cryptography

Pushpa kumar
MENTOR : Sharvari Medhe

June 25, 2024

Contents

1 Introduction **3**

2 Stream Ciphers **4**
2.1 Encryption and Decryption in Stream Ciphers 5
2.2 One time pad 6
2.3 Linear Feedback Shift Registers (LFSR) 6
2.4 **A5/1** 7

3 DES(Data Encryption Standard) **9**
3.1 Encryption 9
3.1.1 Initial and final permutation 11
3.1.2 The f-Function 11
3.1.3 Key Schedule 13
3.2 Decryption 14

4 Advanced Encryption Standard(AES) **17**
4.1 Encryption 18
4.1.1 Byte Substitution Layer 21
4.1.2 Diffusion Layer 22
4.1.3 MixColumn Sublayer 22

5 More About Block Ciphers **24**
5.1 Electronic Codebook Mode (ECB) 24
5.2 Cipher Block Chaining Mode (CBC) 24
5.3 More about brute force attack 25
5.3.1 Double Encryption and Meet-in-the-Middle Attack 26

6 Public-Key Cryptography **27**
6.1 The RSA Cryptosystem 27

7 Public-Key Cryptosystems Based on the Discrete Logarithm Problem **31**
7.1 Diffie–Hellman Key Exchange 31
7.1.1 Cryptographic explanation 31
7.2 The Discrete Logarithm Problem 32
7.2.1 Attacks Against the Discrete Logarithm Problem 32

1 Introduction

The study and application of methods for safe communication when hostile activity is present is known as cryptography.Creating and evaluating techniques that stop the public or other parties from reading private messages is the broad definition of cryptography.
Practical applications of cryptography include electronic commerce, chip-based payment cards, digital currencies, computer passwords, and military communications.

Until recently, cryptography was essentially the same as encryption. It worked by translating legible text (plaintext) into unintelligible gibberish (ciphertext), which could only be read by undoing the process (decryption). Encrypted messages are coded, and the sender only provides the decryption mechanism to the intended recipients in order to prevent adversaries from accessing the communication.
Cryptanalysis is the term used for the study of methods for obtaining the meaning of encrypted information without access to the key normally required to do so; i.e., it is the study of how to "crack" encryption algorithms or their implementations.

Terminology

- plaintext - ordinary information.
- ciphertext - unintelligible form.
- The set of all possible keys is called the key space.
- encryption - is the process of converting plaintext to ciphertext.
- decryption - reverse of encryption.
- A cipher (or cypher) is a pair of algorithms that carry out the encryption and the reversing decryption.
- Operation on cipher is controlled by key,which should be kept secret.

Cryptosystem

1. Symmetric
    - Stream cipher
    - Block cipher
2. asymmetric

In symmetric systems,the same secret key encrypts and decrypts a message.Data manipulation in symmetric systems is significantly faster than in asymmetric systems.

Asymmetric systems use a "public key" to encrypt a message and a related "private key" to decrypt it. The advantage of asymmetric systems is that the public key can be freely published, allowing parties to establish secure communication without having a shared secret key.

Some examples of symmetric ciphers:

Substitution cipher is simply interchanging letters ex(a to f,b to h ....).Its quiet simple to crack by using brute force and mostly with word frequency (as frequency doesn't change in ciphertext).
Shift cipher is shifting all letters by k.(easily cracked with letter frequency analysis).
In cryptography modular arithmetic is used .
note:
let $y = a.x + b \mod 26$,then $x = a^{-1}.(y - b) \mod 26$, here a^{-1} exists only if $gcd(a, 26) = 1$.

2 Stream Ciphers

A symmetric key cipher, a **stream cipher** is an algorithm that processes a single bit, byte, or character at a time from plain text.
It is said to be faster than block ciphers and less hardware complexity.

Stream Ciphers

- Synchronous stream ciphers
- Self-synchronizing stream ciphers
- One time pads

Synchronous stream ciphers own an autonomously produced keystream derived from the ciphertext and planetext.The decoding of the remaining characters remains unchanged if the ciphertext is altered; however, if a character is added or removed, synchronization is broken, making the ciphertext unintelligible.

Self-synchronizing stream ciphers possess a keystream produced by the key and a predetermined number of prior characters in the ciphertext.Here, adding or removing a character can cause some parts of the decryption to fail, but after a certain number of characters—the same number as those used to generate the keystream—the decryption succeeds.

Uses

- A5/1 stream cipher is used in GSM phones.

- RC4 stream cipher has been used in the security system for wireless local area networks (WLANs).

2.1 Encryption and Decryption in Stream Ciphers

Each bit x_i is encrypted by adding a secret key stream bit s_i modulo 2
Modulo 2 addition is same as XOR gate

$$\textbf{Encryption}: y_i = e_{s_i}(x_i) = x_i + s_i mod 2.$$
$$\textbf{Decryption}: x_i = d_{s_i}(yi) = y_i + s_i mod 2$$

Plain text	Xor	Key	Result
1	$\oplus$	1	0
1	$\oplus$	0	1
0	$\oplus$	1	1
0	$\oplus$	0	0

Main thing in stream cipher is generation of keystream.It is suppossed to be random so that attacker cannot guess.
Random numbers

- True Random Number Generators (TRNG).
- Pseudorandom Number Generators (PRNG).
- Cryptographically Secure Pseudorandom Number Generators (CSPRNG).

True Random Number Generators (TRNG) like flipping a coin, rolling a die, etc., are incredibly unpredictable. TRNGs are useful in cryptography for a variety of tasks, including the generation of session keys that are sent back and forth between sender and recipient.

Pseudorandom Number Generators (PRNG) are computed from an initial seed value.We can say next random number is some function of previous numbers.

These can be reproduced again with same seed value .Random numbers which we use in programming languages have some seed initialized and all of those are not cryptographically secure,we can easily crack with limited data.

Cryptographically Secure Pseudorandom Number Generators (CSPRNG) are PRNG with some extra properties.
If n consecutive bits of the key stream is known there is no polynomial time algorithm that can predict the next bit s_{n+1},such type of PRNG is called as CSPRNG.

2.2 One time pad

It is an encryption technique that cannot be cracked.Cipher text is shared through internet or some other means and keystream is shared directly.

Conditions for keystream to be OTP.

- The key must be at least as long as the plaintext.
- The key must be random (uniformly distributed in the set of all possible keys and independent of the plaintext), entirely sampled from a non-algorithmic random number generator.
- The key must never be reused in whole or in part.
- The key must be kept completely secret by the communicating parties and preferably destroy after decrypting.

Drawbacks are key length is same as text length ,for large messages it becomes difficult to transfer key and to store also.

Now lets see how keystreams are generated

2.3 Linear Feedback Shift Registers (LFSR)

linear-feedback shift register (LFSR) is a shift register whose input bit is a linear function of its previous state(taps).

Exclusive-or (XOR) is the most widely used linear function of single bits. As a result, an LFSR is typically a shift register with an input bit driven by the XOR of a few bits from the shift register's total value.

The initial value of the LFSR is called the seed.
An LFSR of length m consists of m stages numbered $0, 1, \ldots, m-1$,each capable of storing one bit, and a clock controlling data exchange. A vector with entries $s_0, \ldots, s_{m-1}$would initialize the shift register. At time i, the following operations are performed

- s_i (the content of stage 0) forms part of the output.
- the content of stage i is shifted to stage $i-1$,for $1 \leq i \leq m-1$.
- the new content (the feedback bit) of the stage $m-1$ would be obtained by xoring a subset of the content of the m stages.

The polynomial $P(x) = 1 + p_1x + p_2x^2 + \cdots + p_{m-1}x^{m-1} + x^m$ is the feedback polynomial.
If $p_i = 1$ that means Xor input of s_i is given back to first tap.

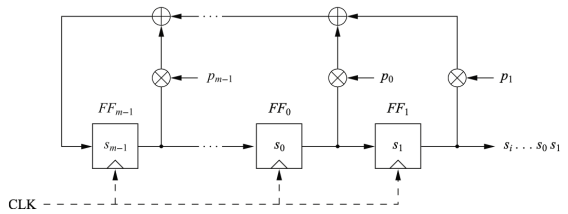


Figure 1: General LFSR with feedback coefficients p_i

$$s_m = s_{m-1}p_{m-1} + \cdots + s_1p_1 + s_0p_0 \mod 2$$

Next output can be calculated as

$$s_{m+1} = s_mp_{m-1} + \cdots + s_2p_1 + s_1p_0 \mod 2$$

In general output can be written as

$$s_{i+m} = \sum_{j=0} m - 1p_j.s_{j+i} \mod 2 \qquad ; s_i, p_j \in 0, 1; \qquad i = 0, 1, 2, \ldots$$

The maximum sequence length generated by an LFSR of degree m is $2^m - 1$,and they are generated only if feedback polynomial is primitive polynomial(see wiki for primitive polynomials).

LSFR is cryptographically not secure just if we know 2m plaintext and cipher text we can easily crack (It reduces to m equations and m unknowns by using above formulae which is easily solvable using matrices and other techniques.)

2.4 A5/1

A5/1 is a stream cipher used to provide over-the-air communication privacy in the GSM cellular telephone standard.

A GSM transmission is set up as a series of short bursts. One burst, with 114 bits available for information, is sent every 4.615 milliseconds in a typical channel and in a single direction. A5/1 is used to generate a 114 bit keystream sequence for each burst, which is XORed with the 114 bits before modulation.A5/1 is initialized with a publicly known 22-bit frame number and a 64-bit key.

Three linear-feedback shift registers (LFSRs) with erratic clocking are the foundation of A5/1. These are the specifications for the three shift registers:

LFSR number	Length in bits	Feedback polynomial	Clocking bit	Tapped bits
1	19	$x^{19} + x^{18} + x^{17} + x^{14} + 1$	8	13,16,17,18
2	22	$x^{22} + x^{21} + 1$	10	20,21
3	23	$x^{23} + x^{22} + x^{21} + x^8 + 1$	10	7,20,21,22

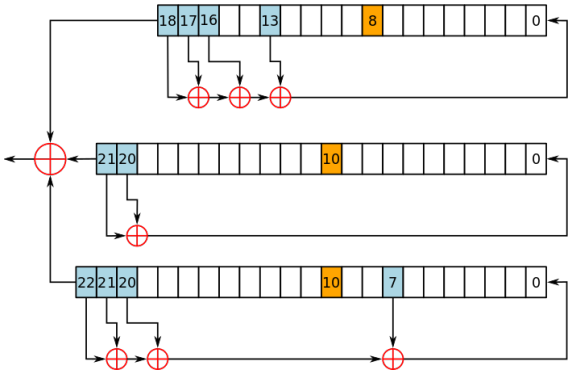


Figure 2: The A5/1 stream cipher uses three LFSRs. A register is clocked if its clocking bit (orange) agrees with the clocking bit of one or both of the other two registers.

Register is clocked if its clocking bit is maximum repeated .
Example: Lets take $c_1 = 0, c_2 = 1, c_3 = 0$,maximum repeated bit is 0 so register 1 and 3 is clocked (note that it c_i changes continuously).

Initialising registers
First all bits are set to 0,then we take a 64 bit key[K].
Here we will perform 64 cycles,in each cycle least significant bit (LSB) is modified with xor operation(in i^{th} cycle with i th key i.e K[i]),and then it is clocked based on above table.
After all cycles we get some random bits in register based on initial key provided.
Now task is to generate keystream
Note that registers are clocked based on clocking bit values ,probability of each register to clock is 3/4.
After generating keystream , plaintext is modified with xor operation with keystream and sends it to reciever or something.
First 114 bits are for upstream and next 114 bits are for downstream and so on...

3 DES(Data Encryption Standard)

A symmetric-key algorithm called **Data Encryption Standard (DES)** is used to encrypt digital data. Though it is too insecure for contemporary applications due to its 56-bit short key length.
Unlike stream cipher 64 bits are encrypted at a time.
There are two important things to know before we start.

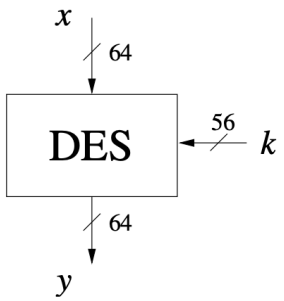


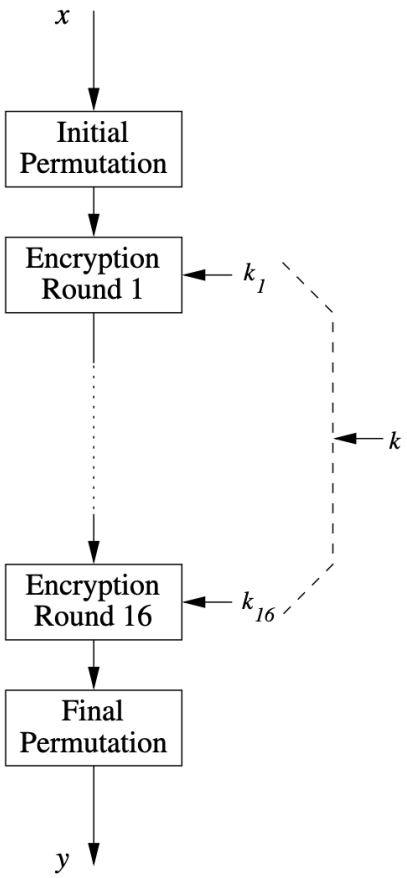
Figure 3: DES block cipher

- **Confusion** is an encryption operation where the relationship between key and ciphertext is obscured.
- **Diffusion** influenceofoneplaintextsymbol is spread over many ciphertext. (Modern block ciphers possess excellent diffusion properties. On a cipher level this means that changing of one bit of plaintext results on average in the change ofhalf the output bits.)

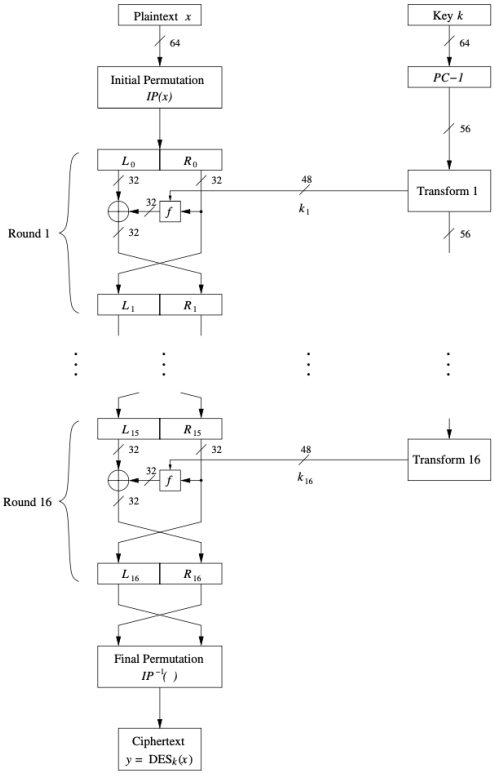
3.1 Encryption

Additionally, DES employs a key to tailor the transformation, ostensibly limiting the ability to decrypt to individuals who possess knowledge of the specific encryption key. Although the algorithm uses only 56 of the 64 bits in the key, the key is actually composed of 64 bits. Eight bits are only used to verify parity before being discarded. Thus, 56 bits is the effective key length.

1. In this encryption is handled in 16 rounds,in every round a different subkey is used and all subkeys k_i are derived from the main key k.
   What is happening inside DES??
   - First initial Permutation.
   - Next plaintext is split into two halves L_0,R_0 . These 32 bits are input to the Feistel network, which consists of 16 rounds.



(a) Iterative structure of DES



(b) The Feistel structure of DES

- The right half R_i is fed into the function f . The output of the f function is **XORed** with the left 32-bit half L_i .Finally, the right and left half are swapped. This process repeats in the next round and can be expressed as: (**NOTE :** only left half encrypts in each round.)

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

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

- After round 16, the 32-bit halves L_{16} and R_{16} are swapped again, and the final permutation IP^{-1} is the last operation of DES.

2. Round key k_i is derived from the main 56-bit key using what is called the key schedule.

3. Once the f -function has been designed securely, the security of a Feistel cipher increases with the number of key bits used and the number of rounds.

3.1.1 Initial and final permutation

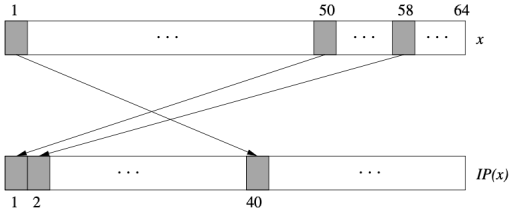
Initial permutation IP and the final permuta- tion IP^{-1} are bitwise permutations. A bitwise permutation can be viewed as simple crosswiring. Interestingly, permutations can be very easily implemented in hardware but are not particularly fast in software.

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

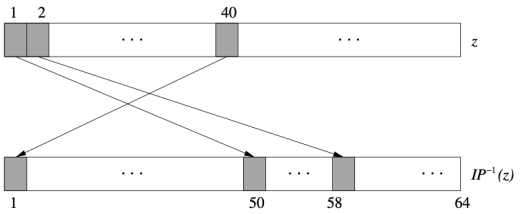
(a) Initial permutation IP

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

(b) Final permutation IP^{-1}



(a) Examples for the bit swaps of the initial permutation



(b) Examples for the bit swaps of the final permutation

How to read the table?? input bit 58 is mapped to output position 1, input bit 50 is mapped to the second output position, and so forth.

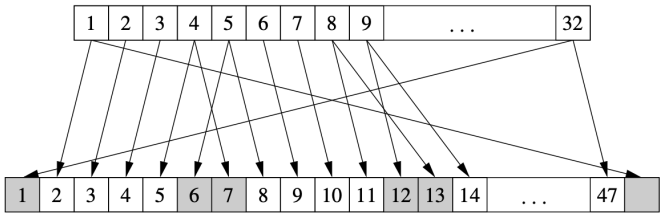
3.1.2 The f-Function

f -function plays a crucial role for the security of DES. In round i it takes the right half R_{i-1}.The output of the f -function is used as an XOR-mask for encrypting the left half input bits L_{i-1}.

1. First, the 32-bit input is ex- panded to 48 bits,this happens in E-box.
2. Exactly 16 of the 32 input bits appear twice in the output.The expansion box increases the diffusion behavior of DES.
3. 48-bit result of the expansion is XORed with the round key ki, and the eight 6-bit blocks are fed into eight different substition boxes, which are often referred to as S-boxes.S-box is a lookup table that maps a 6-bit input to a 4-bit output.

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

(a) Expansion permutation E



(b) Examples for the bit swaps of the expansion function E

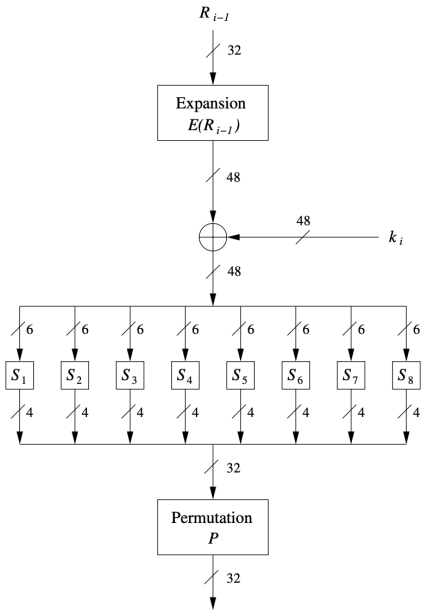


Figure 8: Block diagram of the f -function

each 6-bit input, the most significant bit (MSB) and the least significant bit (LSB). select the table's row, and the four inner bits, the columnEach table entry's integers 0,1,..., 15 represent a 4-bit value's decimal notation.

Ex: The S-box input b = $(100101)_2$ indicates the row $11_2 = 3$ (i.e., fourth row, numbering starts with 002) and the column $0010_2 = 2$ (i.e., the third column). If the input b is fed into S-box 1, the output is $S_1(37 = 100101_2) = 8 = 1000_2$.
finally 32-bit output is permuted bitwise according to the P permutation,

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

(a) S-box S_1

S_2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	01	08	14	06	11	03	04	09	07	02	13	12	00	05	10
1	03	13	04	07	15	02	08	14	12	00	01	10	06	09	11	05
2	00	14	07	11	10	04	13	01	05	08	12	06	09	03	02	15
3	13	08	10	01	03	15	04	02	11	06	07	12	00	05	14	09

(b) S-box S_2

S_3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	00	09	14	06	03	15	05	01	13	12	07	11	04	02	08
1	13	07	00	09	03	04	06	10	02	08	05	14	12	11	15	01
2	13	06	04	09	08	15	03	00	11	01	02	12	05	10	14	07
3	01	10	13	00	06	09	08	07	04	15	14	03	11	05	02	12

(c) S-box S_3

S_4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	07	13	14	03	00	06	09	10	01	02	08	05	11	12	04	15
1	13	08	11	05	06	15	00	03	04	07	02	12	01	10	14	09
2	10	06	09	00	12	11	07	13	15	01	03	14	05	02	08	04
3	03	15	00	06	10	01	13	08	09	04	05	11	12	07	02	14

(d) S-box S_4

S_5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	02	12	04	01	07	10	11	06	08	05	03	15	13	00	14	09
1	14	11	02	12	04	07	13	01	05	00	15	10	03	09	08	06
2	04	02	01	11	10	13	07	08	15	09	12	05	06	03	00	14
3	11	08	12	07	01	14	02	13	06	15	00	09	10	04	05	03

(e) S-box S_5

S_6	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	01	10	15	09	02	06	08	00	13	03	04	14	07	05	11
1	10	15	04	02	07	12	09	05	06	01	13	14	00	11	03	08
2	09	14	15	05	02	08	12	03	07	00	04	10	01	13	11	06
3	04	03	02	12	09	05	15	10	11	14	01	07	06	00	08	13

(f) S-box S_6

S_7	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	04	11	02	14	15	00	08	13	03	12	09	07	05	10	06	01
1	13	00	11	07	04	09	01	10	14	03	05	12	02	15	08	06
2	01	04	11	13	12	03	07	14	10	15	06	08	00	05	09	02
3	06	11	13	08	01	04	10	07	09	05	00	15	14	02	03	12

(g) S-box S_7

S_8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	02	08	04	06	15	11	01	10	09	03	14	05	00	12	07
1	01	15	13	08	10	03	07	04	12	05	06	11	00	14	09	02
2	07	11	04	01	09	12	14	02	00	06	10	13	15	03	05	08
3	02	01	14	07	04	10	08	13	15	12	09	00	03	05	06	11

(h) S-box S_8

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

Figure 10: The permutation P within the f -function

3.1.3 Key Schedule

The key schedule derives 16 round keys k_i, each consisting of 48 bits, from the original 56-bit key.DES is a 56-bit cipher, not a 64-bit one(64-bit key is first reduced to 56 bits by ignoring every eighth bit).
Key is initially permuted with PC-1
The resulting 56-bit key is split into two halves C_0 and D_0.
The two 28-bit halves are cyclically shifted, i.e., rotated, left by one or two bit positions depending on the round i according to the following rules:

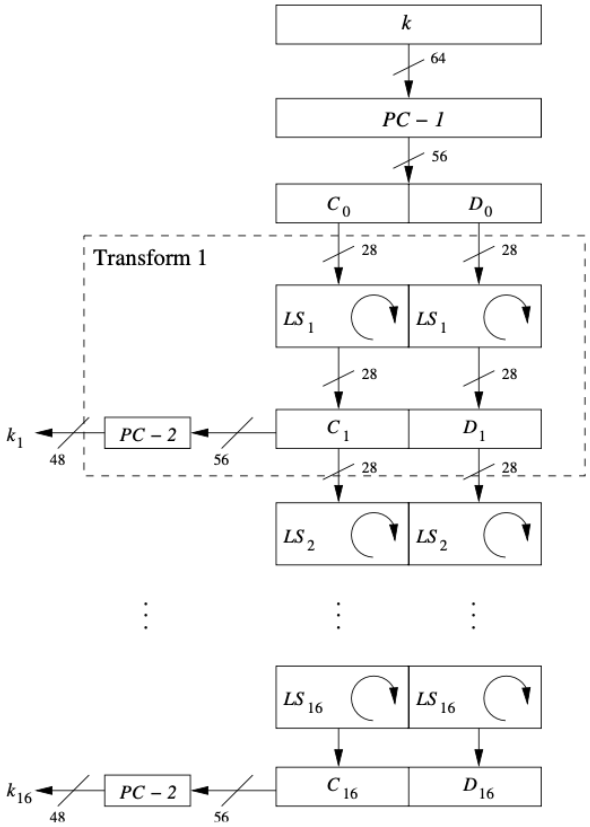
- In rounds i = 1, 2, 9, 16, the two halves are rotated left by one bit.
- In the other rounds where $i \neq 1, 2, 9, 16$, the two halves are rotated left by

two bits.

- $C_0 = C_{16}$ and $D_0 = D_{16}$ as total no of rotations is 28.

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

(a) InitialkeypermutationP ◡ 1



(b) Key schedule for DES encryption

To derive the 48-bit round keys k_i, the two halves, C_i and D_i, are permuted bitwise again with PC-2 (permuted choice 2). PC-2 permutes the 56 input bits coming from C_i and D_i and discards 8 of them. The exact bit-connections of PC-2 are given in the table.

3.2 Decryption

One advantage of DES is that decryption is essentially the same function as encryption. This is because DES is based on a Feistel network. Compared to encryption, only the key schedule is reversed, i.e., in decryption round 1, subkey 16 is needed; in round 2, subkey 15; etc. Thus, when in decryption mode, the key schedule algorithm has to generate the round keys as the sequence $k_{16}, k_{15}, ..., k_1$.
We know $k_0 = k_{16}$
Here key shift is done opposite to encryption.The basic idea is that the decryption function reverses the DES encryption in a round-by-round manner.That means that

$PC-2$							
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

Figure 12: RoundkeypermutationPC-2

decryp- tion round 1 reverses encryption round 16, decryption round 2 reverses encryption round 15, and so on.Let's first look at the initial stage of decryption by looking at Figure 13. Note that the right and left halves are swapped in the last round of DES:

$$(L_0^d, R_0^d) = IP(Y) = IP(IP^{-1}(R_{16}, L_{16})) = (R_{16}, L_{16})$$

$$L_0^d = R_{16}$$

$$R_0^d = L_{16} = R_{15}$$

The first one is easy:

$$L_1^d = R_0^d = L_{16} = R_{15}$$

We now look at how R_1^d is computed:

$$R_1^d = L_0^d \oplus f(R_0^d, k_{16}) = R_{16} \oplus f(L_{16}, k16)$$

$$R_1^d = [L_{15} \oplus f(R_{15}, k16)] \oplus f(R_{15}, k_{16})$$

$$R_1^d = L_{15} \oplus [f(R_{15}, k_{16}) \oplus f(R_{15}, k_{16})] = L_{15}$$

Hence here we shown decryption in round1 reverses 16 th round in encryption.By continuing for rest of the rounds we can get plain text.

Security

Its key length is less, this algorithm is cracked using exhaustive key search and hence its not secure.
However 3DES i.e encrypting 3 times with 3 keys is highly secure.
AES(advanced encryption standard) is highly secure ,which is most commonly used in modern world.

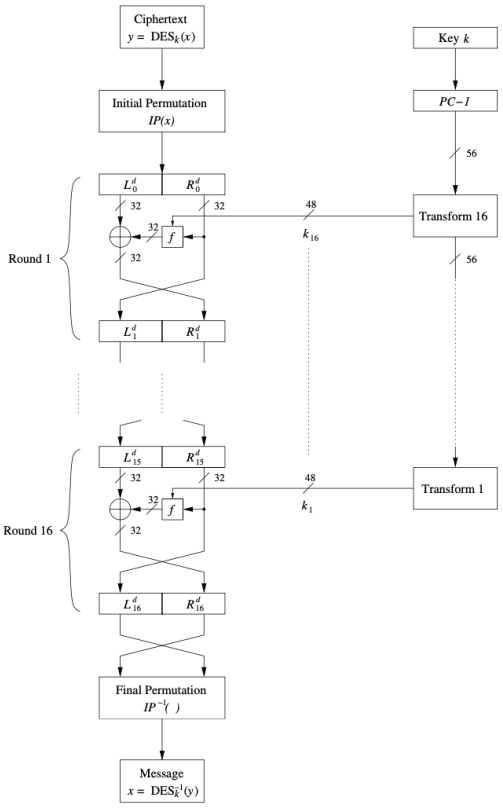


Figure 13: DES decryption

4 Advanced Encryption Standard(AES)

The Advanced Encryption Standard (AES), also known by its original name **Rijndael** is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001.It is the most widely used symmetric cipher today.

AES is a variant of Rijndael, with a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits.

The key size used for an AES cipher specifies the number of transformation rounds that convert the input, called the **plaintext**, into the final output, called the **ciphertext**. The number of rounds are as follows:

- 10 rounds for 128-bit keys.
- 12 rounds for 192-bit keys.
- 14 rounds for 256-bit keys.

Useful Math

In mathematics, a **finite field** or **Galois field** (so-named in honor of Évariste Galois) is a field that contains a finite number of elements. As with any field, a finite field is a set on which the operations of multiplication, addition, subtraction and division are defined and satisfy certain basic rules.
A field with order m only exists if m is a prime power, i.e., $m = p^n$, for some positive integer n and prime integer p. p is called the characteristic of the finite field.
If n is 1 its called as prime fields(integers 0 to p-1) and reast are called as extension fields.
In this cipher we are interested only in extension fields specifically **p=2 and n=8**(it can be seen as 8 bits).
let A(x) denotes bits.

$$A(x) = a^7x^7 + \ + a^1x + a^0, a_i \in GF(2) = 0, 1.$$

Ex: $x^7 + x^4 + 1$ represents 10010001_2
Addition and subtraction:
Let addition and subtration of A(x),B(x)

$$C(x) = A(x) + B(x) = \sum_{i=0}^{m-1} c_i x^i \qquad , c_i = a_i + b_i \mod 2$$

$$C(x) = A(x) - B(x) = \sum_{i=0}^{m-1} c_i x^i \qquad , c_i = a_i - b_i = a_i + b_i \mod 2$$

Multiplication

$$P(x) = \sum_{i=0}^{m} p_i x^i, p_i \in GF(2)$$

Let P(x) be an irreducible polynomial(cannot be factorised).

$$C(x) = A(x) \cdot B(x) \mod P(x)$$

Inversion

$$A^{-1}(x) \cdot A(x) = 1 \mod P(x)$$

calculation of inverse is discussed later.
Its clear that Multiplication and inversion depends on P(x), as there can be multiple irreducible polynomials.AES has a fixed P(x) i.e

$$P(x) = x^8 + x^4 + x^3 + x + 1$$

4.1 Encryption

AES consists of so-called layers. Each layer manipulates all 128 bits of the data path. The data path is also referred to as the state of the algorithm. There are only three different types of layers. Each round, with the exception of the first, consists of all three layers as shown in Figure 14: the plaintext is denoted as x, the ciphertext as y and the number of rounds as nr. Moreover, the last round nr does not make use of the MixColumn transformation, which makes the encryption and decryption scheme symmetric

- **KeyAdditionlayer** A128 bit round key,orsub key,which has been derived from the main key in the key schedule, is XORed to the state.
- **Byte Substitution layer (S-Box)** This introduces confusion to the data.
- Diffusionlayer
  - The **ShiftRows** layer permutes the data on a byte level.
  - The **MixColumn** layer is a matrix operation which combines (mixes) blocks of four bytes.

Internal Structure of AES

The 16-byte input $A_0, A_1, \ldots, A_{15}$ is fed byte-wise into the S-Box. The 16-byte output $B_0, ..., B_{15}$ is permuted byte-wise in the ShiftRows layer and mixed by the MixColumn transformation $c(x)$.Finally, the 128-bit subkey k_i is XORed with the intermediate result.

we first imagine that the state A (i.e., the 128-bit data path) consisting of 16

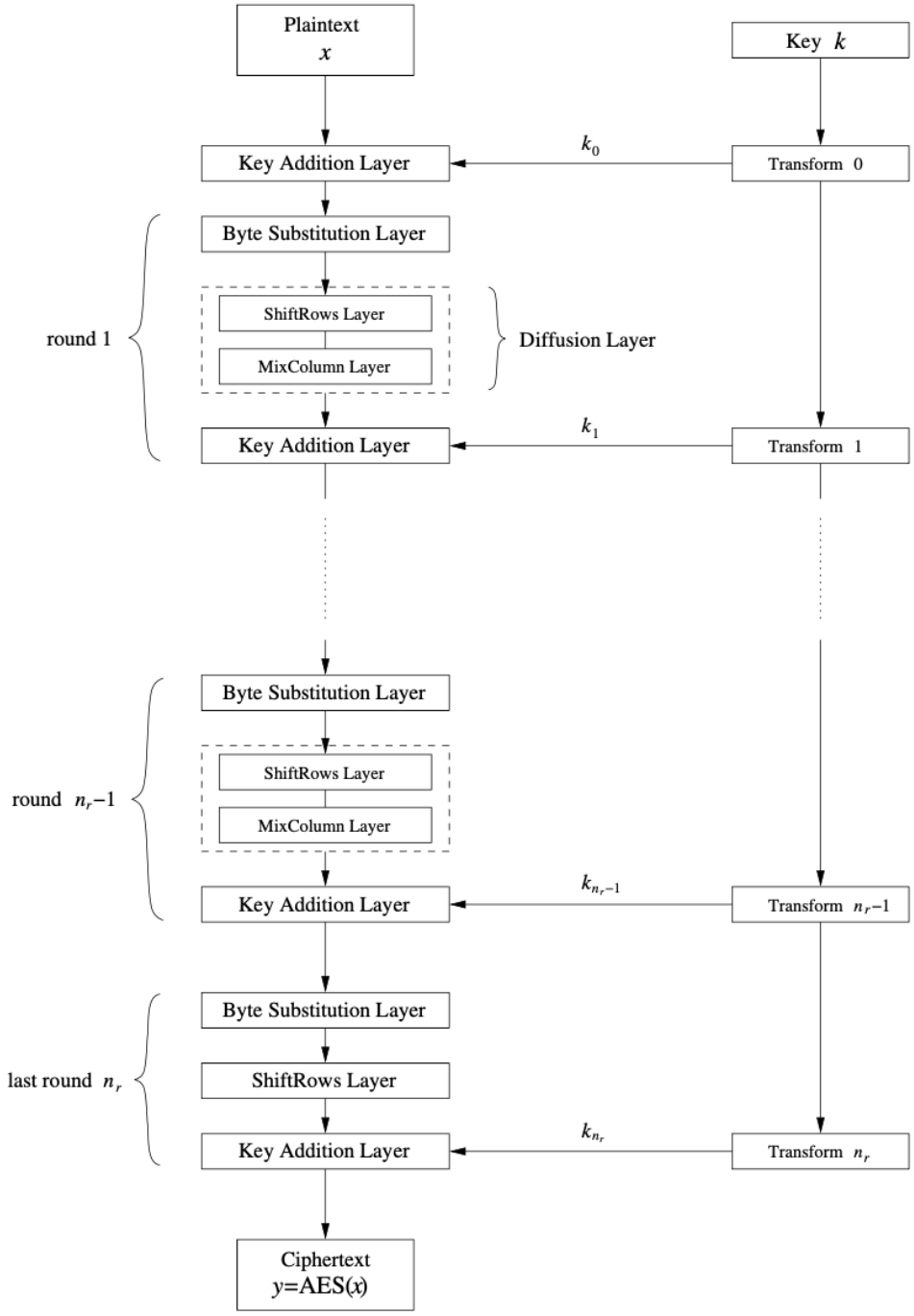


Figure 14: AES encryption block diagram

bytes $A_0, A_1, \ldots, A_{15}$ is arranged in a four-by-four byte matrix,the key bytes are arranged into a matrix with four rows and four (128-bit key), six (192-bit key) or eight (256-bit key) columns.

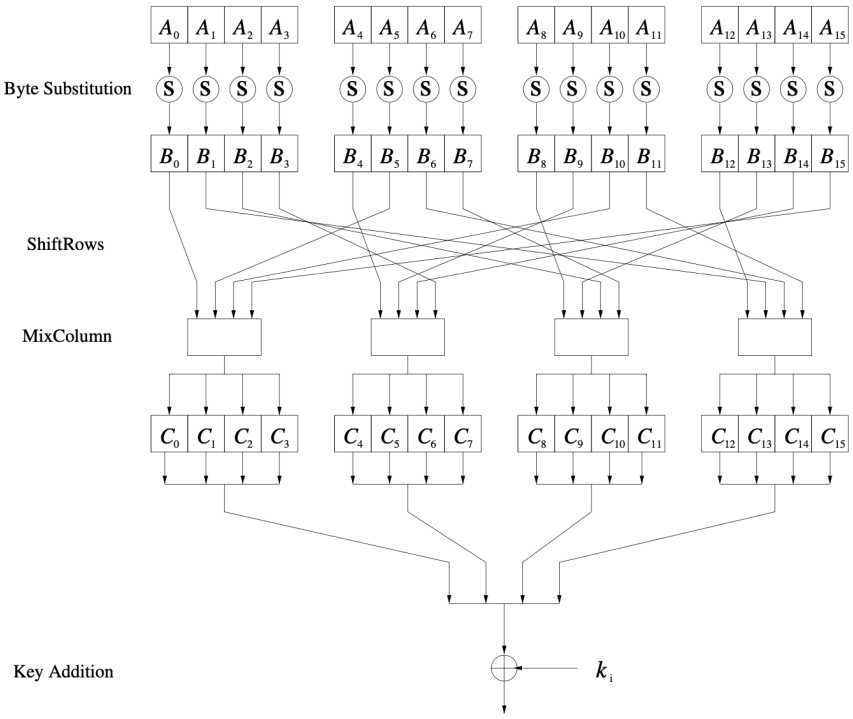


Figure 15: AES round function for rounds 1,2,...,$n_r - 1$

A_0	A_4	A_8	A_{12}
A_1	A_5	A_9	A_{13}
A_2	A_6	A_{10}	A_{14}
A_3	A_7	A_{11}	A_{15}

4.1.1 Byte Substitution Layer

As shown in figure 15;the first layer in each round is the Byte Substitution layer.It can be viewed as 16 parallel S-Boxes,each with 8 input bits and 8 output bits,unlike DES all S-Boxes here are similar.

$$S(A_i) = B_i$$

The S-Box substitution is a bijective mapping which uniquely reverses this substitution in decryption.

									y							
		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	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
x	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Figure 16: AES S-Box: Substitution values in hexadecimal notation for input byte (xy)

How to read the table?? lets see an example.
Let $A_i = (C2)_{hex}$ for B_i see C^{th} row and 2^{nd} column which is 25.

$$S((C2)_{hex}) = (25)_{hex}.$$

$$S(11000010) = (00100101).$$

How table is constructed?? its just the inverse taken with standard P(x) of AES($P(x) = x^8 + x^4 + x^3 + x + 1$) and the result is multiplied with constant bit matrix followed by addition of constant 8-bit vector(if its simply based on mathematical function it can easily be cracked so doing extra operations.)

$$B_i' = A^{-1}$$

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} \equiv \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \begin{pmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \text{ mod 2.}$$

4.1.2 Diffusion Layer

In **AES**, the Diffusion layer consists of two sublayers, the **ShiftRows** transformation and the **MixColumn** transformation(diffusion is the spreading of the influence of individual bits over the entire state).

ShiftRows Sublayer

It just diffuses B_i

B_0	B_4	B_8	B_{12}
B_5	B_9	B_{13}	B_1
B_{10}	B_{14}	B_2	B_6
B_{15}	B_3	B_7	B_{11}

4.1.3 MixColumn Sublayer

The MixColumn step is a linear transformation,it is the major diffusion element in AES.
It diffuses each column in the above matrix and produces new column.
Next columns are computed with corresponding columns in B matrix.

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{pmatrix} \begin{pmatrix} B_0 \\ B_5 \\ B_{10} \\ B_{15} \end{pmatrix}.$$

02 is in hexa decimal system.

$$(01)_{hex} = (00000001)_2 = 1$$
$$(02)_{hex} = (00000010)_2 = x$$
$$(03)_{hex} = (00000011)_2 = x + 1$$

ex: for instance all $B_i = 25$

$$02 \cdot 25 = x(x^5 + x^2 + 1) \mod P(x)$$

similarly you can calculate rest of the values and find C matrix.

5 More About Block Ciphers

The different ways of encryption are called modes of operation.
There are several ways of encrypting long plaintexts with a block cipher.

- Electronic Code Book mode (ECB).
- Cipher Block Chaining mode (CBC).
- Cipher Feedback mode (CFB).

The five modes all work toward the same objective, which is to encrypt data and guarantee Alice and Bob's message's confidentiality. In reality, Bob frequently wants to know if the message is truly coming from Alice in addition to wanting to keep the information private. We refer to this as authentication.

5.1 Electronic Codebook Mode (ECB)

The simplest method of message encryption is the Electronic Code Book (ECB) mode.Assume for the moment that the block cipher encrypts and decrypts blocks with a bit size of b. Messages longer than b bits are divided into blocks of b bits each. The message must be padded to a multiple of b bits before encryption if its length is not a multiple of b bits.
Block synchronization between Alice and Bob, the encryption and decryption parties, is not required (that is, if the recipient doesn't receive all of the bits for some reason, it won't affect the bits that remain).
main problem of the ECB mode is that it encrypts highly deterministically.

Deterministic

Means that identical plaintext blocks result in identical ciphertext blocks, as long as the key does not change.

probabilistic

If same plaintext is encrypted twice we get different ciphertext to do this we introduce initial vector (IV)
Due to deterministic behaviour ECB is susceptible to substitution attacks.(even though we cannot crack AES if we are able to find some pattern we can modify some bits)

5.2 Cipher Block Chaining Mode (CBC)

The encryption of all blocks are "chained together" such that ciphertext y_i depends not only on block x_i but on all previous plaintext blocks as well. Second, the encryption is randomized by using an initialization vector (IV). Ciphertext y_i, which

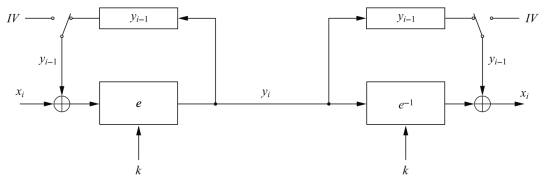


Figure 17: Encryption and decryption in CBC mode

is the result of the encryption of plaintext block x_i, is fed back to the cipher input and XORed with the succeeding plaintext block x_{i+1}. This XOR sum is then encrypted, yielding the next ciphertext y_{i+1}.

CBC encryption nondeterministic. Note that the first ciphertext y_1 depends on plaintext x_1 (and the IV). The second ciphertext depends on the IV, x_1 and x_2. The third ciphertext y_3 depends on the IV and x_1, x_2, x_3, and so on. The last ciphertext is a function of all plaintext blocks and the IV.

Definition 5.1.2 Cipher block chaining mode (CBC)
Let $e()$ be a block cipher of block size b; let x_i and y_i be bit strings of length b; and IV be a nonce of length b.
Encryption (first block): $y_1 = e_k(x_1 \oplus IV)$
Encryption (general block): $y_i = e_k(x_i \oplus y_{i-1}), \quad i \geq 2$
Decryption (first block): $x_1 = e_k^{-1}(y_1) \oplus IV$
Decryption (general block): $x_i = e_k^{-1}(y_i) \oplus y_{i-1}, \quad i \geq 2$

choosing IV it need not be secret,it is preferable to use one number only ones,so counter and time can be good idea.

5.3 More about brute force attack

Because the key length of DES is too short, it is possible to break the encryption multiple times. However, triple encryption is impenetrable and double encryption is not very useful.
Somewhat surprisingly, a brute-force attack can produce false positive results.

For illustration purposes we assume a cipher with a block width of 64 bit and a key size of 80 bit. If we encrypt x_1 under all possible 2^{80} keys, we obtain 2^{80} ciphertexts. However, there exist only 264 different ones, and thus some keys must map x_1

to the same ciphertext. If we run through all keys for a given plaintext–ciphertext pair, we find on average $2^{80}/2^{64} = 2^{16}$

5.3.1 Double Encryption and Meet-in-the-Middle Attack

Let's assume a block cipher with a key length of k bits. For double encryption, a plaintext x is first encrypted with a key k_L, and the resulting ciphertext is encrypted again using a second key k_R.

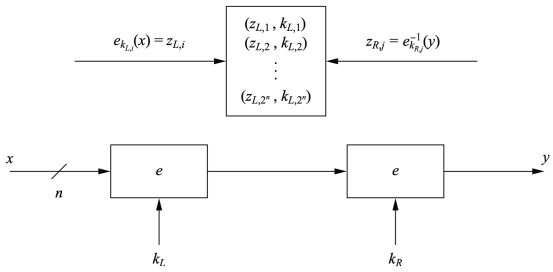


Figure 18: Double encryption and meet-in-the-middle attack

It is impossible to launch a direct brute force attack because the key length (for double DES) will be 112. But we can crack it if we manage to deal two DES separately.

For this to work we need to know some plain text and cipher text which is possible to know easily (it can be header etc).
computation of table with the known text we will try all possible keys i.e 2^{56} and store those result in a table.
We will now decrypt the known cipher text using all possible keys and check the table for math matches (there might be more than one match, so double check with other known texts). At this point, you have both keys.

triple encryption is unbreakable because using similar method as above because we cannot brute force 112 bits.

6 Public-Key Cryptography

It is not required that the key held by the individual decrypting the message—in our case, Alice—be secret. The fact that Bob, the recipient, can only decrypt with a secret key makes this crucial. Bob releases a publicly known encryption key in order to implement such a system. Bob also possesses a secret key that matches and is needed for decryption. Bob's key k is therefore divided into two sections: a public part, k_{pub}, and a private one, k_{pr}.
It is not an alternative for symmetric cipher because of its slow computation(ex: key for AES can be sent through RSA)

some maths

Euclidean Algorithm

$$gcd(r_0, r_1) = gcd(r_0 - r_1, r_1)$$

Extended Euclidean Algorithm

$$gcd(r_0, r_1) = s \cdot r_0 + t \cdot r_1$$

Euler's Phi Function

We consider the ring Z_m, i.e., the set of integers 0,1,...,m-1.
The number of integers in Z_m relatively prime to m is denoted by $\phi(m)$

Theorem 6.3.1 *Let m have the following canonical factorization*

$$m = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_n^{e_n},$$

where the p_i are distinct prime numbers and e_i are positive integers, then

$$\Phi(m) = \prod_{i=1}^{n} (p_i^{e_i} - p_i^{e_i - 1}).$$

6.1 The RSA Cryptosystem

One of the earliest and most popular public-key cryptosystems for safe data transfer is called **RSA (Rivest–Shamir–Adleman)**. The initials "RSA" are derived from the last names of the three people who published the algorithm's 1977 description: Ron Rivest, Adi Shamir, and Leonard Adleman. A similar system was created behind closed doors at Government Communications Headquarters in 1973 **(GCHQ)**, the British signals intelligence agency, by the English mathematician Clifford Cocks. That system was declassified in 1997.

The encryption key in a public-key cryptosystem is made available to the public, while the decryption key is kept confidential. A public key is generated and

made public by an RSA user using two large prime numbers and an additional value. The prime numbers remain unidentified. Anyone can encrypt a message using the public key, but only a person with access to the private key can decrypt it.

Because it is practically impossible to factor the product of two large prime numbers—also known as the "factoring problem"—RSA security is based on this difficulty. The term "RSA problem" refers to cracking RSA encryption. It remains to be seen if it is as challenging as the factoring problem. If a large enough key is used, there are no known ways to break the system.

RSA is a relatively slow algorithm. Because of this, it is not commonly used to directly encrypt user data. More often, RSA is used to transmit shared keys for symmetric-key cryptography, which are then used for bulk encryption-decryption.

RSA Key Generation

Output: public key: $k_{pub} = (n, e)$ and private key: $k_{pr} = (d)$

1. Choose two large primes p and q.
2. Compute $n = p \cdot q$.
3. Compute $\phi(n) = (p-1)(q-1)$.
4. Select the public exponent $e \in 1, 2, ..., \phi(n)$–1 such that

$$gcd(e, \phi(n)) = 1.$$

5. Compute the private key d such that

$$d \cdot e \equiv \text{mod } \phi(n)$$

RSA Encryption

Given the public key $(n, e) = k_{pub}$ and the plaintext x, the encryption function is:

$$y = e_{k_{pub}}(x) = x^e \mod n$$

where $x, y \in \mathbb{Z}_n$

RSA Decryption

Given the private key $d = k_{pr}$ and the ciphertext y, the decryption function is:

$$x = d_{k_{pr}}(y) = y^d \mod n$$

where $x, y \in Z_n$.

Here is an example of RSA encryption and decryption:

1. Choose two distinct prime numbers, such as $p = 61$ and $q = 53$.
2. Compute $n = pq$ giving $n = 61 \times 53 = 3233$.
3. Compute the Carmichael's totient function of the product as $\lambda(n) = lcm(p-1, q-1)$ giving $\lambda(3233) = lcm(60, 52) = 780$.
4. Choose any number $2 < e < 780$ that is coprime to 780. Choosing a prime number for e leaves us only to check that e is not a divisor of 780. Let $e = 17$.
5. Compute d, the modular multiplicative inverse of $e \pmod{\lambda(n)}$, yielding $d = 413$, as $1 = (17 \times 413) \bmod 780$.

The public key is $(n = 3233, e = 17)$. For a padded plaintext message m, the encryption function is $c(m) = m^e \bmod n = m^{17} \bmod 3233$.

The private key is $(n = 3233, d = 413)$. For an encrypted ciphertext c, the decryption function is $m(c) = c^d \bmod n = c^{413} \bmod 3233$.

For instance, in order to encrypt $m = 65$, one calculates

$c = 6517 \mod 3233 = 2790$.

To decrypt $c = 2790$, one calculates $m = 2790^{413} \mod 3233 = 65$.

Both of these calculations can be computed efficiently using the square-and-multiply algorithm for modular exponentiation. In real-life situations the primes selected would be much larger; in our example it would be trivial to factor $n = 3233$ (obtained from the freely available public key) back to the primes p and q. e, also from the public key, is then inverted to get d, thus acquiring the private key. Practical implementations use the Chinese remainder theorem to speed up the calculation using modulus of factors ($\mod pq$ using $\mod p$ and $\mod q$).

The values d_p, d_q and q_{inv}, which are part of the private key are computed as follows:

$d_p = d \mod (p-1) = 413 \mod (61-1) = 53$, $d_q = d \mod (q-1) = 413 \mod (53-1) = 49$, $q_{inv} = q^{-1} \mod p = 53^{-1} \mod 61 = 38 \Rightarrow (q_{inv} \times q) \mod p =$

$38 \times 53 \mod 61 = 1$.

Here is how d_p, d_q and q_{inv} are used for efficient decryption (encryption is efficient by choice of a suitable d and e pair):

$m_1 = c^{d_p} \mod p = 2790^{53} \mod 61 = 4$, $m_2 = c^{d_q} \mod q = 2790^{49} \mod 53 = 12$, $h = (q_{inv} \times (m_1 - m_2)) \mod p = (38 \times -8) \mod 61 = 1$, $m = m_2 + h \times q = 12 + 1 \times 53 = 65$.

7 Public-Key Cryptosystems Based on the Discrete Logarithm Problem

RSA is based on the hardness of factoring large integers.
The security of many cryptographic schemes relies on the computational in- tractability of finding solutions to the Discrete Logarithm Problem (DLP)

7.1 Diffie–Hellman Key Exchange

The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel.This key can then be used to encrypt subsequent communications using a symmetric-key cipher.

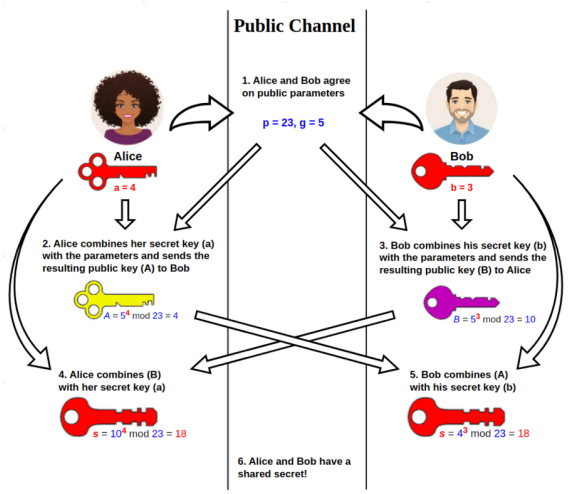


Figure 19: With Diffie-Hellman key exchange, two parties arrive at a common secret key, without passing the common secret key across the public channel.

7.1.1 Cryptographic explanation

The simplest and the original implementation,[2] later formalized as Finite Field Diffie-Hellman in RFC 7919,[9] of the protocol uses the multiplicative group of integers modulo p, where p is prime, and g is a primitive root modulo p. These two values are chosen in this way to ensure that the resulting shared secret can take on any value from 1 to p–1. Here is an example of the protocol, with non-secret values in blue, and secret values in red.

1. Alice and Bob publicly agree to use a modulus $p = 23$ and base $g = 5$ (which is a primitive root modulo 23).

2. Alice chooses a secret integer $a = 4$, then sends Bob $A = g^a \mod p$
   - $A = 5^4 \mod 23 = 4$ (in this example both A and a have the same value 4, but this is usually not the case)
3. Bob chooses a secret integer b = 3, then sends Alice $B = g^b \mod p$
   - $B = 5^3 mod 23 = 10$
4. Alice computes $s = B^a \mod p$
   - $s = 10^4 \mod 23 = 18$
5. Bob computes $s = A^b \mod p$
   - $s = 4^3 \mod 23 = 18$
6. Alice and Bob now share a secret (the number 18).

Only $a$and b are kept secret. All the other values – p, g, $g^a \mod p$, and $g^b \mod p$ – are sent in the clear. The strength of the scheme comes from the fact that $g^{ab} \mod p = g^{ba} \mod p$ take extremely long times to compute by any known algorithm just from the knowledge of $p, g, g^a \mod p$, and $g^b \mod p$. Such a function that is easy to compute but hard to invert is called a one-way function. Once Alice and Bob compute the shared secret they can use it as an encryption key, known only to them, for sending messages across the same open communications channel.

7.2 The Discrete Logarithm Problem

Let G be any group. Denote its group operation by multiplication and its identity element by 1. Let b be any element of G. For any positive integer k, the expression b^k denotes the product of b with itself k times:

$$b^k = \underbrace{b \cdot b \cdots b}_{k \text{ factors}}$$

Similarly, let b^{-k} denote the product of b^{-1} with itself k times. For $k = 0$, the kth power is the identity: $b^0 = 1$.

Let a also be an element of G. An integer k that solves the equation $b^k = a$ is termed a **discrete logarithm** (or simply **logarithm**, in this context) of a to the base b. One writes $k = \log_b a$.

7.2.1 Attacks Against the Discrete Logarithm Problem

- Brute force

- Baby-step giant-step

  Input: A cyclic group G of order n, having a generator α and an element β. Output: A value x satisfying $\alpha^x = \beta$.

  - $m \leftarrow \lceil \sqrt{n} \rceil$
  - For all j where $0 \leq j < m$:
  - Compute α^j and store the pair (j, α^j) in a table. (See § In practice)
  - Compute α^{-m}.
  - $\gamma \leftarrow \beta$. (set $\gamma = \beta$)
  - For all i where $0 \leq i < m$:
    - Check to see if γ is the second component (α^j) of any pair in the table.
    - If so, return $im + j$.
    - If not, $\gamma \leftarrow \gamma \cdot \alpha^{-m}$.

References

- **Understanding Cryptography** by Christof Paar and Jan Pelzl
- You tube letcures by christof paar
- wikipedia of respective topics.

Plan Of Action

Week 1

1. Introduction to cryptography.
2. Stream ciphers.

Week 2

1. Data encryption standard(encryption and key).
2. Introduction to Galois field for the AES.
3. Advanced encryption standards.

Week 3

1. Modes of operation for block ciphers.
2. Multiple encryption and brute force attacks.
3. The RSA crypto system

Week 4

1. Diffie-Hellman key exchange and the discrete log problem.
2. The generalised discrete log problem and the security of diffie-hellman.