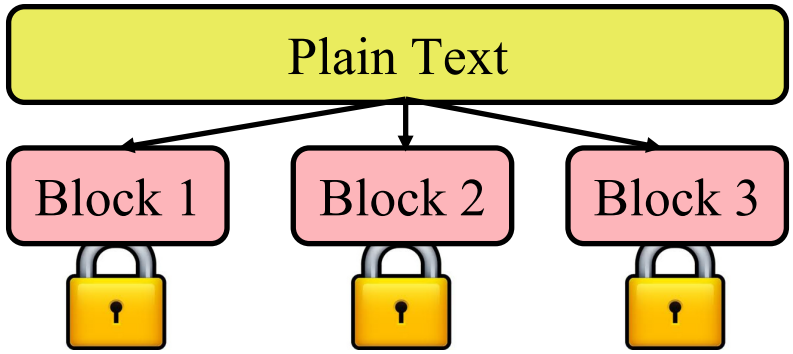
Block Ciphers and DES



Raj Jain
Washington University in Saint Louis
Saint Louis, MO 63130
Jain@cse.wustl.edu

Audio/Video recordings of this lecture are available at:

http://www.cse.wustl.edu/~jain/cse571-11/

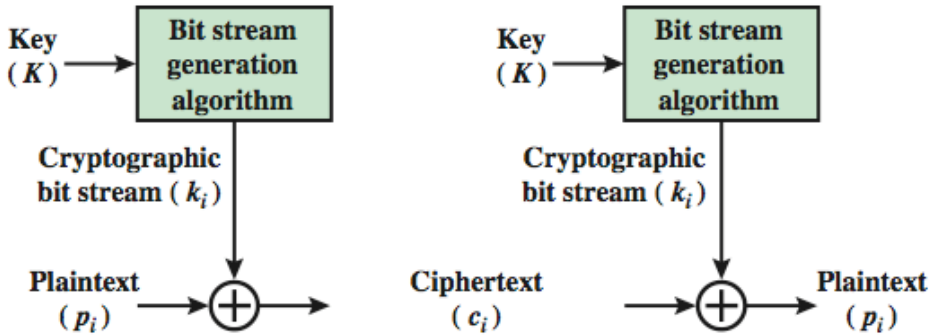
Overview

1. Block Cipher Principles
2. Data Encryption Standard (DES)
3. Differential and Linear Cryptanalysis
4. Block Cipher Design Principles

These slides are based partly on Lawrie Brown's slides supplied with William Stalling's book "Cryptography and Network Security: Principles and Practice," 5th Ed, 2011.

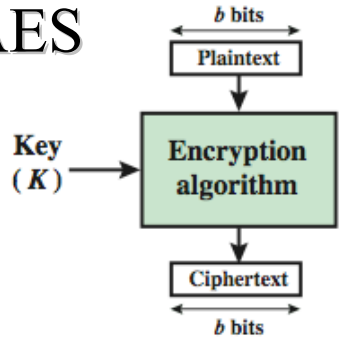
Block vs Stream Ciphers

- **Stream**: Bits and bytes are processed as they arrive
  Example: RC4



(a) Stream Cipher Using Algorithmic Bit Stream Generator

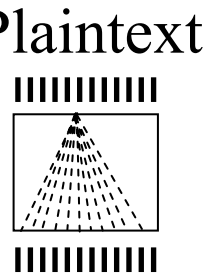
- **Block**: Messages are broken into blocks of 64-bit, 512-bit, …
  Example: DES, AES



