L6: Basic Cryptography II

Hui Chen, Ph.D.
Dept. of Engineering & Computer Science
Virginia State University
Petersburg, VA 23806

Acknowledgement

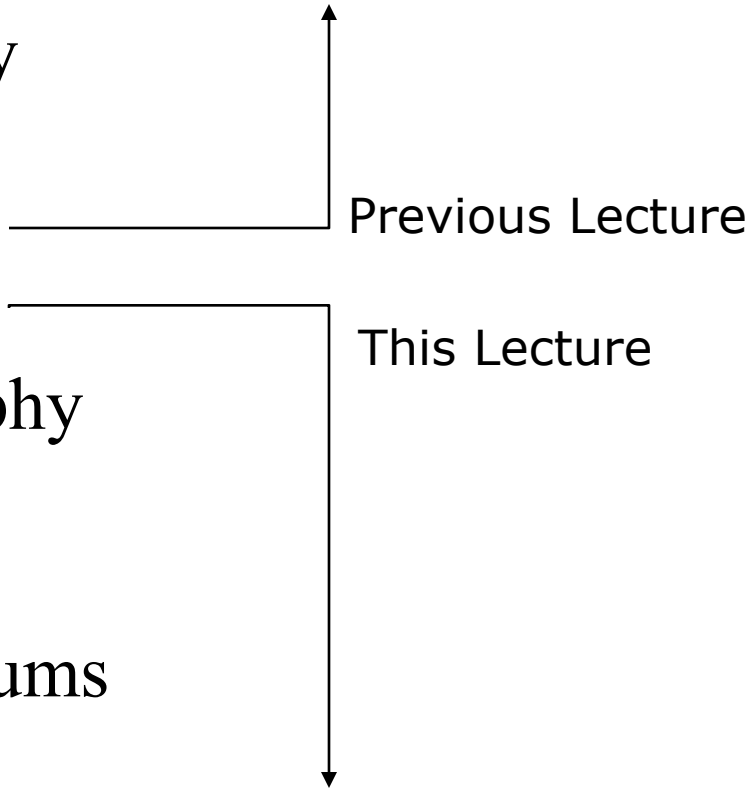
- Many slides are from or are revised from the slides of the author of the textbook
  - Matt Bishop, Introduction to Computer Security, Addison-Wesley Professional, October, 2004, ISBN-13: 978-0-321-24774-5. Introduction to Computer Security @ VSU's Safari Book Online subscription
  - http://nob.cs.ucdavis.edu/book/book-intro/slides/

Overview

- A deep mathematical subject
- In the context of this class
  - A supporting tool for confidentiality and integrity
- Concepts
  - Cryptography, crytoanalysis, cryptoanalysis
- Basic Cryptography
  - Classical Cryptography
  - Public Key Cryptography
  - Cryptographic Checksums

Overview

- Classical Cryptography
  - Caesar cipher
  - Vigènere cipher
  - DES
- Public Key Cryptography
  - Diffie-Hellman
  - RSA
- Cryptographic Checksums
  - HMAC



The Data Encryption Standard

- DES = The Data Encryption Standard
  - A Product Cipher: uses both transposition and substitution
- In 1977 the National Bureau of Standards announced a Data Encryption Standard to be used in unclassified U.S. Government applications
  - For sensitive but unclassified U.S. government data
  - Unclassified U.S. Government data: information not concerned with national security
  - In wide international use
    - e.g., banks used it for funds transfer security

DES: A Block Cipher

- Input, output, and key are each 64 bits long
  - divide data into 64-bit blocks
  - use a 64 bit key (i.e., a key block) supplied by user
  - encrypts the 64-bit blocks of data
  - outputs 64 bits of ciphertext

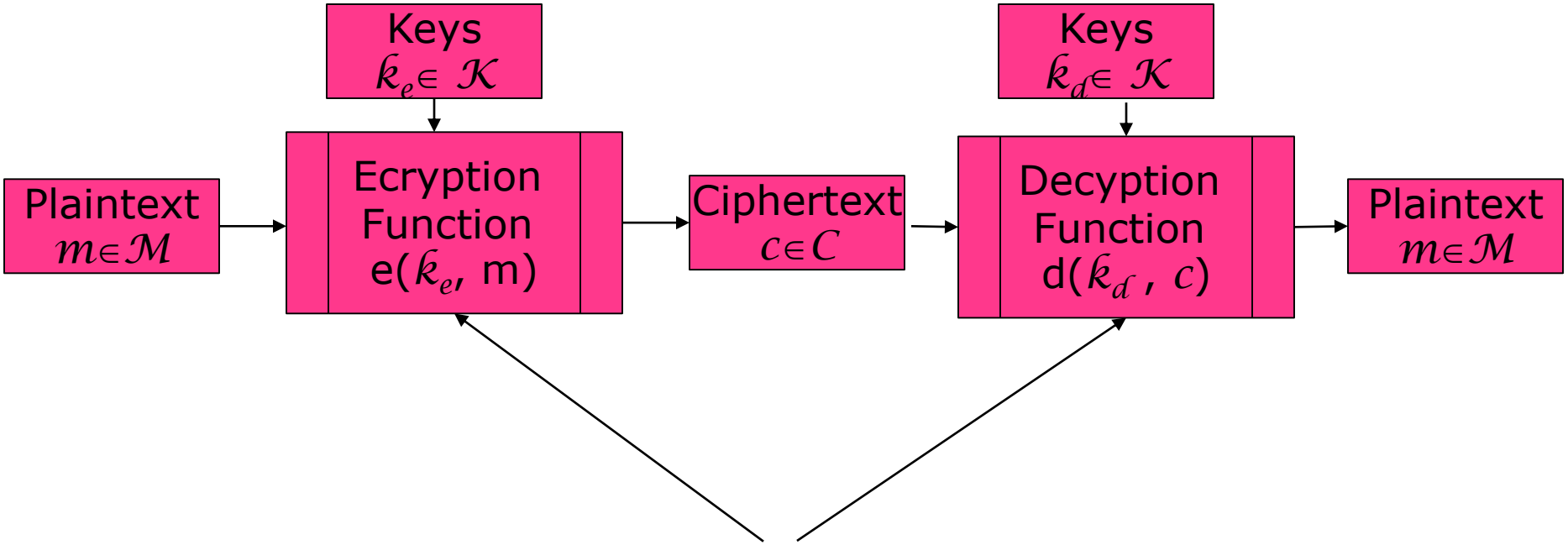
DES Key Block

- 64 bit key block
  - 8 bytes
  - Each byte
    - 7 bits + 1 parity bit
- 56 bit key
  - $8 \times 7 = 56$ bits
  - Drop 8 parity bits

