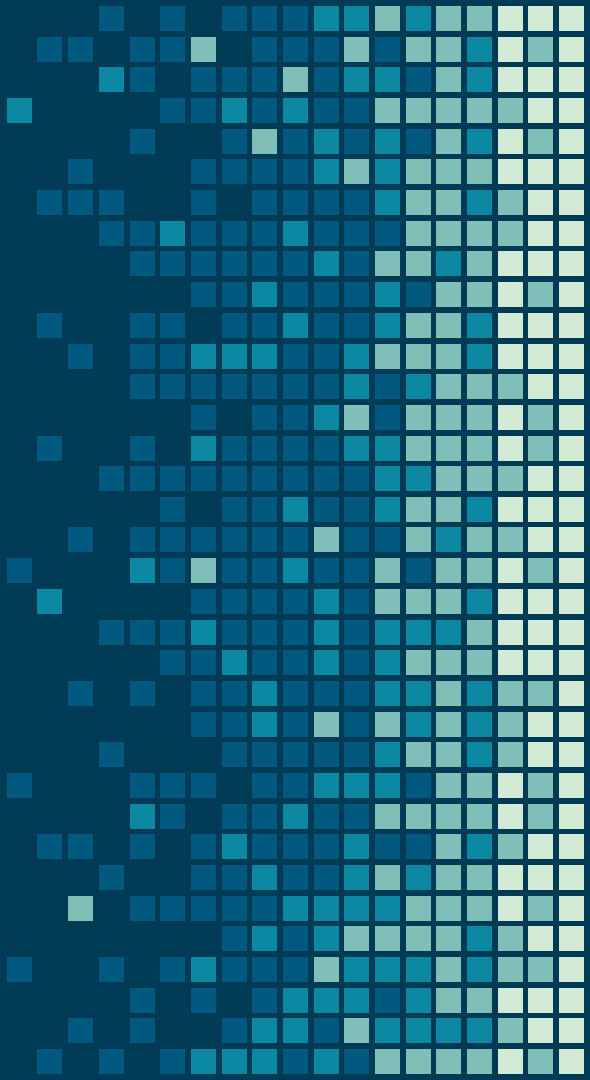
Message Authentication Codes



ROAD MAP

1. **Message Authentication Requirements**
2. Message Authentication Functions
    a. Message Encryption
    b. Message Authentication Code
3. Requirements for Message Authentication Codes
4. Security of MACs
    a. Brute-Force Attacks
    b. Cryptanalysis

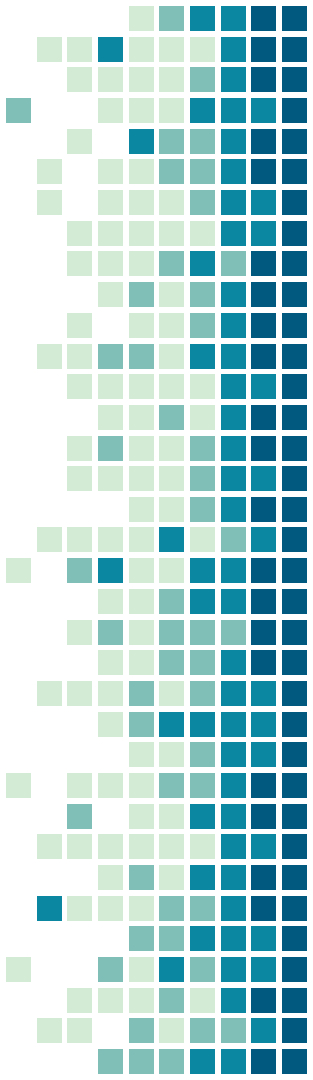
Message Authentication Requirements

In the context of communications across a network, the following attacks can be identified.

	Description	Measures
Disclosure	Release of message contents to any person or process not possessing the appropriate cryptographic key.	Measures to deal with the first two attacks are in the realm of message confi-dentiality and are dealt with in Part One.
Traffic analysis	Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.	

Message Authentication Requirements

	Description	Measures
Masquerade	Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent acknowledgments of message receipt or non-receipt by someone other than the message recipient.	Measures to deal with these items are generally regarded as message authentication. Generally, a digital signature technique will also counter some or all of the attacks here.
Content modification	Changes to the contents of a message, including insertion, deletion, transposition, and modification.	



Message Authentication Requirements

	Description	Measures
Sequence modification	Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.	Measures to deal with these items are generally regarded as message authentication. Generally, a digital signature technique will also counter some or all of the attacks here.
Timing modification	Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., datagram) could be delayed or replayed.	

Message Authentication Requirements

	Description	Measures
Source repudiation	Denial of transmission of message by source.	Mechanisms for dealing specifically with this comes under the heading of digital signatures.
Destination repudiation	Denial of receipt of message by destination.	Dealing with this item may require a combination of the use of digital signatures and a protocol designed to counter this attack.

Message Authentication Requirements

Message authentication is a procedure to verify that received messages come from the alleged source and have not been altered. Message authentication may also verify sequencing and timeliness. A digital signature is an authentication technique that also includes measures to counter repudiation by the source.

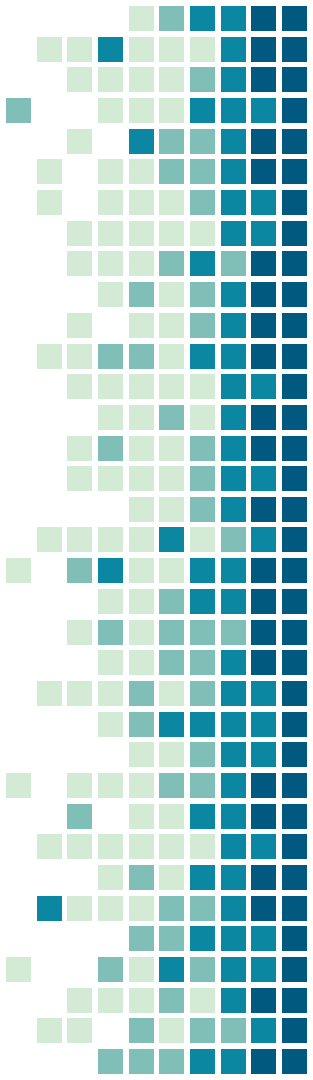
MESSAGE AUTHENTICATION FUNCTIONS:

- Any message authentication or digital signature mechanism has two levels of functionality.
  - At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message.
  - This lower-level function is then used as a primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.

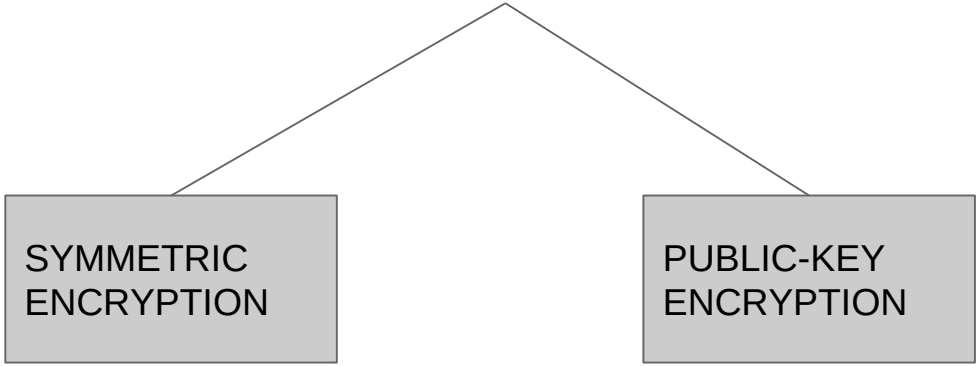
MESSAGE AUTHENTICATION FUNCTIONS:

The types of functions that may be used to produce an authenticator may be grouped into three classes.

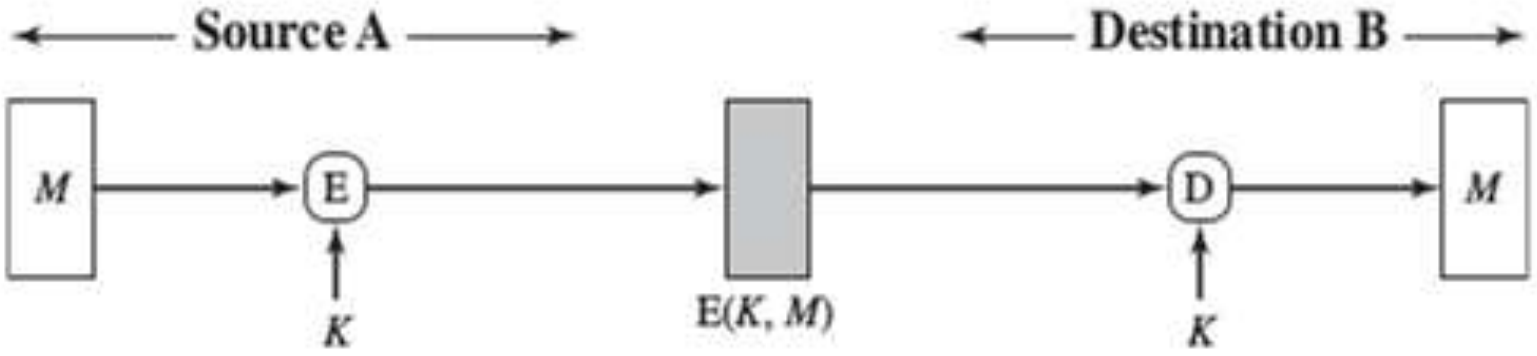
- **Hash function:** A function that maps a message of any length into a fixed-length hash value, which serves as the authenticator
- **Message encryption:** The ciphertext of the entire message serves as its authenticator
- **Message authentication code (MAC):** A function of the message and a secretkey that produces a fixed-length value that serves as the authenticator



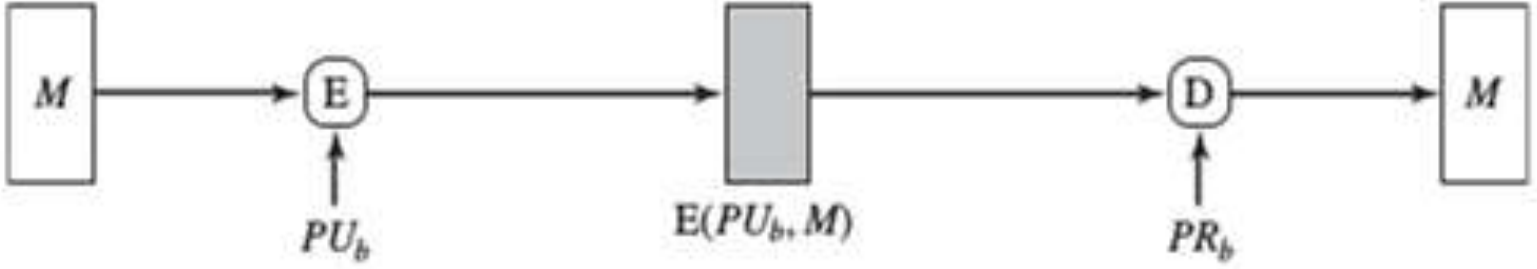
Message Encryption



MESSAGE ENCRYPTION USES:

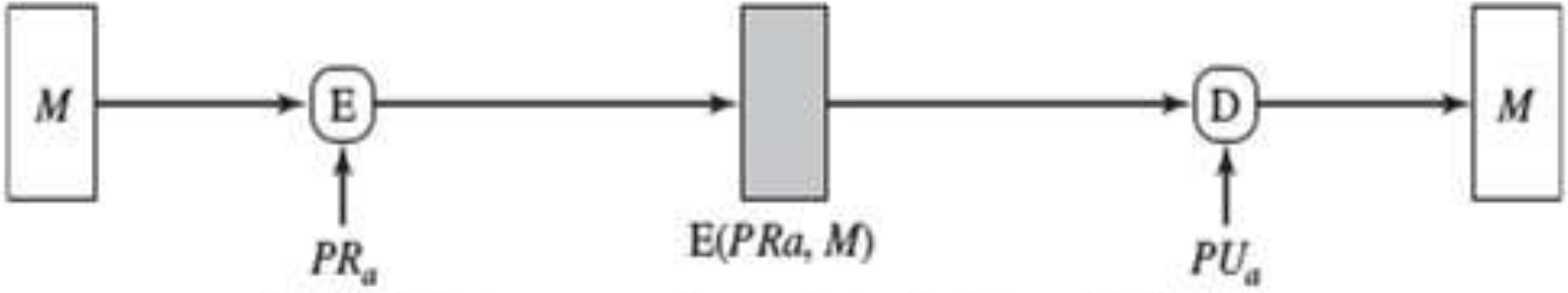


(a) Symmetric encryption: confidentiality and authentication

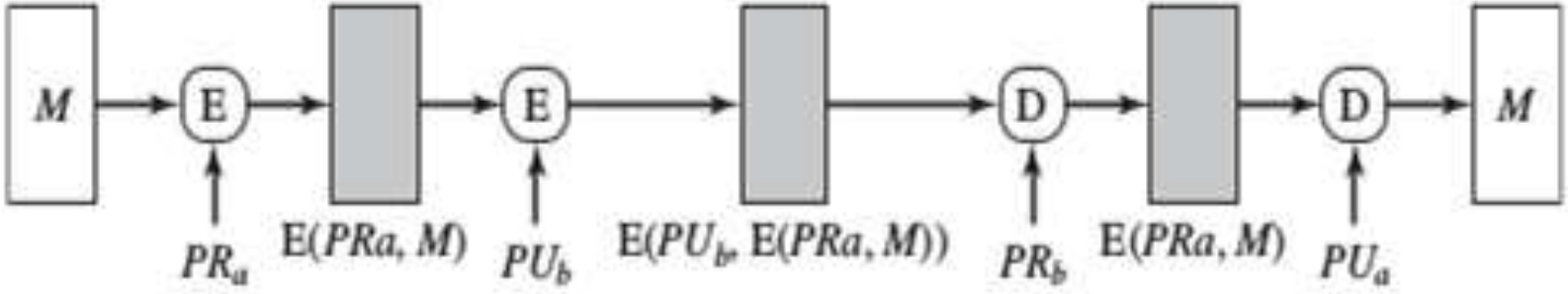


(b) Public-key encryption: confidentiality

MESSAGE ENCRYPTION USES:

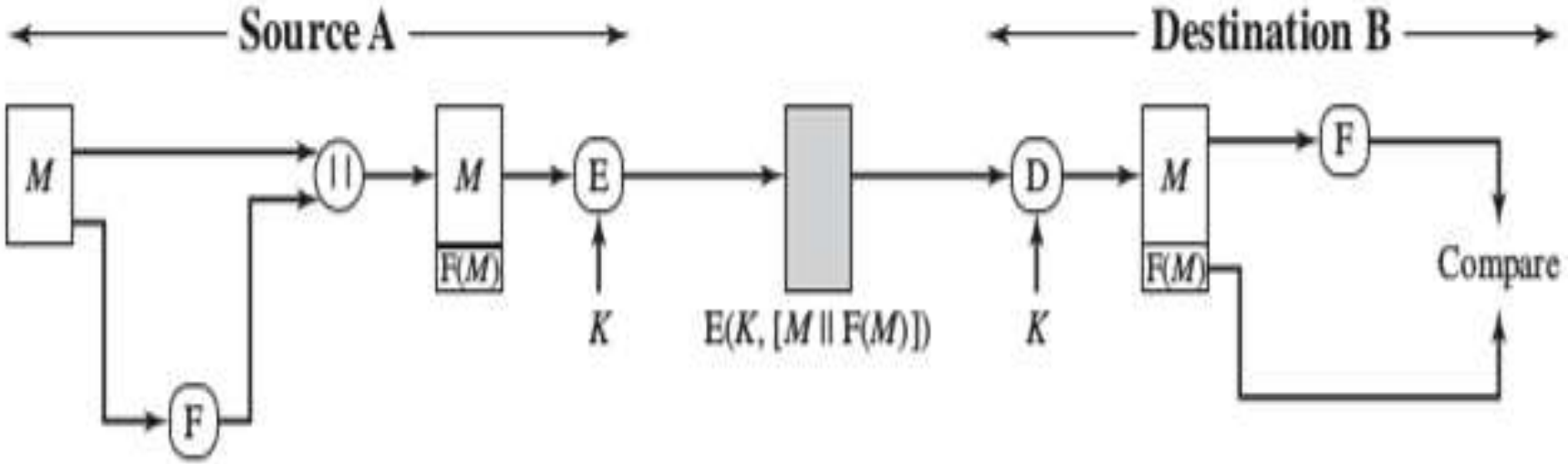


(c) Public-key encryption: authentication and signature



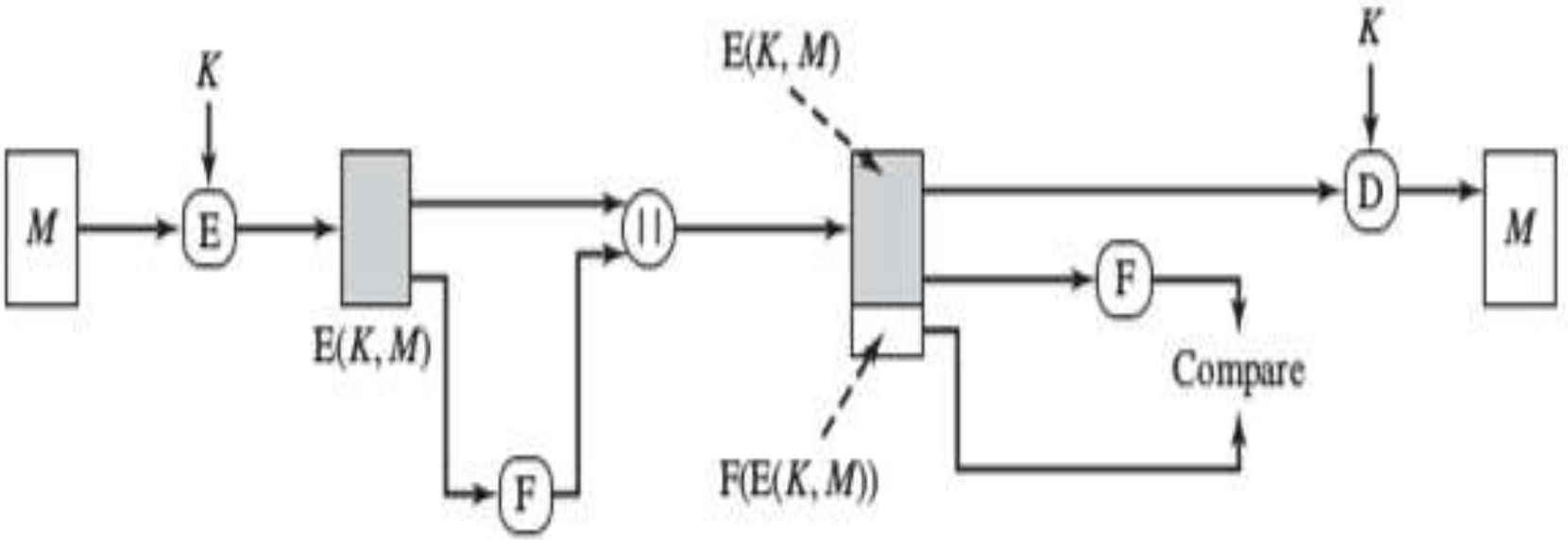
(d) Public-key encryption: confidentiality, authentication, and signature

MESSAGE ENCRYPTION ERROR CONTROL:



(a) Internal error control

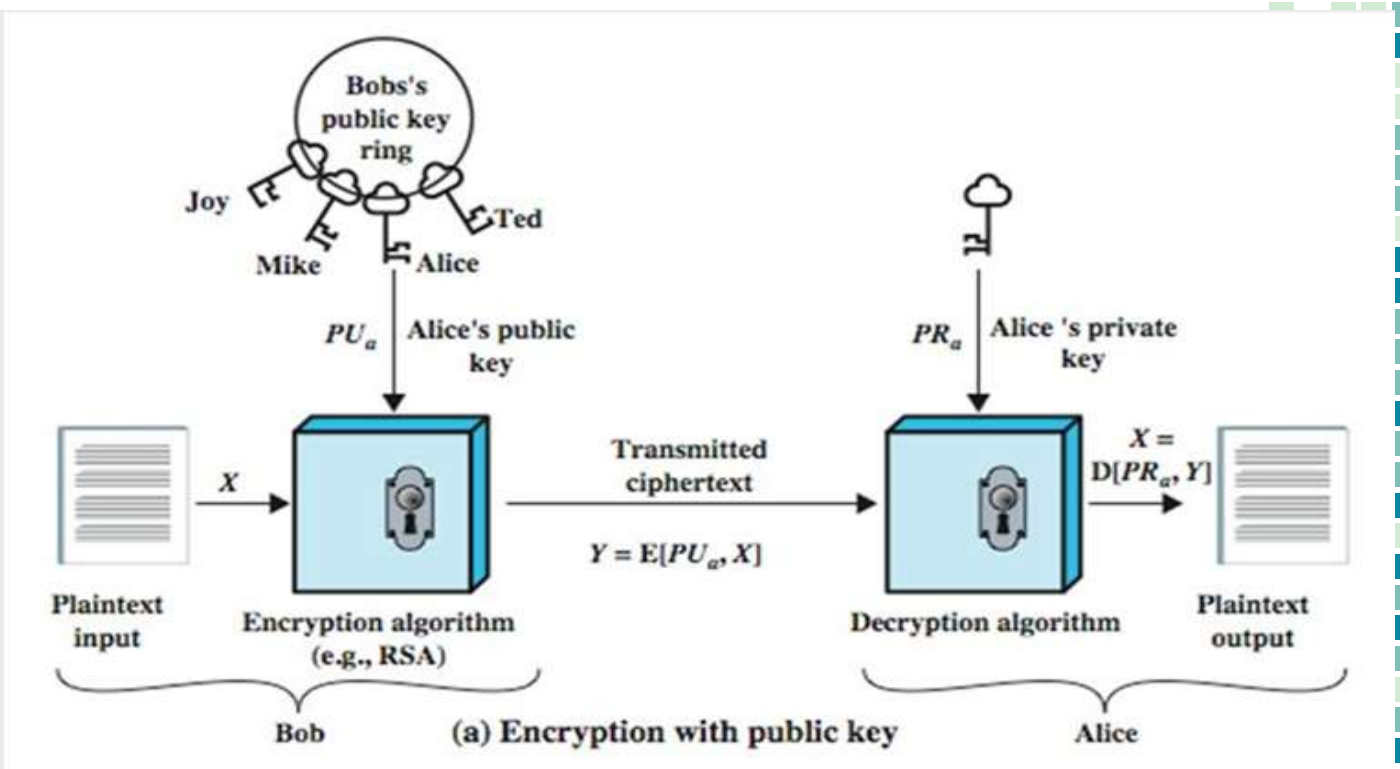
MESSAGE ENCRYPTION ERROR CONTROL:



(b) External error control

Public key Encryption

- Public key encryption provides confidentiality but not authentication
- The source (A) uses the public key of the destination (B) to encrypt M.
- Because only B has the corresponding private key , only B can decrypt the message.



Message authentication code(MAC)

- A message authentication code (MAC), also known as a **cryptographic checksum**, is an authentication technique involves the use of a secret key to generate a small fixed-size block of data.
- When A has a message to send to B, it calculates the MAC as a function of the message and the key:

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

Where:

M= input message

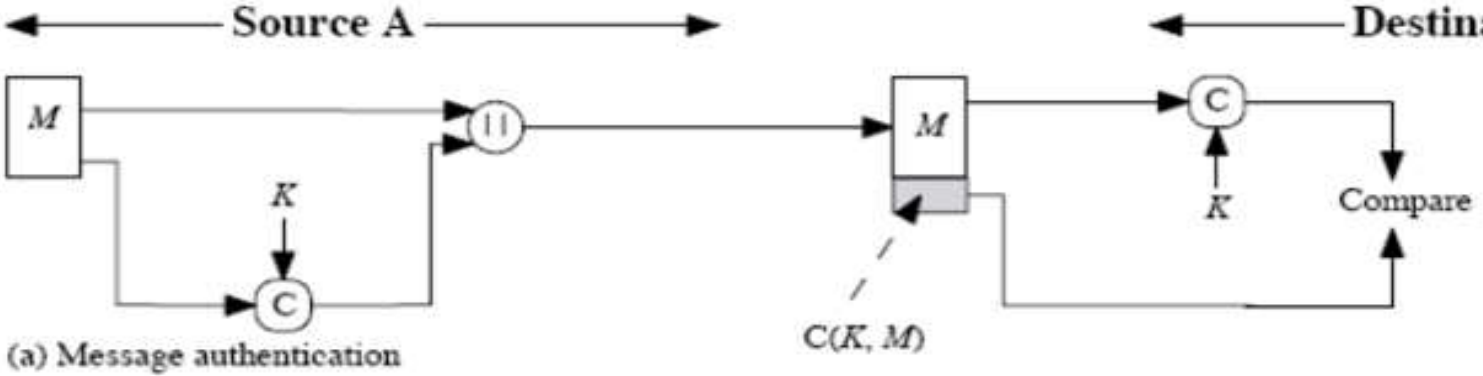
C = MAC function

K= shared secret key

- The tag is appended to the message at the source.
- The message and the MAC are transmitted to the recipient.
- The receiver recomputes the MAC from the secret key and the message data received.
- The receiver MAC is compared to the calculated MAC for message authentication
- MAC function is similar to encryption and is Many to one in nature.

- If we assume that only the receiver and the sender know the identity of the secret key, and if the received MAC matches the calculated MAC, then:
  - The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC. Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.
  - The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper MAC.
  - If the message includes a sequence number (such as is used with HDLC, X.25, and TCP), then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number

Message Authentication Code(MAC)



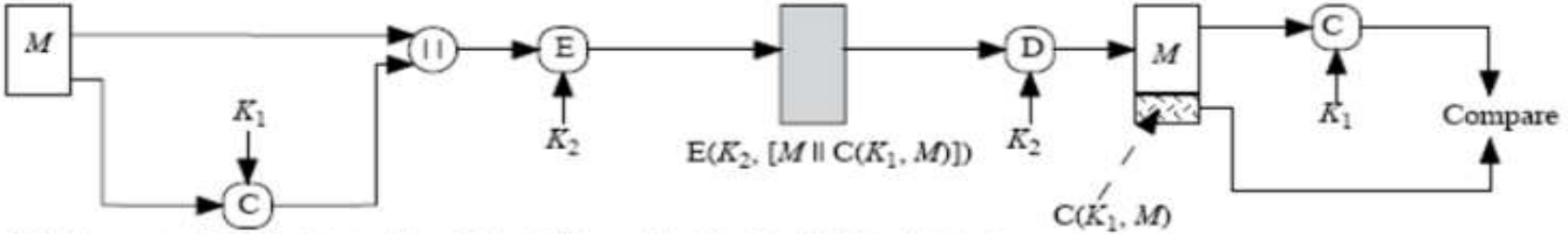
(a) Message authentication

A → B: $M \| C(K, M)$

- Provides authentication
  - —Only A and B share K

(a) Message authentication

Message Authentication Code(MAC)



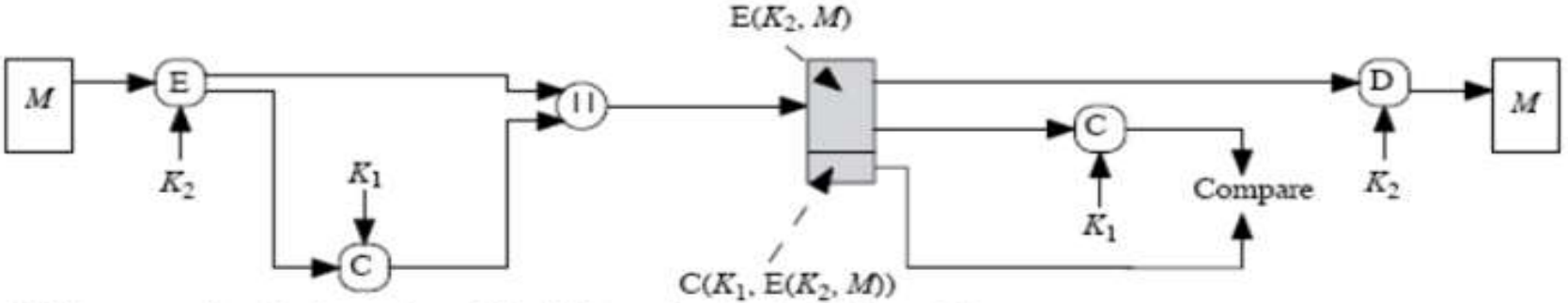
(b) Message authentication and confidentiality; authentication tied to plaintext

$A \rightarrow B: E(K_2, [M \| C(K, M)])$

- Provides authentication
  - —Only A and B share K_1
- Provides confidentiality
  - —Only A and B share K_2

(b) Message authentication and confidentiality: authentication tied to plaintext

Message Authentication Code(MAC)



(c) Message authentication and confidentiality; authentication tied to ciphertext

$A \rightarrow B: E(K_2, M) \| C(K_1, E(K_2, M))$

- Provides authentication
  - Using K_1
- Provides confidentiality
  - Using K_2

(c) Message authentication and confidentiality: authentication tied to ciphertext

Mac Requirements

MAC Function

- If an opponent observes M and MAC(K, M) , it should be computationally infeasible for the opponent to construct a message M' such that

  MAC(K, M') = MAC(K, M)

- MAC(K, M) should be uniformly distributed in the sense that for randomly chosen messages M' and M, the probability that

  MAC(K, M) = MAC(K, M') is 2^{-n} ,where

  n is the number of bits in the tag.

MAC REQUIREMENTS

- **Round 1**

  Given: $M_1, T_1 = \text{MAC}(K, M_1)$
  Compute $T_i = \text{MAC}(K_i, M_1)$ for all 2^k keys
  Number of matches $\approx 2^{(k-n)}$

- **Round 2**

  Given: $M_2, T_2 = \text{MAC}(K, M_2)$
  Compute $T_i = \text{MAC}(K_i, M_2)$ for the $2^{(k-n)}$ keys resulting from Round 1
  Number of matches $\approx 2^{(k-2\times n)}$

- On average, α rounds will be needed if . For example, if an 80-bit key is used and the tab is 32 bits, then the first round will produce about possible keys. The second round will narrow the possible keys to about possibilities.
- The third round should produce only a single key, which must be the one used by the sender.

Points to Note for MAC:

- An opponent is able to construct a new message to match a given tag, even though the opponent does not know and does not learn the key.
- The need to thwart a brute-force attack based on chosen plaintext.
- The authentication algorithm should not be weaker with respect to certain parts or bits of the message than others.

SECURITY OF MAC:

- Brute-Force Attacks
- Cryptanalysis

Brute-Force Attacks

- A brute-force attack on a MAC is a more difficult undertaking than a brute-force attack on a hash function because it requires known message-tag pairs
- To attack a hash code, we can proceed in the following way. Given a fixed message x with n-bit hash code **h = H(x)**, a brute-force method of finding a collision is to pick a random bit string y and check if **H(y) = H(x)**
- The attacker can do this repeatedly off line
- Whether an off-line attack can be used on a MAC algorithm depends on the relative size of the **key** and the **tag**

To proceed, we need to state the desired security property of a MAC algorithm, which can be expressed as follows.

- **Computation resistance:**

Given one or more text-MAC pairs ,[xi, MAC(K,xi)]

it is computationally infeasible to compute any text-MAC pair

[x, MAC(K, x)] for any new input x ≠ xi

There are two lines of attack possible:

1. **Attack the key space**
2. **Attack the MAC value**

Attack the key space:

- If an attacker can determine the **MAC key**, then it is possible to generate a valid MAC value for any input x
- Suppose the key size is k bits and that the attacker has **one known text–tag pair**
- Then the attacker can compute the n-bit tag on the known text for all possible keys
- At least one key is guaranteed to produce the correct tag, namely, the valid key that was initially used to produce the known text–tag pair
- This phase of the attack takes a level of effort proportional to 2^k (that is, one operation for each of the **2^k possible key values**)
- However, as was described earlier, because the MAC is a **many-to-one mapping**, there may be other keys that produce the correct value
- Thus, if more than one key is found to produce the correct value, additional text–tag pairs must be tested

Attack the MAC value:

- An attacker can also work on the tag without attempting to recover the key
- Here, the objective is to generate a **valid tag** for a given message or to find a message that matches a given tag
- In either case, the level of effort is comparable to that for attacking the one-way or weak collision-resistant property of a hash code, or 2^k
- In the case of the MAC, the attack cannot be conducted off-line without further input
- The attacker will require chosen text–tag pairs or knowledge of the key

SUMMARY OF BRUTE FORCE ATTACK

- To summarize, the level of effort for brute-force attack on a MAC algorithm can be expressed as **min (2^k, 2^n)**
- The assessment of strength is similar to that for symmetric encryption algorithms
- It would appear reasonable to require that the key length and tag length satisfy a relationship such as **min(k,n) > N** , where N is perhaps in the range of 128 bits

CRYPTANALYSIS

- As with encryption algorithms and hash functions, cryptanalytic attacks on MAC algorithms seek to exploit some property of the algorithm to perform some attack other than an exhaustive search.
- The way to measure the resistance of a MAC algorithm to cryptanalysis is to compare its strength to the effort required for a brute-force attack.
- That is, an ideal MAC algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort.
- There is much more variety in the structure of MACs than in hash functions, so it is difficult to generalize about the cryptanalysis of MACs.
- Furthermore, far less work has been done on developing such attacks.

HMAC

Keyed-Hashing for Message Authentication,

a variation on the MAC algorithm, has emerged as an Internet standard for a variety of applications.

It makes use of existing Message digest algorithms such as SHA-512 ,MD5 etc

Objectives Of HMAC

- To use Hash Functions that are available, without modification
- To allow replaceability of embedded Hash functions
- To use and handle keys in simple ways
- Preserving Original Performance of hash functions
- To have a well understood analysis of authentication mechanism strength.

Elements of the HMAC Algorithm

- H = embedded hash function (e.g., MD5, SHA-1, RIPEMD-160)
- IV = initial value input to hash function
- M = message input to HMAC
- Yi = ith block of M, 0 i (L - 1)
- L = number of blocks in M
- b = number of bits in a block
- n = length of hash code produced by embedded hash function
- K= secret key recommended length is n; (k>=b)
- K+ = K padded with zeros on the left so that the result is b bits in length
- ipad = 00110110 (36 in hexadecimal) repeated b/8 times
- opad = 01011100 (5C in hexadecimal) repeated b/8 times

HMAC Procedure

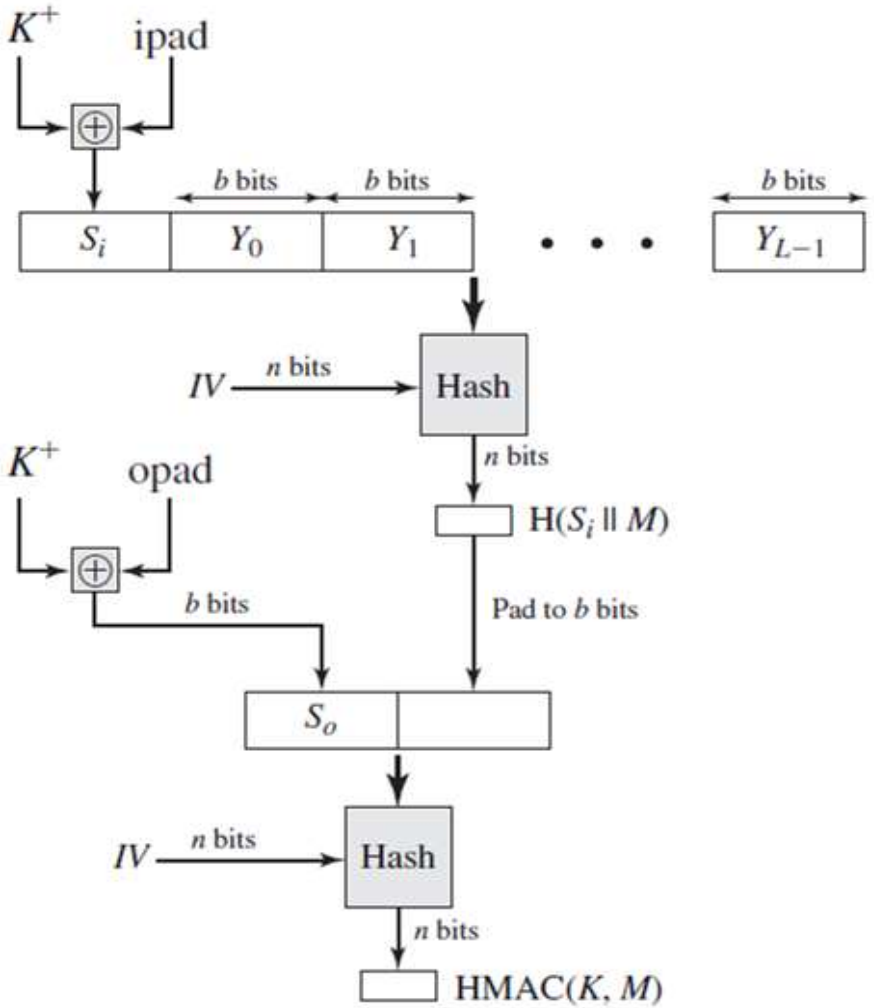


Figure 12.5 HMAC Structure

$$\text{HMAC}(K, m) = \text{H}\Big(\big(K' \oplus opad\big) \,\|\, \text{H}\big(\big(K' \oplus ipad\big) \,\|\, m\big)\Big)$$

$$K' = \begin{cases} \text{H}(K) & K \text{ is larger than block size} \\ K & \text{otherwise} \end{cases}$$

Steps Involved In the HMAC algorithm

1) Make the Size of **K greater than OR equal to 'b'** to obtain **K+**
2) XOR (bitwise exclusive-OR) K+ with **ipad** to produce the b-bit block Si.
3) Append M to Si.
4) Apply H to the stream generated in step 3..
5) XOR K+ with **opad** to produce the b-bit block So
6) Append the hash result from step 4 to So
7) Apply H to the stream generated in step 6 and output the result.

HMAC: Conclusion

The HMAC algorithm is a Hashed - Message Authentication Code generator, wherein the end product in turn is a MAC (message authentication Code). It uses available message Digest algorithms like SHA-512

HMAC makes use of Library Hash codes which are easily available. Cryptographic hash functions such as MD5 and SHA-1 generally execute faster in software than symmetric block ciphers such as DES. The aforementioned point contains factors which were the main motives for the interest in HMAC.

Thank You!