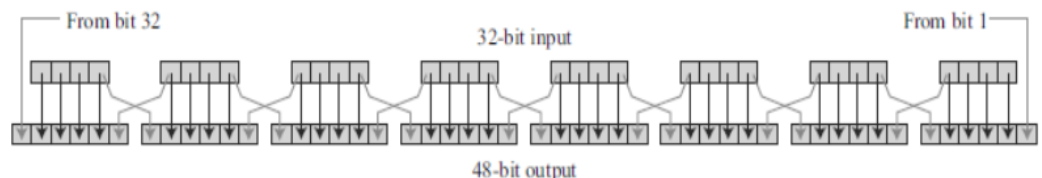


# Information Security

## 1) DES::

- a) DES key generation: parity drop to 56 bit then dividing it into 2 parts of 28 bits each applying circular left shift to each part adding them again and feeding them to compression p box to form a 48 bit key
- b) DES function: expansion of the right side plaintext from 32 to 48 bits using p box and then xoring with the 48 bit key of which the output is then fed to an s box to make it a 32 bit text which is in turn fed to a straight p box.
- c) After all this is done the final text from above is fed to the xor again with the left side of the plaintext and then the output (32 bit) is swapped and sent to the right while the right side plaintext (which hasn't undergone any functions or such) goes to the left side.
- d) Points to remember:
  - i) The last round(16th) doesn't have a swapper so the right side stays the right side and the left side remains the left.
  - ii) Initial permutation box and final permutation box are used before any round takes place and after every round takes place respectively.
  - iii) In the Expansion P-Box (the rule mentioned in point B), this is the expansion of the right side of the plaintext, the 8 4-bit parts are made into 8 6-bit parts by feeding the last bit of the 4-bit parts to the second box's 1st bit of the 6-bit part.



- iv) S-Box: The 6 bit input can be converted to 4 bit by using the S-Box the first and the sixth bit will give the row number and the middle bits will give the column number for getting the value which is to be substituted.
- v) Properties of DES include: Avalanche and completeness.
- vi) Weakness:
  - (1) Cipher design: S-Box and P-Box
  - (2) Weakness in key size
    - (a) Key size:  $2^{56}$  possible combinations to check which is kinda ez.
    - (b) Weak keys: 4 definite weak and 48 possible weak.
    - (c) Semi weak keys: 6 key pairs or 12 keys.

DES decryption: Just remember that the last key of encryption is used as the first key of decryption and so on. Rest of the steps are exactly the same as that of encryption.

# AES

$(x^8 + x^4 + x^3 + x + 1)$ ---- irreducible polynomial galva field 2

01. **AES has 3 types:** 128 bit with 10 rounds, 192 bit with 12 rounds, 256 bit with 14 rounds.

- a. There is a preroundkey which happens before any of the rounds actually happen so the key is used 11 times in the AES algorithm, in this round the plaintext is exord with the untouched cipherkey to generate the first state for the first round.

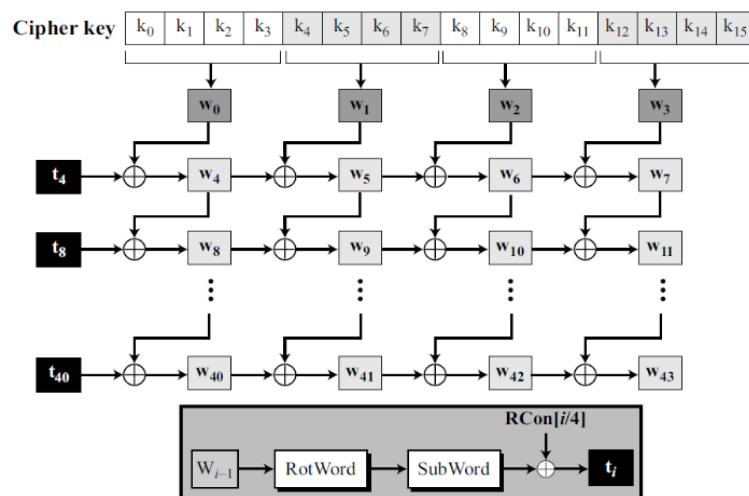
02. **Data types:**

- a. A **byte** is an 8 bit binary number represented mostly in hexadecimal so basically two digits.
- b. A **word** is 4 bytes or 32 bits which are represented as a column in the state.
- c. A **state** is a 16 byte or 4 word or 128 bit 4x4 matrix which is used in all of the transformations that are done in the rounds.