DES Rounds

- The DES block cipher consists of 16 rounds (iterations)
  - each round with a round key generated from the user-supplied key
  - basic unit is the bit
  - each round is a product cipher, i.e., each round performs both substitution and transposition (permutation) on the bits
- The rounds are executed sequentially
  - The input of round $i+1$ is the output of round i

Overview of DES



- 3 major steps in both encipherment and decipherment
- $k_e = k_d$

Encipherment & Decipherment

- Apply an initial permutation (IP) to the input block

  $(L_0, R_0) \leftarrow IP\ (Input\ Block)$

- Iterate 16 rounds

  $L_i \leftarrow R_{i-1}$

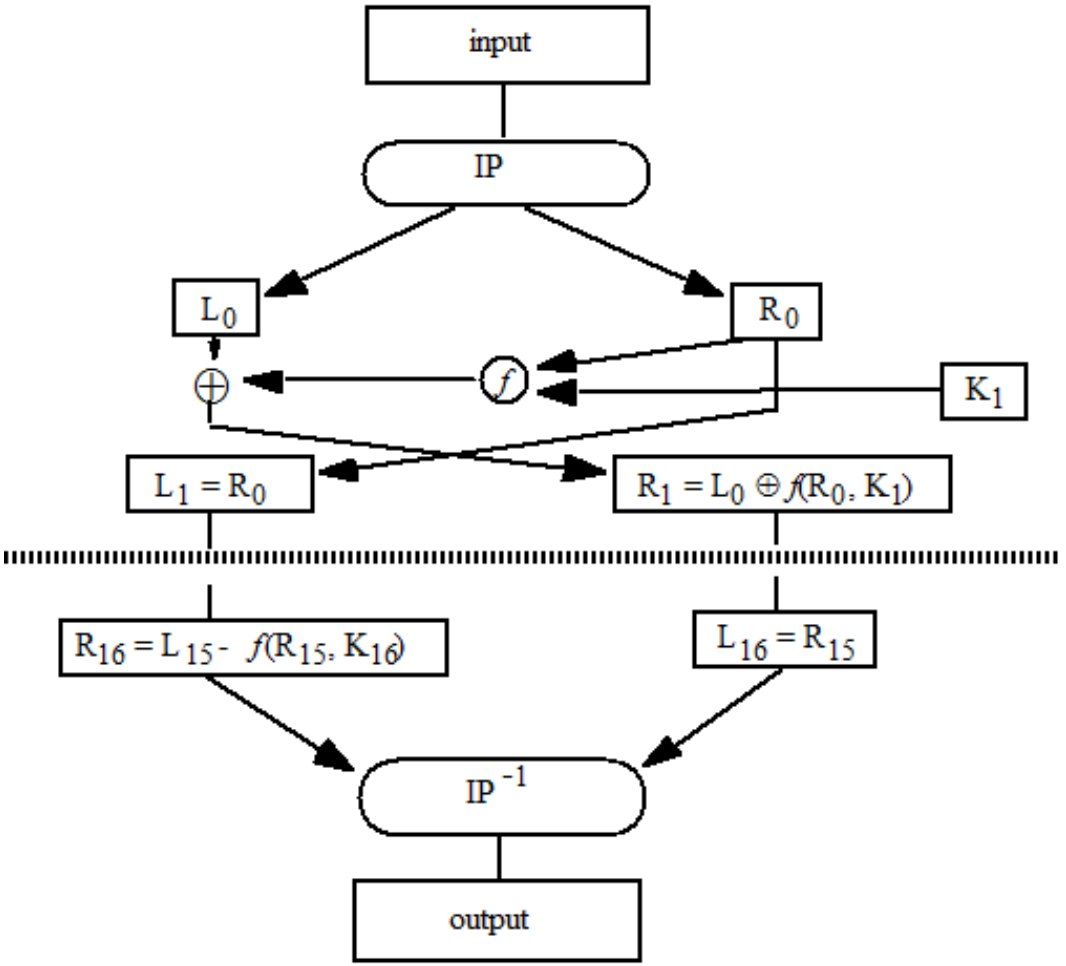
  $R_i \leftarrow L_{i-1} \oplus f(R_{i-1}, K_i)$

  - K_i is a round key, a substring of the 56-bit input key
  - *f is called S-Box function: f provides the strength of DES*

- Apply the inverse of IP to the output of round 16

  $Output\ Block \leftarrow IP^{-1}\ (R_{16}, L_{16})$

Encipherment & Decipherment



16 rounds

Initial & Final Permutations

- From Schneier 1996
- Designed to load plaintext and ciphertext data into a DES chip in byte-sized pieces
- Does not affect DES's strength
- Bit-wise permutation trivial in hardware, but difficult (*inefficient*) in software
  - Many software implementations leave the input & final permutations out (they should *not* be called DES though)

IP and its Inverse

- Initial Permutation and its inverse (from Denning 1982)

TABLE 2.3(a) Initial permutation IP.

