CO 487: Applied Cryptography

Chapter 3: Message Authentication Codes and Authenticated Encryption

December 5, 2019

Contents

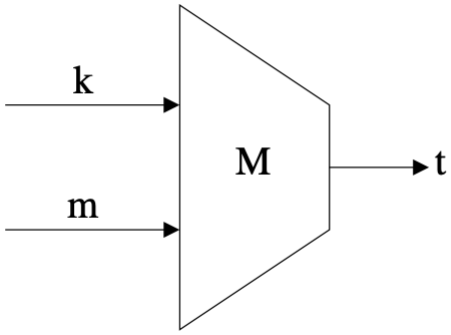
1 Message Authentication Codes

1.1 Basic Ideas

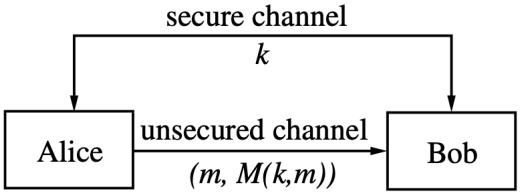
Definition: A **message authentication code (MAC)** schemes are used for providing (symmetric-key) data integrity and data origin authentication. It is an efficiently computable function $M : \{0,1\}^l \times \{0,1\}^l \to \{0,1\}^n$.

This function can also be written in $M(k, m) = t$ where

- k: key
- m: message
- t: tag



Applications of Message Authentication Codes Schemes:



To provide data integrity and data origin authentication:

1. The sender and the receiver establish a secret key $k \in \{0,1\}^l$.
2. The sender computes MAC $t = M(k, m)$ and sends (m, t) to the receiver.
3. The receiver verifies that $t = M(k, m)$

A timestamp or sequence number is added to avoid replay.

The **Security** of MAC:

- Assumption: The adversary knows everything about the MAC scheme except the value of k, the secret key shared by users.
- Definition: A MAC scheme is **secure** if it is existentially unforgeable against chosen-message attack, which means,
  - Interaction: Some MAC tags $M(k, m_i)$ are given for message m_i chosen by the adversary.
  - Computational Resources: it is computationally infeasible.
  - Goal: to compute the value of $M(k, m)$ for any message $m \neq m_i$.
- A secure MAC scheme can be used to provide data integrity and data origin authentication.

1.2 Generic Attacks

How to guess the MAC of a message m:

Select $y \in \{0,1\}^n$ and guess that $M(k, m) = y$. Assuming that $M(k, \cdot)$ is a random function, the probability of success is $\frac{1}{2^n}$. Guesses cannot be directly checked without interaction. $n \geq 80$ is preferred.

Exhaustive Search on the Key Space:

Given r known message-MAC pairs: $(m_1, t_1), ..., (m_r, t_r)$, one can check whether a guess k of the key is correct by verifying that $M(k, m_i) = t_i$, for $i = 1, 2, ..., r$.

Assuming that the $M(k, \cdot)$'s are random functions, the expected number of keys for which the tags verify is $K = \frac{2^l}{2^{nr}}$. Exhaustive search is infeasible if $\ell \geq 80$.

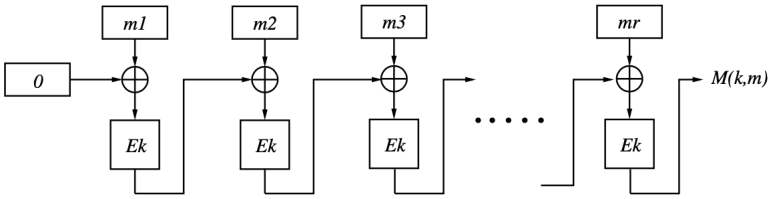
Example: If $l = 56$, $n = 64$, $r = 2$, then $K \approx \frac{1}{2^{72}}$.

2 Different Types of MACs

2.1 CBC-MAC: MACs Based on Block Ciphers

Definition: A **cipher block chaining message authentication code (CBC-MAC)** is a technique for constructing a message authentication code from a block cipher.

- Definition: Let E be an n-bit block cipher with key space $\{0,1\}^l$.
- Assumption: Suppose that plaintext messages all have lengths that are multiplies of n.
- To compute $M(k,m)$
  1. Divide m into n-bit blocks m_1, m_2,...,m_r.
  2. Compute $H_1 = E(k, m_1)$.
  3. For $2 \leq i \leq r$, compute $H_i = E(k, H_{i-1} \oplus m_i)$.
  4. Then $M(k,m) = H_r$.



