Data Encryption Standard (DES)

1

6.1.1 History

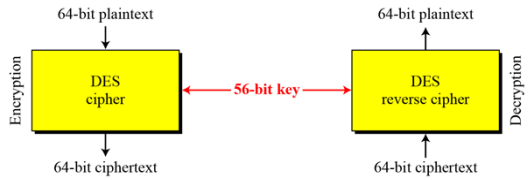
- DES was adopted as a US federal standard for commercial encryption in 1975.
- Feistel Cipher: the fundamental building block of DES designed by IBM.
- Design requirements:
  - must provide high level of security (commercial standard)
  - Security must not depend on secrecy of algorithm (Kerckhoff's principle)
  - Must be easily and economically implemented

6.2

6.1.2 Overview

DES is a block cipher, as shown in Figure 6.1.

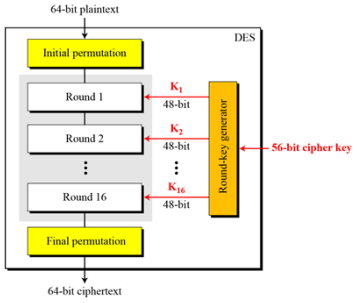
Figure 6.1 *Encryption and decryption with DES*



6.3

DES Structure

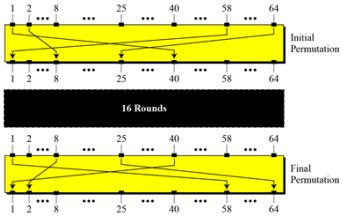
The encryption process is made of two permutations (P-boxes), which we call initial and final permutations, and sixteen Feistel rounds.



6.4

6.2.1 Initial and Final Permutations

Figure 6.3 *Initial and final permutation steps in DES*



6.5

6.2.1 Continue

Table 6.1 *Initial and final permutation tables*

Initial Permutation	*Final Permutation*
58 50 42 34 26 18 10 02	40 08 48 16 56 24 64 32
60 52 44 36 28 20 12 04	39 07 47 15 55 23 63 31
62 54 46 38 30 22 14 06	38 06 46 14 54 22 62 30
64 56 48 40 32 24 16 08	37 05 45 13 53 21 61 29
57 49 41 33 25 17 09 01	36 04 44 12 52 20 60 28
59 51 43 35 27 19 11 03	35 03 43 11 51 19 59 27
61 53 45 37 29 21 13 05	34 02 42 10 50 18 58 26
63 55 47 39 31 23 15 07	33 01 41 09 49 17 57 25

How to read this table?

The 58th bit of input x will be the first bit of output IP(x), the 50th bit of x is the second bit of IP(x), etc.

The initial and final permutations are straight P-boxes that are inverses of each other. They have no cryptography significance in DES.

6.6

6.2.1 Continued

Example 6.1

Find the output of the initial permutation box when the input is given in hexadecimal as:

0x0000 0080 0000 0002

Solution

Only bit 25 and bit 64 are 1s; the other bits are 0s. In the final permutation, bit 25 becomes bit 64 and bit 63 becomes bit 15. The result is

0x0002 0000 0000 0001

6.7

6.2.2 Rounds

Figure 6.4
A round in DES (encryption site)

DES uses 16 rounds. Each round of DES is a Feistel cipher.

–Separate message block into two 32-bit halves, L_i and R_i

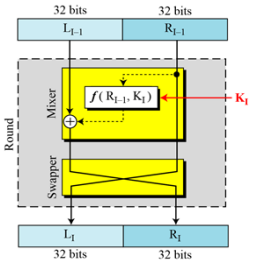
–Introduce **confusion** by using a "complex" nonlinear function f

–f has two inputs: R_i and a 48-bit round key, K_i

–Introduce **diffusion** by "adding" L_i and the output of f

$L_{i+1} = R_i$

$R_{i+1} = L_i \oplus f(R_i, K_{i+1})$



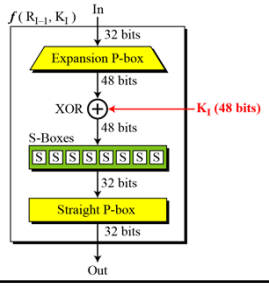
6.8

6.2.2 Continued

DES Function

The heart of DES is the DES function. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.