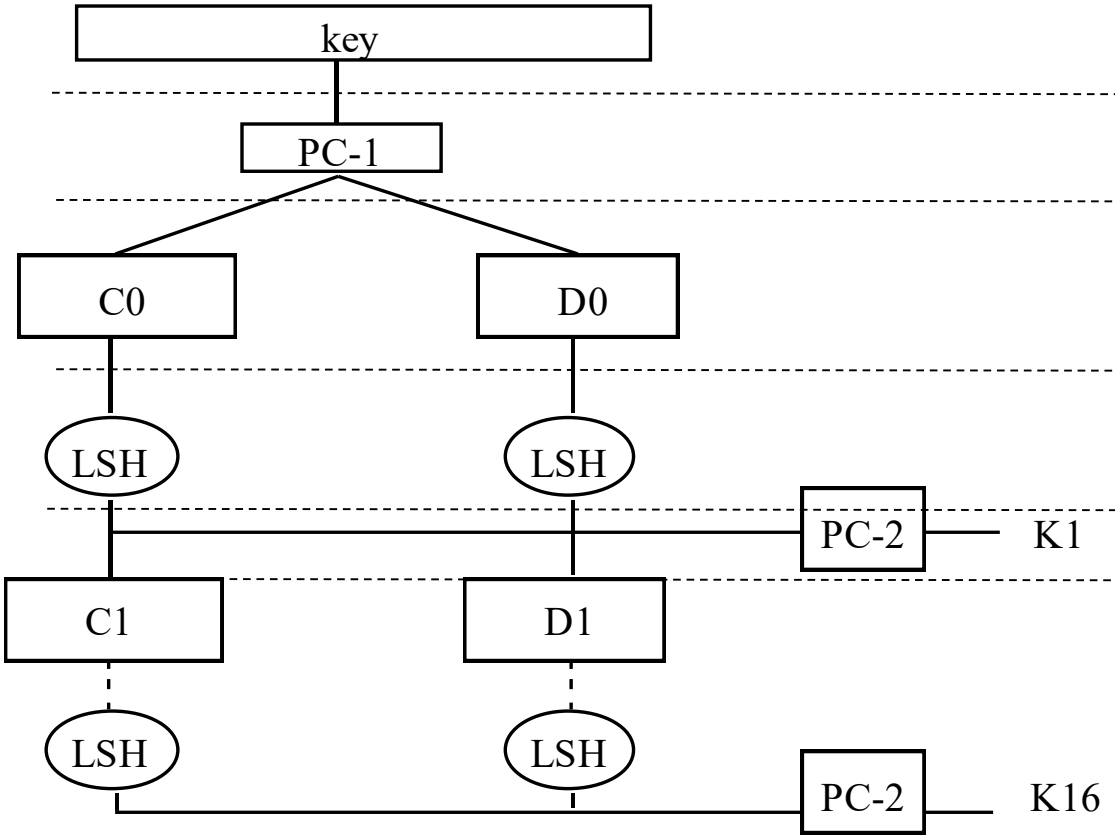
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

TABLE 2.3(b) Final permutation IP^{-1}.

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

Generation of Round Keys

- Round keys are 48 bits each



56 bits (drop parity bits from the 64 bit key block)

A permutation (next slide)

Two 28-bit halves

Apply left circular shift
C1 = LS1(C0) D1=LS1(D1)

K1 = PC-2(C1 D1)

Repeat for next round key

Note: PC-2 (C1 D1) takes 48 bits from (C1 D1) after a permutation

Generation of Round Keys: Permutations

- PC-1 and PC-2 are two permutations (from Denning 1982)

TABLE 2.7 Key permutation PC-1.

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
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

PC-1: 56-bit input and output

TABLE 2.9 Key permutation PC-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

PC-2: 56-bit input and 48-bit output

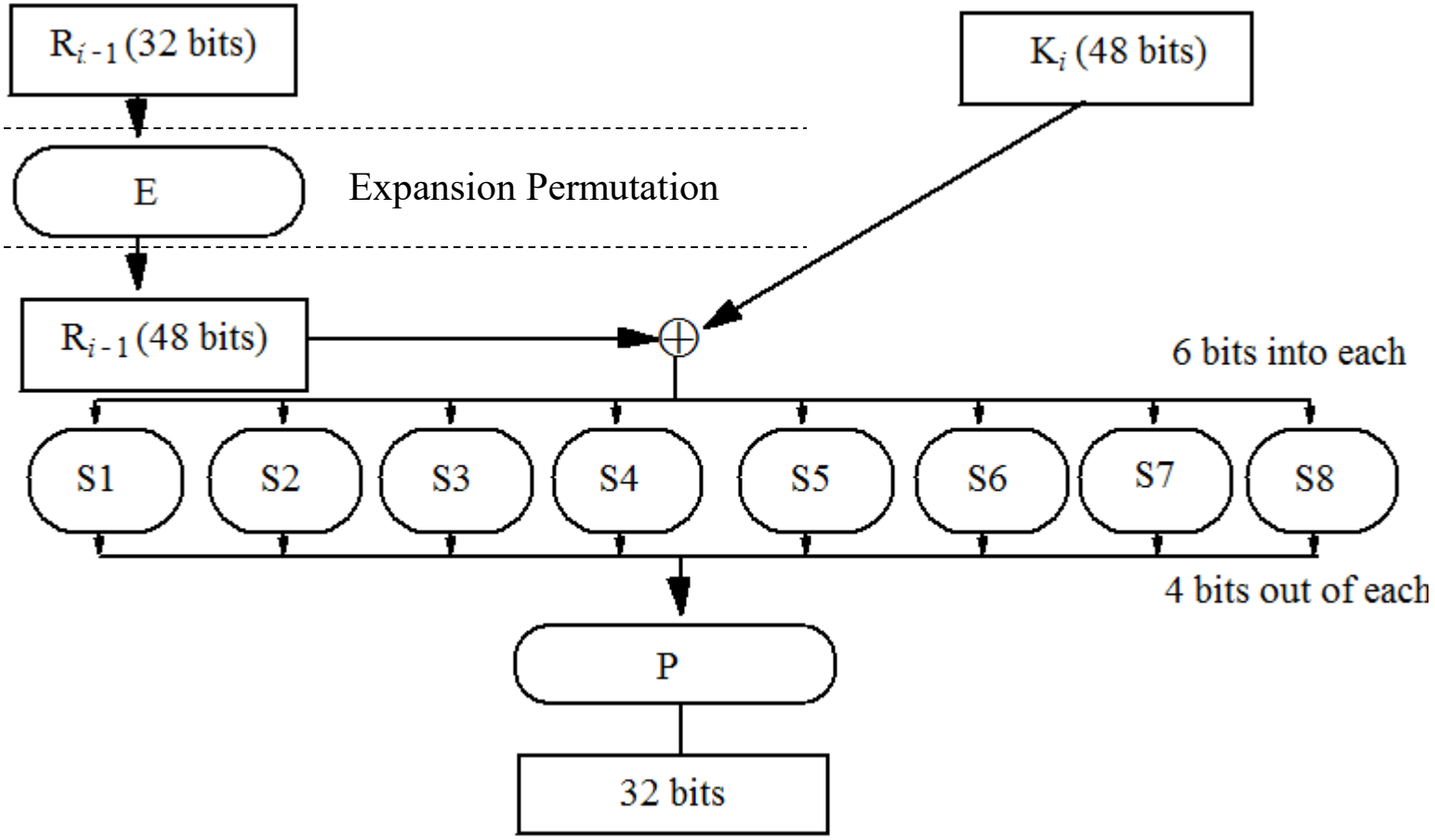
Generation of Round Keys: Left Circular Shift