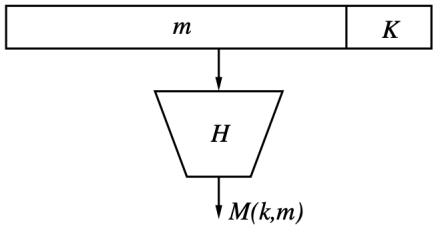
- Security:
  - CBC-MAC is not secure if variable length message are allowed. Here is a chosen-message attack on CBC-MAC.
    1. Let m_1 be an n-bit block.
    2. Let (m_1, t_1) be a known message-MAC pair.
    3. Request the MAC t_2 of the message t_1.
    4. Then t_2 is also the MAC of the 2-block message $(m_1, 0)$. Since $t_2 = E_k(E_k(m_1))$.
  - **Theorem**: Suppose that E is an ideal encryption scheme, which means for each $k \in \{0,1\}^\ell$ $E_k : \{0,1\}^n \to \{0,1\}^n$ is a random permutation), then CBC-MAC with fixed-length inputs is a secure MAC algorithm.
  - **Definition**: **Encrypted CBC-MAC (EMAC)** is a solution for variable-length message. EMAC encrypts the last block under a second key s: $M((k,s),m) = E_s(H_r)$, where H_r is the output of CBC-MAC.
    - **Theorem**: Suppose that E is an ideal encryption scheme, then EMAC is a secure MAC algorithm for inputs of any length.

2.2 MACs Based on Hash Functions: Secret Prefix Method

Hash functions were not originally designed for message authentication. In particular they are not keyed primitives.

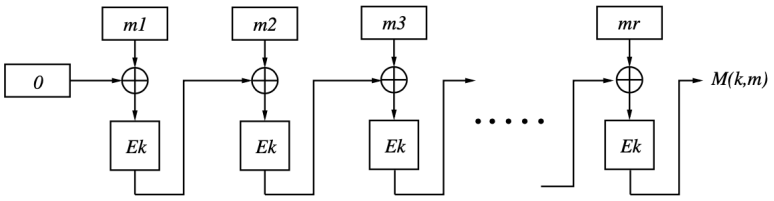
2.2.1 Secret Prefix Method

- **Definition**: **MACs in Secret Prefix Method**



$\underline{M(k,m) = H(K||m)}$

  - H: an iterated n-bit hash function without. the final length block consisting of padding.
  - $n + r$: the length of the input of the compression function $f : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^n$.
  - k: $k \in \{0,1\}^n$
  - K: K denote k padded with $(r - n)0$'s. So K has bitlength r.



However, this is **insecure**.

Example: A **length extension attack** against MACs in Secret Suffix Method

1. Suppose that $(m, M(k,m))$ is known.
2. Suppose that the bit length of m is a multiple of r.
3. Then $M(k, m||m')$ can be computed for any m' without knowledge of k.

Example: If a length block is postpended to $K||m$ prior to application of H, it is also **insecure**.

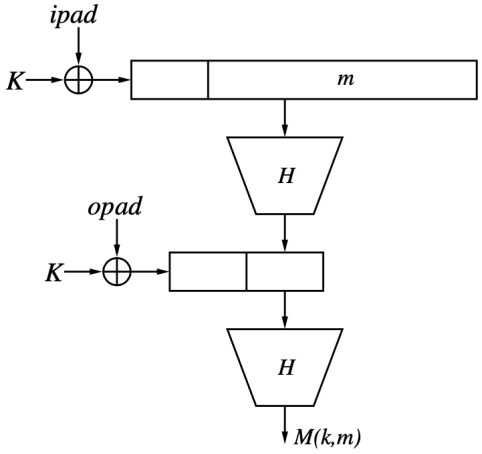
Example: An attack against MACs in Secret Suffix Method

1. Suppose that a collision (m_1, m_2) can be found for H, which means $H(m_1) = H(m_2)$. Assume that m_1 and m_2 both have bitlengths that are multiples of r.
2. Then, $H(m_1||K) = H(m_2||K)$.
3. Then, $M(k, m_1) = M(k, m_2)$.
4. The MAC for m_1 can be requested and be given to the MAC for m_2.

2.3 Hash-Based MAC (HMAC)

How Hash-Based MAC works: Define two r-bit strings. Each repeated $\frac{r}{8}$ times.

Definition: $M(k, m) = H(K \oplus opad, H(K \oplus ipad, m))$



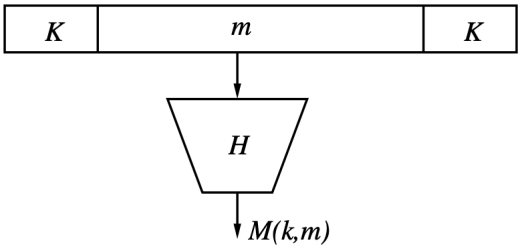
Definition: Suppose that the compression function used in H is a secure MAC with fixed length messages and a secret IV as the key. Then HMAC is a **secure** MAC algorithm.

2.4 Envelope Method

Definition: **Envelope Method/Sandwich MAC**

- $M(k, m) = H(K||m||K)$

- The MAC key is used both
  at the start and end of the MAC computation.

