[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy — Winter 2022-2023
Scribed by: Akshay (202051018) — Lecture (Week 1)

1 Introduction

Cryptography: The part where we develop algorithms to get security (Designing the algorithm).

Cryptanalysis: We try to break the security of the designed algorithm.

$$\text{Cryptology} = \text{Cryptography} + \text{Cryptanalysis}$$

NIST is an organisation which standardizes cryptographic algorithms.

An Easiest Example:
Suppose I want to keep the pins of my ATM cards. One super easy way is to write the pin on the card itself but this type of implementation is highly insecure. If somebody steals my card then he/she will easily get the pin. To keep the pin secure what I can do is choose a specific number (key) add it to each pin number and write the resultant number (cipher) on the corresponding ATM card.

$$\begin{gathered}\text{ATM 1} \rightarrow \text{PIN 1} + \text{X} = \text{Y1} \\ \text{ATM 2} \rightarrow \text{PIN 2} + \text{X} = \text{Y2} \\ . \\ . \\ . \\ \text{ATM n} \rightarrow \text{PIN n} + \text{X} = \text{Yn}\end{gathered}$$

Now, to retrieve my pin I will just subtract X from the number written on the card. Keeping the X secret no none will be able to steal my pin even if he/she knows that I performed addition. Nomenclature:

$$\begin{gathered}\text{PIN : Plain text (P)} \\ \text{X : Secret key (K)} \\ \text{Y : Cipher text (C)}\end{gathered}$$

Encryption: It is a function which takes plain text and key as argument and returns cipher text.

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

Decryption: It is a function which takes cipher text and key as argument and returns meaningful plain text.

$$\text{D(C,K)} = \text{P}$$

2 Classification of Cryptography Algorithms

2.1 Symmetric Key Cryptography

Under this encryption and decryption keys are kept **same** and **secret**.

2.2 Asymmetric Key Cryptography

- Encryption and decryption keys are kept different but not random. They are related somehow.
- Some keys are kept public while others are kept secret.

3 Cryptography provides following security services:

1. Confidentiality (secrecy): Hiding the message from undesirable person. To achieve confidentiality encryption and decryption algorithms are used.
2. Integrity: Refers to the assurance that a message has not been tampered with or altered during transmission.
3. Authentication: Process of verifying the identity of a person or device. This is typically done through the use of a username and password, or by using a digital certificate.
4. Non-repudiation: A mechanism to prove that the sender really sent this message.

4 CAESAR Cipher

This cipher is named after Julius Caesar. It refers on shifting the letters of a message by an agreed number.

agreed number = 3

We map the English alphabets with integers. A to 0, B to 1, ... Z to 25.
if 'x' is a plain text:
E(x,3) = $(x + 3)\%26$
D(c,3) = (c - 3 + 26)%26 [26 is added to avoid negative output]

5 Revisiting Maths

5.1 Function

f:A $\rightarrow$ B, it is a relation between the elements of A and B with the property that if a,b $\in$ A and a = b then f(a) = f(b).

5.2 One to One Function

f(a) = f(b) $\implies$ a = b

5.3 Onto Function

f : A $\rightarrow$ B then $\forall$ b $\in$ B $\exists$ a $\in$ A such that f(a) = b.

5.4 Bijective Function

f : A $\rightarrow$ is bijective function iff f is one to one and onto.

5.5 Permutation

Let π be a permutation on a set S then π : S $\rightarrow$ S is a bijection from S to S.
E.g:
$\pi : \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 1 & 4 \end{pmatrix}$

5.6 One way Function

f : X $\rightarrow$ Y is called a one way f given x $\in$ X it is easy to compute f(x) but given it is hard to find x.
E.g.
Given that p and q are large prime numbers, computing N = p*q is easy.
Given N, finding p and q (large prime numbers) s.t. N = p*q is hard.

5.7 Substitution Box

S : A $\rightarrow$ B with $|B| \leq |A|$
E.g.:
S : $1, 2, 3, 4 \rightarrow 1, 2, 3$
S(1) = 1, S(2) = 3, S(3) = 2, S(4) = 1

6 Transposition Cipher

$$M = m_1 m_2 m_3 .....m_t \text{ (plain text)}$$

e : **permutation** on t elements $\rightarrow$ Secret key

6.1 Encryption

Cipher text, $C = m_{e(1)} m_{e(2)} .....m_{e(t)} = c_1 c_2 .....c_t$
If e(1) = 5 then m_5 is placed at position 1.

6.2 Decryption

$M = c_{e^{-1}(1)} c_{e^{-1}(2)} ......c_{e^{-1}(t)}$

E.g.:
Plaintext : CAESAR = $m_1 m_2 .....m_6$

Secret key, e : $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 4 & 1 & 3 & 5 & 2 \end{pmatrix}_{encryption}$

Cipher text : RSCEAA = $c_1 c_2 c_3 ....c_6$

$d = e^{-1} : \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 6 & 4 & 2 & 5 & 1 \end{pmatrix}_{decryption}$

7 Substitution Cipher

$M = m_1 m_2 .....m_t$
$S = \{A, B, C, ...Z\},\ m_i \in S$
e : Substitution from S to S, e $\rightarrow$ Secret key.
Encryption: $C = e(m_1)e(m_2).....e(m_t)$

E.g.:
e(A) = Z, e(B) = D, .. e(C) = A
plain text : ABC
cipher text : ZDA

8 Affine Cipher

A is mapped to 0, B to 1, ... , Z to 25.
S : set of alphabets
$S \longrightarrow \mathbb{Z}_{26}$
K = secret key = (a, b) $\in \mathbb{Z}_{26} \times \mathbb{Z}_{26}$

Encryption : e(x, K) = (a*x + b)mod26 = C

Decryption : d(C, K) = $((c - b) * a^{-1}) mod 26$

Inverse of a number 'a' under modulo 26 exists only when gcd(a, 26) = 1

8.1 Proving that modular inverse of x only exists when gcd(x, m)=1

If y is the inverse of x under moudulo m then:
xy = 1 mod m
$\implies$ m divides (xy-1)
$\implies$ xy - 1 = t*m
$\implies$ 1 = x*y + m*t
Now, using Extended Euclidean Algorithm we can say that gcd(x, m) = 1
[$gcd(a, b) = a * x + b * y$]

8.2 Basic Euclidean Algorithm to find gcd(a, b)

- while one of the number does not become 0
  - subtract the multiple of smaller number from the bigger number. Resultant will the remainder. We can do so by taking bigger number modulo smaller number
  - Replace bigger number with this remainder

Reasoning:
a = gx (say) ; b= gy (say) , where g is gcd of a and b then gcd(x,y) = 1 because after taking out the highest common factor from two numbers (which is g here) , nothing common lefts in them.
now if $b < a$, a-b = g(x-y) $\implies$ gcd(a-b,b) = g = gcd(a,b)
If we keep on doing this then both number a, b will become equal to gcd (g) at some point.

When we are subtracting b from a till $a > b$ ultimately we are removing multiple of b from a and leaving the remainder. For $a > b$, let's say a = n*b + r, where n is the number of times b is present in a and r is the rest part. So, instead of removing b one by one from a till $a > b$, we can optimize this process by directly reaching r via the modulus operator.

8.3 For how many numbers does inverse exist under modulo m?

Answer is **Euler Totient Function ($\phi(m)$)**
$\phi(m)$ = Number of positive integers less than 'm' that are relatively prime to m (i.e., gcd(integer, m) = 1).

	Criteria of 'm'	**Formula**
$\phi(m)$	'm' is prime	$\phi(m) = m - 1$
	m=p*q p and q are primes	$\phi(m) = (p-1) * (q-1)$
	m=a*b Either a or b is composite. Both a and b are composite	$\phi(m) = m * (1 - \frac{1}{p_1})(1 - \frac{1}{p_2})$..... where $p_1, p_2, ...$ are distinct primes.

26 = 13*2

Thus we have 12 meaningful choices for 'a' in the Affine cipher. Still 26 choices for 'b'

∴
Total valid keys possible = 12*26 = 312

9 Playfair Cipher

Plain text : HIDE
Secret Key : PLAYFAIR EXAMPLE

Key Schedule :

P	**L**	**A**	**Y**	**F**
I/J	**R**	**E**	**X**	**M**
B	C	D	G	H
K	N	O	Q	S
T	U	V	W	Z

- Create a 5x5 table
- First write the secret key, drop any duplicate letters
- Fill the rest of the table with the remaining letters of the alphabet.
- I and J are put in the same space since we have 26 letters

We pick two letters at a time from our plain text and apply following rules :

1. If both letters are the same (or only one letter is left), add X after the first letter

2. If the letters appear on the same row of your table, replace them with the letters to their immediate right respectively (wrapping around to the left side of the row if a letter in the original pair was on the right side of the row.

3. If the letters appear on the same column of your table, replace them with the letters immediately below respectively (wrapping around to the top side of the column if a letter in the original pair was on the bottom side of the column.

4. If the letters are not on the same row or column, replace them with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair. The order is important - the first letter of the encrypted is the one that lies on the same row as the first letter of the plain-text pair.

Plain text :	HI	DE
Cipher text :	BM	OD

[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy — Winter 2022-2023
Scribed by: Akshay (202051018) — Lecture (Week 2)

1 PlayFair Cipher contd..

We pick two letters at a time from our plain text and apply following rules :

1. If both letters are the same (or only one letter is left), add X after the first letter
   (a) Playfair cipher has no rule when the same letters are X. So, it avoid the infinite loop of adding X, we mutually decide any other letter apart from X to b/w two Xs.
   Note: Playfair cipher didn't mention. We can do this modification to use playfair cipher for plaintexts having XX pair(s).
2. If the letters appear on the same row of your table, replace them with the letters to their immediate right respectively (wrapping around to the left side of the row if a letter in the original pair was on the right side of the row.
3. If the letters appear on the same column of your table, replace them with the letters immediately below respectively (wrapping around to the top side of the column if a letter in the original pair was on the bottom side of the column.
4. If the letters are not on the same row or column, replace them with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair. The order is important - the first letter of the encrypted is the one that lies on the same row as the first letter of the plain-text pair.