(b) Block Cipher

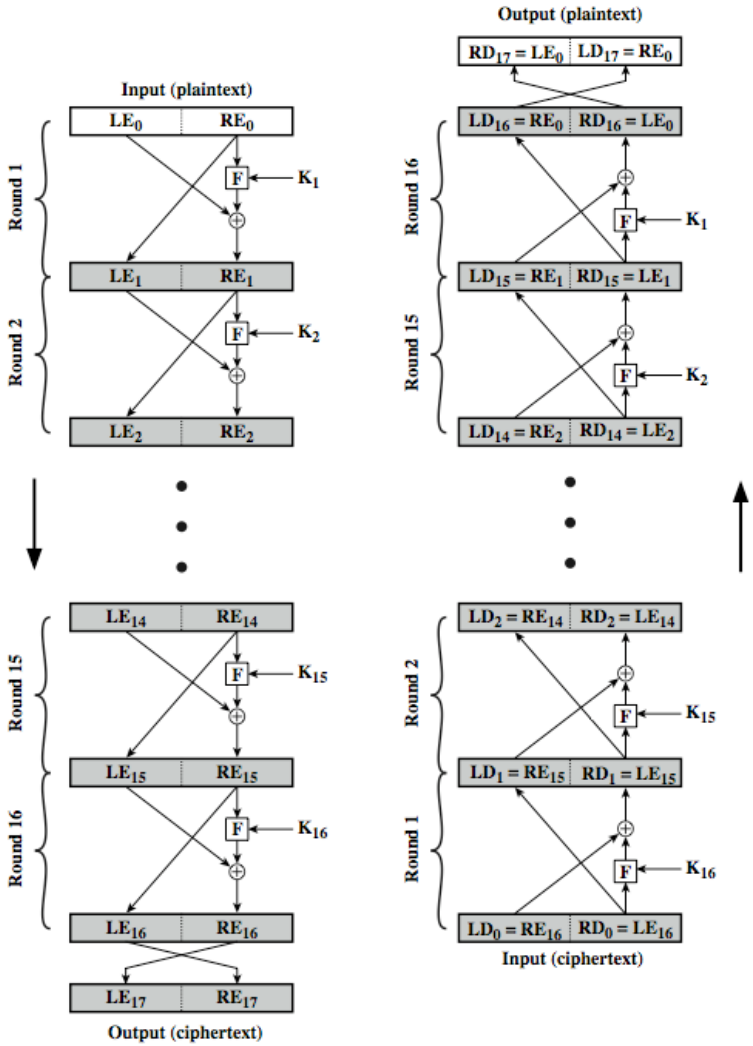
Shannon's S-P Networks

- Claude Shannon introduced idea of substitution-permutation (S-P) networks in his 1949 paper
- Two primitive cryptographic operations:
  - ***Substitution*** **(S-box)** = Replace n-bits by another n-bits
    ⇒ **Diffusion**: Dissipate statistical structure of plaintext over bulk of ciphertext.
    One bit change in plaintext changes many bits in ciphertext.
    Can not do frequency analysis.
  - ***Permutation*** **(P-box)** = Bits are rearranged.
    No bits are added/removed.
    ⇒ **Confusion**: Make relationship between ciphertext and key as complex as possible
- Combination S-P = Product cipher



Feistel Cipher Structure

- A practical implementation of Shanon's S-P Networks
- Partitions input block in 2 halves
  - Perform a substitution on left data half based on a function of right half & subkey (Round Function or Mangler function)
  - Then permutation by swapping halves
  - Repeat this "round" of S-P many times
- Invertible



Feistel Cipher Design Elements

Most modern block ciphers are a variation of Feistel Cipher with different:

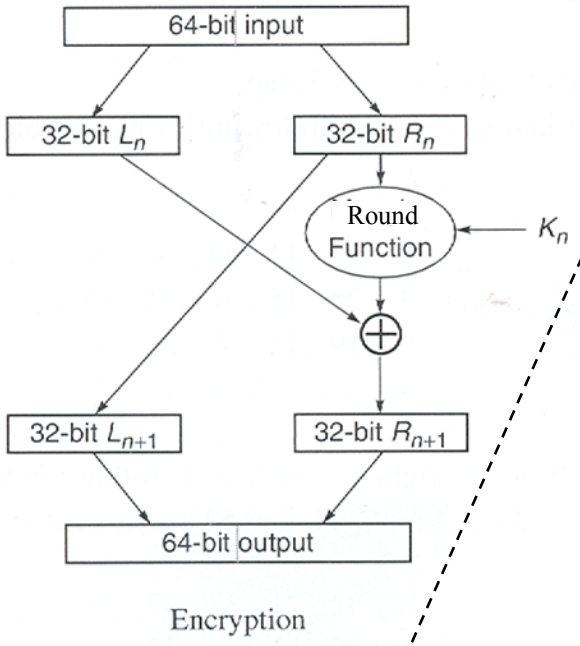
1. Block size
2. Key size
3. Number of rounds
4. Subkey generation algorithm
5. Round function
6. Fast software en/decryption
7. Ease of analysis