Figure 6.5
DES function



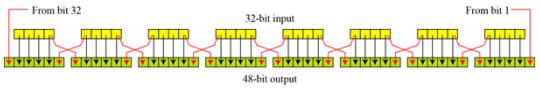
6.9

6.2.2 Continue

Expansion P-box

Since R_{I-1} is a 32-bit input and K_I is a 48-bit key, we first need to expand R_{I-1} to 48 bits.

Figure 6.6 ***Expansion permutation***



6.10

6.2.2 Continue

Although the relationship between the input and output can be defined mathematically, DES uses Table 6.2 to define this P-box.

Table 6.6 ***Expansion P-box table***

32	01	02	03	04	05
04	05	06	07	08	09
08	09	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	31	31	32	01

6.11

6.2.2 Continue

Whitener (XOR)

After the expansion permutation, DES uses the XOR operation on the expanded right section and the round key. Note that both the right section and the key are 48-bits in length. Also note that the round key is used only in this operation.

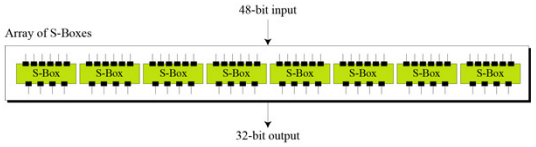
6.12

6.2.2 Continue

S-Boxes

The S-boxes do the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. See Figure 6.7.

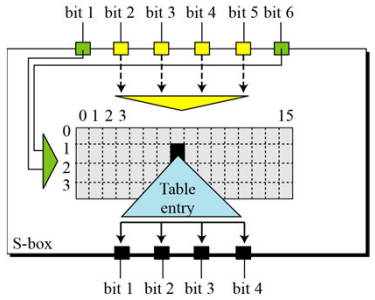
Figure 6.7 *S-boxes*



6.13

6.2.2 Continue

Figure 6.8 *S-box rule*



6.14

6.2.2 Continue

Table 6.3 shows the permutation for S-box 1. For the rest of the boxes see the textbook.

Table 6.3 *S-box 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	10	03	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

6.15

6.2.2 Continued

Example 6.3

The input to S-box 1 is 100011. What is the output?

Solution

If we write the first and the sixth bits together, we get 11 in binary, which is 3 in decimal. The remaining bits are 0001 in binary, which is 1 in decimal. We look for the value in row 3, column 1, in Table 6.3 (S-box 1). The result is 12 in decimal, which in binary is 1100. So the input 100011 yields the output 1100.

6.16

6.2.2 Continue

Straight Permutation

Table 6.11 *Straight permutation table*

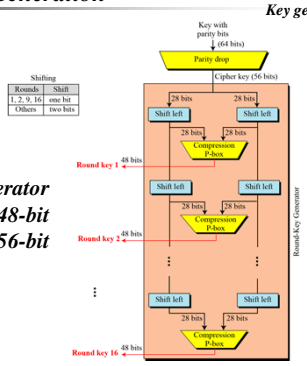
16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

6.17

6.2.3 Key Generation

Figure 6.10 *Key generation*

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key.



6.18

6.2.3 Continued

Table 6.12 *Parity-bit drop table*

57	49	41	33	25	17	09	01
58	50	42	34	26	18	10	02
59	51	43	35	27	19	11	03
60	52	44	36	63	55	47	39
31	23	15	07	62	54	46	38
30	22	14	06	61	53	45	37
29	21	13	05	28	20	12	04

64 → 56

Table 6.13 *Number of bits shifts*

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

6.19

6.2.3 Continued

Table 6.14 *Key-compression table*

14	17	11	24	01	05	03	28
15	06	21	10	23	19	12	04
26	08	16	07	27	20	13	02
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

56 → 48

6.20

6.3.1 Properties

Two desired properties of a block cipher are the avalanche effect and the completeness.

Example 6.7

To check the avalanche effect in DES, let us encrypt two plaintext blocks (with the same key) that differ only in one bit and observe the differences in the number of bits in each round.

Plaintext: 0000000000000000 Key: 22234512987ABB23
Ciphertext: 4789FD476E82A5F1

Plaintext: 0000000000000001 Key: 22234512987ABB23
Ciphertext: 0A4ED5C15A63FEA3

6.21

6.3.1 Continued

Example 6.7 *Continued*

Although the two plaintext blocks differ only in the rightmost bit, the ciphertext blocks differ in 29 bits. This means that changing approximately 1.5 percent of the plaintext creates a change of approximately 45 percent in the ciphertext.

Table 6.17 *Number of bit differences for Example 6.7*

Rounds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit differences	1	6	20	29	30	33	32	29	32	39	33	28	30	31	30	29

6.22

6.3.1 Continued

Completeness effect

Completeness effect means that each bit of the ciphertext needs to depend on many bits on the plaintext.

6.23

6.3.2 Design Criteria

S-Boxes

The design provides confusion and diffusion of bits from each round to the next.

P-Boxes

They provide diffusion of bits.

Number of Rounds

DES uses sixteen rounds of Feistel ciphers. the ciphertext is thoroughly a random function of plaintext and ciphertext.

6.24

6.3.3 DES Weaknesses

During the last few years critics have found some weaknesses in DES.

Weaknesses in Cipher Design

1. Weaknesses in S-boxes

- *Two specifically chosen inputs to an S-box can create same output*

2. Weaknesses in P-boxes

- *initial and final permutations have no security benefits*
- *the first and fourth bits of every 4-bit series are repeated*

3. Weaknesses in Key

- *Weak keys create same 16 round keys*
- *Semi-weak keys create 2 different round keys*
- *Possible weak keys create 4 distinct round keys*
- *Key complement*

6.25

6.3.3 DES Weaknesses

- *There are four weak keys.*
- *After parity drop operation, a key consists either of all 0s, all 1s, or half 0s and half 1s.*
- *Weak keys create same 16 round keys.*

Table 6.18 *Weak keys*

Keys before parities drop (64 bits)	*Actual key (56 bits)*
0101 0101 0101 0101	0000000 0000000
1F1F 1F1F 0E0E 0E0E	0000000 FFFFFFF
E0E0 E0E0 F1F1 F1F1	FFFFFFF 0000000
FEFE FEFE FEFE FEFE	FFFFFFF FFFFFFF

6.26

6.3.3 Continued

Example 6.8

Let us try the first weak key in Table 6.18 to encrypt a block two times. After two encryptions with the same key the original plaintext block is created. Note that we have used the encryption algorithm two times, not one encryption followed by another decryption.

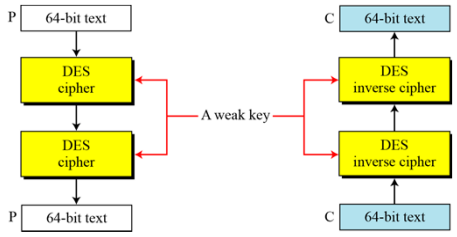
Key: 0x0101010101010101
Plaintext: *0x1234567887654321* Ciphertext: 0x814FE938589154F7

Key: 0x0101010101010101
Plaintext: 0x814FE938589154F7 Ciphertext: *0x1234567887654321*

6.27

6.3.3 Continued

Figure 6.11 ***Double encryption and decryption with a weak key***



$$E_k(E_k(P)) = P$$

6.28

6.3.3 Continued

Semi-weak keys create only 2 different round keys; k1, k2

Table 6.19 *Semi-weak keys*

First key in the pair	*Second key in the pair*
01FE 01FE 01FE 01FE	FE01 FE01 FE01 FE01
1FE0 1FE0 0EF1 0EF1	E01F E01F F10E F10E
01E0 01E1 01F1 01F1	E001 E001 F101 F101
1FFE 1FFE 0EFE 0EFE	FE1F FE1F FE0E FE0E
011F 011F 010E 010E	1F01 1F01 0E01 0E01
E0FE E0FE F1FE F1FE	FEE0 FEE0 FEF1 FEF1

6.29

6.3.3 Continued

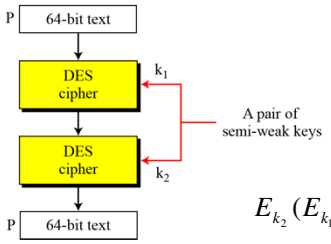
Round key 1	9153E54319BD	6EAC1ABCE642
Round key 2	6EAC1ABCE642	9153E54319BD
Round key 3	6EAC1ABCE642	9153E54319BD
Round key 4	6EAC1ABCE642	9153E54319BD
Round key 5	6EAC1ABCE642	9153E54319BD
Round key 6	6EAC1ABCE642	9153E54319BD
Round key 7	6EAC1ABCE642	9153E54319BD
Round key 8	6EAC1ABCE642	9153E54319BD
Round key 9	9153E54319BD	6EAC1ABCE642
Round key 10	9153E54319BD	6EAC1ABCE642
Round key 11	9153E54319BD	6EAC1ABCE642
Round key 12	9153E54319BD	6EAC1ABCE642
Round key 13	9153E54319BD	6EAC1ABCE642
Round key 14	9153E54319BD	6EAC1ABCE642
Round key 15	9153E54319BD	6EAC1ABCE642
Round key 16	6EAC1ABCE642	9153E54319BD

Semi-week keys create 2 different round keys

6.30

6.3.3 Continued

Figure 6.12 *A pair of semi-weak keys in encryption and decryption*



$$E_{k_2}(E_{k_1}(P)) = P$$

6.31

6.3.3 Continued

Example 6.9

What is the probability of randomly selecting a weak, a semi-weak, or a possible weak key?

Solution

DES has a key domain of 2^{56}. The total number of the above keys are 64 (4 + 12 + 48). The probability of choosing one of these keys is 8.8×10^{-16}, almost impossible.

6.32

6.3.3 Continued

Key Complement In the key domain (2^{56}), definitely half of the keys are complement of the other half. A **key complement** can be made by inverting (changing 0 to 1 or 1 to 0) each bit in the key. Does a key complement simplify the job of the cryptanalysis? It happens that it does. Eve can only half of the possible keys (2^{55}) to perform brute-force attack. This is because

$$C = E(K, P) \rightarrow \overline{C} = E(\overline{K}, \overline{P})$$

In other words, if we encrypt the complement of plaintext with the complement of the key, we get the complement of the ciphertext. Eve does not have to test all 2^{56} possible keys, she can test only half of them and then complement the result.

6.33

6.3.3 Continued

Example 6.10

Let us test the claim about the complement keys. We have used an arbitrary key and plaintext to find the corresponding ciphertext. If we have the key complement and the plaintext, we can obtain the complement of the previous ciphertext (Table 6.20).

Table 6.20 *Results for Example 6.10*

	Original	*Complement*
Key	1234123412341234	EDCBEDCBEDCBEDCB
Plaintext	12345678ABCDEF12	EDCBA987543210ED
Ciphertext	E112BE1DEFC7A367	1EED41E210385C98

6.34

6-4 Multiple DES

Major limitation of DES

- Key length is too short (56 bits).
- Question: So can we apply DES multiple times to increase the strength of encryption?
- Advantage: We could then preserve the existing investment in software and equipment.

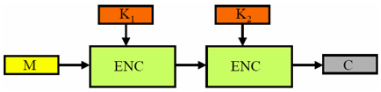
Topics discussed in this section:

6.4.1 Double DES
6.4.4 Triple DES

6.35

Double DES (I)

Apply two iterations of DES with two keys K1 and K2



What if DES has a structure of an algebraic group, such that for each K1 and K2 there is a K3 with the property:

$$E_{K2}(E_{K1}(P)) = E_{K3}(P)$$

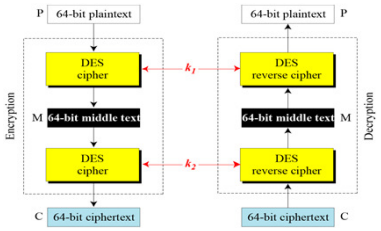
Meet-in-the-Middle Attack

However, using a known-plaintext attack called meet-in-the-middle attack proves (1992) that double DES improves this vulnerability slightly (to 2^{57} tests), but not tremendously (to 2^{112}).

6.36

6.4.1 Continued

Figure 6.14 *Meet-in-the-middle attack for double DES*

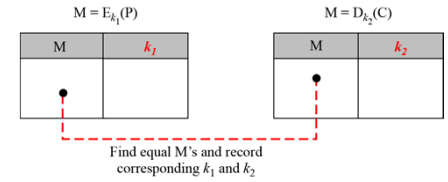


- For given P and C – search only O(2^56) pairs of keys K1 and K2 at the intermediate message M
- Encrypt P under all 2^56 options for K1
  - Denote the results by M1, M2, . . ., Mn
- Decrypt C under all 2^56 options for K2
  - Denote the results by M1', M2', . . ., Mn'

6.37

6.4.1 Continued

Figure 6.15 *Tables for meet-in-the-middle attack*



- Sort the values M1, M2, . . ., Mn
- Sort the values M1', M2', . . ., Mn'

Eve will find at least one match of M with two keys (k1 and k2). If there is only match, Eve found the key. If there is more than one, Eve takes another intercepted plain-text-cipher text pair. This is repeated until she finally finds a unique pair.

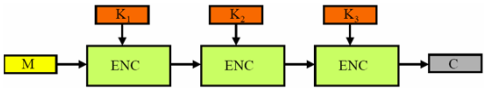
6.38

Triple-DES (I)

EEE Mode:

– DES Encrypt-Encrypt-Encrypt with three keys K_1, K_2, K_3 (168 bits) and strength O (2^{110}) against Meet-in-the-Middle

– Not compatible with regular DES

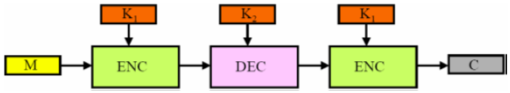


6.39

Triple-DES (II)

EDE Mode:

- DES Encrypt-Decrypt-Encrypt with two keys K_1, & K_2
- Properties:
  - Two keys (112 bits)
  - Strength O(2^{110}) against Meet-in-the-Middle
  - Compatible with regular DES when $K_1 = K_2$



6.40

E-D-E *versus* E-E-E

Why E-D-E?

- Initial and final permutations would cancel each other out with EEE (minor advantage to EDE)
- EDE compatible with single DES if same keys.
- Only 2 different Keys needed with E-D-E

The possibility of known-plaintext attacks on triple DES with two keys has enticed some applications to use triple DES with three keys. Triple DES with three keys is used by many applications such as PGP. New candidates numerous - RC5, IDEA, two-fish, CAST, etc.

6.41

6-5 Security of DES

DES, as the first important block cipher, has gone through much scrutiny.

- The size of the key space, 256, is "too small" to be really secure. Brute-Force Attack: Combining short cipher key in DES with the key complement weakness, it is clear that DES can be broken using 2^55 encryptions.
- Security of DES mainly relies on the nonlinearity of the f (i.e. the S-boxes)

6.42

6-5 Security of DES

- Differential cryptanalysis: Designed S-boxes and 16 rounds aim to make DES specifically resistant to this type of attack.
- Linear cryptanalysis: DES is more vulnerable to linear cryptanalysis than to differential cryptanalysis. S-boxes are not very resistant to linear cryptanalysis. It has been shown that DES can be broken using 243 pairs of known plaintexts. However, from the practical point of view, finding so many pairs is very unlikely.

6.43

Exhaustive Key Search

- In 1993, Michael Wiener presented a pipelined chip which does 16 encryptions simultaneously and tests 5×10^7 DES keys per second.
- Each chip could be built for US$10 using current technology.
- A frame consisting of 5760 chips can be built for $100K.

Machine Unit Cost	Expected Time
$100,000	35 hours
$1,000,000 (10 frames)	3.5 hours
$10,000,000 (100 frames)	21 minutes

- In 1997, cost cut by a factor of 6
- Software version of DES cracking effort can be found at http://www.distributed.net/des/
- Current Record: 22 hrs and 15 mins to break DES by distributed software cracking effort.

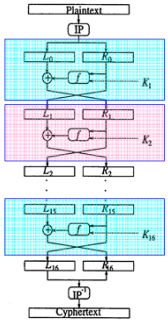
44

Overview of DES

C = DES (K, M)

- Block size = 64 bits
- Key size = 56 bits
- Number of rounds = 16
- IP - Initial Permutation
- IP^{-1} - The inverse of IP
- f - A nonlinear function
- K_i - Round i subkey (48 bits)
- Each Feistel block can be described as

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



45

Advanced Encryption Standard (AES)

46

7-1 INTRODUCTION

The Advanced Encryption Standard (AES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST) in December 2001.

7.47

7.1.2 Criteria

The criteria defined by NIST for selecting AES fall into three areas:

1. Security
2. Cost
3. Implementation.

7.48

Rounds.

AES is a non-Feistel cipher that encrypts and decrypts a data block of 128 bits. It uses 10, 12, or 14 rounds. The key size, which can be 128, 192, or 256 bits, depends on the number of rounds.

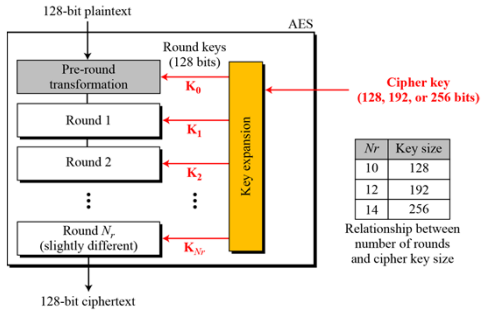
Note

AES has defined three versions, with 10, 12, and 14 rounds.
Each version uses a different cipher key size (128, 192, or 256), but the round keys are always 128 bits.

7.49

Rounds.

Figure 7.1 *General design of AES encryption cipher*



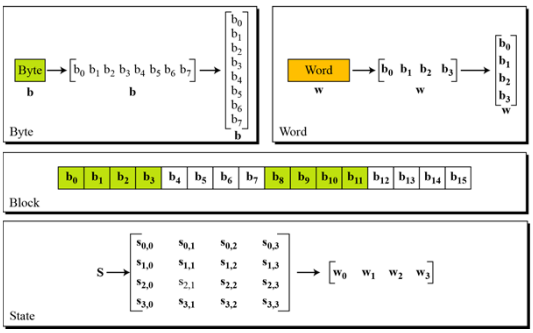
Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

7.50

7.1.4 Data Units.

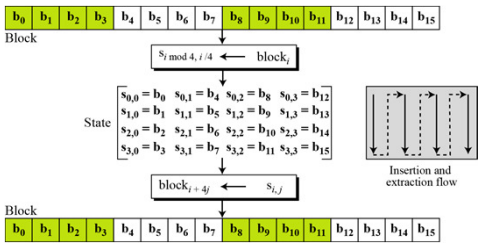
Figure 7.2 *Data units used in AES*



7.51

7.1.4 Continue

Figure 7.3 *Block-to-state and state-to-block transformation*

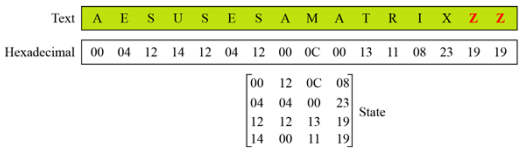


7.52

7.1.4 Continue

Example 7.1 *Continue*

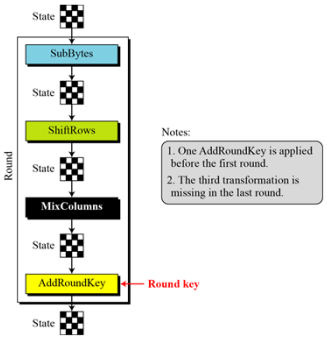
Figure 7.4 *Changing plaintext to state*



7.53

Structure of Each Round

Figure 7.5 *Structure of each round at the encryption site*



7.54

7-2 TRANSFORMATIONS

To provide security, AES uses four types of transformations: substitution, permutation, mixing, and key-adding.

Topics discussed in this section:

7.55

7.2.1 Substitution

AES, like DES, uses substitution. AES uses two invertible transformations.

SubBytes

The first transformation, SubBytes, is used at the encryption site. To substitute a byte, we interpret the byte as two hexadecimal digits.

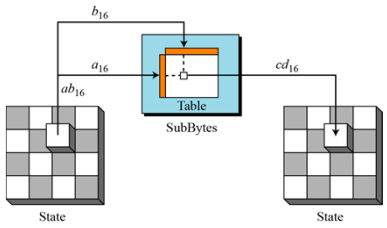
Note

The SubBytes operation involves 16 independent byte-to-byte transformations.

7.56

7.2.1 Substitution

Figure 7.6 *SubBytes transformation*



7.57

7.2.1 Substitution

Table 7.1 *SubBytes transformation table*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8

7.58

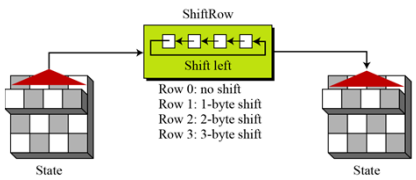
7.2.2 Permutation

Another transformation found in a round is shifting, which permutes the bytes.

ShiftRows

In the encryption, the transformation is called ShiftRows.

Figure 7.9 *ShiftRows transformation*



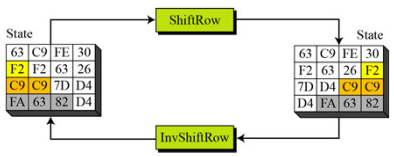
7.59

7.2.2 Permutation

Example 7.4

Figure 7.10 shows how a state is transformed using ShiftRows transformation. The figure also shows that InvShiftRows transformation creates the original state.

Figure 7.10 *ShiftRows transformation in Example 7.4*

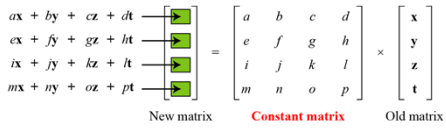


7.60

7.2.3 Mixing

We need an interbyte transformation that changes the bits inside a byte, based on the bits inside the neighboring bytes. We need to mix bytes to provide diffusion at the bit level.

Figure 7.11 *Mixing bytes using matrix multiplication*



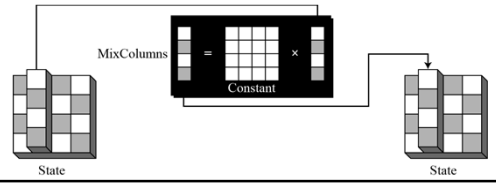
7.61

7.2.3 Mixing

MixColumns

The MixColumns transformation operates at the column level; it transforms each column of the state to a new column.

Figure 7.13 *MixColumns transformation*



7.62

7.2.4 Key Adding

AddRoundKey

AddRoundKey proceeds one column at a time. AddRoundKey adds a round key word with each state column matrix; the operation in AddRoundKey is matrix addition.

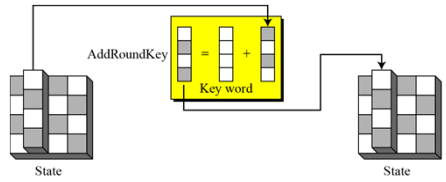
Note

The AddRoundKey transformation is the inverse of itself.

7.63

7.2.4 Key Adding

Figure 7.15 *AddRoundKey transformation*

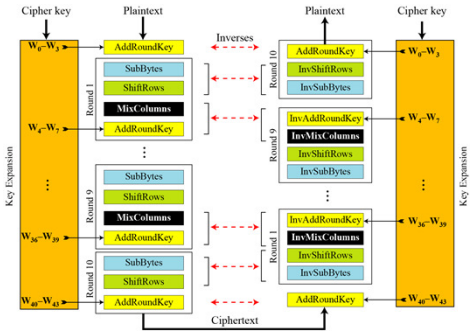


Algorithm 7.4 *Pseudocode for AddRoundKey transformation*

```
AddRoundKey (S)
{
    for (c = 0 to 3)
        s_c ← s_c ⊕ w_round + 4c
}
```

7.64

Cipher and reverse cipher of AES



7.65

AES Security

AES was designed after DES. Most of the known attacks on DES were already tested on AES.

Brute-Force Attack

AES is definitely more secure than DES due to the larger-size key.

Statistical Attacks

Numerous tests have failed to do statistical analysis of the ciphertext.

Differential and Linear Attacks

There are no differential and linear attacks on AES as yet.

7.66

Simplicity and Cost

The algorithms used in AES are so simple that they can be easily implemented using cheap processors and a minimum amount of memory.

7.67

Cryptographic APIs

1. Cryptlib (http://www.cryptlib.com/)
2. OpenSSL (http://www.openssl.org)
3. Crypt++ (http://www.cryptopp.com/)
4. BSAFE (http://www.rsa.com/node.aspx?id=1204)
5. Cryptix (http://www.cryptix.org/)
6. Crypt:: CPAN modules (http://www.cpan.org/, http://search.cpan.org/dist/Crypt-SSLeay/)

7.68

Supported Ciphers

1. Range of MAC algorithms
   Almost all include MD5, SHA-1
2. Range of symmetric algorithms
   Almost all include AES, DES
3. Range of public key algorithms
   Almost all include RSA, Diffie-Hellman, DSA

7.69

Cryptographic APIs

Cryptlib
- easy to use
- free for noncommercial use

OpenSSL
- poorly documented
- open source
- popular

7.70

Cryptographic APIs

Crypto++
- C++ library
- open source

BSAFE
- well documented, Java, C/C++
- most popular commercial library
- Was commercial SDK from RSA
- free from 2009 under RSA Share Project https://community.emc.com/community/edn/rsashare?view=tags&tags=java

7.71

Cryptographic APIs

Cryptix: JCA, JCE
- open source Java library, C# library
- http://www.bouncycastle.org/java.html

Python Cryptographic Toolkit
- open source crypt, hash, rand modules
- http://www.amk.ca/python/code/crypto

Crypt:: CPAN modules for Perl
- well documented
- many different libraries

7.72