

Cryptography and Network Security Chapter 2

Block Ciphers and the Data Encryption
Standard

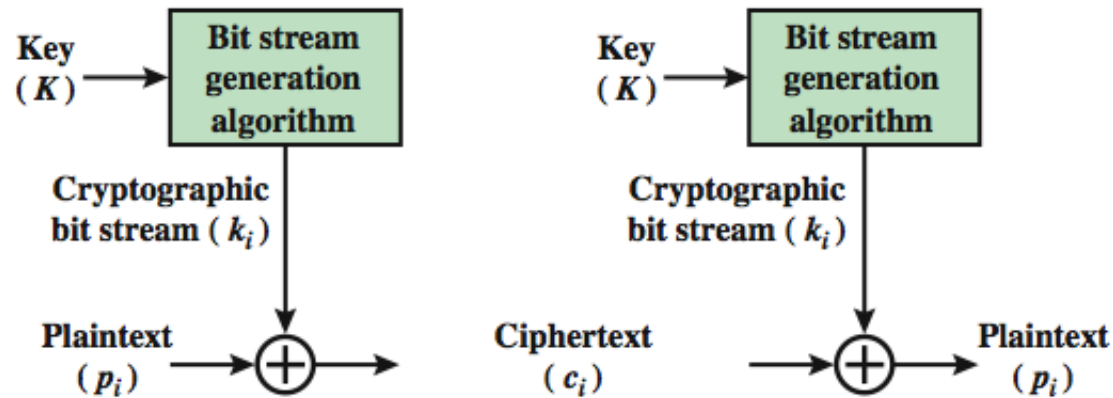
Modern Block Ciphers

- now look at modern block ciphers
- one of the most widely used types of cryptographic algorithms
- provide secrecy /authentication services
- focus on DES (Data Encryption Standard)
- to illustrate block cipher design principles based on a structure referred to as Feistel block cipher

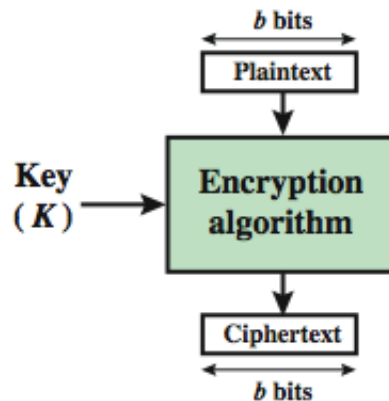
Block vs Stream Ciphers

- block ciphers process messages in blocks, each of which is then en/decrypted to produce a ciphertext/plaintext of equal length.
- like a substitution on very big characters_(64-bits or more)
- stream ciphers process messages a bit or byte at a time when en/decrypting where the two users share a symmetric encryption key
 - Autokeyed Vigenère cipher
 - One-time pad version of Vernam cipher:
 - The keystream (k_i) is as long as the plaintext bit stream (p_i)
 - If the cryptographic keystream is random => the cipher is unbreakable by any means other than acquiring the keystream.
- many current ciphers are **block ciphers**
 - better analyzed
 - broader range of applications