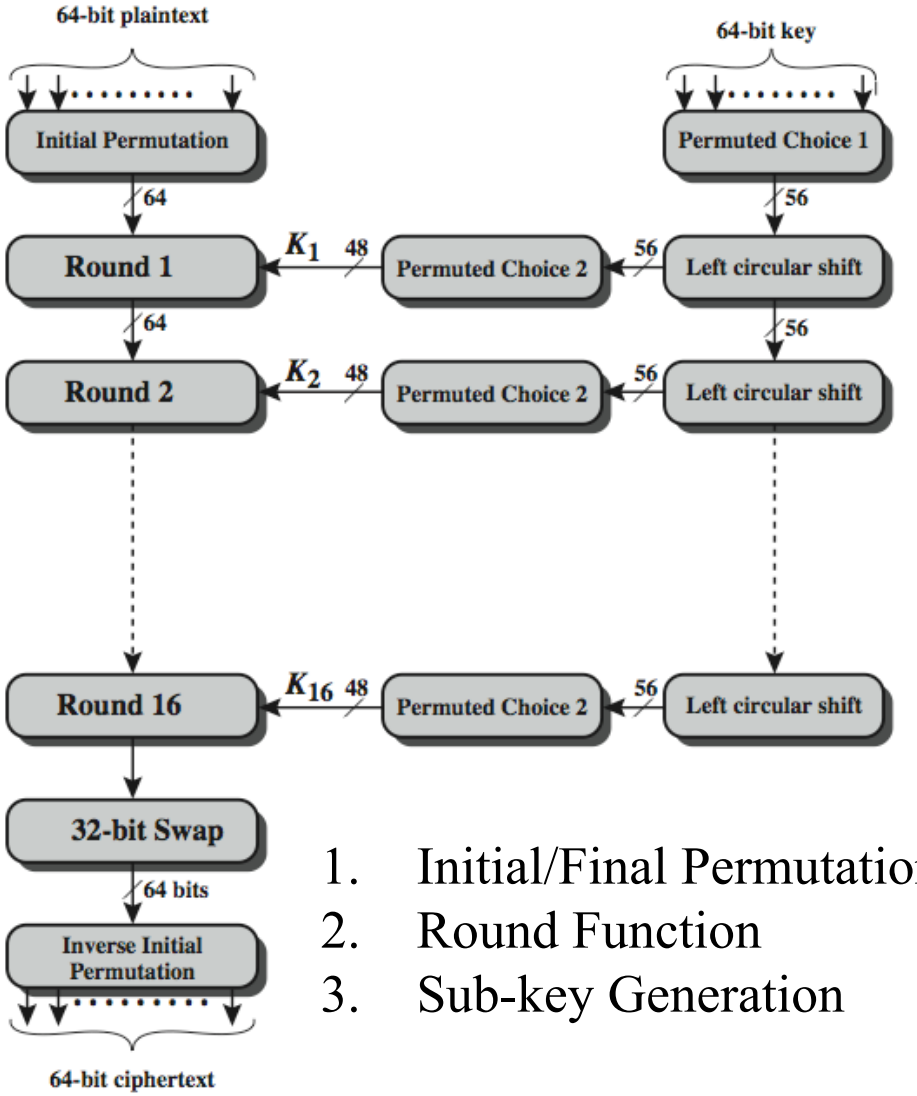
Data Encryption Standard (DES)

- Published by National Bureau of Standards in 1977
- A variation of IBM's Lucifer algorithm developed by Horst Feistel
- For commercial and *unclassified* government applications
- 8 octet (64 bit) key.
  Each octet with 1 odd parity bit ⇒ 56-bit key
- Efficient hardware implementation
- Used in most financial transactions
- Computing power goes up 1 bit every 2 years
- 56-bit was secure in 1977 but is not secure today
- Now we use DES three times ⇒ Triple DES = 3DES

DES Encryption Overview

- 16 rounds using 64-bit block and 48-bit subkey





1. Initial/Final Permutation
2. Round Function
3. Sub-key Generation

1. Initial and Final Permutation

Initial Permutation (IP)

