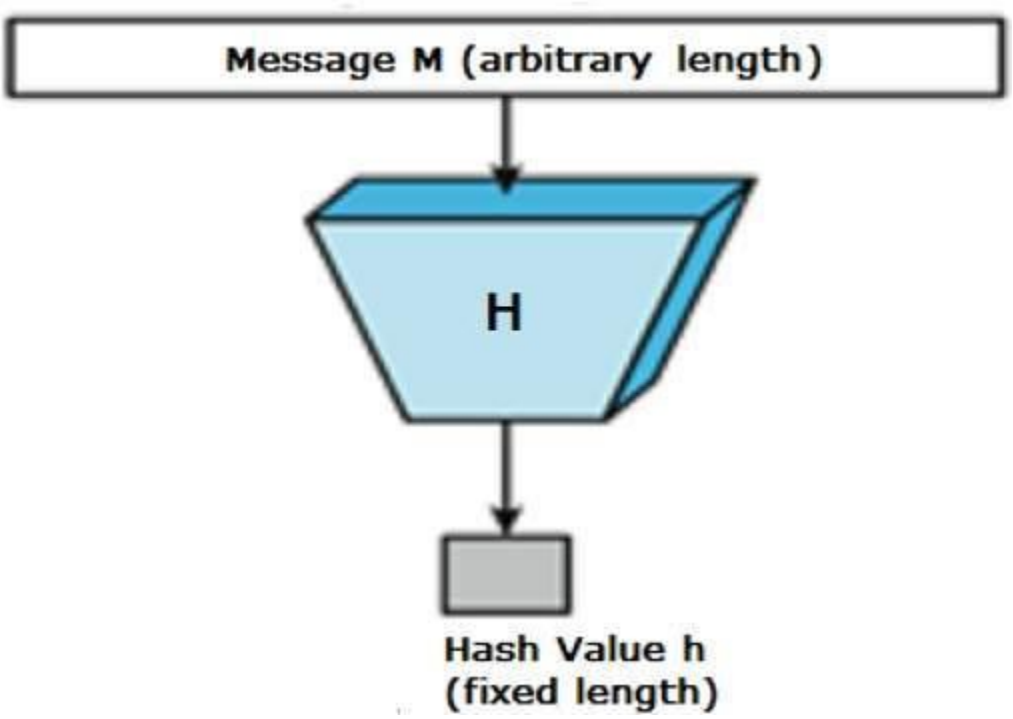
Hash Function

A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length but output is always of fixed length.Values returned by a hash function are called **message digest** or simply **hash values**. The following picture illustrated hash function



Features of Hash Functions

The typical features of hash functions are −

- **Fixed Length Output (Hash Value)**

- Hash function coverts data of arbitrary length to a fixed length. This process is often referred to as **hashing the data**.
- In general, the hash is much smaller than the input data, hence hash functions are sometimes called **compression functions**.
- Since a hash is a smaller representation of a larger data, it is also referred to as a **digest**.
- A message digest algorithm or a hash function, is a procedure that maps input data of an arbitrary length to an output of fixed length. Output is often known as hash values, hash codes, hash sums, checksums, message digest, digital fingerprint or simply hashes.

- **Efficiency of Operation**
  - Generally for any hash function h with input x, computation of h(x) is a fast operation.

- Computationally hash functions are much faster than a symmetric encryption.

Properties of a Strong Hash Algorithm:-

- **Determinism** — A hash algorithm should be **deterministic**, meaning that it always gives you an output of identical size regardless of the size of the input you started with. This means that if you're hashing a single sentence, the resulting output should be the same size as one you'd get when hashing an entire book.
- **Pre-Image Resistance** — The idea here is that a strong hash algorithm is one that's preimage resistance, meaning that it's infeasible to reverse a hash value to recover the original input plaintext message. Hence, the concept of hashes being irreversible, one-way functions.
- **Collision Resistance** — A collision occurs when two objects collide. Well, this concept carries over in cryptography with hash values. If two unique samples of input data result in identical outputs, it's known as a collision. This is bad news and means that the algorithm you're using to hash the data is broken and, therefore, insecure. Basically, the concern here is that

someone could create a malicious file with an artificial hash value that matches a genuine (safe) file and pass it off as the real thing because the signature would match. So, a good and trustworthy hashing algorithm is one that is resistant to these collisions.

- **Avalanche Effect** — What this means is that any change made to an input, no matter how small, will result in a massive change in the outputhence the term “avalanche effect.”
- **Hash Speed** — Hash algorithms should operate at a reasonable speed. In many situations, hashing algorithms should compute hash values quickly; this is considered an ideal property of a cryptographic hash function.

1.it is similar to MAC but it does not use a key.

2.it takes in variable size message and produce a fixed output called hash code/hash value/message digest.

3. The only input is the message

4. A hash value ‘h’ is generated by function H.

H(M)=fixed length code ‘h’

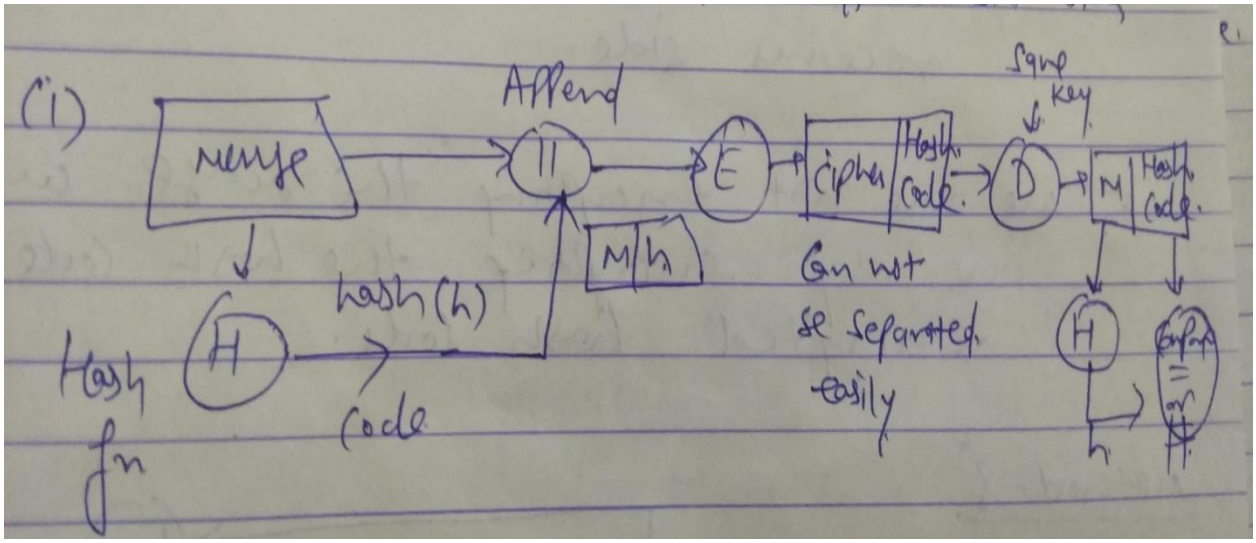
{here M-variable length msg,

‘h’-hash code}

5. it is also called compression fn.

6.There are different method to provide authentication in different situation.

(i)

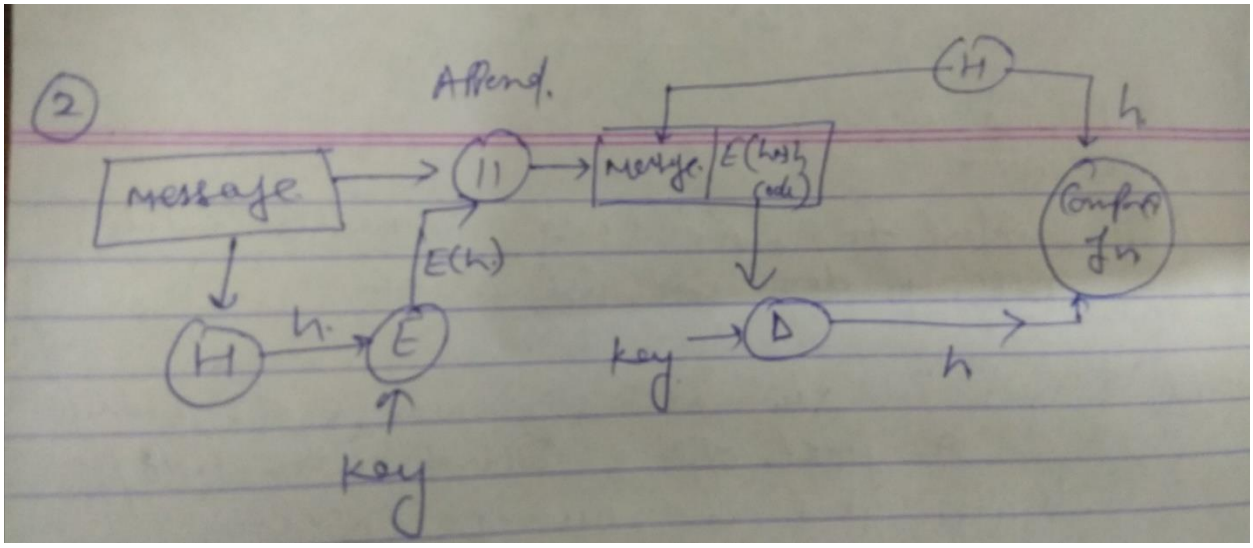


in this authentication and confidentiality maintained.

Authentication:- if both the hash code are equall in the end.because A and B share the secret key,the message must have come from A and key has not been altered.

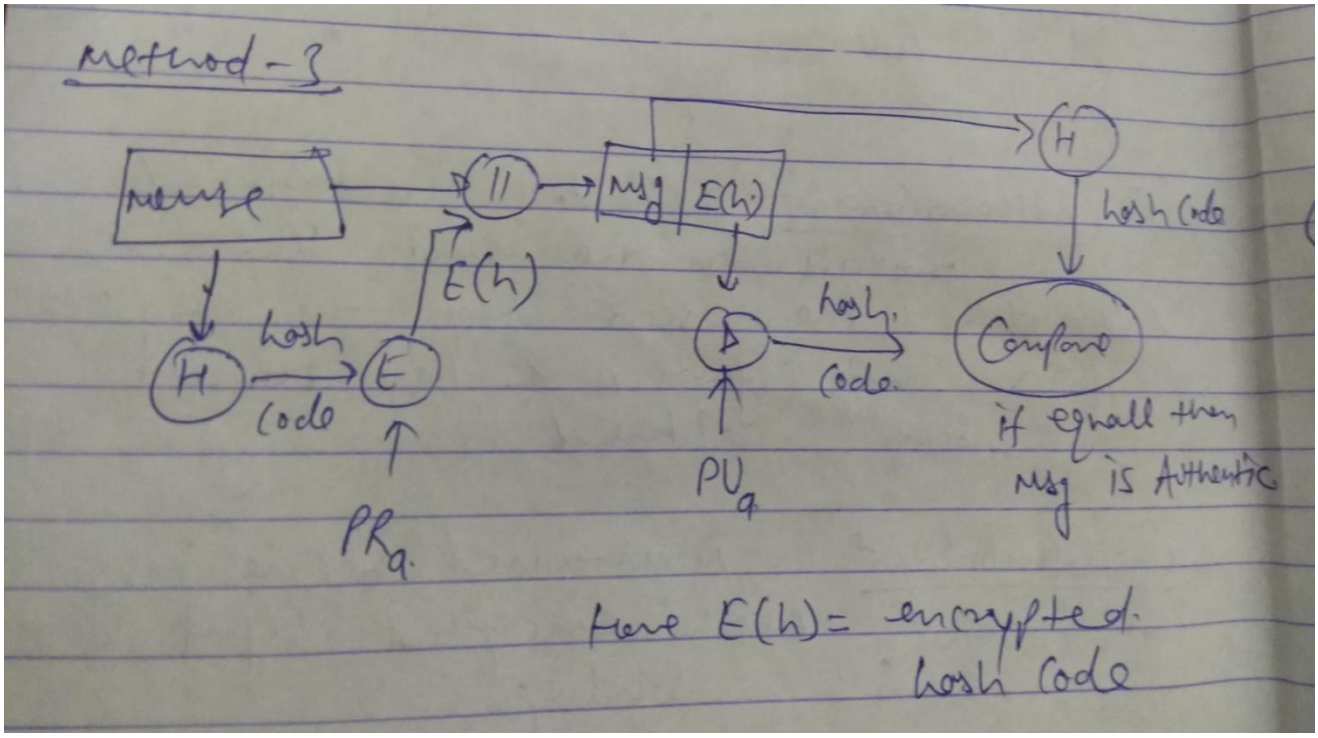
Confidentiality:-it maintained because message was encrypted before sending.

(ii)



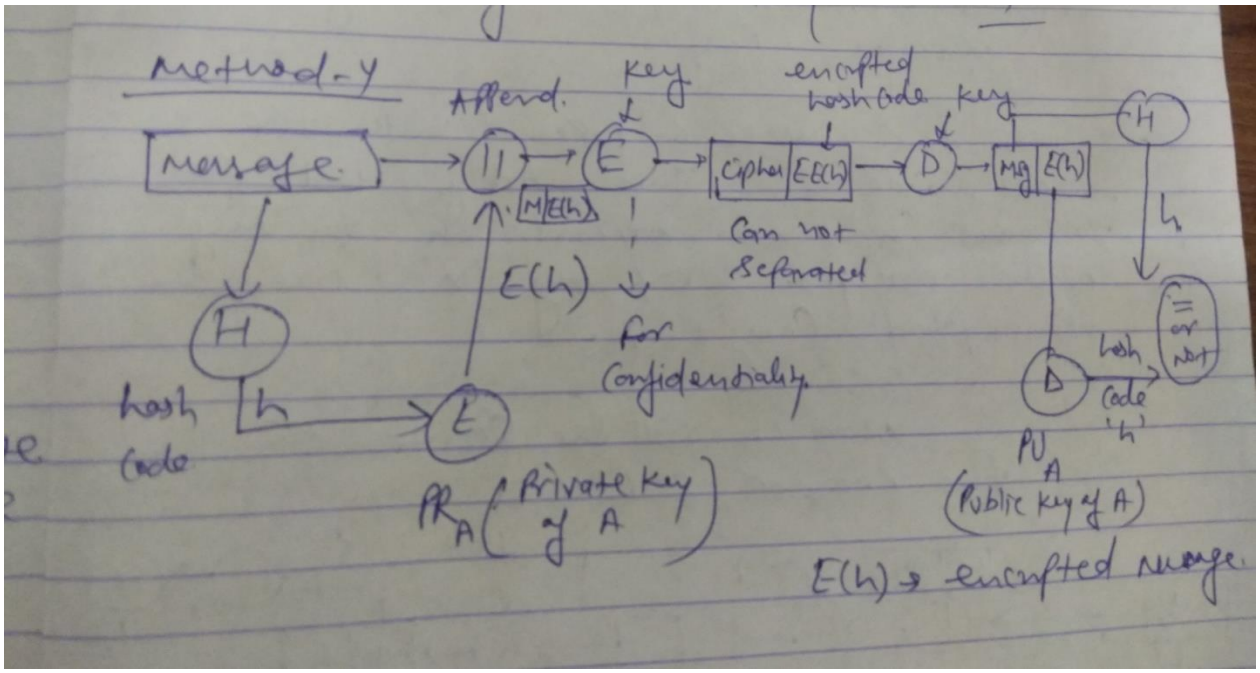
1. in this only authentication achieved when both the hash code are equall.but no confidentiality because any one can steal the message.
2. Only hash code is encrypted using symmetric encryption.
3. if messge is not private message,so can use it because the processing time will be sent to receiver side.
4. in this not encrypting the message .only encrypting hash code .

(iii)



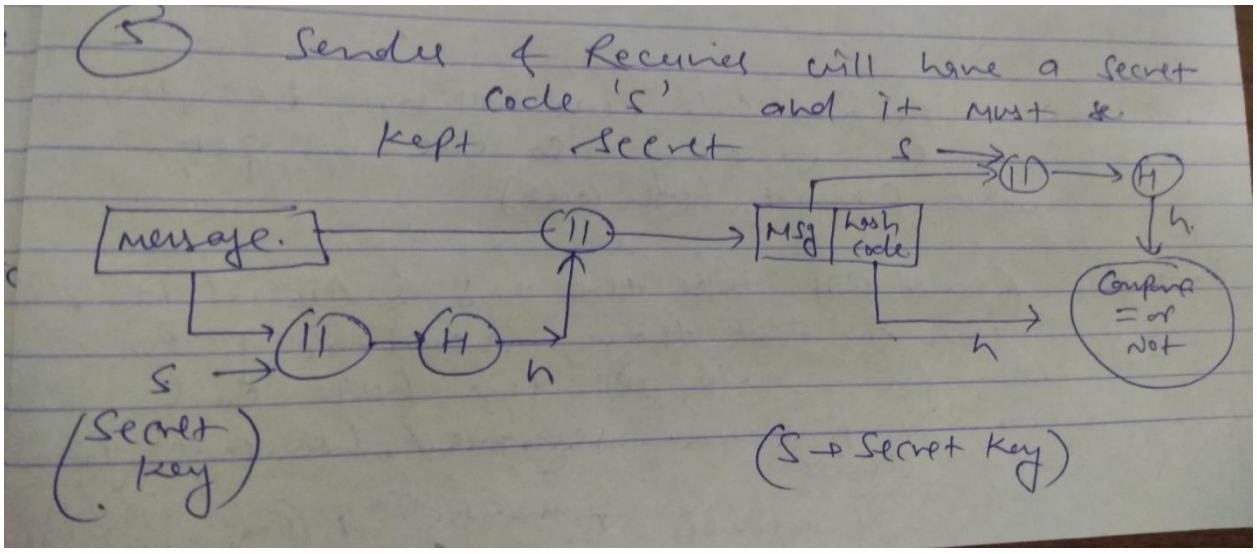
1. in this no confidentiality ,only authentication maintained.
2. processing time will be less as the message is not encrypted.
3. in this asymmetric key encryption is used.

(iv)



1. it is used when Authentication and confidentiality achieved by using symmetric key.

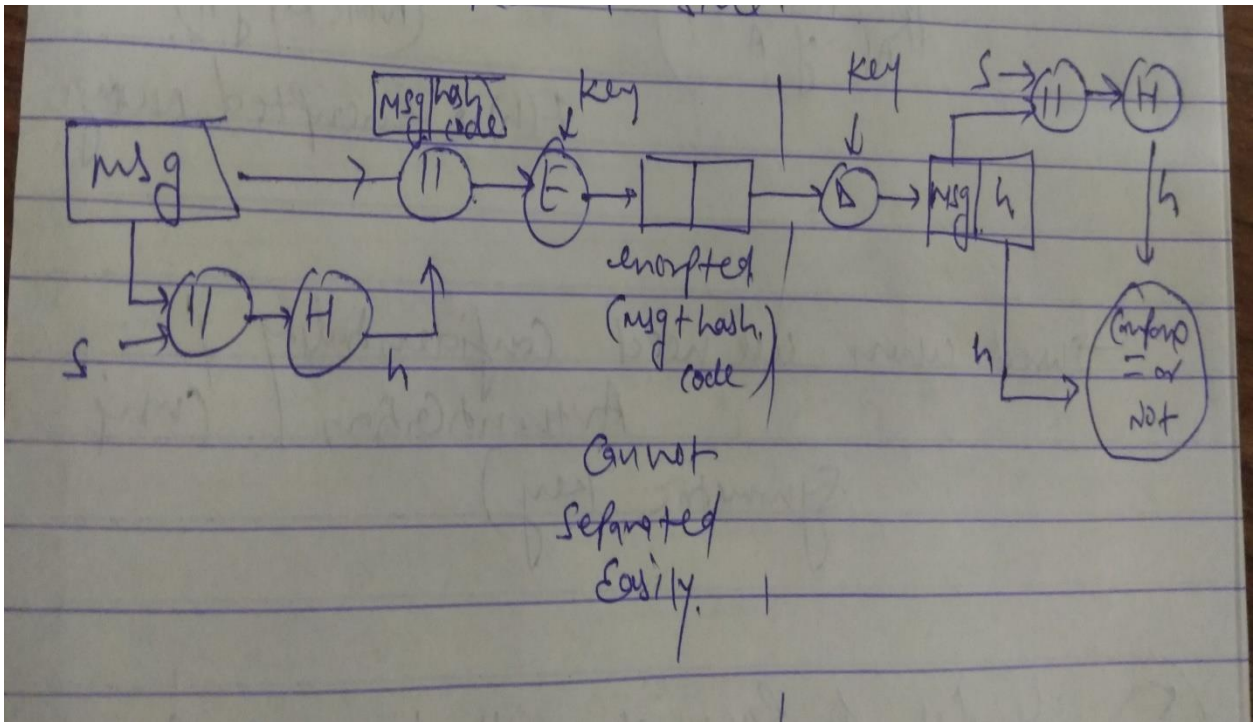
(v) Sender and Receiver will have a secret code ‘s’and it must be kept secret



1. In this only authentication ,no confidentiality achieved.
2. no confidentiality(any one can attack) and authentication when hash value are equall.

(vi)

1. in this confidentiality can be added to the previous approach by encrypting the entire message.
2. take the message and append with the secret code ‘s’ then apply Hash fn. it gives ‘h’.now append ‘h’and message .now encrypt using key ‘x’ .we get encrypted (message+h).



3. now it will be sent to the Receiver side.
4. now at receiver side we will use decryption algo & same key .so we will get message+hash code.
5. now Receiver will take the message and append it with s and then ,now apply to hash fn to get the hash code
6. Now compare the separated hash code from this message and compare.if equall then authenticate and confidentiality also achieved.