Block vs Stream Ciphers



(a) Stream Cipher Using Algorithmic Bit Stream Generator



(b) Block Cipher

Block Cipher Principles

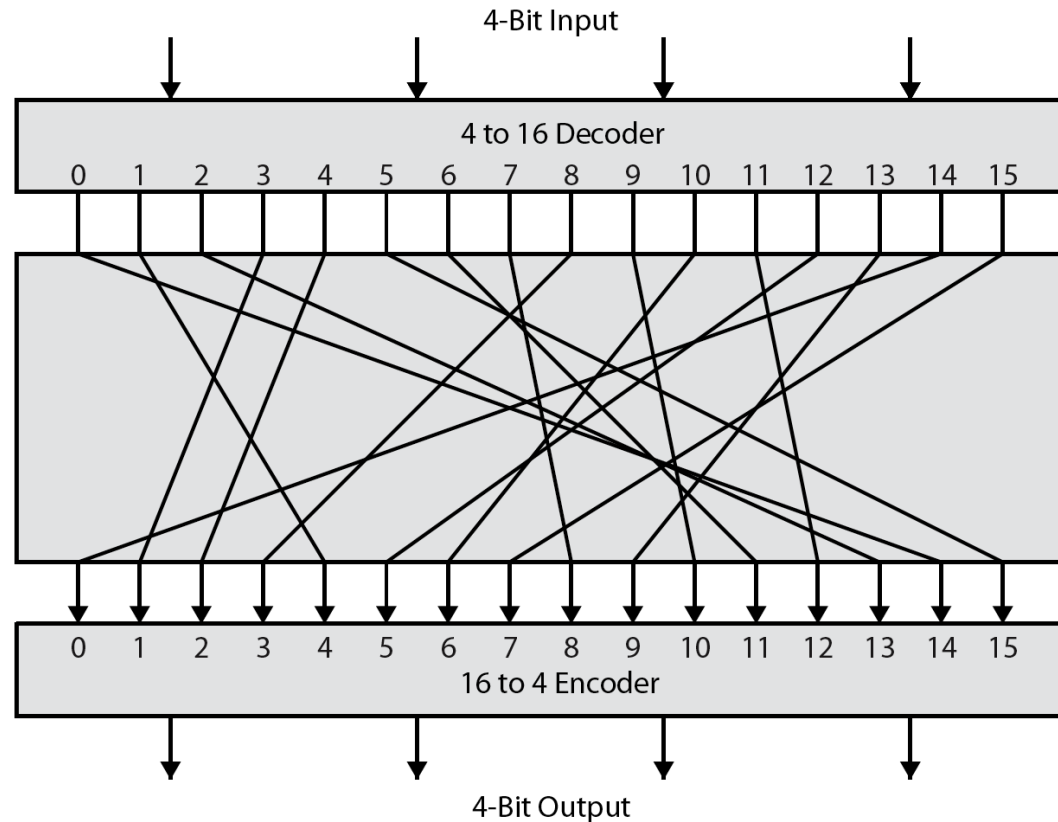
- Block ciphers look like an extremely large substitution
- A block cipher operates on plaintext of n bits to produce a ciphertext block of n bits $\Rightarrow 2^n$ possible different plaintext blocks.
 - would need table of 2^{64} entries for a 64-bit block
 - For n -bits general substitution block cipher, the size of the key $n \times 2^n$.
- instead create from smaller building blocks
 - 64 or 128 bits.
- Reversible (or nonsingular) encryption $\Rightarrow$ each plaintext block must produce a unique ciphertext block $\Rightarrow 2^n!$ different transformations (reversible mappings).
 - needed since must be able to **decrypt** ciphertext to recover messages efficiently
 - Why $2^n!$?? For the first plaintext, we can choose any of the 2^n ciphertext blocks. For the second plaintext, we choose from among $2^n - 1$ remaining ciphertext blocks, and so on.
- most symmetric block ciphers are based on a **Feistel Cipher Structure** (*ideal block cipher*)
 - using idea of a **product** cipher

Ideal Block Cipher (IDC)

Ideal block cipher allows for the maximum number of possible encryption mappings from the plaintext block.

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

The encryption and decryption mappings can be defined by a tabulation.



Tabulation

The mapping between plaintext and ciphertext can be defined by the entries in the second column.

The required key length is
(4bits) x (16 rows) = 64 bits

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

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

reversible mappings between plaintext and ciphertext

IDC weaknesses

- If a small block size (e.g., $n = 4$) is used, then the system is equivalent to a classical substitution cipher
 - Vulnerable to statistical analysis of the plaintext
- This weakness is not inherent in the use of a substitution cipher but rather results from the use of a small block size.
- If n is sufficiently large & arbitrary reversible substitution between the plaintext and ciphertext is used
 - = > the statistical characteristics of the source plaintext are masked
 - => such cryptanalysis is infeasible.