For decryption we can apply these rules in opposite manner.

2 Hill Cipher

As usual map the alphabets with integers from 0 to 25.
Secret Key: $A = (a_{ij})_{n\times n}$ (invertible matrix)
$a_{ij} \in \mathbb{Z}_{26}$
Plaintext: $M = m_1 m_2 .....m_n$
n = dimension of key and n alphabets are encrypted at a time.

2.1 Encryption

E(M,A) = A*M = $[c_1\ c_2....c_n]^T$ = C
where M is plaintext vector, A is secret key (matrix) and C is cipher vector.

2.2 Decryption

D(C,A) = $A^{-1}C = M$

Note: All the arithmetic operation are done under modulo 26

2.3 Cryptanalysis

It follows S-box, $S : \{A, B, ...Z\} \rightarrow \{A, B, C....Z\}$

$$\text{C} = \text{S(P)}, \ \text{C is known}$$

Total possible S-boxes: $26^{26} \approx 2^{122}$
Brute force attack / exhaustive search

Kerchoff's rule: Design has to be public.

---

3 Shannon's Notion of Perfect Secrecy

If your crypto system is not giving any extra advantage in guessing (probability) the message from cipher text then you system is secure.
E : Encryption algo M : Message
C : Cipher text E(M) = C (going via public channel)
E will be providing perfect secrecy iff the ciphertext does not reveal any information regarding the plaintext/message.

$$Pr[M = m|C = c] = Pr[M = m]$$
$$Pr[message|ciphertext] = Pr[message]$$

4 Symmetric Key Ciphers (classification)

4.1 Block Ciphers

Given any message, it will be divided into multiple blocks of fixed size.
$M = M_0||M_1||.....M_n$ (divided into n+1 blocks)
Encryption is done block-wise.

$$C = Enc(M_0, K)||Enc(M_1, K)||.....||Enc(M_n, K) = C_0||C_1||.....C_n$$

4.2 Stream Ciphers

In this we do the encryption bit-wise.
$M = m_0m_1.....m_l$, where $m_i \in \{0, 1\}$

$$C = (m_0 \oplus Z_0, m_1 \oplus Z_1, ..., m_l \oplus Z_l)$$

To get the message back we have to xor again with same Z :

$$M = (c_0 \oplus Z_0, c_1 \oplus Z_1, ..., c_l \oplus Z_l)$$

$F(K) \rightarrow Z_i \in \{0, 1\}$ (some function to get Zs from secret key K)

5 Product Ciphers

A product cipher combines two or more transformations (functions) in a manner intending that the resulting cipher is more secure than the individual transformation.

5.1 Substitution Permutation Network (SPN)

It is a product cipher based in substitution box and permutation box.

$S : \{0,1\}^n \to \{0,1\}^m$
$P : \{0, ..., m_{r-1}\} \to \{0, \ ...., \ m_{r-1}\}$

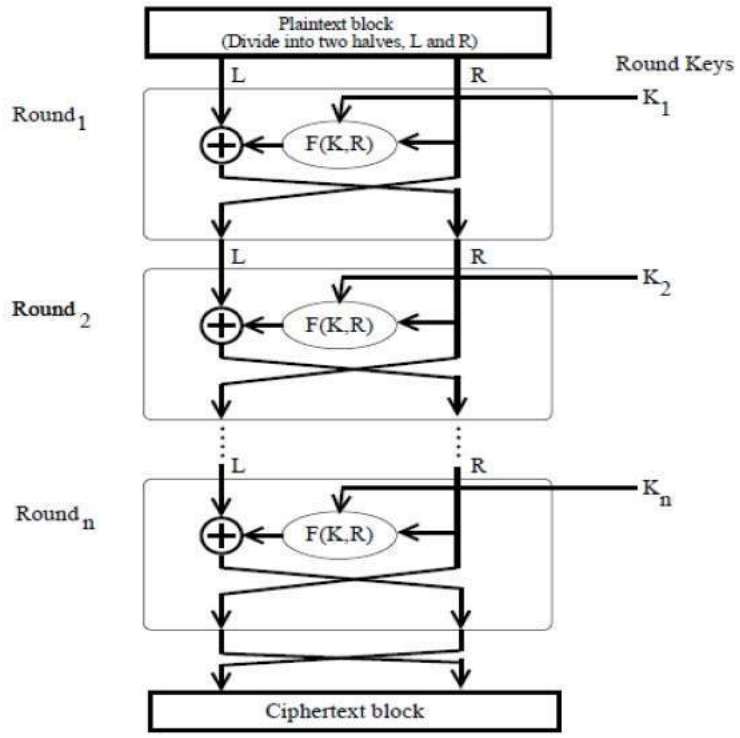
In a SPN based ciphers we have S-box and P-box along with some other functionalities/transformations. In simplest one, we first apply S-box and then P-box to encrypt.

5.2 Fiestal Network

It is also an other type of methodology to design my enc/dec algos.
P $\to$ plaintext of size 2n - bits.

$$P = L_0 || R_0$$

L_0 is left half and R_0 is right half of plaintext.
K : secret key len(K) = l-bits



$F : \{0,1\}^l \times \{0,1\}^n \to \{0,1\}^n$

For i^{th} round:
$L_i = R_{i-1}$
$R_i = L_{i-1} \oplus f(K, R_{i-1})$

6

Data Encryption Standard (DES) is based on Fiestal Network (FN).

Advanced Encryption Standard (AES) is based on SPN.

7 Iterated Block Cipher

An iterated block cipher is block cipher involving the sequential repetition of an internal function (called as Round function). The parameters include the number of rounds r, the block size n, and the round keys K_i of length l generated from the original secret key K.

8 OTP (One Time Padding)

OTP provides perfect secrecy under some conditions.

Encryption: P : plaintext, K: secret key
Enc(P,K) = P$\oplus$K = C
Decryption: Dec(C,K) = C $\oplus$ K = P

Aim: Pr[message | ciphertext] = Pr[message]

Conditions:

- The secret key K cannot be used to encrypt two messages.
- length(K) $\geq$ length(P)
- K is uniformly selected from the key space.

[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy
Scribed by: Akshay (202051018)

Winter 2022-2023
Lecture (Week 3)

1 One Time Padding

One time padding provides perfect secrecy under some conditions.

Encryption:

P : Plaintext
K : Secret key

Enc(P,K) = P⊕K = C

Decryption:
Dec(C,K) = C⊕K = P

$$Pr[message|ciphertext] = Pr[message]$$

Conditions:

- The secret key K cannot be used to encrypt two messages. (No repeatition of K)
- length(K) ≥ length(P)
- K is uniformly selected from the key space.

1.1 OTP on 1 Bit Enc.

message: m ∈ {0,1} K ∈ {0,1}

Pr[m=0] = p Pr[K=0] = $\frac{1}{2}$
Pr[m=1] = 1-p Pr[K=1] = $\frac{1}{2}$

(Key is uniformly distributed, that's why 0 and 1 have equal probability)

Encryption:
C = m⊕K
C = 0 $\implies$ (m=0 and K=0) or (m=1 and K=1)
Pr[C=0] = Pr[m=0,K=0] + Pr[m=1,K=1]
Pr[C=0] = Pr[m=0].P[K=0] + Pr[m=1].Pr[K=1]

$$= (p \times \frac{1}{2}) + (1-p) \times \frac{1}{2}$$
$$= \frac{1}{2}$$

Similarly, Pr[C=1] = $\frac{1}{2}$

This should hold for perfect secrecy:

$$Pr[M = m|C = c] = Pr[M = m]$$

Let's check:

$$Pr[M = 0|C = 0] = \frac{Pr[M = 0, C = 0]}{Pr[C = 0]}$$
$$= \frac{Pr[C = 0|M = 0].Pr[M = 0]}{\frac{1}{2}}$$
$$= \frac{Pr[K = 0].Pr[M = 0]}{\frac{1}{2}}$$
$$= \frac{\frac{1}{2} \times Pr[M = 0]}{\frac{1}{2}}$$
$$= Pr[M = 0]$$
$$Pr[M = 0|C = 0] = Pr[M = 0]$$

Thus it provides perfect secrecy.

1.2 Unfavourable Conditions

1)

$$M_1 \oplus K = C_1$$
$$M_2 \oplus K = C_2$$

(This will reveal information on messages)

$$C_1 \oplus C_2 = (M_1 \oplus K) \oplus (M_2 \oplus K)$$
$$= M_1 \oplus M_2$$

Ciphertext difference will give you message difference. i^{th} bit $= 0 \implies$ both message have same i^{th} bit (revealed some information).

2) len(K) < len(P)
Let len(P) = n, and len(K) = t
One way to use the key to encrypt P is by padding n-t zeroes at starting or ending but that will explicitly reveal some part of message. We must avoid using constants in the key.
What we do is we repeat bits of the same key to fill the gap.

$P = P_1....P_l....P_n$
$\oplus K = K_1....K_lK_1....K_t$

$C = (P_1 \oplus K_1)\ (P_2 \oplus K_2)\ ....\ (P_l \oplus K_l)\ (P_{l+1} \oplus K_1)\ ....\ (P_n \oplus K_t)$

If we xor C_i bit from first l bits with C_{l+i} bit then we will get $(P_i \oplus P_{l+i})$ which is revealing information about the message.

OTP is not usable in real life because we will not be able to reuse the secret key.

2 Data Encryption Standard (DES)

- It is a Block Cipher.
- Designed by IBM. Initially design was kept secret by IBM. When the design was made public it was immediately broken because of certain weaknesses.

1. Block size = 64 bit
2. Number of rounds = 16
3. Secret key size = 64 bit with 8 parity check bits (means there are 8 bits depending on the 56 bits).
4. It is based on Feistel Network.

