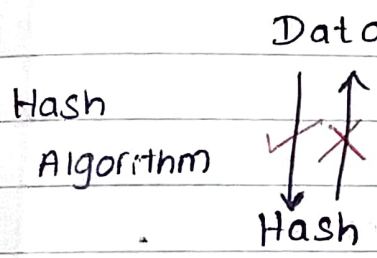
# Hash functions and Message Authentication Code.

## Hash functions

Condense arbitrary sized message to a fixed size.
Can be used to check (detect) changes to messages
Integrity Checking.
Often used with password, digital signatures, cryptocurrency, blockchain.

Hash collisions - when more than one message corresponds to same hash value.

Hash functions are irreversible / one-way algorithm.




Length of hash depends on algorithm.
Purpose of hash function is to produce a fingerprint.
Simplest Hash function - XOR

Properties

$h = H(M)$ h - hash
H - hashing algorithm
M - Message

- Can be applied to any sized message M.
- Produces fixed length hash
- Easy to compute $h = H(M)$ for any message M.

- Given $h$ it is infeasible to find $x$ s.t. $H(x) = h$
  one-way property
- Given $x$ it is infeasible to find $y$ s.t. $H(x) = H(y)$
  weak collision resistance.

It is infeasible find any $x, y$ s.t. $H(x) = H(y)$
strong collision resistance

## MD5

Message Digest 5

1. Pad msg so its length is 448 mod 512 (The length is 64 bits lesser than any multiple of 512 (block size))
2. Append 64 bit length value to msg (Make it a multiple of 512)

(takes input% 512 == 0)

3. Initialize 4-word (128 bits) MD buffer (A, B, C, D) (buffers of size 32 bit each)

4. Process message in 16-word (512-bit) blocks. (Break the msg in 512 bits each chunks)
   - Each 512 block is broken down to 16 sub blocks of 32 bits using 4 rounds (4 buffers) of 16 bit operations on message block and buffer.
   - add output to buffer input to form new buffer value

* With advancements in computing, MD5 is proven to be vulnerable.

| | SHA-1 | MD5 | RIPEMD-160 |
|---|---|---|---|
| Digest length | 160 bits | 128 bits | 160 bits |

| | | |
|---|---|---|
| linux | md5sum | sha1sum |
| mac | md5 | shasum |

} Hash of files

md5deep / shadeep can be used to get the hash value of dir

Application

- for storing passwords.

Passwords are not saved in plain text but as a hashed valu
weak passwords and dictionary attacks.
People have calculated hash values of common passwords / words
so that when they come across a hash value, they can see
it they can find that hash value in that precomputed
table. If it is found, then the password is compromised.

Public salt -

When setting a password, pick a random n-bit salt s.
When verifying pw, test if $H(pw, s_A) = h_A$

Recommended salt length: 64 bits.

PW Table

| id | Salt | h |
|---|---|---|
| Alice | $s_A$ | $H(pw_A, s_A)$ |
| Bob | $s_B$ | $H(pw_B, s_B)$ |

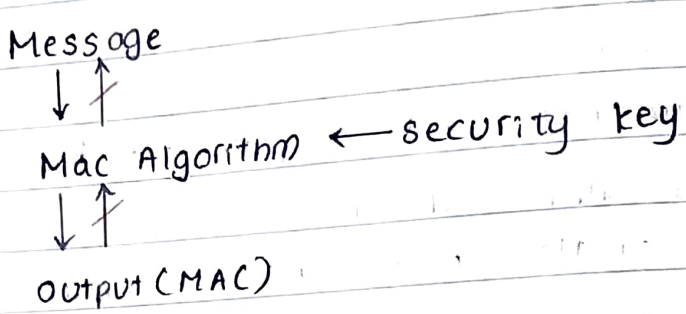
- Authenticate evidence

At the time they acquire evidence, they have to hash
the files in front of security people.
The signature is the hash.
Alterations could be proven.