Use of linear equations

For $n = 4$, suppose we define the mapping in terms of linear equations

$$y_1 = k_{11}x_1 + k_{12}x_2 + k_{13}x_3 + k_{14}x_4$$

$$y_2 = k_{21}x_1 + k_{22}x_2 + k_{23}x_3 + k_{24}x_4$$

$$y_3 = k_{31}x_1 + k_{32}x_2 + k_{33}x_3 + k_{34}x_4$$

$$y_4 = k_{41}x_1 + k_{42}x_2 + k_{43}x_3 + k_{44}x_4$$

x_i : the 4 binary digits of the ciphertext block

k_{ij} : the binary coefficients

Arithmetic : mode 2

$\Rightarrow$ The key size is $n^2 = 16$ bits.

⚠ vulnerable to cryptanalysis by an attacker aware of the structure of the algorithm

Claude Shannon and Substitution-Permutation Ciphers

- Claude Shannon introduced idea of substitution-permutation (S-P) networks in 1949 paper
- form basis of modern block ciphers
- S-P nets are based on the two primitive cryptographic operations seen before:
 - *substitution* (S-box)
 - *permutation* (P-box)
- provide *confusion* & *diffusion* of message & key
 - Feistel is a practical application

Confusion and Diffusion

- cipher needs to completely obscure statistical properties of original message to thwart cryptanalysis based on statistical analysis.
- a one-time pad does this
- more practically Shannon suggested combining S & P elements to obtain:
- **diffusion** – dissipates statistical structure of plaintext over bulk (long-range of statistics) of ciphertext.
 - Achieved by having each plaintext digit affect the value of many ciphertext digits
- **confusion** – makes relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible

Feistel Cipher

- The Feistel cipher proposed that we can approximate the ideal block cipher by utilizing the concept of product cipher
 - The execution of 2 or more simple ciphers in sequence in such a way that the final result or product is **cryptographically stronger** than any of the component ciphers.
 - to develop a block cipher with a key length of k bits and a block cipher of n bits allowing a total of 2^k possible transformations instead of $2^n!$
- Feistel proposed the use of a cipher that alternates substitutions and permutations
 - **Substitution:** *each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.*
 - **Permutation:** *a sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed.*
 - <https://www.youtube.com/watch?v=FGhj3CGxl8I>

Feistel Cipher Design Elements

- block size ($n \text{ bits} = \{64, 128 \text{ (AES)}\}$)
- key size ($k \text{ bits} = 128$)
- number of rounds (*a typical value is 16 rounds*)
- subkey generation algorithm (*greater complexity => greater difficulties of cryptanalysis*)
- round function (*greater complexity => greater difficulties of cryptanalysis*)
- fast software en/decryption (*hardware v/s software*)
- ease of analysis (*algorithm can be concisely and clearly explained => easier to analyze for cryptanalytic vulnerabilities => develop a higher level of assurance as to its strength*)

Feistel Cipher Structure

- Horst Feistel devised the **Feistel cipher**
 - based on concept of invertible product cipher
- Partitions input block into two halves
 - process through multiple rounds which
 - perform a substitution on left data half
 - based on round function of right half & subkey
 - then have permutation swapping halves
- Implements particular form of Shannon's S-P network (SPN) concept

Feistel Cipher Structure

Input = plaintext block of length $2w$ bits divided into two halves LE_0 and RE_0 and a key k .

The data passes through n rounds of processing and then combine to produce the ciphertext.

For the i th iteration:

- Encryption:

$$LE_i = RE_{i-1}$$

$$RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i)$$

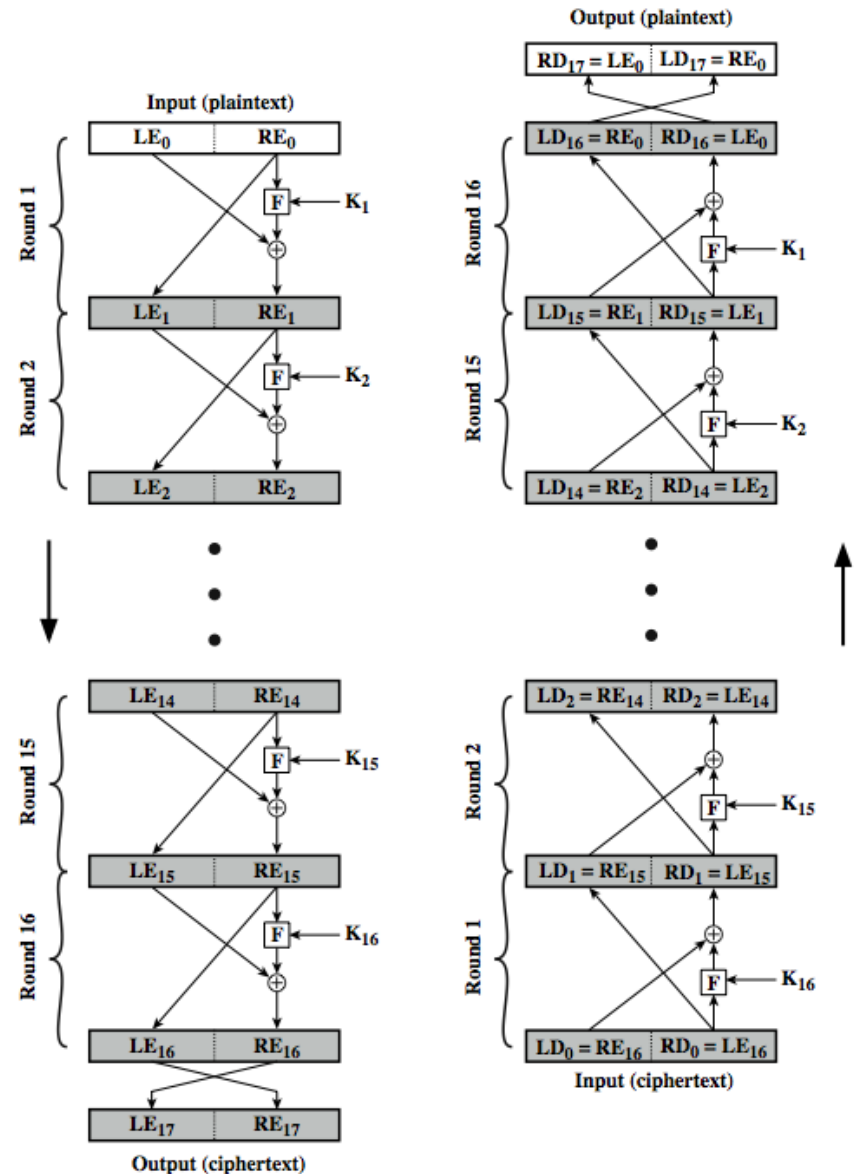
- Rearranging terms:

$$RE_{i-1} = LE_i$$

$$LE_{i-1} = RE_i \oplus F(RE_{i-1}, K_i) = RE_i \oplus F(LE_i, K_i)$$

- decryption:

Same algorithm as encryption



Data Encryption Standard (DES)

- most widely used block cipher in world
- adopted in 1977 by NBS (now NIST)
 - as FIPS PUB 46
- encrypts 64-bit data using 56-bit key
- has widespread use
- has been considerable controversy over its security

DES History

- IBM developed Lucifer cipher
 - by team led by Feistel in late 60's
 - used 64-bit data blocks with 128-bit key
- then redeveloped as a commercial cipher with input from NSA and others
- in 1973 NBS issued request for proposals for a national cipher standard
- IBM submitted their revised Lucifer which was eventually accepted as the DES