03. **Rounds details:**

- a. **SubBytes:** SubBytes is where substitution bytes are made for the already existing state, the unit digit is the row number and the other digit is the column number in the subbyte matrix which is 16x16 in size to accommodate for 256 bit AES as well.
  - i. If a question is asked to convert the byte to subbyte then we will need to perform euler's division thing which sets  $a_1 = 1$ ,  $a_2 = 0$ ,  $a_3 = m(x)$ ;  $b_1 = 0$ ,  $b_2 = 1$ ,  $b_3 = p(x)$ , where  $m(x)$  is the **irreducible polynomial** and  $p(x)$  is the polynomial form of the byte to be converted.
  - ii. After this we need to perform  $a_3/b_3$  and then find  $t_1$ ,  $t_2$  and  $t_3$  which can be found by the formula  $a_1 + (\text{Quotient} * b_1)$  for  $t_1$  and so on, remember that if this results in a polynomial with degree greater than 7 then we need to divide it by the **irreducible polynomial** (remember that all pluses in this are ex or operations).
  - iii. After all this for the next iteration we shift the value of  $b_1$ ,  $b_2$ ,  $b_3$  from the previous iteration onto  $a_1, a_2, a_3$  in the current round and set the value of  $b_1$ ,  $b_2$ ,  $b_3$  for the current round as  $t_1$ ,  $t_2$ ,  $t_3$  of the previous round.
  - iv. This continues till we get 0 or 1 as the remainder of the long division. After we get 1 or 0 we take the  $t_2$  and convert it into polynomial form and then binary form which we multiply with the 8x8 matrix with first row as 5 1's and 3 0's which are circularly shifted down till the 8th column is reached.
  - v. After this we multiply the final answer with a predetermined 8x1 matrix (63 in hexadecimal) to find get the final answer.
- b. **ShiftRows:** The first row of the state is not shifted, the second row is shifted one time, the third row is shifted two times, and the fourth row is shifted three times, all these shifts are left circular shifts that shift the two bytes present in that place on the 4x4 matrix.

- c. **Mixed Columns:** The columns of the state are then multiplied to a constant 4x4 matrix which mixes up the values in all of the columns.
- Remember that all the calculations need to be carried out in the Galva field of 2, which means that if the polynomial which results after the multiplication of the two bytes has a degree greater than 7 then it needs to be divided by  $(x^8 + x^4 + x^3 + x + 1)$ .
- d. **AddRoundKey:**
- The key is numbered from  $w_0, \dots, w_{(Nr+1)4}$  so the first key is from  $w_0$  to  $w_3$ . Where  $w$  is a word.
  - The addround key is where the key generation takes place, we calculate the key for the rounds as follows:
    - if  $(i \bmod 4) \neq 0$ ,  $w_i = w_{i-1} + w_{i-4}$
    - if  $(i \bmod 4) = 0$ ,  $w_i = t + w_{i-4}$
    - $T$  is a temporary word which is found by calculating  $t = \text{subword}(\text{rotword}(w_{i-1})) + R_{\text{con}}[i/4]$
    - Subword operation is the same as subbyte operation so refer the subbyte matrix for it.
    - Rotword is where the 4 bytes are shifted left by 1 byte circularly.
    - $R_{\text{con}}$  is a constant matrix that is provided.
  - After the key is generated like this it is XORed with the state and one round is completed.



Decryption side has something like this:

### ◆ AES Decryption Steps

For decryption, you apply the **inverse transformations in reverse order**:

- **AddRoundKey** → (same as encryption, XOR is its own inverse)
- **InvMixColumns** → (inverse of MixColumns)
- **InvShiftRows** → (shift right instead of left)
- **InvSubBytes** → (inverse S-box substitution)

## Security goals:

1. Confidentiality
2. Integrity
3. Availability

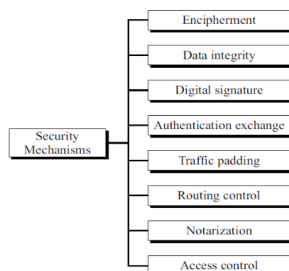
## Attacks:

1. **Threat to Confidentiality also are passive attacks:**
  - a. Snooping: Interception of data
  - b. Traffic Analysis: Monitoring online traffic
2. **Threat to Integrity:**
  - a. Modification: Intercepting information and then modifying the information to make it beneficial for the attacker.
  - b. Masquerading: or spoofing, happens when the attacker impersonates someone
  - c. Replaying: obtains a copy of a message sent by a user and tries to replay it.
  - d. Repudiation: sender of the message denies that she has sent the message or receiver denies that he received it.
3. **Threat to availability:**
  - a. Denial of Service: send so many bogus requests to the server that it crashes leading to denial of service or some other similar method.