- From Denning 1982

TABLE 2.8 Key schedule of left shifts LS.

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

f function


R_{i-1} (32 bits)
K_i (48 bits)
E
Expansion Permutation
R_{i-1} (48 bits)
6 bits into each
S1
S2
S3
S4
S5
S6
S7
S8
4 bits out of each
P
32 bits

Inside *f* function: Expansion Permutation

- From Schneier 1996
  - Repeating some bits to achieve *avalanche effect*, i.e., to have every bit of the ciphertext depend on every bit of the plaintext and every bit of the key as quickly as possible.

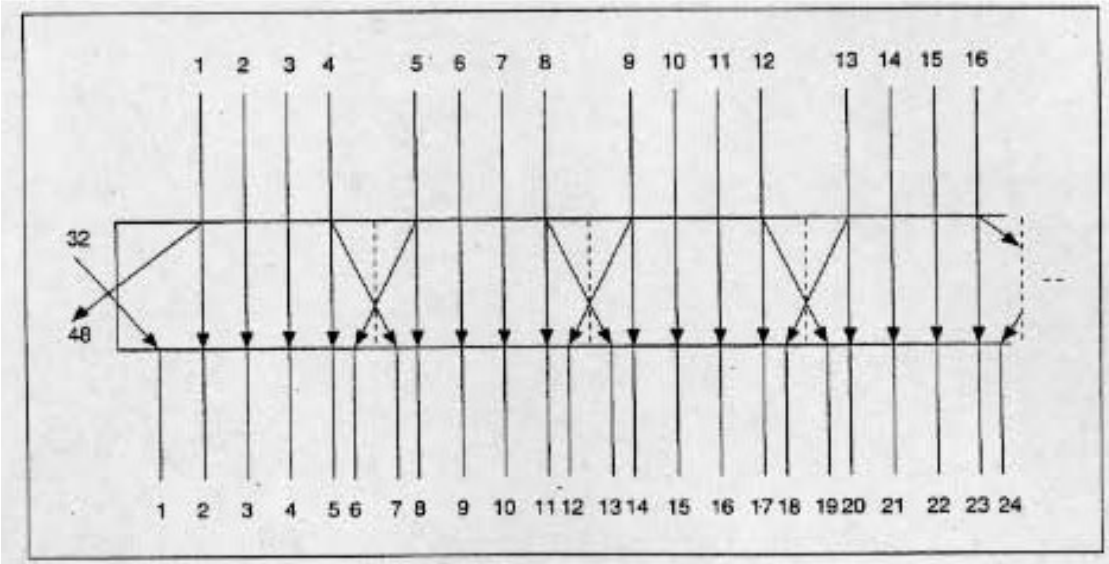


Figure 12.3 Expansion permutation.

Inside *f* function: Substitution Boxes

- S-Boxes: From Denning 1982 (and for complete table)
- 6 bit input

  b1b2b3b4b5b6

  b1b6 selects row

  b2b3b4b5 selects column
- Example

  Input: $(010011)_2$

  b1b6 = $(01)_2 = 1$

  b2b3b4b5 = $(1001)_2 = 9$

  Select 6 = $(0110)_2$

TABLE 2.6 Selection functions (S-boxes).