Encryption:

$$\text{P (64 bit)} \rightarrow \underset{\downarrow \text{K (64 bit)}}{\boxed{\text{DES}}} \rightarrow \text{C of 64 bit}$$

Decryption:

$$\text{C (64 bit)} \rightarrow \underset{\downarrow \text{K (64 bit)}}{\boxed{DES^{-1}}} \rightarrow \text{P of 64 bit}$$

Secret key is 64 bit with 8 parity check bits.
01101010 11010001 01
Parity check bit is placed after every 7 bits. It is equal to the xor of previous seven bits. Parity check bits are used to check if the key is altered. After discarding 8 parity check bits the final secret key will be of 56 bit. DES should provide 56 bit security.

In DES we have 16 round keys $K_1, K_2, ...K_{16}$ (sub keys).
length(K_i) = 48 bits
They (sub keys) are generated using key scheduling algorithm. Key scheduling algorithm will take the secret key as input and output 16 sub keys, G(K).

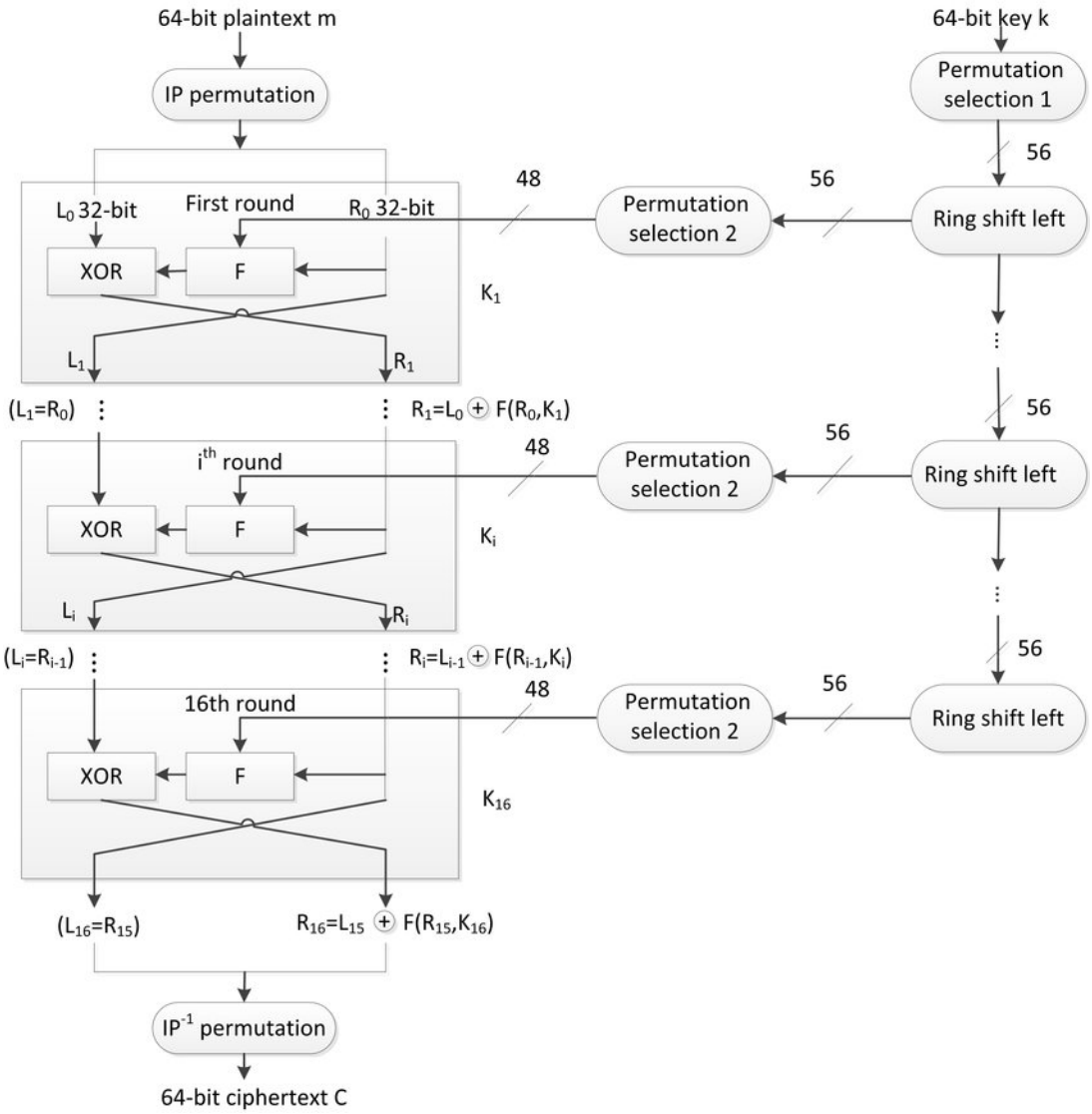


Image source: ResearchGate

$F : \{0,1\}^{32} \times \{0,1\}^{48} \to \{0,1\}^{32}$ (takes 32 bit right half of IP output and 48 bit key as argument and outputs 32 bit).
$L_i = R_{i-1}$; $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$
Note: In decryption we do not require the F to be invertible.

We have to learn the followings:

1. IP, IP^{-1}
2. F (round function)
3. How $K_1, K_2, ...K_{16}$ (round keys) are generated.

2.1 IP (Initial Permutation)

$$IP : \{0,1\}^{64} \to \{0,1\}^{64}$$

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

This table is basically having the 64 positions for 64 bits. We do the permutation in our message according to this table. Take the positions from the table row wise.

$$IP(m_1\ m_2\ ....\ m_{64}) = (m_{58}\ m_{50}\ .....\ m_{15}\ m_7)$$

58^{th} is placed at first position.
$L_0 = m_{58}\ m_{50}\ ....\ m_8$; $R_0 = m_{57}\ m_{49}\ ....\ m_7$

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

2.2 Round Function

$F(R_{i-1}, K_i) = X_i$; $F : \{0,1\}^{32} \times \{0,1\}^{48} \rightarrow \{0,1\}^{32}$

$$\boxed{\mathrm{F(R}_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))}$$

- $E : \{0,1\}^{32} \rightarrow \{0,1\}^{48}$ (Expansion function)
- $S : \{0,1\}^{48} \rightarrow \{0,1\}^{32}$ (Substitution box)
- $P : \{0,1\}^{32} \rightarrow \{0,1\}^{32}$ (Permutation box)

2.2.1 Expansion Function

$E : \{0,1\}^{32} \rightarrow \{0,1\}^{48}$

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

$$E(x_1\ x_2\ x_3\ x_4\ ....\ x_{32}) \rightarrow (x_{32}\ x_1\ x_2\ ....\ x_{32}\ x_1)$$

Some bits are repeated to expand the size to 48 from 32.

2.2.2 S Box

$S : \{0,1\}^{48} \rightarrow \{0,1\}^{32}$
S(X) = Y
X = $B_1\ B_2\ B_3\ B_4\ B_5\ B_6\ B_7\ B_8$, where length of B_i is 6 bit
$S_1,\ S_2,\ S_3,\ S_4,\ S_5,\ S_6,\ S_7,\ S_8$
$S_i : \{0,1\}^6 \rightarrow \{0,1\}^4 \qquad \forall\ \ i = 1, 2, ...8$
$S_i(B_i) = C_i$ (4 bit)

These S_i are fixed (pre-defined).

> $S_i(B_i)$ maps $B_i = b_1b_2....b_6$ to the 4-bit entry in row r and column c of S_i in table given below.
> Where, $r = 2.b_1 + b_6$ and $b_2b_3b_4b_5$ is radix-2 representation of $0 \leq c \leq 15$.
> E.g.: $S_1(011011)$ yields r=1, c=13, and output 5, i.e., binary 0101.

$0 \leq r \leq 3$; Basically r is the radix-2 representation of (b_1b_6)

row	column number															
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]
S_1																
[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
S_2																
[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
S_3																
[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
S_4																
[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	1	13	8	9	4	5	11	12	7	2	14
S_5																
[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
S_6																
[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
S_7																
[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
S_8																
[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

Image source: Handbook of Applied Cryptography

2.2.3 Permutation Box

$P : \{0,1\}^{32} \rightarrow \{0,1\}^{32}$

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

$$P(x_1\ x_2\ ....\ x_{32}) = (x_{16}\ x_7\ x_{20}\ ....\ x_4\ x_{25})$$

2.3 Key Scheduling Algorithm

Input: 64 bit key K
Output: 16 round keys K_i , $1 \leq i \leq 16$, where length of K_i is 48 bit.

- Define v_i, $1 \leq i \leq 16$
  - $v_i = 1$ if $i \in \{1, 2, 9, 16\}$
  - else $v_i = 2$
- Discard 8 parity check bits from K.
- T = PC1(K)
  PC1 : $\{0,1\}^{56} \rightarrow \{0,1\}^{56}$
- (C_0, D_0) = T
  where C_0 is of 28 bit (most significant part) and D_0 is of 28 bit (least significant part)
- for i=1 to 16:
  - C_i = leftCircularShift(C_{i-1}) by v_i unit
  - D_i = leftCircularShift(D_{i-1}) by v_i unit
  - $K_i = PC2(C_i, D_i)$
    PC2: $\{0,1\}^{56} \rightarrow \{0,1\}^{48}$

PC1 and PC2 are substitution tables.

Table 1: **DES key schedule bit selections (PC1)**

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
above for C_i ; below for D_i						
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

Table 2: **DES key schedule bit selections (PC2)**

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

[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy
Scribed by: Akshay (202051018)

Winter 2022-2023
Lecture (Week 4)

1 Attack Models

1.1 Cipher-text only Attack

Attackers Knows only ciphertexts.
Goal:- Recover the plaintexts corresponding to the ciphertexts or recover the secret key.

1.2 Known Plaintext Attack

Attacker knows some plaintexts and corresponding ciphertexts.