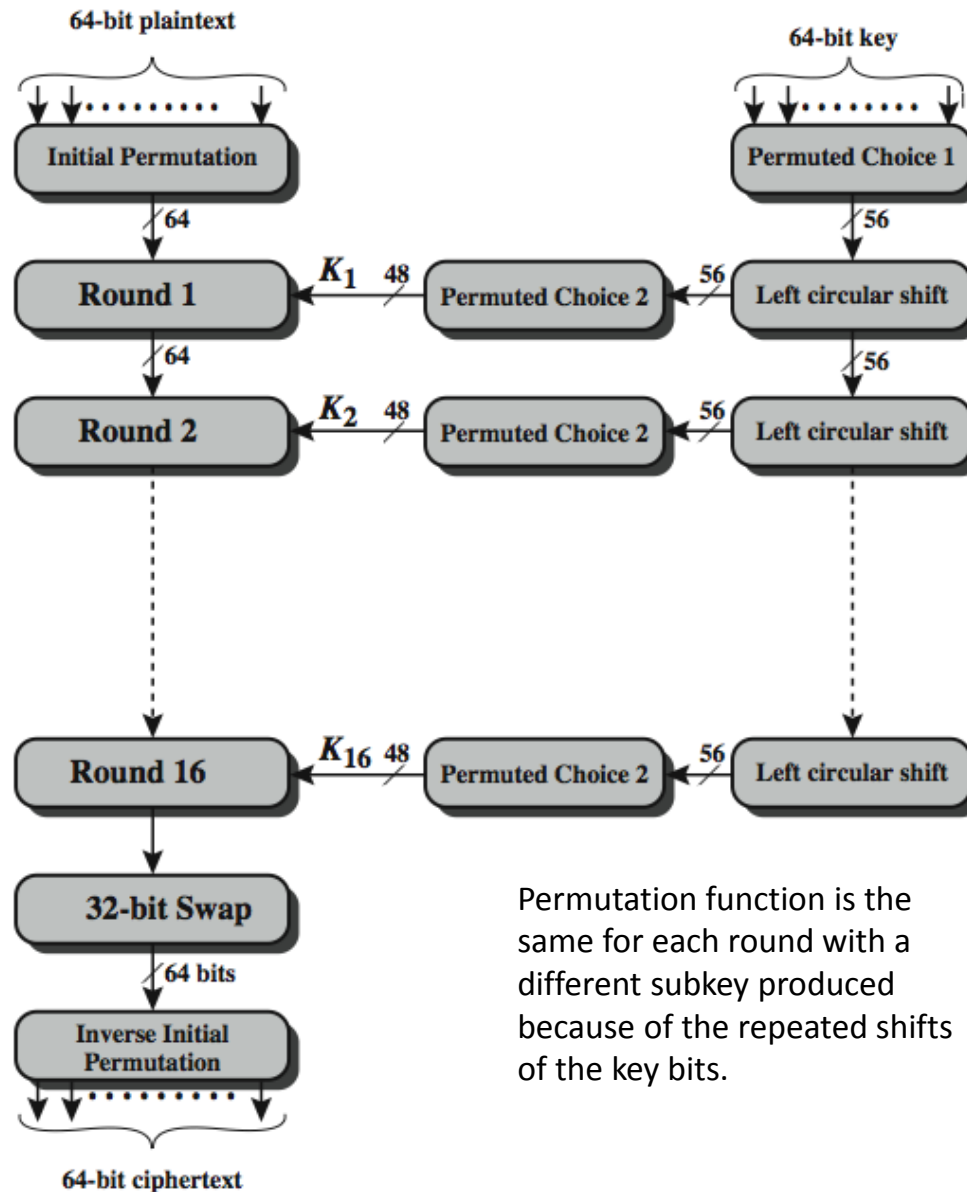
DES Design Controversy

- although DES standard is public
- was considerable controversy over design
 - in choice of 56-bit key (vs Lucifer 128-bit)
 - and because design criteria were classified
- subsequent events and public analysis show in fact design was appropriate
- use of DES has flourished
 - especially in financial applications
 - still standardised for legacy application use
- In 1999, NIST issued a new version (FIPS Pub 46-3)
 - 3DES (in essence involves repeating the DES algorithm three times on the plaintext using two or three different keys to produce ciphertext).

DES Encryption Overview

The left side shows the basic process for enciphering a 64-bit data block which consists of:

- an initial permutation (IP) which shuffles the 64-bit input block
- 16 rounds of a complex key dependent round function involving substitutions & permutations; left and right halves of the 16th round are swapped
- a final permutation, being the inverse of IP



The right side shows the handling of the 56-bit key and consists of:

- an initial permutation of the key (PC1) which selects 56-bits out of the 64-bits input, in two 28-bit halves
- 16 stages to generate the 48-bit subkeys using a left circular shift and a permutation of the two 28-bit halves.

