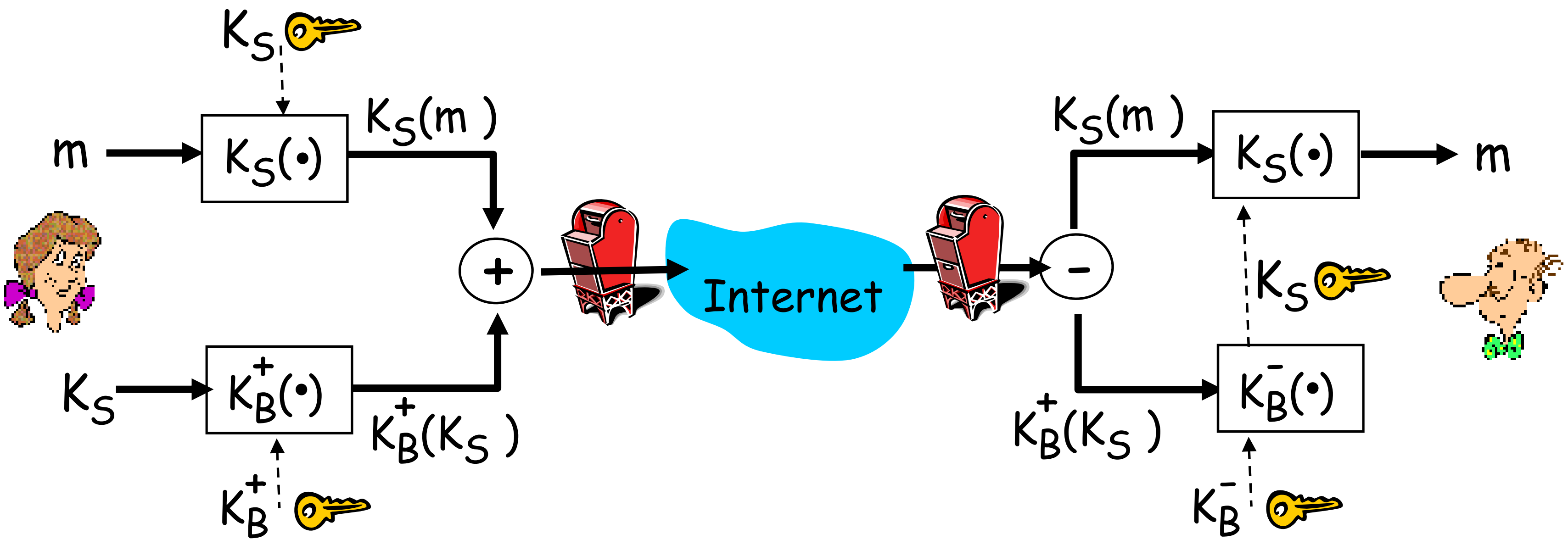


Outline

- What is network security?
- Principles of cryptography
- Authentication
- Integrity
- Key Distribution and certification
- Access control: firewalls
- Attacks and counter measures
- Security in many layers
  - Secure email
  - Secure sockets
  - IPsec
  - 802.11 WEP

Secure e-mail using public key

- Alice wants to send confidential e-mail, m, to Bob.

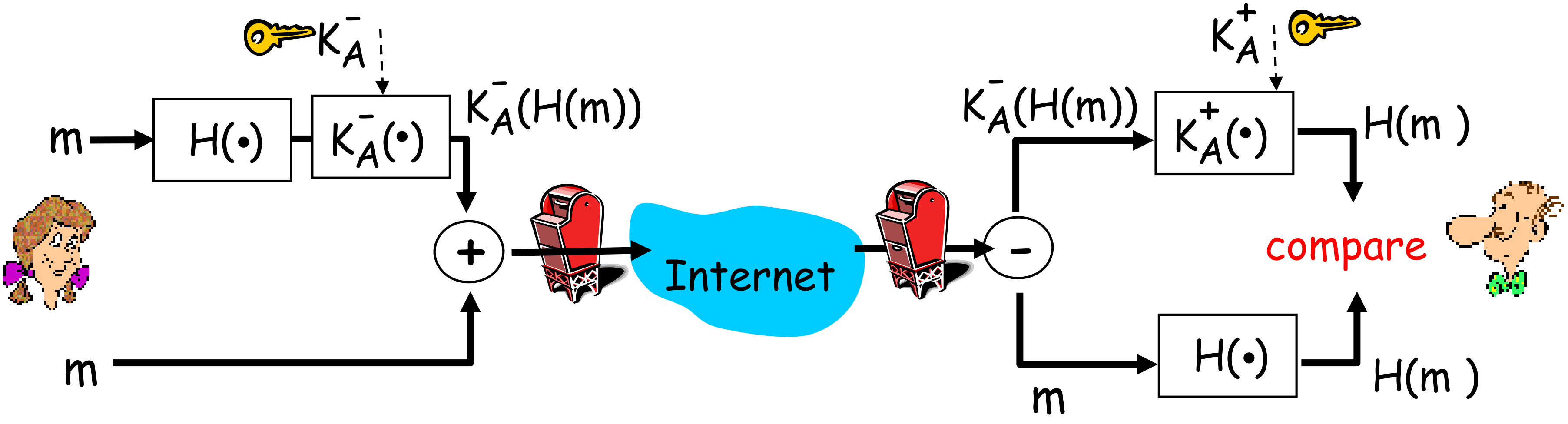


Alice:

- generates random *symmetric* private key, K_S.
- encrypts message with K_S (for efficiency)
- also encrypts K_S with Bob's public key.
- sends both $K_S(m)$ and $K_B(K_S)$ to Bob.

Secure e-mail (continued)

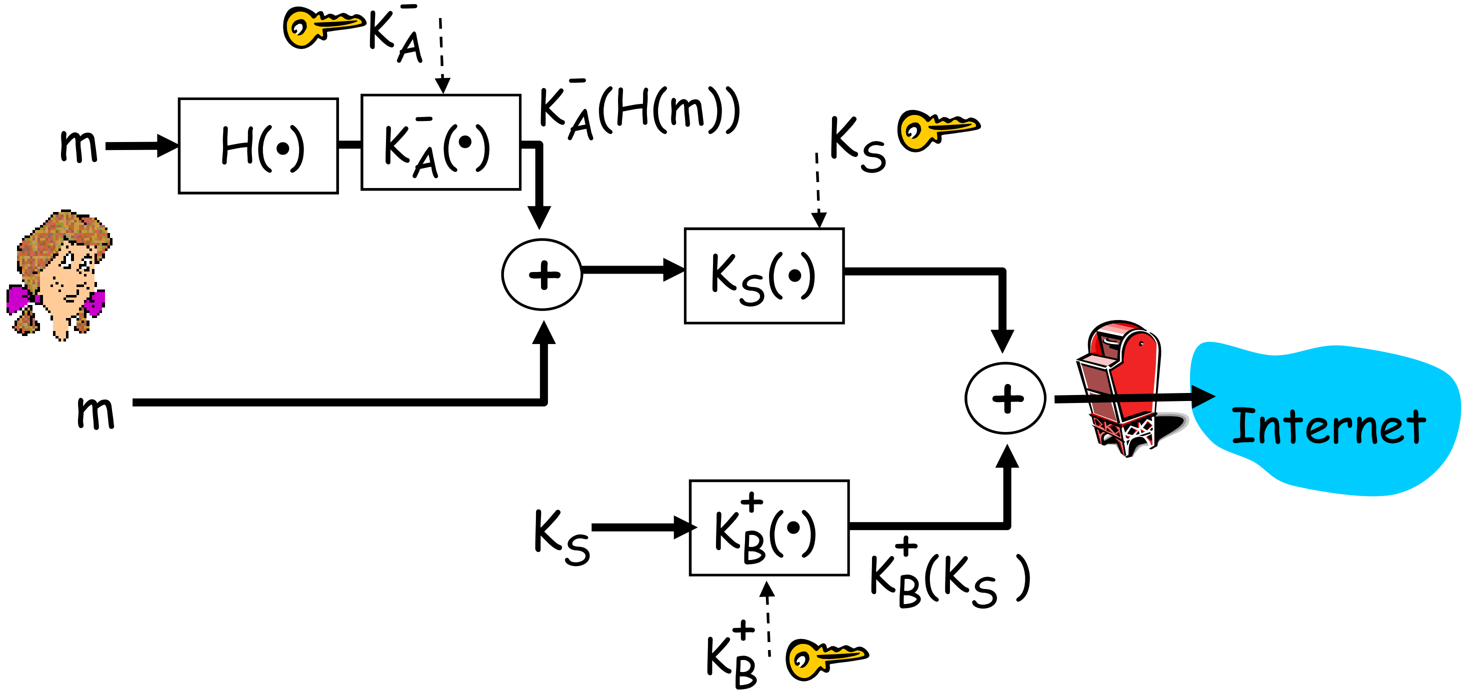
- Alice wants to provide sender authentication message integrity.



- Alice digitally signs message.
- sends both message (in the clear) and digital signature.

Secure e-mail (continued)

- Alice wants to provide secrecy, sender authentication, message integrity.



Alice uses three keys: her private key, Bob's public key, newly created symmetric key

Pretty good privacy (PGP)

- Internet e-mail encryption scheme, de-facto standard.
- uses symmetric key cryptography, public key cryptography, hash function, and digital signature as described.
- provides secrecy, sender authentication, integrity.
- inventor, Phil Zimmerman, was target of 3-year federal investigation.