1.3 Chosen Plaintext Attack

Attackers chooses plaintexts according to hi/her choice and (s)he will be provided the corresponding ciphertexts.
Goal:- Generate new plaintext, ciphertext pair or recover the secret key.

1.4 Chosen Ciphertext Attack

Attacker chooses some ciphertext and he/she is allowed to get the corresponding plaintexts.
Goal:- Generate a new plaintext and ciphertext pair or recover the secret key.

Order of strongness of above attacks:
$1 < 2 < 3 < 4$

2

$$DES(M, K) = C$$
$$DES(\overline{M}, \overline{K}) = \overline{C}$$

Key = 56; Brute force attack/Exhaustive search = 2^{56}

Attacker chooses two plaintexts:
1) M 2) $\overline{M}$
Provided by corresponding ciphers C_1 and C_2, both encrypted with the same key.

$$C_1 = DES(M, K)$$

$$C_2 = DES(\overline{M}, K)$$

Challenge is find the key K (CPA, Chosen plaintext attack).

From the design of DES:

$$DES(\overline{\overline{M}}, \overline{K}) = \overline{C_2}$$
$$\implies DES(M, \overline{K}) = \overline{C_2}$$
$$\text{Keys} = \{K_1, K_2, ..., K_{2^{56}}$$

Attacker selects $K_1 \in$ Keys.
He knows that $\overline{K_1} \in$ Keys.

Attacker performs $DES(M, K_1) = C$
If $\tilde{C} \neq C_1$ or $\tilde{C} \neq \overline{C_2}$
then discard K_1 and $\overline{K_1}$

Because:
$\tilde{C} \neq C_1 \implies K_1 = K$
$\tilde{C} \neq \overline{C_2} \implies K_1 \neq \overline{K} \implies \overline{K_1} \neq K$

Search space is reduced to half

DES is not secure due to multiple attacks.

3 Making DES Secure from Attacks

Increase the length of the secret key.

3.1 Idea of Double Encryption

$$K = K_1 || K_2$$

length of $K_1 = 56$ bit
length of $K_2 = 56$ bit
$\implies$ len(K) = 112 bit

1) Enc:

$$\text{P} \rightarrow \overset{\downarrow K_1}{\boxed{Enc_{DES}}} \rightarrow \overset{\downarrow K_2}{\boxed{Enc_{DES}}} \rightarrow \text{C}$$

Dec:

$$\text{C} \rightarrow \overset{\downarrow K_2}{\boxed{Dec_{DES}}} \rightarrow \overset{\downarrow K_1}{\boxed{Dec_{DES}}} \rightarrow \text{P}$$

2) Enc:

$$\text{P} \rightarrow \overset{\downarrow K_1}{\boxed{Enc_{DES}}} \rightarrow \overset{\downarrow K_2}{\boxed{Dec_{DES}}} \rightarrow \text{C}$$

Dec:

$$\text{C} \rightarrow \overset{\downarrow K_2}{\boxed{Enc_{DES}}} \rightarrow \overset{\downarrow K_1}{\boxed{Dec_{DES}}} \rightarrow \text{P}$$

- EE, ED, DE, DD

This Idea of doing double encryption does not provide any extra security
Proof:
$K = K_1||K_2$
Attacker knows plaintext M and the corresponding ciphertext C.

$$C = Enc(Enc(M, K_1), K_2)$$

Keys $= \{SK_1, ..., SK_{2^{56}}\}$ (We have to select K_1 and K_2 from the same set of 2^{56} keys.
$Enc(M, SK_i) = X_i$
$Dec(C, SK_j) = Y_j$
If $X_i = Y_j$ then $SK_i = K_1$ and $SK_j = K_2$
Time complexity is still 2^{56} instead of 2^{112}
Hence, encrypting twice does not provide any extra security. It is true for all algorithms.

3.2 Triple Encryption

$K = K_1||K_2$; 2n-bit security

1) Enc:

$$\text{P} \rightarrow \overset{\downarrow K_1}{\boxed{Enc}} \rightarrow \overset{\downarrow K_2}{\boxed{Dec}} \rightarrow \overset{\downarrow K_1}{\boxed{Enc}} \rightarrow \text{C}$$

Dec:

$$\text{C} \rightarrow \overset{\downarrow K_1}{\boxed{Dec}} \rightarrow \overset{\downarrow K_2}{\boxed{Enc}} \rightarrow \overset{\downarrow K_1}{\boxed{Dec}} \rightarrow \text{P}$$

- EEE, EDE, DED, ...

If DES is used in Triple encryption setup then it is known as Triple DES (3-DES)

4 Maths Pre-requisite for AES

We have to understand certain mathematical results.

4.1 Binary Operation

A binary operation $\star$ on a set S is a mapping from S×S to S.
That is $\star$ is a rule which assigns to each ordered pair of elements from S to an element of S.
$\star : S \times S \rightarrow S$
$\star(a, b) = c$, a,b,c $\in$ S

$\star(b, a) = d \qquad d \in S$
It is not necessary that c = d.

4.2 Group

A group (G, $\star$) consists of a set G with a binary operation $\star$ on G satisfying the following axioms.

1. $\star$ is associative on G.
   $a \star (b \star c) = (a \star b) \star c \ \ \forall$ a,b,c $\in$ G.
2. There is an element e $\in$ G called the identity element such that:
   $a \star e = a = e \star a \ \ \forall$ a $\in$ G.
3. For each $a \in G$ there exists an element $a^{-1} \in G$ called the inverse of 'a' s.t. $a \star a^{-1} = e = a^{-1} \star a$
   $\forall$ a $\in$ G.

A group G is called abelian (or commutative) if:
$a \star b = b \star a \ \ \forall$ a,b $\in$ G.

Example:
1)
$\star$: matrix multiplication over square matrices of order nn
M : set of $n \times n$ matrices over $\mathbb{R}$
(M, $\star$) - Is it a Group? Ans: No, because all square matrices are not invertible.

2)
$\mathbb{Z}$: set of all integers.
$(\mathbb{Z}, +)$: Is it a group? Ans: Yes

1. a+(b+c) = (a+b)+c
2. a + 0 = a = 0 + a $\qquad \forall$ a $\in$ Z
3. $\forall$ a $\in \mathbb{Z} \ \ \exists$ -a $\in \mathbb{Z}$ s.t. a + (-a) = 0 = (-a) + a

3)
$\mathbb{Z}$: set of all integers
$(\mathbb{Z}, +)$: Group? Ans: Yes

4)
$(\mathbb{Z}, \times)$: Not a group.
$a \in \mathbb{Z} \ \nexists \ a^{-1} \in \mathbb{Z}$

5)
Q : set of all rational numbers.
(Q, $\times$) : Not a group
(Q - {0}, $\times$) : group

If $|G|$ is finite then (G, $\times$) is a finite group.
$|G|$: Cardinality of G

Example:
$Z_n = \{0, 1, ..., n-1\}$
$(Z_n, +_n)$: group
$x +_n y = (x + y) mod\ n$

$(Z_n - \{0\}, \times_n)$: not a group because not all numbers have inverse under modulo n.

[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy — Winter 2022-2023
Scribed by: Akshay (202051018) — Lecture (Week 5)

1 Generator

A group (G, $\star$) is closed under $\star$
Let's say, $\alpha \in G$
and, $\alpha^0, \alpha^1, \alpha^2, ... \in G$, ——— (1) where α^0 is identity
α^n is outcome of applying $\star$ b/w α^n and α

> For any $b \in G$ if $\exists\ i \geq 0$ s.t. $b = \alpha^i$
> then, α is called the generator of $(G, \star)$ ——— (2)
> Such groups are called cyclic group.

From (1), $< \alpha > \subseteq G$ ——— (3)
From (2), $G \subseteq < \alpha >$ ——— (4) From (3) and (4) $(G, \star) = < \alpha >$

2 Order

$(G, \star)$ $|G| : finite$ $a \in G$

> Orde of 'a' i.e. O(a) is the leasr postive integer m such that $a^m = e$ (identity)
> O(a) = m

$a^0 (= e), a^1, a^2, ...a^{m-1} \in G$
H = $\{a^0, a^1, a^2, ...a^{m-1}\}$, After a^{m-1} every element will repeat ($\because a^m = e$)

1. $H \subseteq G$
2. H is group under $\star$ (you can check verify with properties)
   - $a^i \star a^{m-i} = a^0 = e \implies$ invertible

H is a sub-group of G.

3 Lagrange's theorem

> If G is a finite group and H is a sub-group of G then $|H|$ divides $|G|$

$O(a)$ *divides* $|G|$ because O(a) is cardinality of cyclic **sub-group** of G

Another Result:

For $a \in G$,

$$O(a^k) = \frac{O(a)}{gcd(O(a),k)} = \frac{t}{gcd(t,k)}$$

where, t = O(a)

If gcd(t,k) =1 then,
$O(a^k) = t = O(a)$
$| < a^k > | = | < a > |$
$\implies < a^k > = < a >$
Reasoning:
$x \in < a^k >$
$\implies x = (a^k)^i = a^{ki} = a^{(some\ integer\ only)} \in < a >$
$\implies < a^k >$ **is also a generator of same cyclic group.**
Result:

If k is co-prime O(a) then a^k is the generator of same set which 'a' generates.

4 Illustration:

$\mathbb{Z}_{19}^* = \{x \mid gcd(x,19) = 1,\ 1 \le x \le 18\}$
$\star_{19}$: multiplication modulo 19.
$x \star_{19} y = (x.y) mod 19$
Find the generator of the group $(\mathbb{Z}_{19}^*, \star_{19})$ **:**
$x \in \mathbb{Z}_{19}^*$
S = $\{x^0, x^1, ...\}$
< 2 > = {1,2,4,8,16,13,7,14,9,18,17,15,11,3,6,12,5,10} = $\mathbb{Z}_{19}^*$
So, 2 is a generator. O(2) = 18
32 (=2^5) is also a generator of $\mathbb{Z}_{19}^*$ because gcd(5,O(2)) = gcd(5,18) = 1.