## Security Services:

1. **Data Confidentiality**
2. **Data Integrity:**
  - a. Anti change
  - b. Anti replay
3. **Authentication:**
  - a. Peer entity
  - b. Data Origin
4. **No repudiation:**
  - a. Proof of origin
  - b. Proof of delivery
5. **Access Control**

## Security mechanisms:



# Asymmetric Ciphers

## 1. RSA:

### a. Process:

- i. Two exponents: **e**(public) and **d**(private)
- ii. Public key: e, n
- iii. P- plaintext C- Ciphertext
- iv.  $C = P^e \bmod n$  and  $P = C^d \bmod n$
- v. p and q are large primes and not equal to each other,  $n = p * q$ .
- vi. We calculate  $\phi(n) = (p-1)(q-1)$
- vii. p, q, phi and d are secret. Once the ciphertext is generated we can get rid of p, q and phi.
- viii. If the value of d is to be found, we use extended Euclidean method from subBytes in AES, just change the values of  $a_3 = \phi(n)$  and  $b_3 = e$ .

## 2. Rabin:

- a. Private key: p, q
- b. Public key: n
- c. d is set as 1/2 and e is set as 2
- d.  $C = P^2 \bmod n$  and  $P = C^{1/2} \bmod n$
- e. Needs to keep p and q
- f. p and q are set in the form of  $4k+3$

## 3. ElGamal:

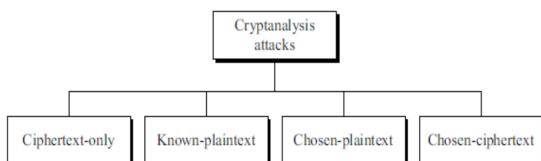
- a. Private key: d
- b. Public key:  $e_1, e_2, p$  : where p is a large prime and  $e_2 = (e_1)^d \bmod p$
- c.  $C_1 = (e_1)^r \bmod p$  and  $C_2 = ((e_2)^r * P) \bmod p$  where P is the plaintext.
- d.  $P = [C_2 * (C_1^d)^{-1}] \bmod p$ , we can use  $P = [C_2 * C_1^{p-1-d}] \bmod p$  to avoid inverse calculation.

## 4. Diffie-Hellman Key Exchange:

- a. a is primitive root of q where: q is a large prime.
- b.  $X_a$  and  $X_b$  are private components.
- c.  $Y_a$  and  $Y_b$  are public components.
- d.  $Y_a = a^{X_a} \bmod q$  and  $Y_b = a^{X_b} \bmod q$ .
- e. So  $X_a$  is a component that the receiver that gave  $Y_a$  has and vice versa for calculation.
- f. Shared session key:
  - i. At a's end:  $K_a = Y_b^{X_a} \bmod q$
  - ii. At b's end:  $K_b = Y_a^{X_b} \bmod q$
  - iii. Note that  $K_a$  and  $K_b$  are same.

## Categories of Traditional Ciphers

4 common types of cryptanalysis attacks.



Two broad categories: **substitution ciphers** and **transposition ciphers**.

**substitution cipher:** Replace one symbol in the ciphertext with another symbol;

**transposition cipher:** Reorder the position of symbols in the plaintext.

Caesar cipher or additive cipher:  $C = (P + k) \bmod 26$  and  $P = (C - k) \bmod 26$

Multiplicative cipher:  $C = (P * k) \bmod 26$  and  $P = (C * k^{-1}) \bmod 26$ , here the key should be co prime of 26.

Affine cipher: combo of multiplicative and additive, meaning first multiplication then addition.

## Cryptanalysis of Affine Cipher

Using chosen-plaintext attack.

Algorithm 1:	Plaintext: et	ciphertext: → WC
Algorithm 2:	Plaintext: et	ciphertext: → WF

e → W	04 → 22
t → C	19 → 02

e → W	04 → 22	$(04 \times k_1 + k_2) \equiv 22 \pmod{26}$
t → C	19 → 02	$(19 \times k_1 + k_2) \equiv 02 \pmod{26}$

$$\begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} 4 & 1 \\ 19 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 22 \\ 2 \end{bmatrix} = \begin{bmatrix} 19 & 7 \\ 3 & 24 \end{bmatrix} \begin{bmatrix} 22 \\ 2 \end{bmatrix} = \begin{bmatrix} 16 \\ 10 \end{bmatrix} \longrightarrow k_1 = 16 \quad k_2 = 10$$

## Autokey Cipher

$$P = P_1 P_2 P_3 \dots$$

$$C = C_1 C_2 C_3 \dots$$

$$k = (k_1, P_1, P_2, \dots)$$

$$\text{Encryption: } C_i = (P_i + k_i) \bmod 26$$

$$\text{Decryption: } P_i = (C_i - k_i) \bmod 26$$

**Playfair Cipher (contd)**

The cipher uses three rules for encryption:

If the two letters in a pair

are located in the same row of the secret key	corresponding encrypted character for each letter is the next letter to the right in the same row  (with wrapping to the beginning of the row if the plaintext letter is the last character in the row).
are located in the same column of the secret key	corresponding encrypted character for each letter is the letter beneath it in the same column  (with wrapping to the beginning of the column if the plaintext letter is the last character in the column).
are not in the same row or column of the secret	corresponding encrypted character for each letter is a letter that is in its own row but in the same column as the other letter