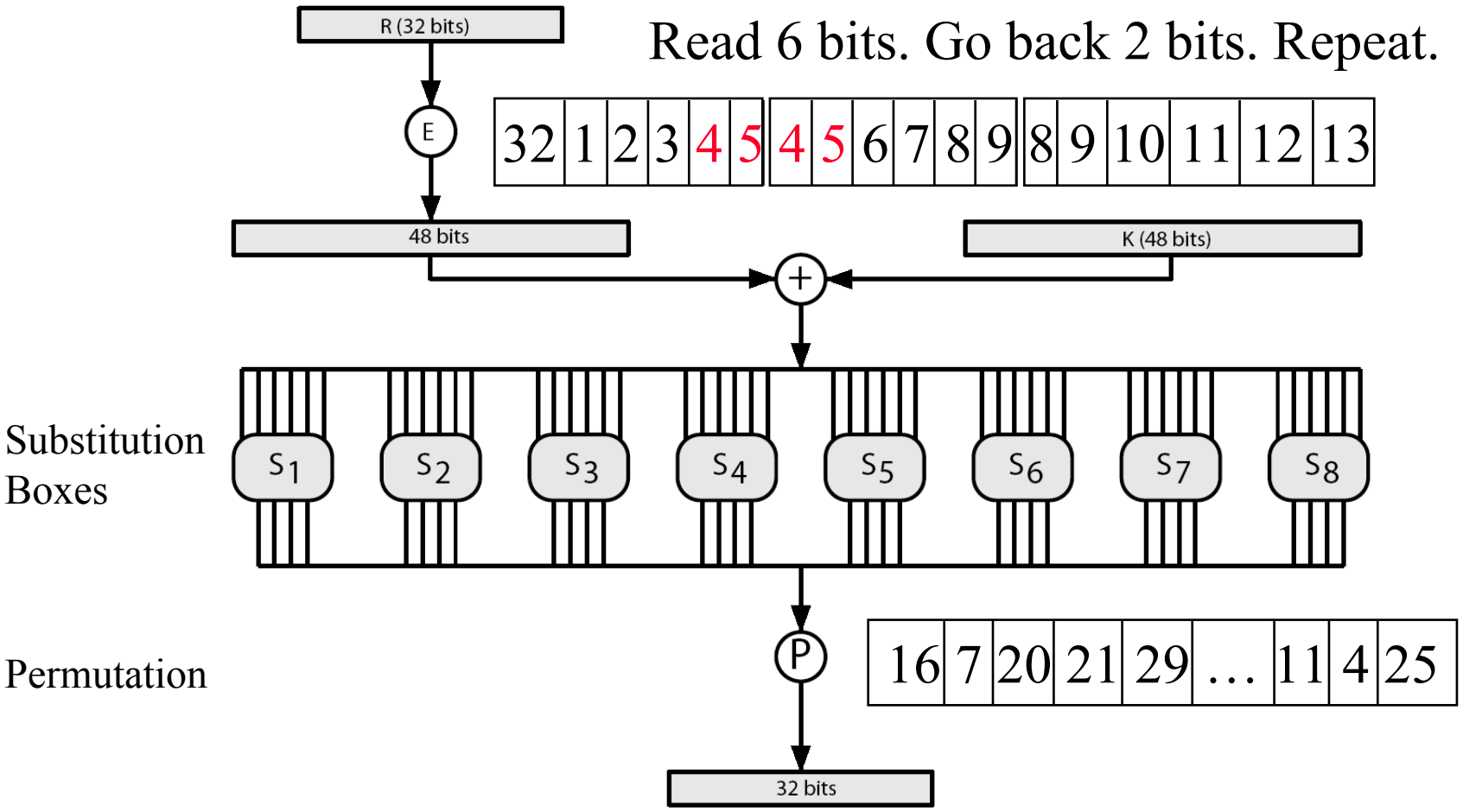
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

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

- Input bit 58 goes to output bit 1
  Input bit 50 goes to output bit 2, …
- Even bits to LH half, odd bits to RH half
- Quite regular in structure (easy in h/w)

2. DES Round Structure



Substitution Boxes

- Map 6 to 4 bits
- Outer bits 1 & 6 (**row** bits) select one row of 4
- Inner bits 2-5 (**column** bits) are substituted
- Example:

Input bits 1 and 6 ↓	Input bits 2 thru 5															
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

3. DES Sub-Key Generation

- Permutation PC1 divides 56-bits in two 28-bit halves
- Rotate **each half** separately either 1 or 2 places depending on the **key rotation schedule** K
- Select 24-bits from each half & permute them by PC2