A PGP signed message:

```
---BEGIN PGP SIGNED MESSAGE---
Hash: SHA1

Bob:My husband is out of town
   tonight.Passionately yours,
   Alice

---BEGIN PGP SIGNATURE---
Version: PGP 5.0
Charset: noconv
yhHJRHhGJGhgg/12EpJ+lo8gE4vB3mqJ
   hFEvZP9t6n7G6m5Gw2
---END PGP SIGNATURE---
```

Secure sockets layer (SSL)

- transport layer security to any TCP-based app using SSL services.
- used between Web browsers, servers for e-commerce (shttp).
- security services:
  - server authentication
  - data encryption
  - client authentication (optional)

- server authentication:
  - SSL-enabled browser includes public keys for trusted CAs.
  - Browser requests server certificate, issued by trusted CA.
  - Browser uses CA's public key to extract server's public key from certificate.
- check your browser's security menu to see its trusted CAs.

SSL (continued)

Encrypted SSL session:

- Browser generates *symmetric session key*, encrypts it with server's public key, sends encrypted key to server.
- Using private key, server decrypts session key.
- Browser, server know session key
  - All data sent into TCP socket (by client or server) encrypted with session key.

- SSL: basis of IETF Transport Layer Security (TLS).
- SSL can be used for non-Web applications, e.g., IMAP.
- Client authentication can be done with client certificates.

What is IPSec (Microsoft Technet)?

“Internet Protocol security (IPSec) is a framework of open standards for helping to ensure private, secure communications over Internet Protocol (IP) networks through the use of cryptographic security services. IPSec supports network-level data integrity, data confidentiality, data origin authentication, and replay protection. Because IPSec is integrated at the Internet layer (layer 3), it provides security for almost all protocols in the TCP/IP suite, and because IPSec is applied transparently to applications, there is no need to configure separate security for each application that uses TCP/IP. “

IPsec: Network Layer Security

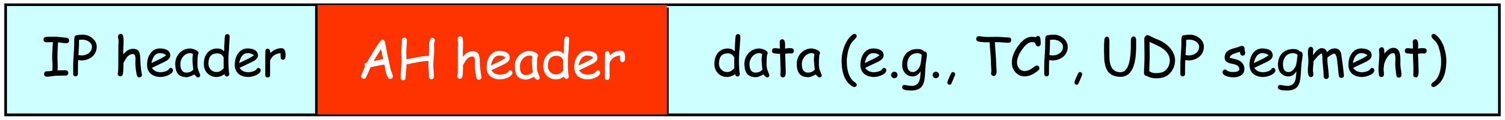
- Network-layer secrecy:
  - sending host encrypts the data in IP datagram
  - TCP and UDP segments; ICMP and SNMP messages.
- Network-layer authentication
  - destination host can authenticate source IP address
- Two principle protocols:
  - authentication header (AH) protocol
  - encapsulation security payload (ESP) protocol
- For both AH and ESP, source, destination handshake:
  - create network-layer logical channel called a security association (SA)
- Each SA unidirectional.
- Uniquely determined by:
  - security protocol (AH or ESP)
  - source IP address
  - 32-bit connection ID

Authentication Header (AH) Protocol

- provides source authentication, data integrity, no confidentiality
- AH header inserted between IP header, data field.
- protocol field: 51
- intermediate routers process datagrams as usual

AH header includes:

- connection identifier
- authentication data: source- signed message digest calculated over original IP datagram.
- next header field: specifies type of data (e.g., TCP, UDP, ICMP)



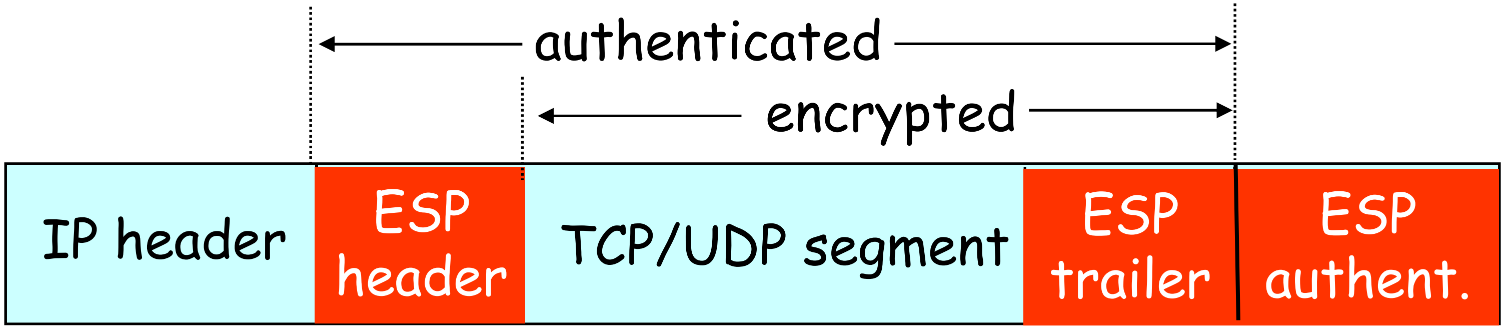
IP header

AH header

data (e.g., TCP, UDP segment)

ESP Protocol

- provides secrecy, host authentication, data integrity.
- data, ESP trailer encrypted.
- next header field is in ESP trailer.
- ESP authentication field is similar to AH authentication field.
- Protocol = 50.