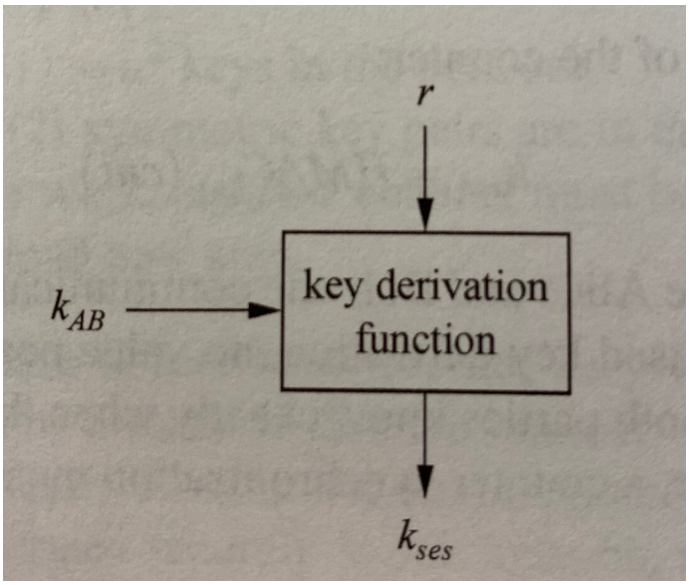


Theorem: Suppose that the compression function used in H is a secure MAC with fixed length messages and a secret IV as the key. Then Envelope MAC with m padded to a multiple of the block length of H is **secure** MAC algorithm.

2.5 Key Derivation Functions (KDF)

Definition: **Key Stretching Function**

- Goal: Turn a short and weak key into a longer and more secure key.
- Basic Idea: Uses an already established joint secret key to derive fresh session key.
  - Using keys which are only valid for a limited time ha several advantages.A major one is there is less damage if the key is exposed. Also, an attack has less ciphertext available that was generated under one key, which can make cryptographic attacks more difficult. KDF is one of the approaches to key update.



- Application:
  - Password storage in data bases
  - Fixed-length keys: user types in a passphrase which is converted into a 256-bit key
  - Key exchange using public-key cryptography

Example: HMAC is commonly used as a key derivation function.

For example, there is a user has a secret key k, and he/she wants to derive several session keys, which means encrypting data in different communication sessions. It can be done by computing $sk_1 = HMAC_k(1)$, $sk_2 = HMAC_k(2)$, $sk_3 = HMAC_k(3)$...

An adversary is unable to learn anything about any particular session key sk_j without knowledge of k, even though it may have learnt some other session keys.

Key Stretching Techniques

Bad	Secret Prefix Method using an iterated hash function, and Secret Suffix Method using an iterated hash function
Okay	HMAC using an iterated hash function, and Secret prefix method, using a non-iterated hash function specifically designed to support this usage.
Best	Key derivation function

Definition: **Password-Based Key Derivation Function 2 (PBKDF2)**

Key = $PBKDF2(F, p, s, c, \ell)$, where

- F: keyed hash function
- p: passphrase-the master password from which a derived key is generated
- s: salt-random data that is used as an additional input a one-way that hashes data.
- c: number of iterations
- ℓ: output length

Key=$PBKDF2(F, p, s, c, \ell)$, where

$$T_i = U_{i,1} \oplus U_{i,2} \oplus ... \oplus U_{i,c}$$
$$U_{i,1} = F([, s||i)$$
$$U_{i,2} = F(p, U_{i,1})$$
$$...$$
$$U_{i,c} = F(p, U_{i,c-1})$$

This function is supposed to be slow. Larger iteration counts yield more security.

3 Authenticated Encryption

3.1 Combining Encryption and Authentication

Given a share secret key:

- A symmetric-key encryption scheme achieves confidentiality.
- A message authentication code achieves data authentication.

Definition: **Composed Primitives** are the methods of Combining Encryption and Authentication. For example,

1. Encrypt-and-MAC (E&M) used by SSL/TLS up to v1.2,
2. MAC-then-Encrypt (MtE) used by IPsec, and
3. Encrypt-then-MAC (EtM) used by SSH.

Security Goals

- **Integrity of Plaintext (INT-PTXT)**

  Produce a valid ciphertext containing an encryption of a plaintext message that was never sent by the legitimate sender.

  (Produce a valid ciphertext means one that passes all authentication checks.)

- **Integrity of Ciphertext (INT-CTXT)**

  Produce a valid ciphertext that was never sent by the legitimate sender.