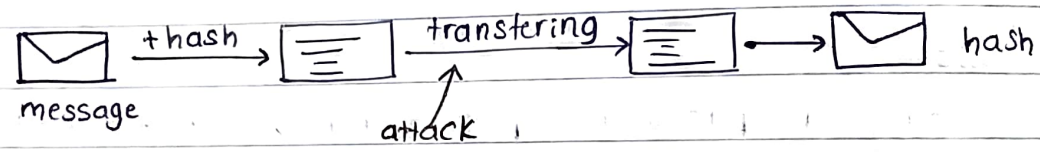
echo -n test123 | sha1sum

-n removes newline

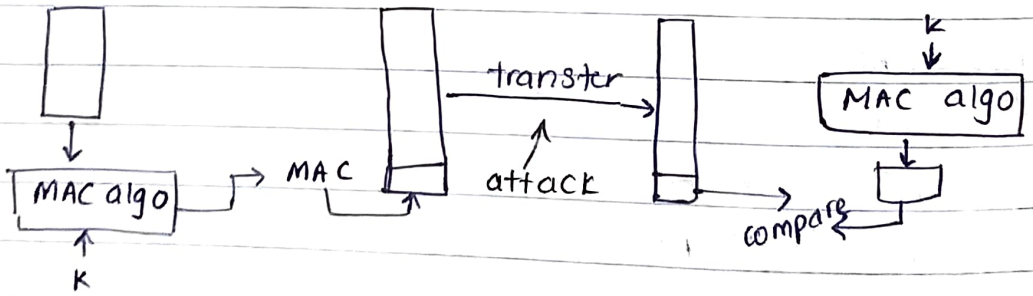
## Message Authentication Code (MAC)




When sending messages over a network, if hashing was used.




The attacker can alter both the hash and message so it gives the expected output




The attacker does not have the key. Therefore, even the message is altered, the MAC cannot be altered as expected.

The security key should be shared between the sender and receiver

$$MAC = F(K, M)$$

can convert a hash function to a MAC algorithm.

Original proposal was,
hashed key = hash(key|Msg)
weaknesses? - is the key a suffix? or prefix?

What is HMAC? Keyed Hash
To use available hash functions without modification.

$HMAC_K$ = Hash [ (K+ XOR opad) || Hash [(K+ XOR ipad) || M]]

K+ - key is repeated until it matches msg length
ipad, opad - specified padding constants.

overhead - 3 more hash functions

Security Attacks
Bruteforce attacks.
Birthday paradox / birthday attacks.
accumulate many msg+MAC pairs, wait for collision.

Implementation of a simple hash class in java

① calculate Digest
a. MessageDigest md = MessageDigest.getInstance("SHA1")
b. feed data
md.upade(buf) // buf is binary data
c. calculate digest
md.digest()