5 Ring

A ring $(R, +_R, \times_R)$ consists of one set R with two binary operations arbitrarily denoted by $+_R$ (addition) and $\times_R$ (multiplication) on R satisfying the following properties:

1. $(R, +_R)$ is an abelian group with the identity element 0_R
2. The operation $\times_R$ is associative i.e., $a \times_R (b \times_R c) = (a \times_R b) \times_R c\ \forall$ a,b,c $\in$ R
3. There is a multiplicative identity denoted by 1_R with $1_R \neq 0_R$ s.t. $1_R \times_R a = a \times_R 1_R = a\ \forall a \in R$
4. The operation $\times_R$ is distributive over $+_R$ i.e.,
   $(b +_R c) \times_R a = (b \times_R a) +_R (c \times_R a)$,
   $a \times_R (b +_R c) = (a \times_R b) +_R (a \times_R c)$

5.1 Examples:

5.2

$(Z, +, \cdot)$: Check - ring or not.
Ring ✓

5.3

$(R, +_R, \times_R)$: Is it a ring? Yes ✓
If $a \times_R b = b \times_R a \ \forall a, b \in R$
then $(R, +_R, \times_R)$ is a commutative ring

Commutativity is meaningful w.r.t second operation in ring

An element 'a' of a ring R is called unit or an invertible element if there is an element $b \in R$ s.t. $a \times_R b = 1_R$
The set of units in a ring R forms a group under multiplication operation. This is known as group of units of R.

6 Field

A Field is a non-empty set F together with two binary operation + (addition) and * (multiplication) for which the following properties are satisfied:

1. (F,+) is an abelian group
2. If 0_F denotes the additive identity element of (F,+) then $(F\backslash\{0_F\}, \star)$ is a commutative/abelian group
3. $\forall a, b, c \in F$, we have:
   a*(b+c) = (a*b)+(a*c) (distributive)

Example:
$(Z_p, +_p, *_p)$, $Z_p = \{0, 1, ..., p-1\}$, p: prime
Field ✓

6.1 Field Extension

Suppose K_2 is a field with addition (+) and multiplication (*). Suppose $K_1 \subseteq K_2$ is closed under both these operation such that K_1 itself is a field with the restriction of + and * to the set K_1
Then K_1 is called a sub-field of K_2, and K_2 is called a field extension of K_1

F : field (F,+,*)

F[x] = $\{a_0 + a_1x + a_2x^2 + ...|a_i \in F\}$
F[x] is a set of all polynomials whose coefficients are from the field F. We can choose any field for our F[x]. If we choose $\mathbb{F}_p$ then the coefficients will be from 0 to p-1. If we choose R field then

coefficients will be real numbers.

Polynomial ring, (F[x], +, *)
If we club F[x] (defined already) with plus and multiply operator then it will become a ring, more specifically, a polynomial ring.

$P_1(x) \in F[x]$ $\quad P_1(x) = a_0 + a_1x + a_2x^2$
$P_2(x) \in F[x]$ $\quad P_2(x) = b_0 + b_1x + b_2x^2$
$P_1(x) + P_2(x) = (a_0 + a_1x + a_2x^2) + (b_0 + b_1x + b_2x^2) = (a_0 + b_0) + (a_1 + b_1)x + (a_2 + b_2)x^2$
$(a_i + b_i)$: Field addition
Additive identity: $0 + 0.x + 0.x^2$
Additive inverse: $-a_0 - a_1x - a_2x^2$

Under addition it is abelian group ✓
* is associative
1 is multiplicative identity
* is distributive over +
Ring ✓

6.2 Irreducible Polynomial

A polynomial $P(x) \in F[x]$ of degree n (n ≥ 1) is called irreducible if it cannot be written in the form of $P_1(x) * P_2(x)$ with $P_1(x)$, $P_2(x) \in F[x]$ and degree of $P_1(x)$, $P_2(x)$ must be ≥ 1

$$P(x) \neq P_1(x) * P_2(x)$$

Example:
$\boldsymbol{x^2 + 1 \in \mathbb{F}_2[x]}$
$(x + 1) * (x + 1) = x^2 + (1 + 1)x + 1 = x^2 + 1$ (Don't forget: operations are under modulo 2)
$\implies (x^2 + 1) = (x + 1) * (x + 1)$ in $\mathbb{F}_2[x]$
$\implies x^2 + 1$ is reducible in $\mathbb{F}_2[x]$

$$I =< P(x) >= \{q(x).P(x) \mid q(x) \in F(x)\}$$

I : ideal generated by P(x)

$$\boldsymbol{F[x]/ < P(x) >}$$

For every $q(x) \in F[x]$, $r(x) \in F[x]/ < P(x) >$
where $r(x)$ is obtained as remainder by dividing $q(x)$ with $P(x)$
$q(x) = d(x) * P(x) + r(x)$

Note: degree(r(x)) < degree(P(x))

A Result:

If $P(x)$ is an irreducible polynomial then $(F[x]/ < P(x) >, +, *)$ becomes a field.

After each operation in this field, outcome is stored as remainder we get by dividing with $< P(x) >$

Example:
$x^2 + x + 1 \in \mathbb{F}_2[x]$, $(F_2 = \{0, 1\})$
$P(x) = x^2 + x + 1$, which is irreducible
$q(x) = d(x).P(x) + r(x)$ $(q(x) \in \mathbb{F}_2[x])$

$$\mathbb{F}_2[x]/ < x^2 + x + 1 >$$

deg(r(x)) < 2
We can construct these polynomials: x, 1, $x + 1$, 0
$r(x) \in \{0, 1, x, x + 1\}$

An instance:

$$\begin{array}{r} 1 \\ x^2+x+1 \overline{)\, x^2+1} \\ \underline{x^2+x+1} \\ x \end{array}$$

$\star$
Observation from programming point of view:

- We can take xor of coefficients of dividend with $x^2 + x + 1$ to get to the remainder
- Remainder can be achieved by replacing x^2 (higher degree term in dividend) with $x + 1$ (part of divisor excluding higher degree term)
  $(x + 1) + 1 = x$

Another instance for clarity:

$$\begin{array}{r} x+1 \\ x^2+x+1 \overline{)\, x^3+1} \\ \underline{x^3+x^2+x} \\ x^2+x+1 \\ \underline{x^2+x+1} \\ 0 \end{array}$$

Now, let's do it with our programming hacks:
Replacer : $x + 1$ (taken from divisor by excluding higher degree term)

$$x^3 + 1$$

$$= x.x^2 + 1$$

$$= x(x+1) + 1$$

$$= x^2 + x + 1$$

$$= (x+1) + x + 1 = 0$$

6.3 Primitive Polynomial

$$\mathbb{F}_2[x]/ < x^2 + x + 1 >$$

$x^2 + x + 1 = 0$
Let α is a root of $x^2 + x + 1 = 0$
So, $\alpha^2 + \alpha + 1 = 0$
$\implies \alpha^2 = \alpha + 1$ (-1 = 1 under mod 2)
$\{0, 1 = \alpha^0, \alpha^1, \alpha^2 = \alpha + 1\}$ $O(\alpha) = 2$
Since, α is generating all the elements of the field (all possible polynomials r(x)), so, polynomial $x^2 + x + 1$ is a primitive polynomial.

$$\mathbb{F}_2[x]/ < x^3 + x + 1 >$$

Maximum number of polynomials degree < 3: $2^3 = 8$
$\{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$
$\alpha^3 + \alpha + 1 = 0 \implies \alpha^3 = \alpha + 1$
$\{0, \alpha^0, \alpha, \alpha^2, \alpha+1, \alpha^2+\alpha, \alpha^3+\alpha^2 = \alpha^2+\alpha+1, \alpha^3+\alpha^2+\alpha = \alpha^2+1\}$
primitive polynomial ✓

7 Advanced Encryption Standard (AES)

It is standardized by NIST
- Rijndael: Winner design of Advanced Encryption Standard competition.
- Winner of competition was named as AES.

AES :

- Iterated block cipher
- It is based on SPN

Variants of AES :-
AES-128 :

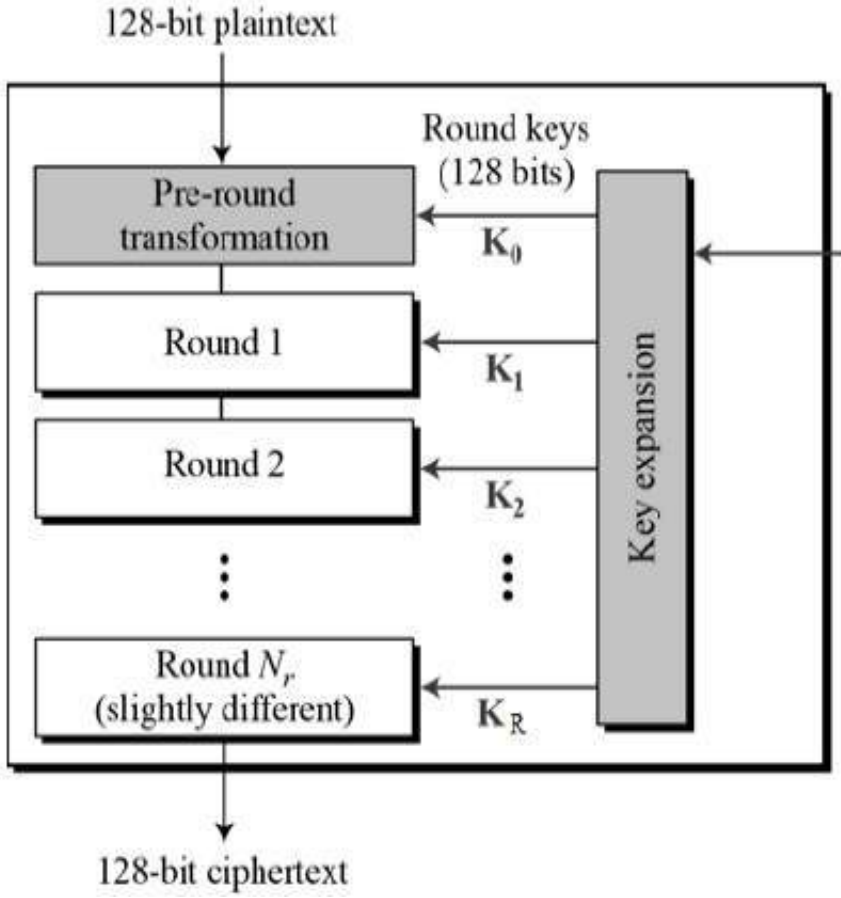
- Block size = 128 bit
- Number of rounds = 10
- Secret key size = 128 bit