	Column															
Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	S_1
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	S_2
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	S_3
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	

Inside f function: P-Box Permutation

- From Schneier 1996

Table 12.7
P-Box 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

Controversy

- Diffie and Hellman claim that in a few years technology would allow DES to be broken in days (Diffie and Hellman, 1977)
- Design of efficient attacks using 1999 technology published
  - See "Chronology" in *https://en.wikipedia.org/wiki/Data_Encryption_Standard*
- Design decisions of S-boxes not public
  - S-boxes may have backdoors

Undesirable Properties

- 4 weak keys
  - They are their own inverses
- 12 semi-weak keys
  - Each has another semi-weak key as inverse
- Complementation property
  - $\mathrm{DES}_k(m) = c \Rightarrow \mathrm{DES}_k(m') = c'$
- S-boxes exhibit irregular properties
  - Distribution of odd, even numbers non-random
  - Outputs of fourth box depends on input to third box

Note: DES key space:
2^{56} = 72,057,594,037,927,936 keys

Choosing weak keys (very unlikely) leads to the same round keys

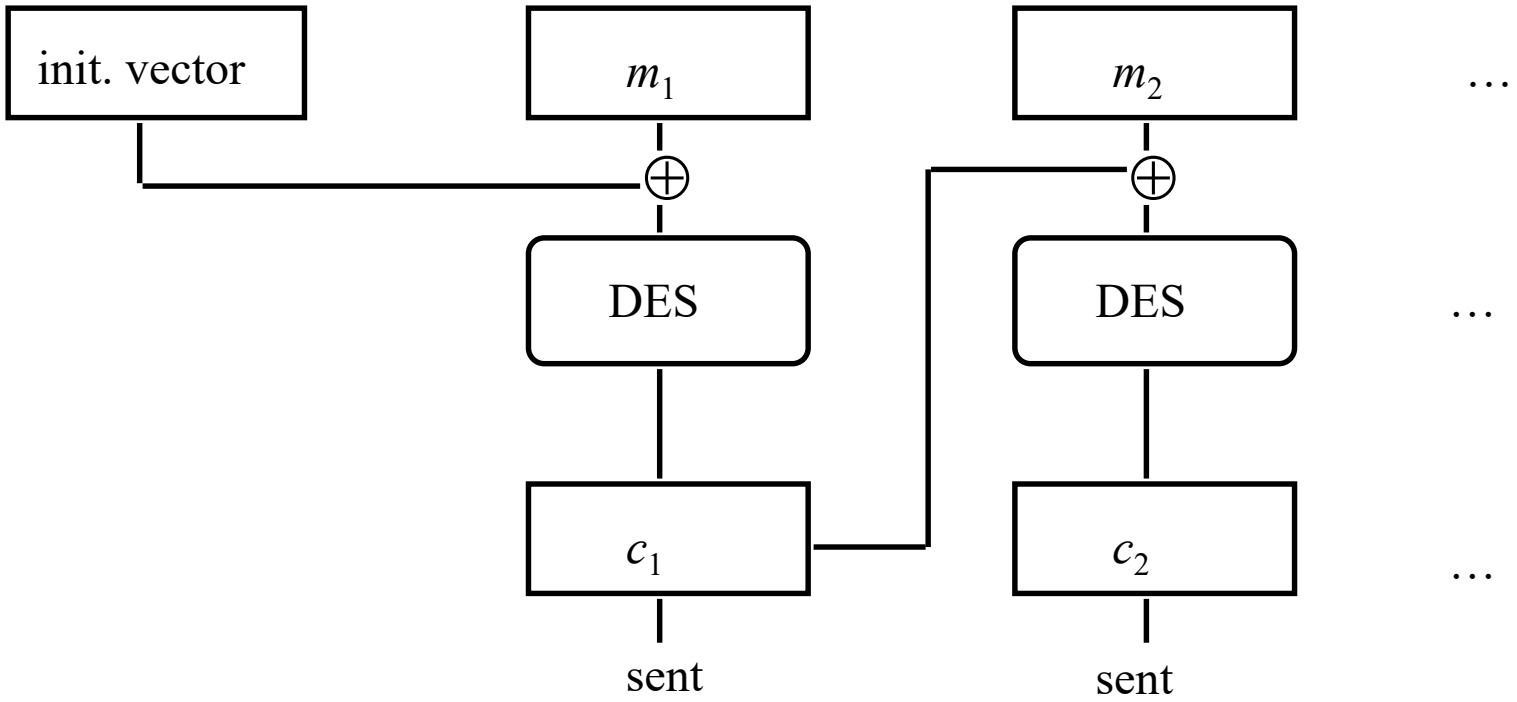
Differential and Linear Cryptanalysis on DES

- Chosen ciphertext attacks
- Differential cryptanalysis: based on how differences in inputs correlate with difference in outputs
  - Requires 2^{47} plaintext-ciphertext pairs
  - Revealed several properties
    - Small changes in S-boxes reduce the number of pairs needed
    - Making every bit of the round keys independent does not impede attack
- Linear cryptanalysis: based on correlations between inputs and outputs
  - improved result, requires 2^{43} plaintext-ciphertext pairs