Initial Permutation IP

- first step of the data computation
- IP reorders the input data bits
- even bits to LH half, odd bits to RH half
- quite regular in structure
- example:

$IP(675a6967 \ 5e5a6b5a) = (ffb2194d \ 004df6fb)$

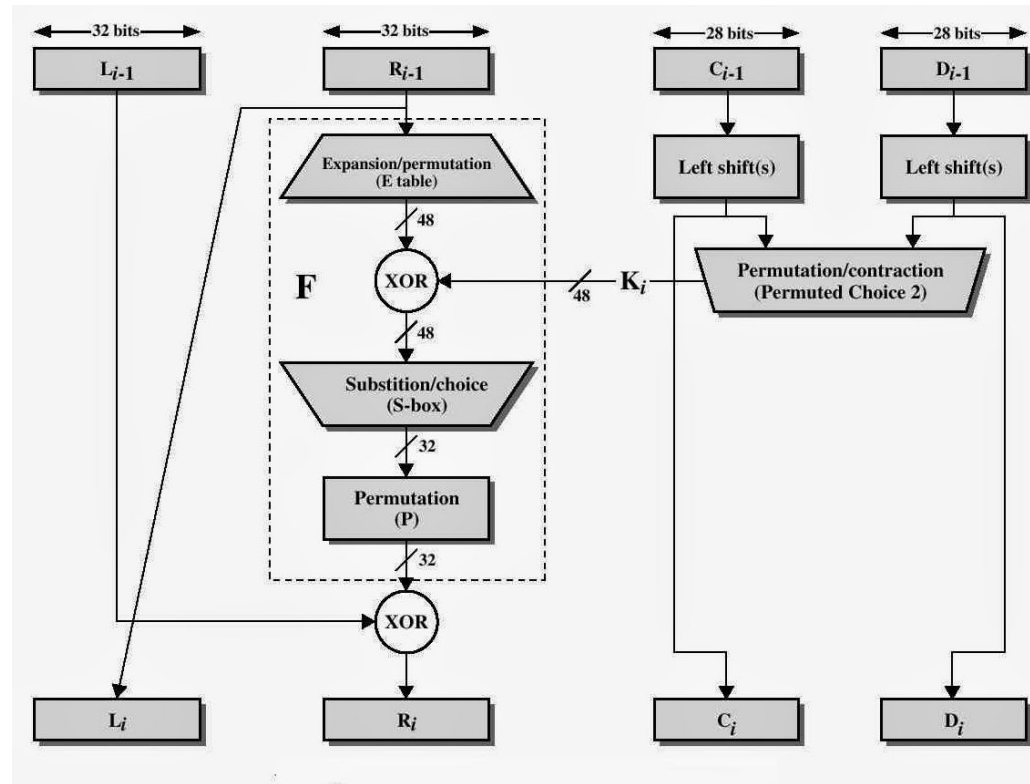
DES Round Structure

- uses two 32-bit L & R halves
- as for any Feistel cipher can describe as:

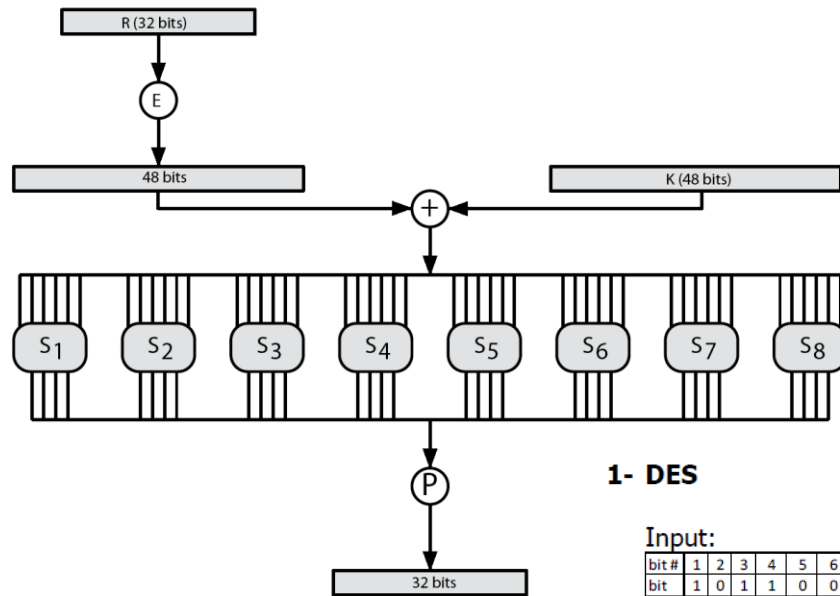
$$L_i = R_{i-1}$$

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

- F takes 32-bit R half and 48-bit subkey:
 - expands R to 48-bits using perm E
 - adds to subkey using XOR
 - passes through 8 S-boxes to get 32-bit result
 - finally permutes using 32-bit perm P