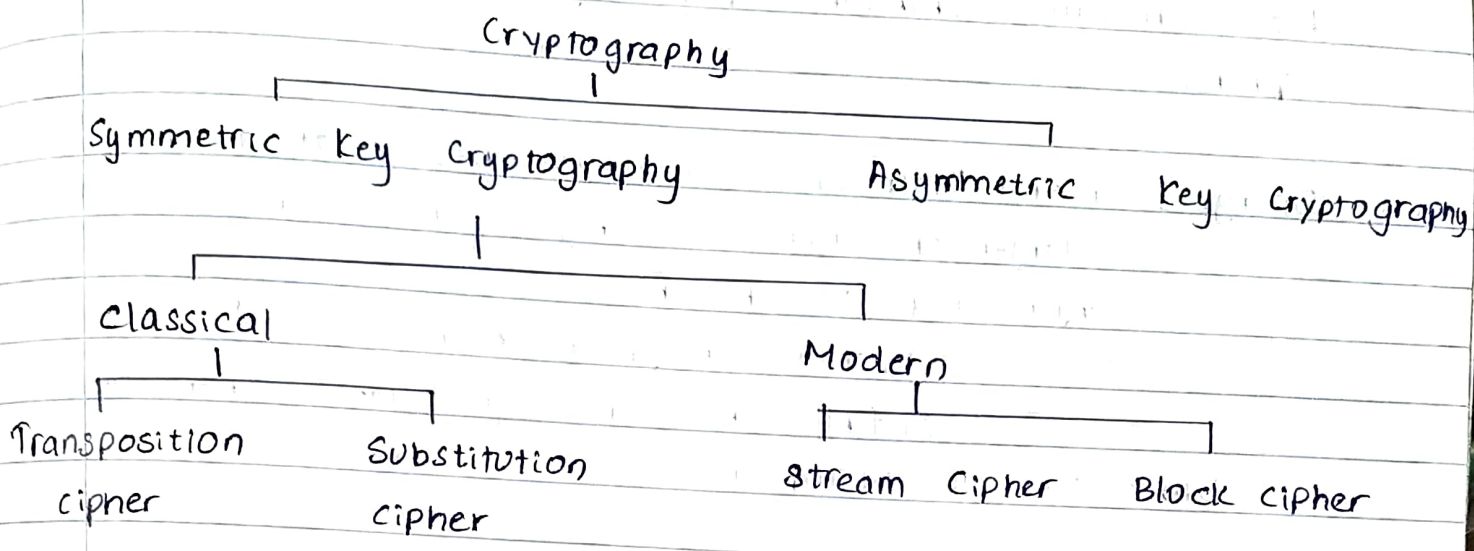
② Verify digest

Use the above steps.

Then compare with the original digest

md.isEqual(newDigest, originalDigest)

# Symmetric key




Symmetric key - same key is used for encryption and decryption.

Difficulty → key distribution

~~Exchanging the key between~~ How difficult it is to share send the key between the sender and recepient.

* Using the same network to send the encrypted data and key is vulnerable for attacks.
* Brute force - complexity increases when the key length increases.

if k is the length of key then $2^k$ possibilities of keys.

* Scalability

If there are 2 people → 1 key, pair
If there are 3 people → 3 keys, 3 pairs, triangular number
If there are 4 people → 6 keys
If there are n people → $\frac{n(n-1)}{2}$

∴ n↑ no. of keys increases exponentially

Issues with modern symmetric key cryptographic algorithms

1) Key Distribution Issue
2) Brute force issue
3) Scalability Issue

Block Ciphers

Most modern symmetric key algorithms are block ciphers.

Encrypts more than one character (block) at a time.

Block size: larger the block size, higher the security.

Key size: larger key size better

Number of rounds: internal encryption rounds

Encryption Modes: Defines how messages larger than the block size is encrypted.

Strengths of Symmetric key / Private key cryptosystems

- speed / efficient algorithms
- Hard to break when key size is large
- Ideal for bulk encryption / decryption

Weaknesses

- Poor key distribution
- Cannot provide authenticity and non-repudiation - only confidentiality
- Poor key management (Scalability)

Requirements for secure use of symmetric encryption

- Strong encryption algorithm
- Secret key

$$Y = E_K(X)$$

$$X = D_K(Y)$$

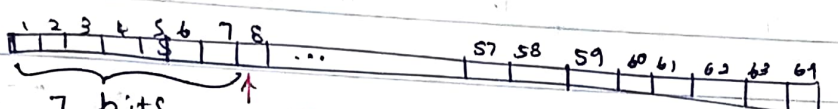
Data Encryption Standard (DES)

Block - 64 bit (8 bytes)

key - 56 bit (7 bytes) [In reality 64 bits are used. But 8 bits are used as parity check bits for error control]

16 rounds

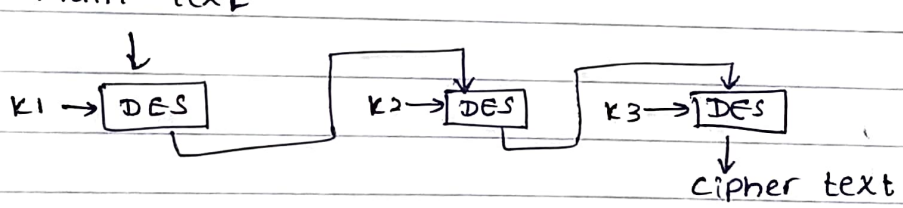
16 intermediary keys of size 48 bits.




first 7 bits → parity check bit → XOR of the previous 7 bits.

$2^{56}$ different keys

Triple DES ~~with 2 keys~~.




- 3 encryptions

$C = E_{K1}[D_{K2}[E_{K3}[P]]]$

if $K1 = K2$, then triple DES can work with single DES.

The first two will cancel out and give original text.

Why not more than 3? Encrypting shifts bits. Therefore encrypting for a large number of rounds may result in cipher text similar or close to plain text.

*using same key for 2 consecutive rounds will downgrade triple DES to DES Backward compatibility.

Triple DES with 2 keys

Then K1 = K3

Advanced Encrytion Standard

Block
Data - 128 bit

key - 128 bit / 192 bit / 256 bit (recommended key size now)

Active life of 20-30 years (+ archival use)

10 rounds (128 bit key) 12 rounds (192) 14 (256)

Decryption is not identical to encryption algo.

Padding Standards

Padding - used when data length is different from block length

~~PKCS5~~

h e l l o 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

11 bits needed : 0b x 11 (hex representation)

h e l l o w o r l d ! 0 0
0 1 2 3 4 5 6 7 8 9 10

PKCS5

Any msg not a multiple of 8 bytes is padded

1 byte needed : 0x 1 ↑ Assume block cipher of 8-bits

2 bytes needed : 0x 2 0x2

:

No padding: 0x8 0x8 0x8 0x8 0x8 0x8 0x8 0x8

* Even if the message size is a multiple of block size, a block with bytes with value of block size is added.

i.e. If block size is 8 bytes

0a 0b 0c 0d 0e 0f 02 03

padding is added

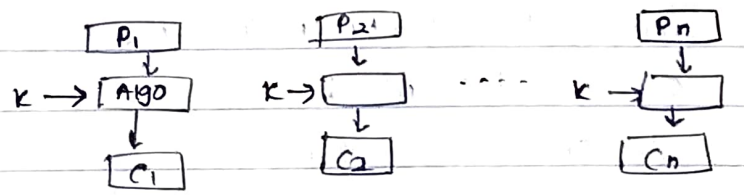
0a 0b 0c 0d 0e 0f 02 03

08 08 08 08 08 08 05 08

Block Cipher Modes of Operation

1) ECB Mode - Electronic Code Book Mode

Message is broken down into independent blocks which are encrypted.




$C_1 = DES_K(P_1)$

Recommended to encrypt small amount of data.

Eg: credit card number

ID number

Not Recommended to encrypt images.

If the same block is encountered again, an attacker can see that the cipher text is identical.

Size of block depends on encryption algo

a) CBC Mode - Cipher Block Chaining

Message is broken down into blocks

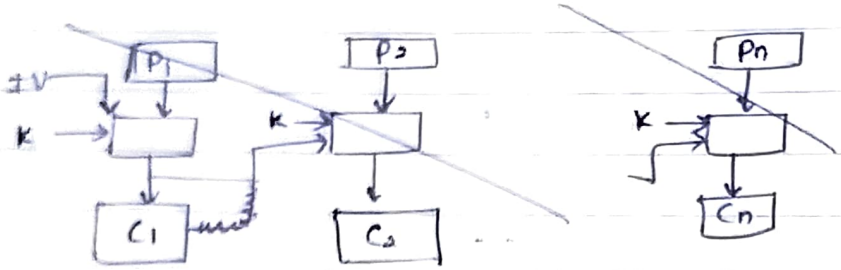
But these linked together in the encryption operation

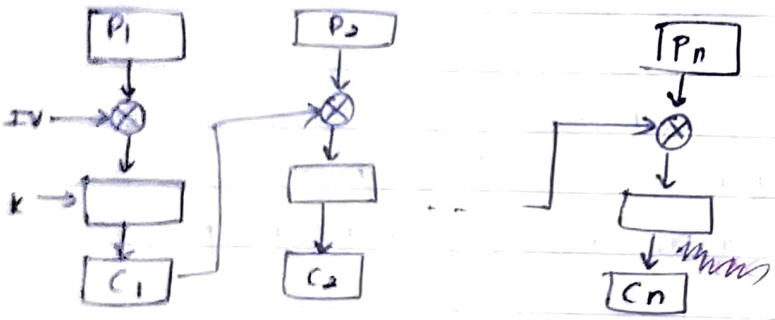
Each previous cipher block is chained with the current plaintext block.

Use initial vector (IV) to start process.

$C_i = DES_K(P_i \text{ XOR } C_{i-1})$

$C_{-1} = IV$




The above can be changed to a MAC based on CBC by using the final output of the last block

So changing one bit will change everything, as previous outputs are used in the current encryption

CBC-MAC vs CBC-Enc

CBC-Enc → Secure encryption

CBC-MAC → Secure MAC

Initialization

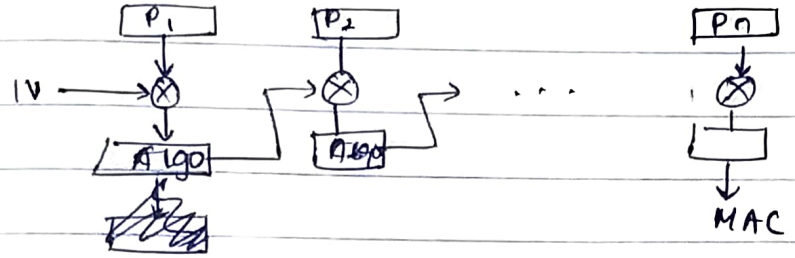
CBC-Enc → uses random IV

CBC-MAC → first block fixed at 0. random IV is insecure.

Output

CBC-Enc → outputs all intermediate blocks (to decrypt)

CBC-MAC → outputs only last block.



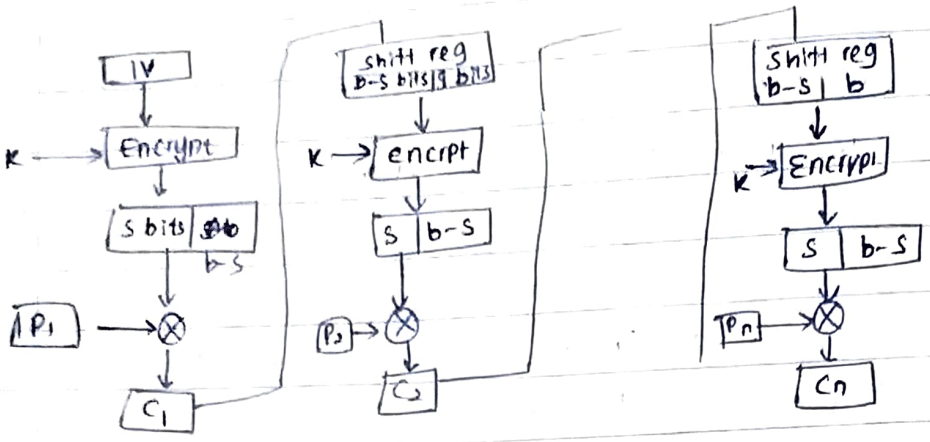

IV needs to be known by both sender and receiver. However if the IV is transmitted as a plaintext, an attacker can change the first block and change IV to compensate. Hence either IV should be fixed or it must be encrypted.

3) Cipher Feedback (CFB)

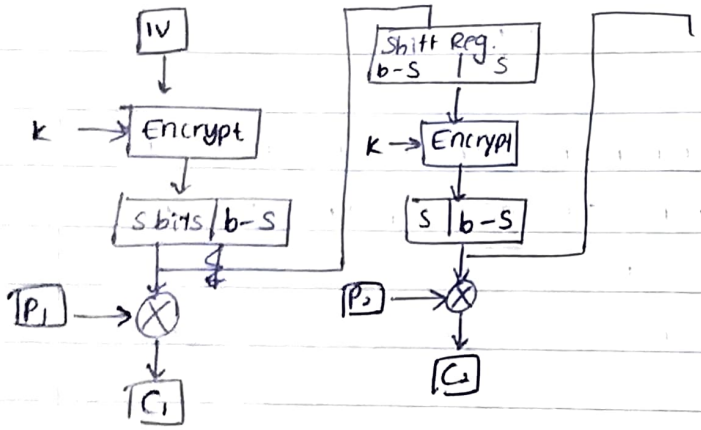
Ciphertext is used as feedback in the key generation source to develop the next key stream.

The ciphertext generated by performing an XOR on the plaintext with the key stream.

Errors will propogate.



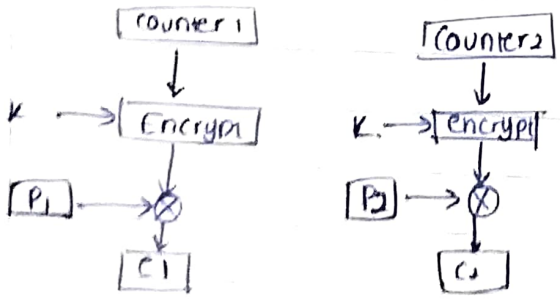

4) Output feedback (OFB) mode



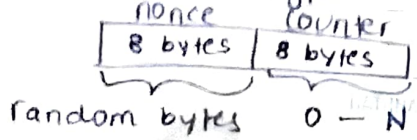

Errors will not propagate.

Suitable for when we need fast communication (with, VOLTE)

5) Counter (CTR) Mode




Counter is of 16 bytes.



Can do parallel encryptions – fast
can preprocess in advance of need.
good for high speed links
random access to encrypted data
Secure
should never reuse key/counter values.

Authenticated Encryption
combine confidentiality and integrity
Security
  Confidentiality: CCM security
  Integrity: attacker cannot create new ciphertexts that decrypt properly.

Decryption returns either
  valid messages
  Invalid symbols

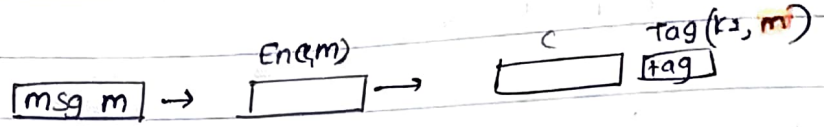
~~WPA-2~~ WPA2 - CCM
Authenticated encryption
  CTR → encryption
  CBC-MAC → check integrity

CCM → CTR + CBC-MAC for confidentiality and integrity.

Combining MAC and ENC.

1. Option 1 Enc and MAC

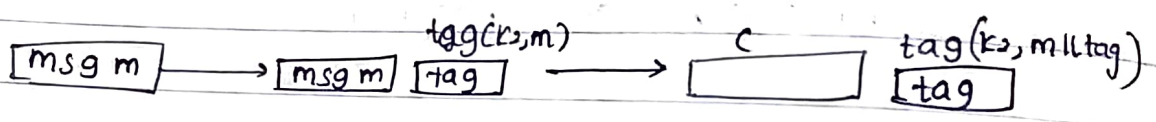
Eg: SSH




Encrypt msg. Then calculate integrity code of msg.

2. Option 2 MAC then ENC

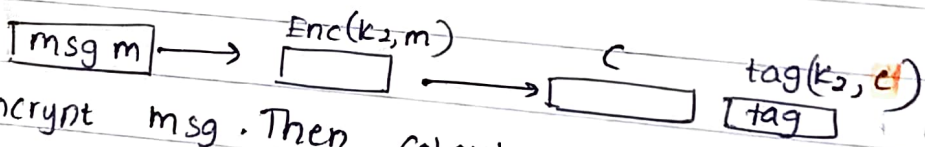
Eg: SSL




Calculate the sec. code first. Encrypt msg and the tag.

3. Option 3 Enc then MAC

Eg: IPsec




Encrypt msg. Then calculate the integrity code of ciper te

GCM mode