AES-192 :

- Block size = 128 bit
- Number of rounds = 12
- Secret Key size = 192 bit

AES-256 :

- Block size = 128 bit
- Number of rounds = 14
- Secret Key size = 256 bit



R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy
Scribed by: Akshay (202051018)

Winter 2022-2023
Lecture (Week 6)

1 AES ....cnt'd

We have to understand the followings:

- Round function
- Key scheduling algorithm

Round functions of AES-128

$$f_1, f_2, ..., f_{10}$$

1. $f_1 = f_2 = f_3 = ... = f_9$
2. f_{10} is different from f_i, $i = 1, 2, ..., 9$

   (In each version of AES, all round functions are equal except the last round function)

The first nine round functions are based on the following functions:

1. Subbytes
2. Shift rows
3. Mix column

The 10th round function is based on:

1. Subbytes
2. Shift rows

$$f_i : \{0,1\}^{128} \to \{0,1\}^{128}$$

In each round function, on input we apply subbytes first, whatever the output we get is passed to shift-rows and output from shift rows will be served as input to Mix Column.

$$\text{MixColumn(ShiftRows(Subbytes(x)))} = f_i(x)$$

128 bit $\to \boxed{f_i} \to$ 128 bit $\qquad i = 1, 2, ..., 9$

128 bit $\to$ Subbytes $\xrightarrow{128 \text{ bit}}$ Shift rows $\xrightarrow{128 \text{ bit}}$ Mix column $\to$ 128 bit

1.1 Subbytes

$$\text{Subbytes} : \{0,1\}^{128} \rightarrow \{0,1\}^{128}$$

s : input

$$s = \begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{bmatrix}_{4\times 4} \qquad s_{ij} =: \text{8-bit} \qquad (8^*(16) = 128 \text{ (total bits)})$$

Plaintext (P, 128-bit) is xored with the round key (128-bit) to get input (s) of subbyte functions.

$P = P_0P_1....P_{15}$

$$P \oplus K_1 = \begin{bmatrix} P_0 & P_4 & P_8 & P_{12} \\ P_1 & P_5 & P_9 & P_{13} \\ P_2 & P_6 & P_{10} & P_{14} \\ P_2 & P_7 & P_{11} & P_{15} \end{bmatrix} \oplus K_1 = \begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{bmatrix}$$

We have a pre-defined S-box which maps 8-bit to 8-bit.
$S : \{0,1\}^8 \rightarrow \{0,1\}^8$

1. $(C_7C_6C_5C_4C_3C_2C_1C_0) = (01100011) = (63)_{16}$ (constant)
2. $\text{S}(s_{ij}) = (a_7a_6a_5a_4a_3a_2a_1a_0)$
3. For i=0 to 7
   $b_i = (a_i + a_{(i+4)\%8} + a_{(i+5)\%8} + a_{(i+6)\%8} + a_{(i+7)\%8} + C_i) mod 2$
4. $(b_7b_6b_5b_4b_3b_2b_1b_0)$
5. $s'_{ij} = (b_7b_6 \ldots. b_0)$

$$\begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{bmatrix} \rightarrow \begin{bmatrix} s'_{00} & s'_{01} & s'_{02} & s'_{03} \\ s'_{10} & s'_{11} & s'_{12} & s'_{13} \\ s'_{20} & s'_{21} & s'_{22} & s'_{23} \\ s'_{30} & s'_{31} & s'_{32} & s'_{33} \end{bmatrix}$$

Now, we have to learn the working of S-box:
S(0) = 0
$X \neq 0 \in \{0,1\}^8$
S(X) = Y $\in \{0,1\}^8$
X = $(a_7a_6a_5a_4a_3a_2a_1a_0)$ $a_i \in \{0,1\}$
P(x) = $a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6 + a_7x^7 \in \mathbb{F}_2[x]$
deg(P(x)) ≤ 7 $P(x) \in \mathbb{F}_2[x]$
$(\mathbb{F}_2[x], +, *)$: Field
$g(x) = x^8 + x^4 + x^3 + x + 1$
g(x) is a primitive polynomial

$$(\mathbb{F}_2[x]/ < g(x) >, +, *)$$

Find the multiplicative inverse of P(x) under modulo $(x^8 + x^4 + x^3 + x + 1)$
P(x).q(x) $\equiv 1 \bmod (x^8 + x^4 + x^3 + x + 1)$
$\implies P(x).q(x) - 1 = h(x).(x^8 + x^4 + x^3 + x + 1)$
$\implies 1 = P(x).q(x) + h(x).(x^8 + x^4 + x^3 + x + 1)$
$\gcd(P(x), (x^8 + x^4 + x^3 + x + 1)) = 1$
How to find q(x) ?
We use Extended Euclidean Algo to find q(x).
q(x): Polynomial of degree 7.
$q(x) = r_0 + r_1x + r_2x^2 + r_3x^3 + r_4x^4 + r_5x^5 + r_6x^6 + r_7x^7$
$q(x) \to (r_7r_6r_5 \ldots r_0) \in \{0,1\}^8$
S(X) = Y = $(r_7r_6r_5 \ldots r_0)$
Example:
$P(x) = x^6 + x^4 + x + 1$
$g(x) = x^8 + x^4 + x^3 + x + 1$

$$
\begin{array}{r}
x^6 + x^4 + x + 1 \overline{)\, x^8 + x^4 + x^3 + x + 1 \quad} \\
\underline{x^8 + x^6 + x^3 + x^2} \\
x^6 + x^4 + x^2 + x + 1 \\
\underline{x^6 + x^4 + x + 1} \\
x^2 \overline{)\, x^6 + x^4 + x + 1} \\
\underline{x^6} \\
x^4 + x + 1 \\
\underline{x^4} \\
x + 1 \overline{)\, x^2 \quad} \\
\underline{x^2 + x} \\
x \\
\underline{x + 1} \\
1
\end{array}
\begin{array}{l}
\big(x^2 + 1 \\
\\
\\
\\
\big(x^4 + x^2 \\
\\
\\
\\
\big(\quad x + 1 \\
\\
\\
\\
\\
\end{array}
$$

Aim: $1 = q(x).P(x) + h(x).g(x)$
where 1 is gcd of P(x) and g(x). Inverse of g(x) is q(x).
We go from bottom to up in the search of aim equation.
Here:

$$remainder = dividend + divisor \times quotient$$

$$1 = x^2 + (x+1)(x+1)$$
$$= x^2 + (x+1)[(x^6 + x^4 + x + 1) + x^2(x^4 + x^2)]$$
$$= x^2 + (x+1)(x^6 + x^4 + x + 1) + (x+1)x^2(x^4 + x^2)$$
$$= (x+1)(x^6 + x^4 + x + 1) + x^2[1 + (x+1)(x^4 + x^2)]$$
$$= (x+1)(x^6 + x^4 + x + 1) + (1 + x^5 + x^3 + x^4 + x^2)x^2$$

Now replace x^2:

$$= (x+1)(x^6 + x^4 + x + 1) + (1 + x^5 + x^4 + x^3 + x^2)[(x^8 + x^4 + x^3 + x + 1) + (x^6 + x^4 + x + 1)(x^2 + 1)]$$

$$= (1+x^5+x^4+x^3+x^2)(x^8+x^4+x^3+x+1)+(x^6+x^4+x+1)[(x+1)+(1+x^5+x^4+x^3+x^2)(x^2+1)]$$

$$= h(x).g(x) + (x^6 + x^4 + x + 1)[x + 1 + x^2 + x^7 + x^6 + x^5 + x^4 + 1 + x^5 + x^4 + x^3 + x^2$$

$$= h(x).g(x) + (x^6 + x^4 + x + 1)(x^7 + x^6 + x^3 + x)$$

$$q(x) = x^7 + x^6 + x^3 + x$$

Which is the multiplicative inverse of $(x^6 + x^4 + x + 1)$
S(01010011) = (11001010) = $(a_7a_6a_5a_4a_3a_2a_1a_0)$
C = (01100011)
$b_i = (a_i + a_{(i+4)\%8} + a_{(i+5)\%8} + a_{(i+6)\%8} + a_{(i+7)\%8} + C_i) mod\ 2$
$(b_7b_6b_5b_4b_3b_2b_1b_0)$ = (11101101)
So, Subbytes(0101 0011) = (1110 1101)
In hexadecimal:
Subbytes(5 3) = E D
For programming we have outputs corresponding to each inputs already precomputed and stored.
Input = (X Y)
Subbyte(Input) = element present in the row number X and column number Y.

	Y															
X	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
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

Source: Stinson Book

[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy — Winter 2022-2023
Scribed by: Akshay (202051018) — Lecture (Week 7)

1 Shift rows

$$ShiftRows : \{0,1\}^{128} \rightarrow \{0,1\}^{128}$$

$$\begin{matrix}0\\1\\2\\3\end{matrix}\begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{bmatrix} \rightarrow \begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{11} & s_{12} & s_{13} & s_{10} \\ s_{22} & s_{23} & s_{20} & s_{21} \\ s_{33} & s_{30} & s_{31} & s_{32} \end{bmatrix}$$

We left rotate i^{th} row by i unit.