DES Round Structure



1- DES

Input:

bit #	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
bit	1	0	1	1	0	0	1	1	0	0	0	0	1	1	1	1	1	0	0	1	0	0	0	1	0	1	0	0	1	1	0	

Round Key:

bit #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
bit	0	0	0	0	1	0	0	1	1	1	0	0	1	1	0	0	0	1	1	0	0	0	1	0

bit #	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
bit	1	1	0	1	1	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	1	1	1	1

Permutation table

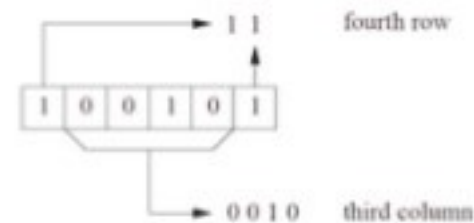
P							
16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

DES Expansion Table

E					
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

Substitution Boxes S

- have eight S-boxes which map 6 to 4 bits
- each S-box is actually 4 little 4 bit boxes
 - outer bits 1 & 6 (**row** bits) select one row of 4
 - inner bits 2-5 (**col** bits) are substituted
 - result is 8 lots of 4 bits, or 32 bits
- row selection depends on both data & key
 - feature known as autoclaving (autokeying)



S_1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

Example of S-box

Substitution Boxes S-Boxes

Box	Row	Column															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S₁																	
	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
	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
S₂																	
	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
	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
S₃																	
	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
	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
S₄																	
	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	13	8	9	4	5	11	12	7	2	14	1
S₅																	
	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S₆																	
	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S₇																	
	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S₈																	
	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

<https://www.youtube.com/watch?v=MGz5gO82qls>

DES Key Schedule

➤ forms subkeys used in each round

- initial permutation of the key (PC1) which selects 56-bits in two 28-bit halves
- 16 stages consisting of:
 - rotating **each half** separately either 1 or 2 places depending on the **key rotation schedule K**
 - selecting 24-bits from each half & permuting them by PC2 for use in round function F

DES Decryption

- decrypt must unwind steps of data computation
- with Feistel design, do encryption steps again using subkeys 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

DES Example

Plaintext (hexadecimal palindrome):	02468aceeca86420
Key:	0f1571c947d9e859
Ciphertext:	da02ce3a89ecac3b

Round	K_i	L_i	R_i
IP		5a005a00	3cf03c0f
1	1e030f03080d2930	3cf03c0f	bad22845
2	0a31293432242318	bad22845	99e9b723
3	23072318201d0c1d	99e9b723	0bae3b9e
4	05261d3824311a20	0bae3b9e	42415649
5	3325340136002c25	42415649	18b3fa41
6	123a2d0d04262a1c	18b3fa41	9616fe23
7	021f120b1c130611	9616fe23	67117cf2
8	1c10372a2832002b	67117cf2	c11bfc09
9	04292a380c341f03	c11bfc09	887fbc6c
10	2703212607280403	887fbc6c	600f7e8b
11	2826390c31261504	600f7e8b	f596506e
12	12071c241a0a0f08	f596506e	738538b8
13	300935393c0d100b	738538b8	c6a62c4e
14	311e09231321182a	c6a62c4e	56b0bd75
15	283d3e0227072528	56b0bd75	75e8fd8f
16	2921080b13143025	75e8fd8f	25896490
IP⁻¹		da02ce3a	89ecac3b

Avalanche Effect

- key desirable property of encryption algo
 - where a small change in either the plaintext or the key should produce a significant change in the ciphertext
- In particular, a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext. This is referred to as the **avalanche effect**. (e.g., a change of **one** input or key bit results in changing approx **half** output bits)
- making attempts to “home-in” by guessing keys impossible
- DES exhibits strong avalanche