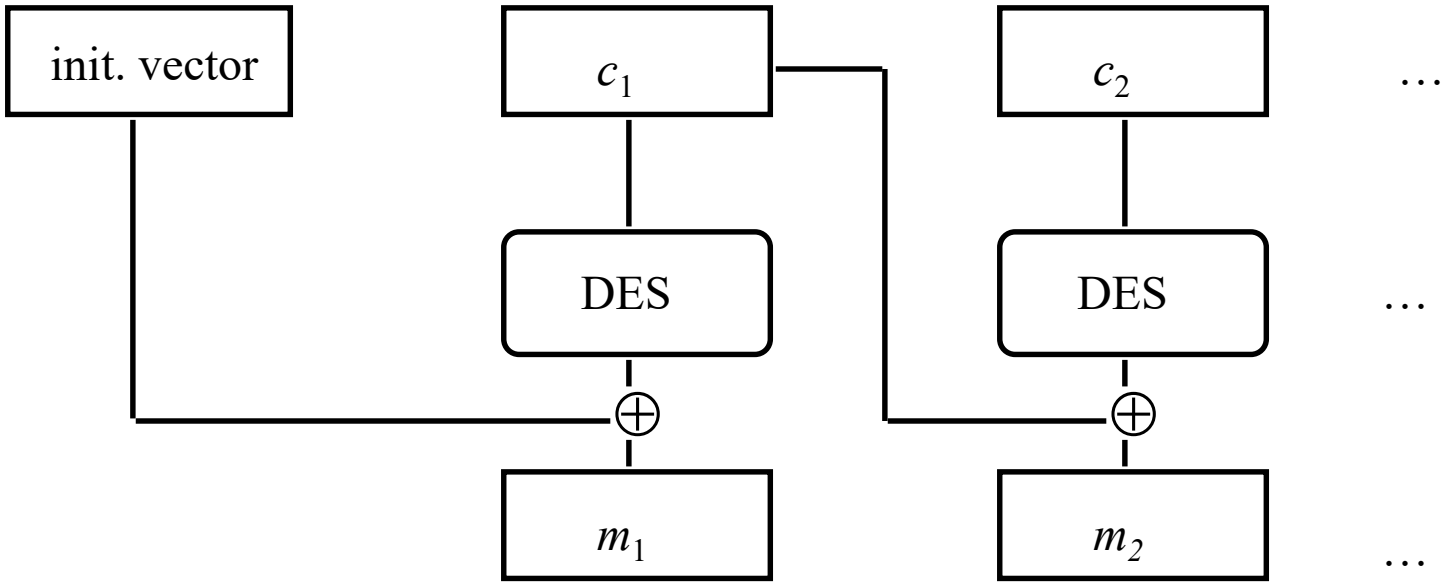
DES Modes

- Electronic Code Book Mode (ECB)
  - Encipher each block independently
- Cipher Block Chaining Mode (CBC)
  - Xor each block with previous ciphertext block
  - Requires an initialization vector for the first one
- Encrypt-Decrypt-Encrypt Mode (2 keys: k, k')
  - $c = \mathrm{DES}_k(\mathrm{DES}_{k'}^{-1}(\mathrm{DES}_k(m)))$
- Encrypt-Encrypt-Encrypt Mode (3 keys: k, k', k'')
  - $c = \mathrm{DES}_k(\mathrm{DES}_{k'}(\mathrm{DES}_{k''}(m)))$

CBC Mode Encryption



CBC Mode Encryption



Self-Healing Property

- Initial message
  - `3231343336353837 3231343336353837`
    `3231343336353837 3231343336353837`
- Received as (underlined 4c should be 4b)
  - `ef7c4cb2b4ce6f3b f6266e3a97af0e2c`
    `746ab9a6308f4256 33e60b451b09603d`
- Which decrypts to
  - `efca61e19f4836f1 3231333336353837`
    `3231343336353837 3231343336353837`
  - Incorrect bytes underlined
  - Plaintext “heals” after 2 blocks

Current Status of DES

- Design for computer system, associated software that could break any DES-enciphered message in a few days published in 1998
- Several challenges to break DES messages solved using distributed computing
- NIST selected Rijndael as Advanced Encryption Standard, successor to DES
  - Designed to withstand attacks that were successful on DES

AES: Result of Open Competition

- NIST held an *open* competition and selected *Rijndael* cipher as Advanced Encryption Standard (AES), a successor to DES
  - NIST issued call for AES cipher in 1997 (http://csrc.nist.gov/archive/aes/pre-round1/aes_9709.htm)
  - 15 candidates accepted in June 1998 (http://csrc.nist.gov/archive/aes/round1/r1report.htm)
  - 5 finalists announced in August 1999 (http://csrc.nist.gov/archive/aes/round1/r1report.htm)
  - *Rijndael* cipher accepted as the winner and AES (http://www.nist.gov/public_affairs/releases/g00-176.cfm and http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf)
  - Designed by Vincent Rijmen and Joan Daemen in Belgium