(a) Input Key

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

(b) Permuted Choice One (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

(c) Permuted Choice Two (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

(d) Schedule of Left Shifts

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES Decryption

- Decrypt with Feistel design: Do encryption steps again using sub-keys in reverse order (SK16 … SK1)
  - IP undoes final FP step of encryption
  - 1st round with SK16 undoes 16th encrypt round
  - ….
  - 16th round with SK1 undoes 1st encrypt round
  - Then final FP undoes initial encryption IP thus recovering original data value

Avalanche Effect

- Key desirable property of encryption algorithm
- A change of **one** input or key bit results in changing approx **half** output bits = **Diffusion**
- Making attempts to "home-in" by guessing keys impossible
- DES exhibits strong avalanche

Avalanche in DES

Round		δ
	02468aceeca86420	1
	12468aceeca86420	
1	3cf03c0fbad22845	1
	3cf03c0fbad32845	
2	bad2284599e9b723	5
	bad3284539a9b7a3	
3	99e9b7230bae3b9e	18
	39a9b7a3171cb8b3	
4	0bae3b9e42415649	34
	171cb8b3ccaca55e	
5	4241564918b3fa41	37
	ccaca55ed16c3653	
6	18b3fa419616fe23	33
	d16c3653cf402c68	
7	9616fe2367117cf2	32
	cf402c682b2cefbc	
8	67117cf2c11bfc09	33
	2b2cefbc99f91153	

Round		δ
9	c11bfc09887fbc6c	32
	99f911532eed7d94	
10	887fbc6c600f7e8b	34
	2eed7d94d0f23094	
11	600f7e8bf596506e	37
	d0f23094455da9c4	
12	f596506e738538b8	31
	455da9c47f6e3cf3	
13	738538b8c6a62c4e	29
	7f6e3cf34bc1a8d9	
14	c6a62c4e56b0bd75	33
	4bc1a8d91e07d409	
15	56b0bd7575e8fd8f	31
	1e07d4091ce2e6dc	
16	75e8fd8f25896490	32
	1ce2e6dc365e5f59	
IP^{-1}	da02ce3a89ecac3b	32
	057cde97d7683f2a	

3+4+3+3+1+0+2+3+2+3+1+2+2+2+1+1=33 bits

Strength of DES

- Bit-wise complement of plaintext with complement of key results in complement of ciphertext
- Brute force search requires 2^{55} keys
- Recent advances have shown, it is possible
  - in 1997 on Internet in a few months
  - in 1998 on dedicated h/w (EFF) in a few days
  - in 1999 above combined in 22hrs!
- Statistical Attacks:
  - Timing attacks: calculation time depends upon the key. Particularly problematic on smartcards
  - Differential cryptanalysis
  - Linear cryptanalysis

Differential Cryptanalysis

- Chosen Plaintext attack: Get ciphertext for a given plaintext
- Get the (ΔX, ΔY) pairs, where ΔX is the difference in plaintext and ΔY is the difference in ciphertext
- Some (ΔX, ΔY) pairs are more likely than others, if those pairs are found, some key values are more likely so you can reduce the amount of brute force search
- Straightforward brute force attack on DES requires 2^{55} plaintexts
- Using differential cryptanalysis, DES can be broken with 2^{47} plaintexts. But finding appropriate plaintexts takes some trials and so the total amount of effort is $2^{55.1}$ which is more than straight forward brute force attack

  $\Rightarrow$ DES is resistant to differential cryptanalysis
- Ref: http://en.wikipedia.org/wiki/Differential_cryptanalysis

Linear Cryptanalysis

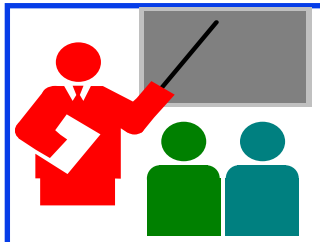
- Bits in plaintext, ciphertext, and keys may have a linear relationship. For example:

$$P1 \oplus P2 \oplus C3 = K2 \oplus K5$$

- In a good cipher, the relationship should hold w probability ½. If any relationship has probability 1, the cipher is easy to break. If any relationship has probability 0, the cipher is easy to break.
- Bias = |Probability of linear relationship – 0.5|
- Find the linear approximation with the highest bias
  $\Rightarrow$ Helps reduce the bruteforce search effort.
- This method can be used to find the DES key given 2^{43} plaintexts.
- Ref: http://en.wikipedia.org/wiki/Linear_cryptanalysis