Avalanche in DES

Number of bits that differs
between two intermediate
values

4th bit changed
in the
plaintext

Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbcb	32
8	67117cf2c11bfc09 2b2cefbcb99f91153	33

Round		δ
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bcla8d9	29
14	c6a62c4e56b0bd75 4bcla8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP ⁻¹	da02ce3a89ecac3b 057cde97d7683f2a	32

Strength of DES – Key Size

- 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values
- brute force search looks hard
- recent advances have shown 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!
- still must be able to recognize plaintext
- must now consider alternatives to DES

Average Time Required for Exhaustive Key Search

Key Size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 Decryptions/s	Time Required at 10^{13} Decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	$2^{55} \text{ ns} = 1.125 \text{ years}$	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	$2^{127} \text{ ns} = 5.3 \times 10^{21} \text{ years}$	$5.3 \times 10^{17} \text{ years}$
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	$2^{167} \text{ ns} = 5.8 \times 10^{33} \text{ years}$	$5.8 \times 10^{29} \text{ years}$
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	$2^{191} \text{ ns} = 9.8 \times 10^{40} \text{ years}$	$9.8 \times 10^{36} \text{ years}$
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	$2^{255} \text{ ns} = 1.8 \times 10^{60} \text{ years}$	$1.8 \times 10^{56} \text{ years}$
26 characters (permutation)	Monoalphabetic	$2! = 4 \times 10^{26}$	$2 \times 10^{26} \text{ ns} = 6.3 \times 10^9 \text{ years}$	$6.3 \times 10^6 \text{ years}$

Strength of DES – Analytic Attacks

- now have several analytic attacks on DES
- these utilise some deep structure of the cipher
 - by gathering information about encryptions
 - can eventually recover some/all of the sub-key bits
 - if necessary then exhaustively search for the rest
- generally these are statistical attacks
 - **linear cryptanalysis:**
 - based on finding linear approximations to describe the transformations performed in DES.
 - **differential cryptanalysis:**
 - study of how differences in information input can affect the resultant difference at the output.
 - **related key attacks:**
 - discovering where the cipher **exhibits non-random behavior** and exploiting such properties to **recover the secret key**

Strength of DES – Timing Attacks

- attacks actual implementation of cipher
- use knowledge of consequences of implementation to derive information about some/all subkey bits
- specifically use fact that calculations can take varying times depending on the value of the inputs to it
- particularly problematic on smartcards

DES Design Criteria

- as reported by Coppersmith in [COPP94]
- 7 criteria for S-boxes provide for
 - non-linearity
 - resistance to differential cryptanalysis
 - good confusion
- 3 criteria for permutation P provide for
 - increased diffusion

Block Cipher Design

- basic principles still like Feistel's in 1970's
- cryptographic strength of a Feistel cipher derives from three aspects of the design:
 - **number of rounds**
 - more is better, exhaustive search best attack
 - greater the number of rounds, the more difficult it is to perform cryptanalysis even for a relatively weak F
 - **function f :**
 - provides "confusion", is nonlinear, avalanche
 - The more difficult to approximate f by a set of linear equation, the more **nonlinear f** is.
 - Strict Avalanche Criterion (SAC): any output j of an S-box should change with probability $\frac{1}{2}$ when any single input i is inverted for all i, j .
 - Bit Independence Criterion (BIC): output j and k should change independently when any single input i is inverted for all i, j and k .
 - have issues of how S-boxes are selected
 - **key schedule algorithm**
 - generate a subkey for each round
 - complex subkey creation(maximize the difficulty of deducing individual subkeys), key avalanche

Summary

- have considered:
 - block vs stream ciphers
 - Feistel cipher design & structure
 - DES
 - details
 - strength
 - Differential & Linear Cryptanalysis
 - block cipher design principles
- https://www.youtube.com/watch?v=Y61qn_SQL40
- <https://asecuritysite.com/encryption/des>
- <https://www.youtube.com/watch?v=0uA86l22FzQ>
- <https://www.youtube.com/watch?v=3BZRBfhplb0>
- <http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm>