Ceng 471 Cryptography

MAC and Secure Channel
Ch. 6 and 7

Asst. Prof. Dr. Serap ŞAHIN

Ref: Cryptography Engineering, Design Principles and Practical Applications

Message Authentication Code - MAC

- MAC is a construction that detects tampering with messages.
- Encryption prevents Eve from reading messages but not prevent her from manipulating the messages.
- We will write the MAC as a result of function **MAC(K,m)**; which **K is a fixed size key** and **m is arbitrarily sized message** m.
- To authenticate a message; Alice sends message with **MAC(K,m) also called the tag**.
- Suppose Bob, also with the key K, receives message m’ and tag T. Bob uses the MAC verification algorithm to verify that T is a valid MAC under key K for message m’.

The Ideal MAC and MAC Security

- The notion of an ideal MAC function, which is very similar to the notion of an ideal block cipher.
- This definition is preferred because it encompasses a broad range of attacks; weak key attacks, related-key attacks and more.
- **Definition:** *An ideal MAC function is a random mapping from all possible inputs to n-bit output.*
  - The **key K, is not known by the attacker**. **There could be a weakness in the rest of the system that provides partial information about K to the attacker**.

CBC-MAC and CMAC

- CBC-MAC: $$H_0 = IV$$ $$H_i = E_K(P_i \oplus H_{i-1})$$ $$MAC = H_k$$
  - The most common definition of CBC-MAC requires the **IV to be fixed at 0**.
  - **In general, one should never use the same key for both encryption and authentication.** Using the same key for both can lead to privacy compromises for CBC encryption and authenticity compromises for CBC-MAC.
- There are number of different collision attacks on CBC-MAC that effectively limit the security to half the length of the block size.

CBC-MAC and CMAC

- A simple collision attack example:
- Let M be a CBC-MAC function.
- If we know M(a)=M(b) then we also know that M(a||c)=M(b||c).
- This due to the structure of CBC-MAC. Let’s illustrate this with a simple case: c consist of a single block:

$$M(a \| c) = E_K(c \oplus M(a))$$

$$M(b \| c) = E_K(c \oplus M(b))$$

$$and \qquad M(a) = M(b)$$

CBC-MAC and CMAC

Attack proceeds in two stages:

1. The attacker collects the MAC values of a large number of messages until a collision occurs.
   - This takes 2^{64} steps for 128-bit block cipher because the birthday paradox.
   - This provides a and b for which M(a)=M(b).
2. If the attacker can now get the sender to authenticate a||c, he can replace the message with b||c without changing the MAC value.

The receiver will check the MAC and accept the bogus message b||c.

- **CBC-MAC would be fine if we could use a block cipher with a 256-bit block.**

CBC-MAC and CMAC

- If you wish to use CBC-MAC, you should instead do following:

1. Construct a string *s* from the concatenation of *l* and *m*, where *l* is the length of message *m* encoded in a fixed length format.
2. Pad *s* until the length is a multiple of the block size.
3. Apply CBC-MAC to the padded string *s*.
4. Output the last cipher text block, or part of the block. Do not output any of the intermediate values.

The **advantages of CBC-MAC** is that it uses the same type of computations as the block cipher encryption modes. In many system encryption and MAC are applied on the bulk data. These are speed critical areas. Due to the same primitive functions makes efficient implementations easier, especially in hardware.

CBC-MAC and CMAC

- **CMAC is based on CBC-MAC and was recently standardized by NIST.**
- CMAC works almost exactly like CBC-MAC, except it treats the last block differently.
  - CMAC, **XORs one of two special values into the last block prior to the last block cipher encryption**.
  - **The special values are derived from the CMAC key**.

HMAC

- **HMAC** computes $\boldsymbol{h(K \oplus a \parallel K \oplus b \parallel m)}$, where **a and b are specified constants**.
- The **message itself is only hashed once, and the output hashed again with the key**.
- HMAC-SHA-256 to 128 bits should be safe.

Properties of a Secure Channel

- The problem is defined as creating a secure connection between Alice and Bob.

1. Roles:

- The most connections are bidirectional.
  - In real systems, one party can be a client and the other is server. (one initiator and one responder).
  - Alice and Bob might be the same person, and the transmission medium could be a backup device or a USB stick.
- There is always Eve, who tries to attack the secure channel in any way possible.
  - Eve can read all of the communications and arbitrarily manipulate these communications. **Eve can delete, insert or modify data that is being transmitted**.

Properties of a Secure Channel

2. Key:

- To implement a secure channel we need a shared secret key.
  - The key is K known only to Alice and Bob.
  - **Every time the secure channel is initialized, a new value is generated for the key K**. Here, a key negotiation protocol is necessary to refresh the secret key K periodically.
  - Alice and Bob assume that knowledge of K is restricted preferably to their selves.

Properties of a Secure Channel

3. Message or Stream

- The communication between parties can be as a sequence of discrete messages (such as emails) or as a continuous stream of bytes (such as streaming media).
- We will consider only discrete messages. These can be converted to handle a stream of bytes by cutting the data stream into separate messages and reassembling the stream at receiver's end.

Properties of a Secure Channel

4.Security Properties

- Alice sends a sequence of messages $m_1, m_2, \ldots$
- There are processed by the secure channel algorithms
- Bob processes the received message $m'_1, m'_2, \ldots$

The following properties are hold:

i. Secrecy; Eve should not learn anything about mesages.
   i. **Eve can see the size and timing of messages** over channel
   ii. **Eve can** find out who is communicating with whom, how much and when. (**Traffic Analysis; as well known problem for SSL/TLS, IPsec, and SSH**).

Properties of a Secure Channel

ii. Bob only gets proper messages with in their correct order. There are no duplicates, modified messages and no bogus messages sent by someone other then Alice.

iii. Bob should exactly learn which messages are missed. Alice wants to ensure that Bob gets all the messages she sent him.

Secure Channel description do not cover this third requirement. **Message acknowledgment and resend are standard communication protocol techniques.**

Order of Authentication and Encryption

Obviously we will apply both encryption and authentication to the messages.

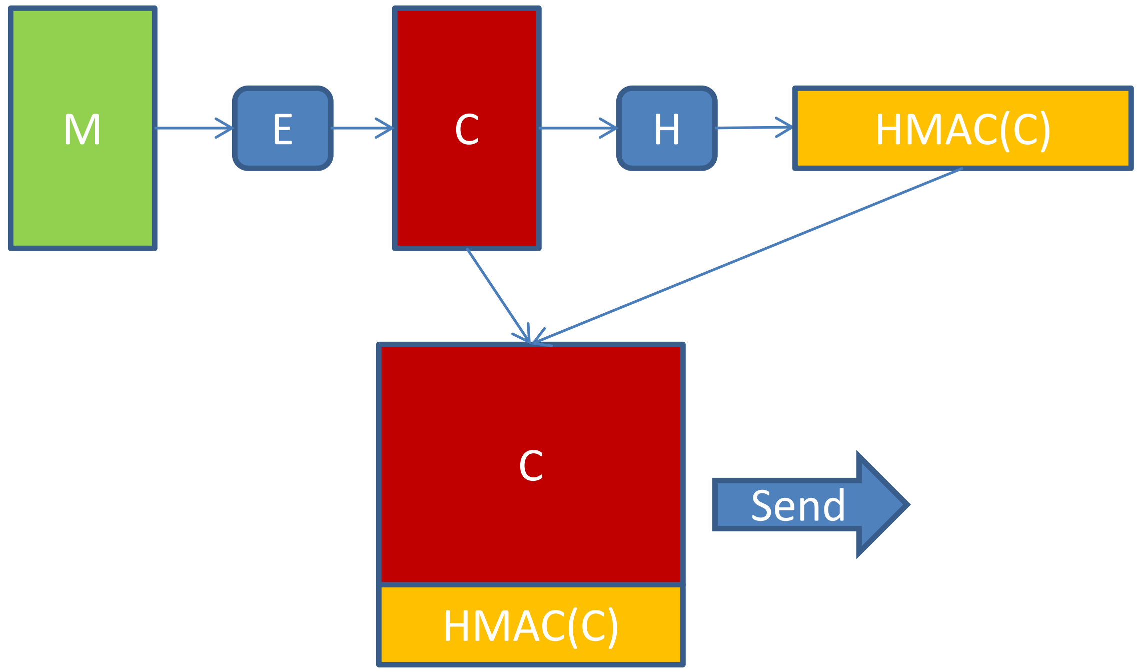
I. We can encrypt first then authenticate the cipher text. (**Encrypt then Authenticate**)

II. Authenticate first and then encrypt both the message and the MAC value. (**Authenticate then Encrypt**)

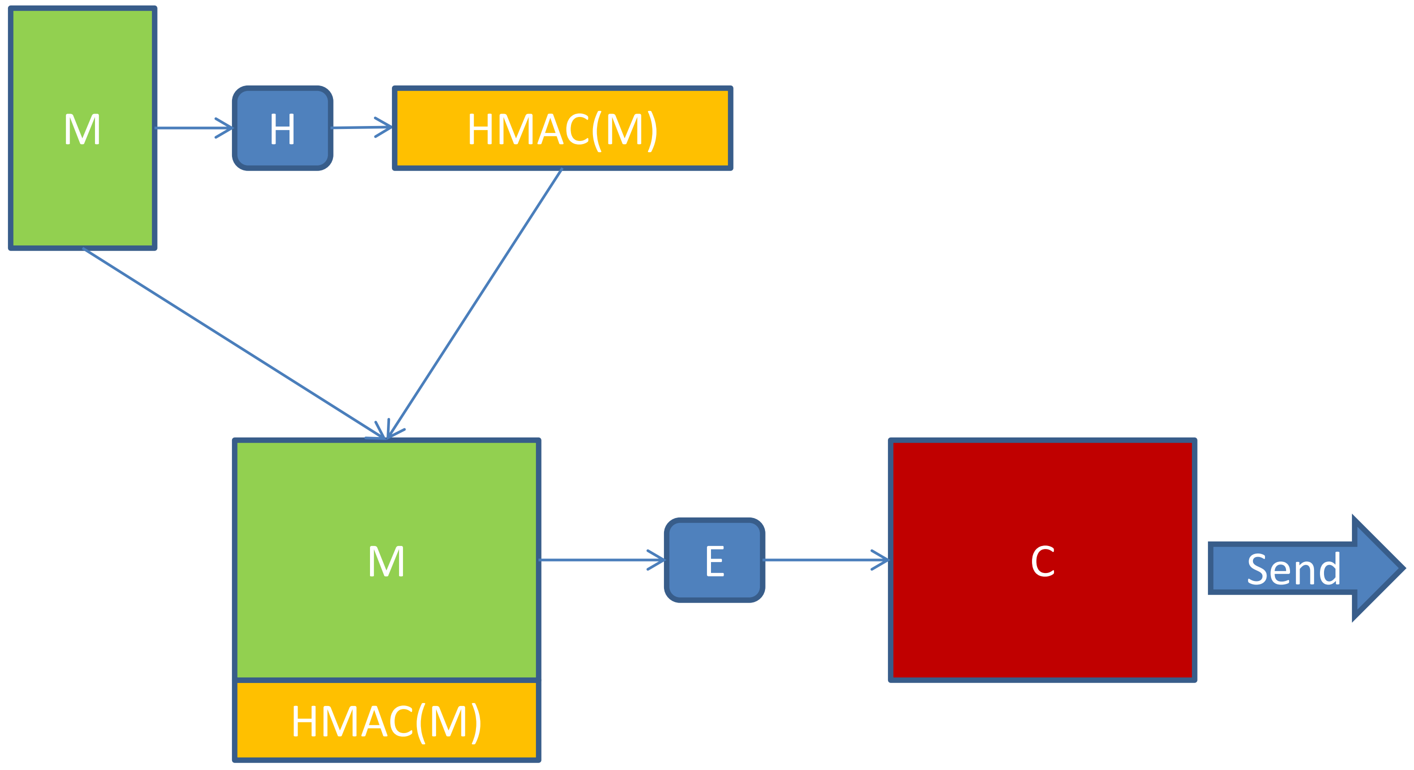
III. Both encrypt the message and authenticate the message and then combine (**concatenate**) **the two results. (Encrypt and Authenticate)**

There is no simple answer for which method is the best.

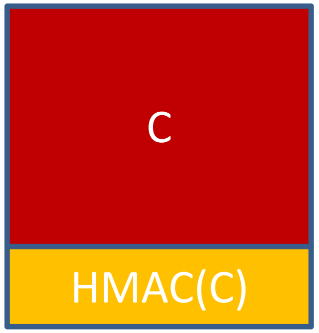
Encrypt then Authenticate



Authenticate then Encrypt



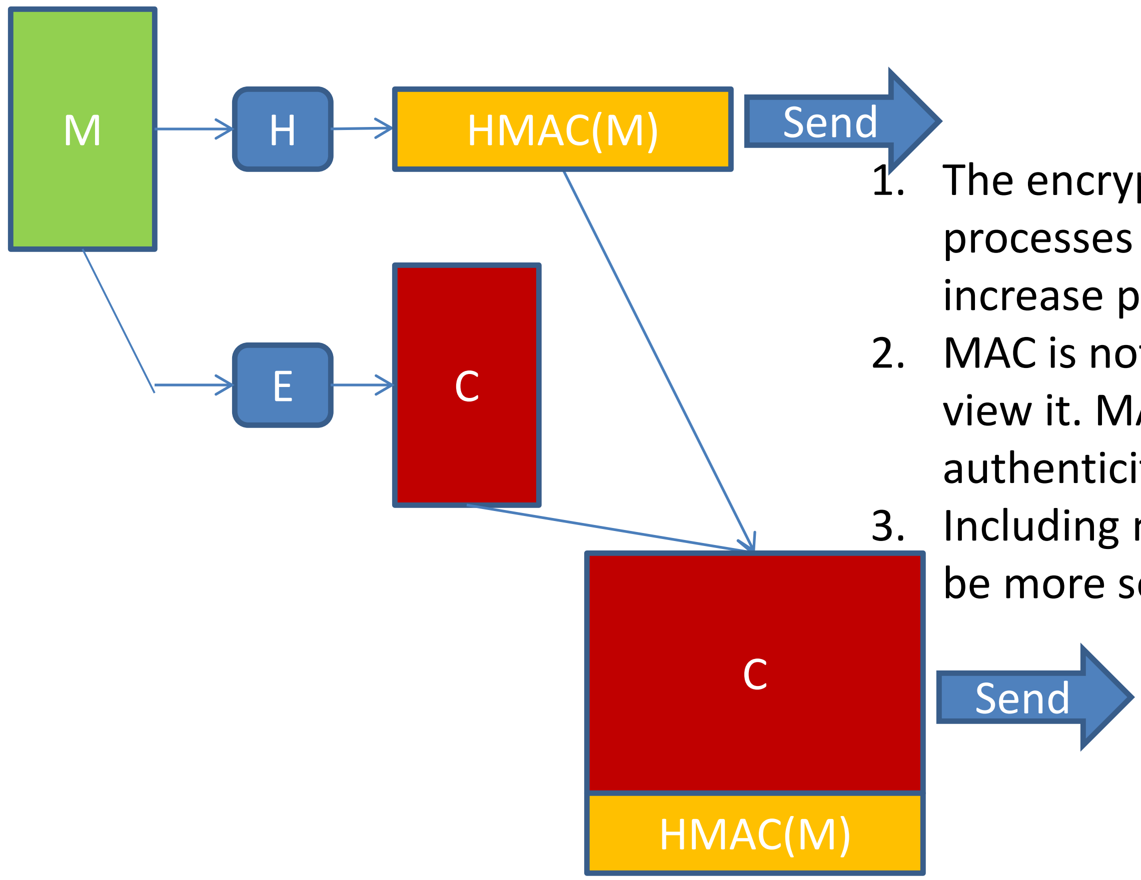
Encryption First



- There are two arguments in favor encryption first:

1. Applying the MAC to the cipher-text of such a weak encryption scheme fixes it and makes it secure.
2. It is more efficient in discarding bogus messages.
   - Normally Bob has to both decrypt the message and check the authentication. With encrypt first, the decryption is done last on the receiver side, and Bob never has to decrypt bogus messages, since he can identify and discard them before decryption.

Encrypt and Authenticate



1. The encryption and authentication processes can happen in parallel. This can increase performance.
2. MAC is not encrypted and attacker can view it. MACs are designed to protect authenticity not privacy.
3. Including nonce value, this approach can be more secure.

Important Question!

- Imagine, how much damage Eve could do if she could read all the traffic. Then think about how much damage Eve could do if she could modify the data being communicated.
- Which one is more important; encryption or authentication?

Important Question!

- In many cases, one can argue that authentication is more important than encryption.
- We therefore prefer to expose the encryption function to Eve’s direct attacks, and protect the MAC as much as possible.
- In most cases, modifying data is a devastating attack, and does more damage than merely reading it.

Harton Principle

- Bob might check that **the cipher text is correctly authenticated**, but then **decrypt the cipher text with a different key than what Alice used to encrypt the message**.
- Bob will get a different plaintext than Alice sent, even though the authentication checked out.
- This shouldn’t hapen but it can.
- **Standard Solution**: to **derive both encryption key and authentication key for the secure channel from a single secure channel key K**. This removes vulnerability and creates cross-dependency. **The authentication depends on the key derivation system**.

Designing a Secure Channel

- The solution consists following components:

1. Message numbering,
2. Authentication,
3. Encryption

Message numbering

- Provide a source for IVs for the ecryption algorithm.
- Allow Bob to reject replayed messages without keeping large databases.
- Tell Bob which messages were lost,
- They ensure that Bob receive the messages in correct order.
- The message numbers must increase monotonically and must be unique.

Authentication

- $a_i = MAC(i \parallel l(x_i) \parallel x_i \parallel m_i)$
- We will use HMAC-SHA-256 with full 256 bit result.
- The input to the MAC consists of the message m_i and extra authentication data x_i. Each x_i is a string and both Alice and Bob have the same value for x_i.
- $l(x_i)$ ensures that the string $i \parallel l(x_i) \parallel x_i \parallel m_i$ uniquely parses into fields and protect from Harton Principible risk.

Encryption

- We use the message number as the unique nonce value that CTR mode. The secure channel uses CTR mode. But, the generation of nonces never expose to external systems.
- We limit the size of each message to 16.2^{32}bytes, which limits the block counter to 64 bits.
- The key stream consists of bytes $k_0, k_1, \ldots$ For a message with nonce i, the key stream is defined by $k_0, k_1, \ldots, k_{2^{32}-1} = E(K, 0 \parallel i \parallel 0) \parallel E(K, 1 \parallel i \parallel 0) \parallel \cdots \parallel E(K, 2^{32} - 1 \parallel i \parallel 0)$ where each plaintext block of the cipher is built from 32bit block number, the 32bit message number and 64bits of zeros.
- The key stream is a very long string. We will use the first $l(m_i) + 32\ bytes$ of the key stream. We concatenate m_i and a_i and XOR these bytes with $k_0, k_1, \ldots, k_{l(m_i)+31}$.