Block Cipher Design Principles

- Nonlinear S-Boxes: Resistant to linear cryptanalysis.
  Linear approximations between input and output bits of the S-boxes should have minimal bias $\Rightarrow P \approx \frac{1}{2}$
- S-Boxes resistant to differential cryptanalysis.
  All (Input bit difference, output bit difference) pairs should be equally likely.
- Any output bit should change with probability ½ when any input bit is changed (strict avalanche criterion)
- Output bits *j* and *k* should change independently when any input bit *i* is inverted for all *i*, *j*, *k* (bit independence criterion)
- Permutation: Adjacent bits should affect different S-Boxes in the next round $\Rightarrow$ Increase diffusion
- More rounds are better (but also more computation)



Summary

1. Goal of ciphers is to increase confusion and diffusion.
   Confusion = Complex relationship
   Diffusion = Each input bit affects many output bits
2. Feistel cipher design divides blocks in left and right halves, mangles the right half with a sub-key and swaps the two halves.
3. DES consists of 16 rounds using a 56-bit key from which 48-bit subkeys are generated. Each round uses eight 6x4 S-Boxes followed by permutation.
4. Differential cryptanalysis analyzes frequency of (ΔP, ΔC) pairs. Linear cryptanalysis analyzes frequency of linear relationships among plaintext, ciphertext, and key.
5. Block ciphers should be nonlinear, complex, maximize diffusion.

Homework 3

- Submit answer to Problem 3.8 (One round version of DES)
  Final Answer = F0AAF0AA 5E1CEC63

Lab Homework 3

This lab consists of using the following tools:

1. SMBdie: A tool to crash windows server described at http://www.windowsecurity.com/articles/SMBDie_Crashing_Windows_Servers_with_Ease.html download from http://packetstormsecurity.org/0208-exploits/SMBdie.zip
2. Snort, vulnerability scanner, http://www.snort.org/
3. Password dump, Pwdump3, http://www.openwall.com/passwords/microsoft-windows-nt-2000-xp-2003-vista-7#pwdump
4. John the ripper, Brute force password attack, http://www.openwall.com/john/

Lab Homework 3 (Cont)

- If you have two computers, you can install these programs on one computer and conduct these *exercises. Your anti-virus programs will prevent you from doing so*.
- Alternately, you can remote desktop via VPN to CSE571XPC2 and conduct exercises 1-3 and then remote desktop to CSE571XPS and conduct exercise 4.
- You need to reserve time in advance.
- Use your last name (with spaces removed) as your user name.

1. PWDump3

- Goal: Get the password hash from the server CSE571XPS
- On CSE571XPC2, open a dos box
- CD to c:\pwdump3
- Run pwdump3 without parameters for help
- Run pwdump3 with parameters to get the hash file from server CSE571XPS
- You will need the common student account and password supplied in the class.
- Open the hash file obtained in notepad. Delete all lines except the one with your last name.
- Save the file as c:\johntheripper\<your_last_name>.txt
- Delete the original full hash file that you downloaded

2. Find your password

- On CSE571XPC2, use the command box
- CD to c:\johntheripper
- Delete john.pot and john.log
- Run johntheripper without parameters to get help
- Run johntheripper with the file you created in step 1
- This will tell you your password. Note down the contents of john.pot file and submit.
- Delete your hash file, john.pot, and john.log
- logout
- Close your remote desktop session.

3. Change your password

- Now remote desktop to CSE571XPS
- Login using your last name as username and the password you obtained in step 2.
- **Change your password** to a strong password.
  Do this from your own account (not the common student account).
- Note the time and date you change the password. Submit the time as homework answer.
- Logout

4. Snort

- Remote desktop to CSE571XPC2. Note down its IP address. Its NETBIOS name is CSE571XPC2
- Delete all the previous logs, if any, c:\snort\bin\log*.*
- Start snort from the desktop icon (or snort.exe in c:\snort\bin\)
- Start another remote desktop session to CSE571XPS
- Log in to the server using your lastname and password
- Run smbdie on the server to attack CSE571XPC2
- When the program stops, logoff from the server.
- Resume your remote desktop to CSE571XPC2
- Use control-C to stop snort
- Type the log file c:\snort\bin\log\alert.ids
- Find one entry that mentions Denial of Service attack. Note the IP address and port of the attacker and submit.
- Delete the logs, c:\snort\bin\log*.* Logout