2 Mix Column

$$MixColumn : \{0,1\}^{128} \rightarrow \{0,1\}^{128}$$
$$(s_{ij})_{4\times 4} \rightarrow (s^{'}_{ij})_{4\times 4}$$

Consider the column $C \in \{0,1,2,3\}$

$$\text{Column} = \begin{pmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{pmatrix}$$

$s_{ic} = (a_7 a_6 a_5 \ .... \ a_0)$
$Poly^n = a_0 + a_1 x + ... + a_7 x^7$

For i=0 to 3:

t_i = Binary to poly(s_{ic})

$$u_0 = \Big[(x * t_0) + (x+1) * t_1 + t_2 + t_3\Big] mod(x^8 + x^4 + x^3 + x + 1)$$
$$u_1 = \Big[(x * t_1) + (x+1) * t_2 + t_3 + t_0\Big] mod(x^8 + x^4 + x^3 + x + 1)$$
$$u_2 = \Big[(x * t_2) + (x+1) * t_3 + t_0 + t_1\Big] mod(x^8 + x^4 + x^3 + x + 1)$$
$$u_3 = \Big[(x * t_3) + (x+1) * t_0 + t_1 + t_2\Big] mod(x^8 + x^4 + x^3 + x + 1)$$

$s^{'}_{ic}$ = Polynomial_to_binary(u_i)

$$\begin{pmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{pmatrix} \rightarrow \begin{pmatrix} s^{'}_{0c} \\ s^{'}_{1c} \\ s^{'}_{2c} \\ s^{'}_{3c} \end{pmatrix}$$

Matrix form:

$$\begin{bmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{bmatrix} * \begin{bmatrix} poly(s_{00}) & poly(s_{01}) & poly(s_{02}) & poly(s_{03}) \\ poly(s_{10}) & poly(s_{11}) & poly(s_{12}) & poly(s_{13}) \\ poly(s_{20}) & poly(s_{21}) & poly(s_{22}) & poly(s_{23}) \\ poly(s_{30}) & poly(s_{31}) & poly(s_{32}) & poly(s_{33}) \end{bmatrix} mod(x^8+x^4+x^3+x+1)$$

$$= \begin{bmatrix} (u_0)_{c=0} & (u_0)_{c=1} & (u_0)_{c=2} & (u_0)_{c=3} \\ (u_1)_{c=0} & (u_1)_{c=1} & (u_1)_{c=2} & (u_1)_{c=3} \\ (u_2)_{c=0} & (u_2)_{c=1} & (u_2)_{c=2} & (u_2)_{c=3} \\ (u_3)_{c=0} & (u_3)_{c=1} & (u_3)_{c=2} & (u_3)_{c=3} \end{bmatrix}$$

$$\begin{bmatrix} to\_binary(\ (u_0)_{c=0}\ ) & to\_binary(\ (u_0)_{c=1}\ ) & to\_binary(\ (u_0)_{c=2}\ ) & to\_binary(\ (u_0)_{c=3}\ ) \\ to\_binary(\ (u_1)_{c=0}\ ) & to\_binary(\ (u_1)_{c=1}\ ) & to\_binary(\ (u_1)_{c=2}\ ) & to\_binary(\ (u_1)_{c=3}\ ) \\ to\_binary(\ (u_2)_{c=0}\ ) & to\_binary(\ (u_2)_{c=1}\ ) & to\_binary(\ (u_2)_{c=2}\ ) & to\_binary(\ (u_2)_{c=3}\ ) \\ to\_binary(\ (u_3)_{c=0}\ ) & to\_binary(\ (u_3)_{c=1}\ ) & to\_binary(\ (u_3)_{c=2}\ ) & to\_binary(\ (u_3)_{c=3}\ ) \end{bmatrix} = (s'_{ij})_{4\times 4}$$

Hexadecimal representation:
$x = (00000010)_2 = (2)_{16}$
$x+1 = (00000011)_2 = (3)_{16}$
$1 = (00000001)_2 = (1)_{16}$

2.1 Illustration

Find $\begin{bmatrix} s'_{00} \\ s'_{10} \\ s'_{20} \\ s'_{30} \end{bmatrix}$ after doing mix-column operation on $\begin{bmatrix} s_{00} \\ s_{10} \\ s_{20} \\ s_{30} \end{bmatrix}$

where, $s_{00} = 95$, $s_{10} = 65$, $s_{20} = FD$, $s_{30} = F3$

Sol:

c = 0 (given)

So, $\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{bmatrix} * \begin{bmatrix} poly(s_{00}) \\ poly(s_{10}) \\ poly(s_{20}) \\ poly(s_{30}) \end{bmatrix} mod(x^8+x^4+x^3+x+1)$

$$\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{bmatrix} * \begin{bmatrix} poly(95) \\ poly(65) \\ poly(FD) \\ poly(F3) \end{bmatrix} mod(x^8+x^4+x^3+x+1)$$

$$= \begin{bmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{bmatrix} * \begin{bmatrix} poly(1001\ 0101) \\ poly(0110\ 0101) \\ poly(1111\ 1101) \\ poly(1111\ 0011) \end{bmatrix} mod(x^8+x^4+x^3+x+1)$$

$$= \begin{bmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{bmatrix} * \begin{bmatrix} x^7+x^4+x^2+1 \\ x^6+x^5+x^2+1 \\ x^7+x^6+x^5+x^4+x^3+x^2+1 \\ x^7+x^6+x^5+x^4+x^1+1 \end{bmatrix} mod(x^8+x^4+x^3+x+1)$$

$$= \begin{bmatrix} x^8+x^7+x^3+x+1 \\ x^8+x^7+x^5+x^3+x+1 \\ x^4+x^3+x^2+x+1 \\ x^7+x^6+1 \end{bmatrix} mod(x^8+x^4+x^3+x+1)$$

Trick to multiply and sum faster:

1. Write x^8 to x^0 as a header of a table
2. After each individual computations of terms, append 1 to the column of corresponding degree
3. Result will have only those degree terms which got odd count of 1s.

To find remainder (or to solve modulo) replace the highest degree term from the dividend with the part of di

Part of divisor: x^4+x^3+x+1

$$\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} x^8+x^7+x^3+x+1 \\ x^8+x^7+x^5+x^3+x+1 \\ x^4+x^3+x^2+x+1 \\ x^7+x^6+1 \end{bmatrix} mod(x^8+x^4+x^3+x+1)$$

$$\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} (x^4+x^3+x+1)+x^7+x^3+x+1 \\ (x^4+x^3+x+1)+x^7+x^5+x^3+x+1 \\ x^4+x^3+x^2+x+1 \\ x^7+x^6+1 \end{bmatrix}$$

$$\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} x^7+x^4 \\ x^7+x^5+x^4 \\ x^4+x^3+x^2+x+1 \\ x^7+x^6+1 \end{bmatrix}$$

Now,

$$\begin{bmatrix} s'_{00} \\ s'_{10} \\ s'_{20} \\ s'_{30} \end{bmatrix} = \begin{bmatrix} to\_binary(u_0) \\ to\_binay(u_1) \\ to\_binay(u_2) \\ to\_binay(u_3) \end{bmatrix}$$

$$\begin{bmatrix} s'_{00} \\ s'_{10} \\ s'_{20} \\ s'_{30} \end{bmatrix} = \begin{bmatrix} to\_binary(x^7+x^4) \\ to\_binay(x^7+x^5+x^4) \\ to\_binay(x^4+x^3+x^2+x+1) \\ to\_binay(x^7+x^6+1) \end{bmatrix}$$

$$\begin{bmatrix} s'_{00} \\ s'_{10} \\ s'_{20} \\ s'_{30} \end{bmatrix} = \begin{bmatrix} 1001\ 0000 \\ 1011\ 0000 \\ 0001\ 1111 \\ 1100\ 0001 \end{bmatrix}$$

$$\begin{bmatrix} s'_{00} \\ s'_{10} \\ s'_{20} \\ s'_{30} \end{bmatrix} = \begin{bmatrix} 90 \\ B0 \\ 1F \\ C1 \end{bmatrix}$$

We are done with round functions of AES with full clarity!!

3 Key Scheduling in AES

Input : 128 bit key
Output : 11 round keys. Length of each round key is 128 bit.
Key = $(Key[15], ..., Key[0])$ (16 bytes)
We will prepare 44 words which are denoted by w[0], ..., w[43].
Size of each word is 32 bits.
44 words $\implies$ 1408 bits = 11*128 bits $\implies$ 11 round keys

Two utility functions:

- ROTWORD$(B_0, B_1, B_2, B_3) = (B_1, B_2, B_3, B_0)$
- SUBWORD$(B_0, B_1, B_2, B_3) = (B'_0, B'_1, B'_2, B'_3)$
  where, B'_i = SUBBYTES$(B_i)\ \forall\ i = 0, ..., 3$

Each B_i is 1 byte.

10 round constants (word) : (Represented in hexadecimal)

- RCon[1] = 01000000
- RCon[2] = 02000000
- RCon[3] = 04000000
- RCon[4] = 08000000
- RCon[5] = 10000000
- RCon[6] = 20000000
- RCon[7] = 40000000
- RCon[8] = 80000000
- RCon[9] = 1B000000
- RCon[10]= 36000000

for i=0 to 3: (generating first 4 words)
    w[i] = concate(Key[4i], Key[4i+1], Key[4i+2], Key[4i+3])
Each such key[] is of 1 byte.
for i=4 to 43:
    temp = w[i-1]
    if i $\equiv$ 0 mod4
        temp = SUBWORD(ROTWORD(temp)) $\oplus$ RCon[i/4]
    w[i] = w[i-4] $\oplus$ temp
return (w[0], ..., w[43])

Round Keys $K_1,\ K_2,\ ....\ , K_{11}$:
K_1 = w[0] || w[1] || w[2] || w[3]
K_2 = w[4] || w[5] || w[6] || w[7]
.
.
.
.
K_{11} = w[40] || w[41] || w[42] || w[43]

4 For Decryption of AES

4.1 Inverse of Subbyte

Subbyte(A) = B, A = X||Y
X = row number and Y = column number of 16 x 16 subbyte table.
For inverse we need to find A corresponding to a B.
Loop through the whole table to locate B in the table (atmost 256 iteration it will take). Now we can get X and Y from the location.

4.2 Inverse Shift Rows

Right rotate i^th row by i unit.

4.3 MixColumn Inverse

$MixCol(S) = S^{'} = M * S$

Where, M is $\begin{bmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{bmatrix}$

Proven result:
MixCol(MixCol(MixCol(MixCol(S)))) = $M^4 * S = I * S$
$M^4 = I$
So, in inverse function of MixCol we can apply MixCol function thrice recursively.

5 Modes of Operation

5.1 ECB (Electronic Code Block mode)

Encryption algorithms like AES and DES cannot encrypt an input of size beyond some maximum size, at a time. So, we can divide our input (or plaintext) in blocks and apply encryption on each block separately. This is what done under ECB mode. Under ECB each block is encrypted independently. But, is it safe?
Problem occurs when we have 2 or more similar blocks in our input. For e.g., while encrypting an image eyes of a person end up in different blocks then cipher of them will become same. This will reveal information about the original image. Pattern of plaintext get revealed. Although the cryptography algorithm is very strong and secure, the way we are using the algorithm creating vulnerabilities.

5.2 CFB (Cipher Feedback mode)

CFB mode also generates a keystream for use in a stream cipher, but this time the resulting stream cipher is asynchronous. We start with $y_0 = IV$ (an initialization vector) and we produce the keystream element z_i by encrypting the previous ciphertext block. That is,

$$z_i = e_K(y_{i-1}),$$

for all $i \geq 1$. As in OFB mode, we encrypt using the formula

$$y_i = x_i \oplus z_i,$$

for all $i \geq 1$. Again, the encryption function e_K is used for both encryption and decryption in CFB mode.

5.3 CBC (Cipher Block Chaining mode) [Most used]

$$M = m_1||m_2||....||m_t$$

Initialization vector = IV (Public variable)
Enc $\rightarrow$ block size l.
len$(m_i) = l$, len(IV) $= l$

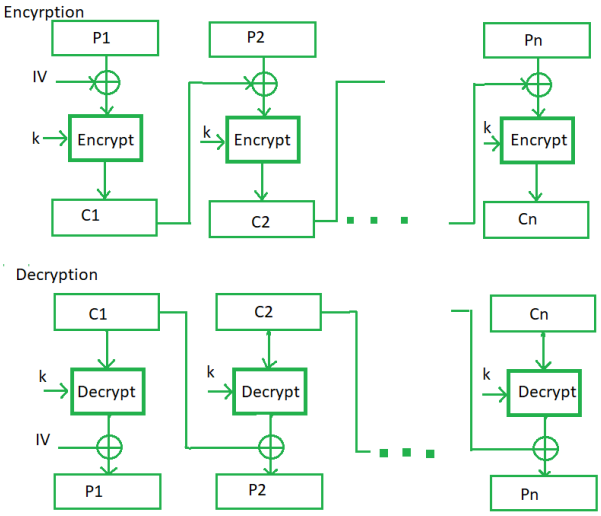
5.3.1 Encryption

$C_0 =$ IV
$C_i = Enc(C_{i-1} \oplus m_i, K)$ $\forall$ i=1, ..., t

Ciphertext:
$C = C_0||C_1||C_2....||C_t$

5.3.2 Decryption

$m_i = Dec(C_i, K) \oplus C_{i-1}$ $\forall$ i=1,....,t



Source: GeeksforGeeks

5.4 OFB (Output Feedback mode)

In OFB mode, a keystream is generated, which is then x-ored with the plaintext. OFB mode is actually a synchronous stream cipher: the keystream is produced by repeatedly encrypting an initialization vector, IV. We define $z_0 = IV$, and then compute the keystream $z_1 z_2 \ldots$ using the rule

$$z_i = e_K(z_{i-1}),$$

for all $i \geq 1$. The plaintext sequence $x_1 x_2 \ldots$ is then encrypted by computing

$$y_i = x_i \oplus z_i,$$

for all $i \geq 1$.

Decryption is straightforward. First, recompute the keystream $z_1 z_2 \ldots$, and then compute

$$x_i = y_i \oplus z_i$$

, for all $i \geq 1$. Note that the encryption function e_K is used for both encryption and decryption in OFB mode.

5.5 Counter mode

Counter mode is similar to OFB mode; the only difference is in how the keystream is constructed. Suppose that the length of a plaintext block is denoted by m. In counter mode, we choose a *counter*, denoted *ctr*, which is a bitstring of length m. Then we construct a sequence of bitstrings of length m, denoted $T_1, T_2, \ldots$, defined as follows:

$$T_i = ctr + i - 1 mod 2^m$$

for all $i \geq 1$. Then we encrypt the plaintext blocks $x_1, x_2, \ldots$ by computing

$$y_i = xi \oplus e_K(T_i),$$

for all $i \geq 1$. Observe that the keystream in counter mode is obtained by encrypting the sequence of counters using the key K.

As in the case of OFB mode, the keystream in counter mode can be constructed independently of the plaintext. However, in counter mode, there is no need to iteratively compute a sequence of encryptions; each keystream element $e_K(T_i)$ can be computed independently of any other keystream element. (In contrast, OFB mode requires one to compute z_{i-1} prior to computing z_i.) This feature of counter mode permits very efficient implementations in software or hardware by exploiting opportunities for parallelism.

5.6 CCM (Counter with Cipher block Chaining Mode)

Basically, CCM mode combines the use of counter mode (for encryption) with CBC-mode (for authentication).

6 Hash Function

$$h : A \to B$$

h(X) = Y

- If X is altered to $X^{'}$ then h($X^{'}$) will be completely different from h
- Given Y it is practically infeasible to find X s.t. h(X) = Y
- Given X and Y = h(X) it is practically infeasible to find $X^{'}$ s.t. h(X) = h($X^{'}$)

Consider the conversation b/w two person (Savitri and Max) :

Savitri **Max**

X=E(M,K) —— X ——→ X_1= Dec($\tilde{X}$, K)

S_1=h(M,K) —— S_1 ——→ S_2= h(X_1, K)

Where,
X is the correct cipher (sent by Savitri to Max)
$\tilde{X}$ is the cipher received by the Max (which may or may not be altered/sent by the Savitri)
X_1 is the message which Max got after decryption
S_1 is the value Savitri got by applying hash function on her message. It is also sent to the Max.
h is some hash function
S_2 is the value Max got by applying same hash function on the X_1

If S_2 is found not to be equal to S_1, then it means X_1 is not equal to X $\implies$ the message which Max received is not sent by the Savitri. (Authentication)

6.1 Defining

A hash family is a four tuple (P, S, K, H) where the following conditions are satisfied.

1. P is the set of all possible messages
2. S is the set of all possible message digests or authentication tags
3. K is the key space
4. H is the set of all hash functions

5. For each $K_1 \in$ K, there is a hash function $h_{K_1} \in$ H s.t.

$$h_{K_1} : P \to S$$

Here, $|P| \geq |S|$
more interestingly, $|P| \geq 2 * |S|$

- If key is involved in the computation of hashed value then that hash function is known as **Keyed hash function**.
- If key is not required to compute the hashed value then that hash function is known as **Unkeyed hash function**.
In general, we will talk about Unkeyed hash function.

6.2 Problem 1

Given $y \in S$, find $x \in P$ such that $h(x) = y$
This problem is known as **preimage finding problem**.
For an hash function h if you cannot find preimage in a feasible time then h is known as **preimage resistant hash function**.
Finding preimage is computationally hard for preimage resistant hash function.

6.3 Problem 2

Given $x \in P$ find $x^{'} \in P$ s.t. $x^{'} \neq x$ and $h(x^{'}) = h(x)$.
The problem is known as **Second pre-image finding problem**.
If finding second preimage is computationally hard for h then h is known as **Second preimage resistant hash function**.

6.4 Problem 3

Find $x, x^{'} \in P$ s.t. $x \neq x^{'}$ and $h(x) = h(x^{'})$
This problem is known as **Collision finding problem**.
For an hash function h if finding collision is computationally hard then h is known as collision resistant hash function.

6.5 Ideal Hash function

h will be called ideal hash function if given $x \in P$ to find h(x) either you have to apply h on x or you have to look into the table corresponding to h (hash table).

6.6 Pre-image finding Algo

$$h : X \to Y; \qquad |Y| = M$$

Given $y \in Y$ find $x \in X$ s.t. $h(x) = y$

Choose any $X_0 \subseteq X$ s.t. $|X_0| = Q$
for each $x \in X_0$:
    compute $y_x = h(x)$
    if $y_x = y$

return x

$X_0 = \{x_1, x_2, ..., x_Q\}$
E_i : event $h(x_i) = y$; $1 \leq i \leq Q$
$Pr[E_i] = \frac{1}{M}$

Reasoning: Favourable case(s): h(x) = given y; All cases: h(x) can take any value out of m values in set Y.

$Pr[E_i^c] = 1 - \frac{1}{M}$

$Pr[success] = Pr[E_1 \cup E_2 \cup E_3 \cup .... \cup E_Q] = 1 - Pr[E_1^c \cap E_2^c \cap E_3^c \cap .... \cap E_Q^c]$

Failure of one event is independent of the other events, so

$$Pr[success] = 1 - \Pi_{i=1}^{Q} Pr[E_i^c] = 1 - \left(1 - \frac{1}{M}\right)^Q$$
$$= 1 - \left[1 - \binom{Q}{1}\frac{1}{M} + \binom{Q}{2}\frac{1}{M^2} - \ ....\ \right]$$
$$\approx \left[1 - \binom{Q}{1}\frac{1}{M}\right]$$
$$= \frac{Q}{M}$$

$\boxed{Pr[Preimage\ finding] \approx \frac{Q}{M}}$

Complexity of finding preimage = $\mathcal{O}(M)$