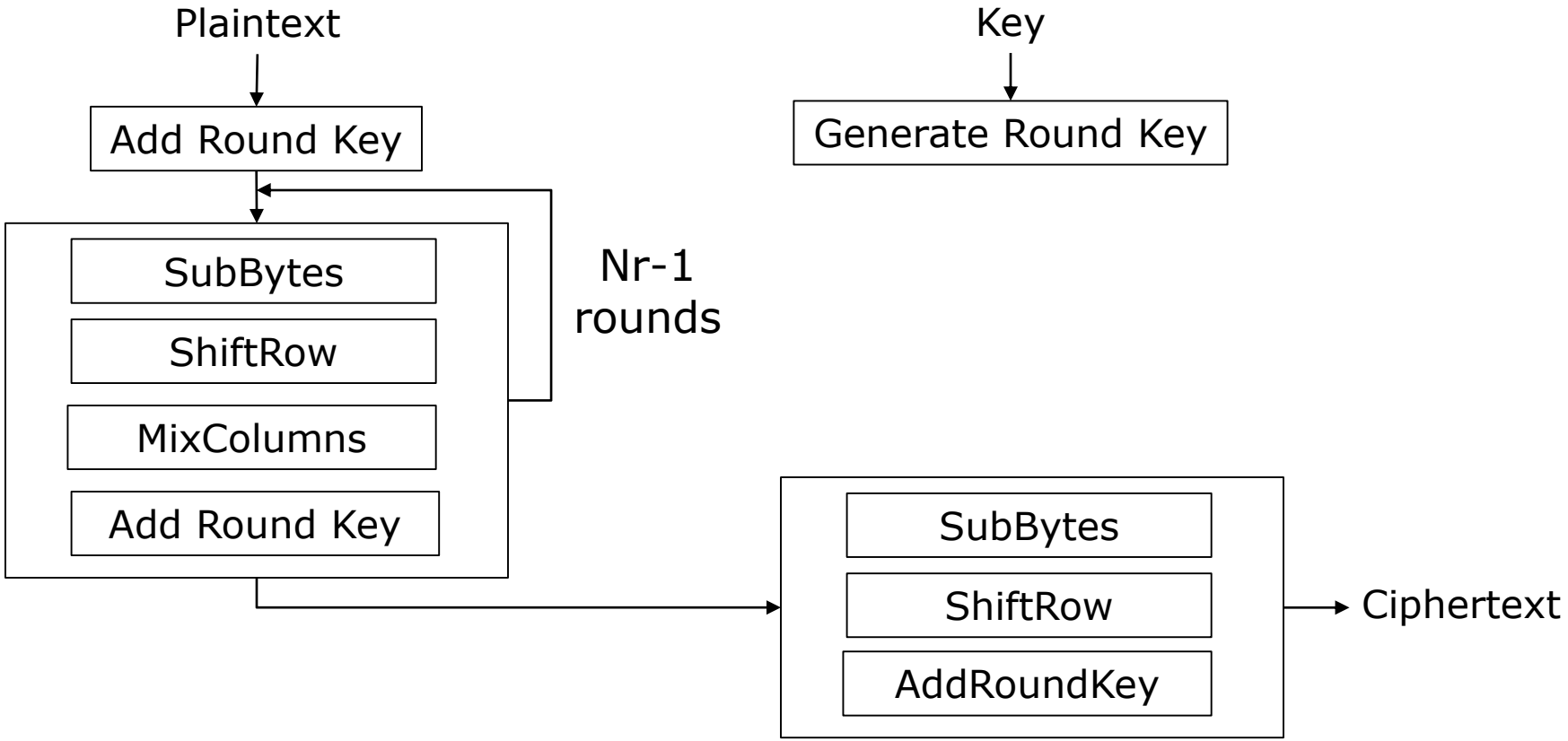
Overview

- Some similarity to DES
  - A product cipher (with transposition and substitution)
  - Operates in rounds
- AES operates on blocks of 128 bits
- AES can use keys of 128, 192, or 256 bits
- Key-block-round combination (a word = 4 bytes = 32 bits): final round slightly different from first Nr – 1 rounds

	Key Length (Nk words)	Block Size (Nb words)	Number of Rounds (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

AES Round

- Final round slightly different from first Nr – 1 rounds



Attacks on AES

- Differential Cryptanalysis
  - High number of rounds increases difficulty of the attack
- Linear Cryptanalysis
  - AES S-box (SubBypes) and MixColumns make the attack difficult

Public Key Cryptography

- Two keys
  - *Private key* known only to individual
  - *Public key* available to anyone
    - Public key, private key inverses
- Idea
  - Confidentiality: encipher using public key, decipher using private key
  - Integrity/authentication: encipher using private key, decipher using public one

Requirements

- It must be computationally easy to encipher or decipher a message given the appropriate key
- It must be computationally infeasible to derive the private key from the public key
- It must be computationally infeasible to determine the private key from a chosen plaintext attack

RSA

- Exponentiation cipher
- Relies on the difficulty of determining the number of numbers relatively prime to a large integer *n*

Background

- Totient function $\phi(n)$
  - Number of positive integers less than *n* and relatively prime to *n*
    - *Relatively prime* means with no factors in common with *n*
- Example: $\phi(10) = 4$
  - 1, 3, 7, 9 are relatively prime to 10
- Example: $\phi(21) = 12$
  - 1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20 are relatively prime to 21

Algorithm

- Choose two large prime numbers p, q
  - Let $n = pq$; then $\phi(n) = (p-1)(q-1)$
  - Choose $e < n$ such that e is relatively prime to $\phi(n)$.
  - Compute d such that $ed \bmod \phi(n) = 1$
- Public key: (e, n); private key: d
- For confidentiality
  - Encipher: $c = m^e \bmod n$
  - Decipher: $m = c^d \bmod n$
- For integrity/authentication
  - Encipher: $c = m^d \bmod n$
  - Decipher: $m = c^e \bmod n$

Example: Confidentiality

- Take $p = 7$, $q = 11$, so $n = 77$ and $\phi(n) = 60$
- Alice chooses $e = 17$, making $d = 53$
- Bob wants to send Alice secret message HELLO (07 04 11 11 14)
  - $07^{17} \bmod 77 = 28$
  - $04^{17} \bmod 77 = 16$
  - $11^{17} \bmod 77 = 44$
  - $11^{17} \bmod 77 = 44$
  - $14^{17} \bmod 77 = 42$
- Bob sends 28 16 44 44 42

Example

- Alice receives 28 16 44 44 42
- Alice uses private key, $d = 53$, to decrypt message:
  - $28^{53} \bmod 77 = 07$
  - $16^{53} \bmod 77 = 04$
  - $44^{53} \bmod 77 = 11$
  - $44^{53} \bmod 77 = 11$
  - $42^{53} \bmod 77 = 14$
- Alice translates message to letters to read HELLO
  - No one else could read it, as only Alice knows her private key and that is needed for decryption

Exercise L6-1

- Take $p = 3$, $q = 5$ and use RSA to encrypt the following message to achieve confidentiality

  H I

Example: Integrity/Authentication

- Take $p = 7$, $q = 11$, so $n = 77$ and $\phi(n) = 60$
- Alice chooses $e = 17$, making $d = 53$
- Alice wants to send Bob message HELLO (07 04 11 11 14) so Bob knows it is what Alice sent (no changes in transit, and authenticated)
  - $07^{53} \bmod 77 = 35$
  - $04^{53} \bmod 77 = 09$
  - $11^{53} \bmod 77 = 44$
  - $11^{53} \bmod 77 = 44$
  - $14^{53} \bmod 77 = 49$
- Alice sends 35 09 44 44 49

Example

- Bob receives 35 09 44 44 49
- Bob uses Alice's public key, $e = 17$, $n = 77$, to decrypt message:
  - $35^{17} \bmod 77 = 07$
  - $09^{17} \bmod 77 = 04$
  - $44^{17} \bmod 77 = 11$
  - $44^{17} \bmod 77 = 11$
  - $49^{17} \bmod 77 = 14$
- Bob translates message to letters to read HELLO
  - Alice sent it as only she knows her private key, so no one else could have enciphered it
  - If (enciphered) message's blocks (letters) altered in transit, would not decrypt properly