IEEE 802.11 security

- *War-driving:* drive around Bay area, see what 802.11 networks available?
  - Thousands accessible from public roadways
  - 85% use no encryption/authentication
  - packet-sniffing and various attacks easy!
- *Wired Equivalent Privacy (WEP):* authentication as in protocol *ap4.0*
  - host requests authentication from access point
  - access point sends 128 bit nonce
  - host encrypts nonce using shared symmetric key
  - access point decrypts nonce, authenticates host

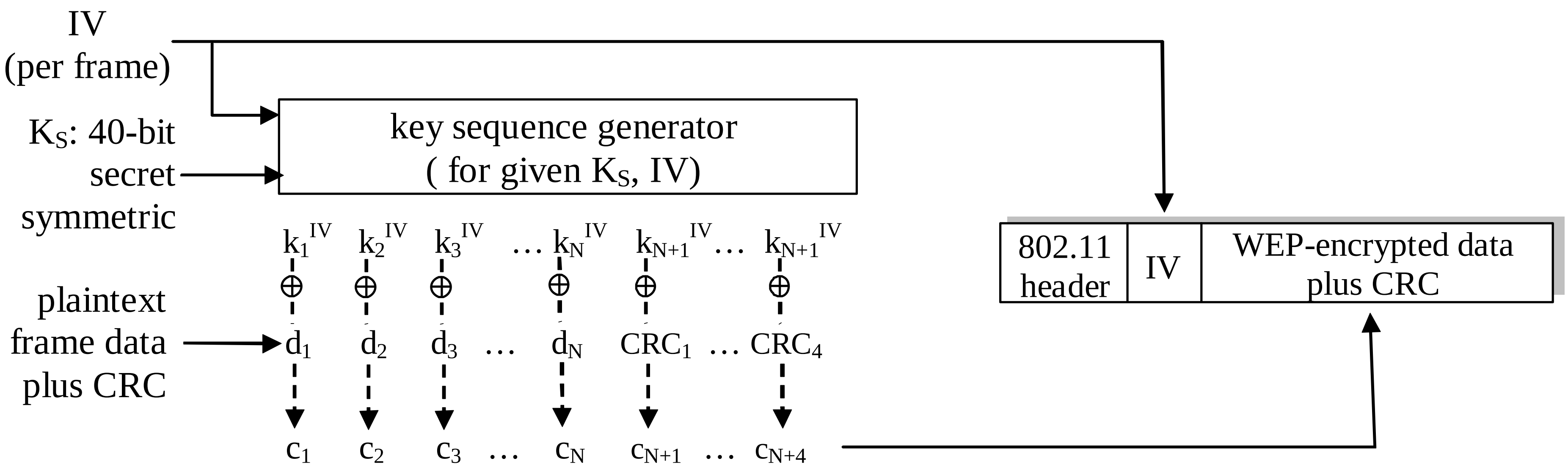
IEEE 802.11 security

- *Wired Equivalent Privacy (WEP):* data encryption
  - Host/AP share 40 bit symmetric key (semi-permanent)
  - Host appends 24-bit initialization vector (IV) to create 64-bit key
  - 64 bit key used to generate stream of keys, k_i^{IV}
  - k_i^{IV} used to encrypt ith byte, d_i, in frame:

$$c_i = d_i \text{ XOR } k_i^{IV}$$

  - IV and encrypted bytes, c_i sent in frame

802.11 WEP encryption



Sender-side WEP encryption

Breaking 802.11 WEP encryption

Security hole:

- 24-bit IV, one IV per frame, -> IV's eventually reused
- IV transmitted in plaintext -> IV reuse detected
- **Attack:**
  - Trudy causes Alice to encrypt known plaintext d_1 d_2 d_3 d_4 …
  - Trudy sees: $c_i = d_i$ XOR k_i^{IV}
  - Trudy knows c_i d_i, so can compute k_i^{IV}
  - Trudy knows encrypting key sequence k_1^{IV} k_2^{IV} k_3^{IV} …
  - Next time IV is used, Trudy can decrypt!

Network Security (summary)

Basic techniques......

- cryptography (symmetric and public)
- authentication
- message integrity
- key distribution

.... used in many different security scenarios

- secure email
- secure transport (SSL)
- IP sec
- 802.11 WEP