Exercise L6-2

- Take $p = 3$, $q = 5$ and use RSA to encrypt the following message to achieve Integrity/Authentication

  H I

Example: Both

- Alice wants to send Bob message HELLO both enciphered and authenticated (integrity-checked)
  - Alice's keys: public (17, 77); private: 53
  - Bob's keys: public: (37, 77); private: 13
- Alice enciphers HELLO (07 04 11 11 14):
  - $(07^{53} \bmod 77)^{37} \bmod 77 = 07$
  - $(04^{53} \bmod 77)^{37} \bmod 77 = 37$
  - $(11^{53} \bmod 77)^{37} \bmod 77 = 44$
  - $(11^{53} \bmod 77)^{37} \bmod 77 = 44$
  - $(14^{53} \bmod 77)^{37} \bmod 77 = 14$
- Alice sends 07 37 44 44 14

Analysis on Example: Security Services (1)

- Confidentiality
  - *Only* the owner of the private key *knows* the private key, so text enciphered with public key cannot be read by anyone except the owner of the private key
- Authentication
  - *Only* the owner of the private key *knows* the private key, so text enciphered with private key must have been generated by the owner

Analysis on Example: Security Services (2)

- Integrity
  - Enciphered letters cannot be changed undetectably without knowing private key
- Non-Repudiation
  - Message enciphered with private key came from someone who knew it

Analysis on Example: Warnings

- Encipher message in blocks should be considerably larger than the examples above
  - If 1 character per block, RSA can be broken using statistical attacks (as in classical cryptosystems)
  - Attacker cannot alter letters, but can rearrange them and alter message meaning
    - Example: reverse enciphered message of text ON to get NO

Cryptographic Checksums

- Mathematical function to generate a set of k bits from a set of n bits (where $k \leq n$).
  - k is smaller then n except in unusual circumstances
- Example
  - ASCII parity bit
    - ASCII has 7 bits; 8th bit is "parity"
    - Even parity: even number of 1 bits
    - Odd parity: odd number of 1 bits

Example Use

- Bob receives "10111101" as bits.
- Sender is using *even* parity
  - Bob counts 6 1-bits and 6 is *even*; no error detected in the character received, assume the received character is correct
  - Note: could still be garbled, but 2 or more bits would need to have been changed to preserve parity
- Sender is using *odd* parity
  - Bob counts 6 of 1-bits. Since 6 is *even*, so character was not received correctly

Cryptographic Checksum

- Strong hash function or strong one-way function
- Cryptographic checksum h: $A \rightarrow B$:

1. For any $x \in A$, $h(x)$ is easy to compute
2. For any $y \in B$, it is computationally infeasible to find $x \in A$ such that $h(x) = y$
3. It is computationally infeasible to find two inputs x, $x' \in A$ such that $x \neq x'$ and $h(x) = h(x')$

Collisions

- If $x \neq x'$ and $h(x) = h(x')$, x and x' are a *collision*
- Pigeonhole principle: if there are n containers for n+1 objects, then at least one container will have 2 objects in it.
- Application: if there are 32 files and 8 possible cryptographic checksum values, at least one value corresponds to at least 4 files

Collision Requirement

3. It is computationally infeasible to find two inputs x, $x' \in A$ such that $x \neq x'$ and $h(x) = h(x')$

Or

4. Given any $x \in A$, it is computationally infeasible to find another $x' \in A$ such that $x \neq x'$ and $h(x) = h(x')$

- Subtle difference between the one: it is considerably harder to find x' meeting the conditions in property 4 than it is to find a pair x and x' meeting the conditions in property 3

Keyless or Keyed Checksum

- Keyless cryptographic checksum
  - Requires no cryptographic key
  - Example
    - MD5 (and MD4), SHA-1 (and SHA-2, SHA-3), HAVAL, and Snefru
- Keyed cryptographic checksum
  - Requires cryptographic key
  - Example
    - DES in chaining mode: encipher message, use last *n* bits. Requires a key to encipher, so it is a keyed cryptographic checksum.

Known Attacks

- MD4 and MD5
  - Dobbertin's attack (1996)
- Snefru
  - Differential cryptanalysis if 4 rounds or less are used (Biham and Shamir, 1993)
- SHA-0, SHA-1, and SHA-2
  - Chabaud and Joux 's attack on SHA-0 (1998)
  - Wang, Yin, and Yu's attack on SHA-1 (2005)
  - Khovratovich, Rechberger and Savelieva's attack on SHA-2 (2011)

HMAC

- Make keyed cryptographic checksums from keyless cryptographic checksums
- h: keyless cryptographic checksum function
  - Input: blocks of b bytes
  - Output: blocks of l bytes
  - k': cryptographic key of length b bytes
  - If short, pad with 0 bytes; if long, hash to length b
  - *ipad* is 00110110 repeated b times
  - *opad* is 01011100 repeated b times
- HMAC-$h(k, m) = h(k' \oplus opad \,||\, h(k' \oplus ipad \,||\, m))$
  - $\oplus$ exclusive or, $||$ concatenation

Strength of HMAC

- Strength of HMAC depends on the strength of the hash function h (Bellare, Canetti, and Krawczyk, 1996)

Exercise L6-3

- In a Linux system, create a checksum for a file.
- Examine how it may be used by surveying a few file downloading sites,
  - e.g., the Fedora Linux project
    - http://fedora.mirrors.tds.net/pub/fedora/releases/20/Fedora/i386/iso/

Summary

- Two main types of cryptosystems: classical and public key
- Classical cryptosystems encipher and decipher using the same key
  - Or one key is easily derived from the other
- Public key cryptosystems encipher and decipher using different keys
  - Computationally infeasible to derive one from the other
- Cryptographic checksums provide a check on integrity