3.2 Encrypt-and-MAC (E&M)

- Compute: $c = Enc(m)$ and $t = MAC(m)$
- Transmit: $c||t$
- Decrypting: The recipient checks the MAC is correct. However,
  - MACs do not ensure confidentiality.
    - **Example**: The MAC might leak one plaintext bit, and still be secure as a MAC. Violates semantic security.
  - Even the SKES and the MAC are secure, the encrypt-and-MAC combination might be insecure.

3.3 MAC-then-Encrypt (MtE)

- Compute: $c = Enc(m)$ and $t = MAC(m)$
- Transmit: c
- Decrypting: The recipient checks the MAC is correct. However,
  - SKESs do not ensure integrity
    - **Example**: Changing the ciphertext might not change the plaintext for certain values of plaintext. Violates INT-CTEXT.
  - Even the SKES and the MAC are secure, the encrypt-and-MAC combination might be insecure.

3.4 Encrypt-then-MAC (EtM)

Definition: An encryption scheme is **ciphertext unforgeable** under chosen plaintext attack (CUF-CPA) if an attacker cannot achieve the following goal:

- Interaction: The attacker may choose any number of messages m_i and obtain their corresponding valid (authenticated) encryptions c_i under the key k.
- Computation: The attacker is computationally bounded.
- Goal: The attacker must produce a valid (authenticated) encryption $c \neq c_i$ (i.e. break INT-CTXT)

3.5 Comparison: E&M, MtE and EtM

	Encrypt-and-MAC (E&M)	MAC-then-Encrypt (MtE)	Encrypt-then-MAC (EtM)
Security	Insecure	Insecure	Secure
Compute	$c = Enc(m)$ and $t = MAC(m)$	$c = MAC(m)$ and $t = MAC(m\|\|t)$	$c = Enc(m)$ and $t = MAC(m)$
Transmit	$c\|\|t$	c	$c\|\|t$
Decrypting	MACs do not ensure confidentiality	SKESs do not ensure integrity	Confidentiality and integrity are both ensured

Definition: **The Cryptographic Doom Principle**
If you perform cryptographic operations before verifying the MAC on a message you've received, it will somehow inevitably lead to doom.

Theorem: Supposes the SKES is semantically secure under chosen plaintext attack, and the MAC is EUF-CMA. Then encrypt-then-MAC is semantically secure and CUF-CPA.

3.6 Padding

Definition: **Padding** refers to a number of distinct practices which all include adding data to the beginning, middle, or end of a message prior to encryption. Often, when using block ciphers, a message needs to be padded before encrypting, to align it with block boundaries.

In classical cryptography, padding may include adding nonsense phrases to a message to obscure the fact that many messages end in predictable ways, e.g. sincerely yours.

Example: Suppose we are designing SSL/TLS using MAC-then-encrypt.

- Apply MAC, then pad, then encrypt
  Compute $t = MAC(m)$ and $c = Encrypt(m||p||t)$, transmit c.
  SSL/TLS uses this, but this is the wrong choice.
- Apply padding, then MAC, then encrypt
  Compute $t = MAC(m||p)$ and $c = Encrypt(m||p||t)$, transmit c.

3.7 Using a Block Cipher Mode of Operation with Authentication Built-In

The fastest way to achieve authenticated encryption is to
use a block cipher mode of operation with authentication built-in.

These modes of operation require a block cipher, but not a separate MAC (authentication functionality is built-in).

- CCM mode: Counter with CBC-MAC
- EAX mode
- GCM mode: Galois/Counter Mode
- OCB Mode: Offset Codebook Mode

Example: **Galois Counter Mode**

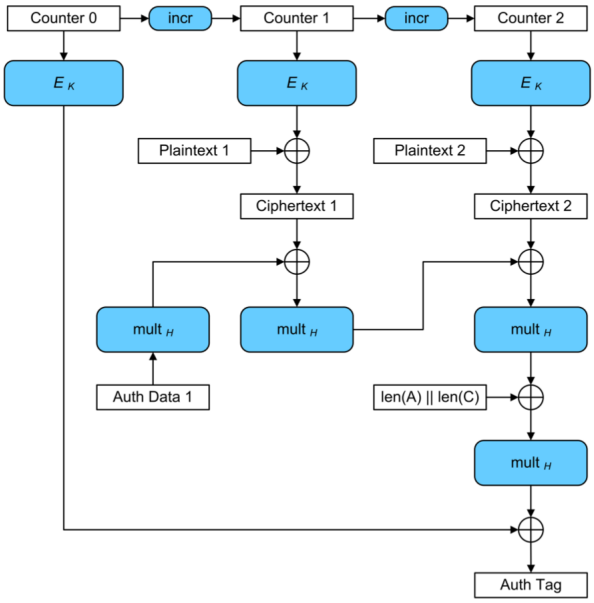
Definition: The **Galois Counter Mode** is an encryption mode which also computes a message authentication code. GCM protects the confidentiality of the plaintext x by using an encryption in counter mode.

Additionally, GCM protects not only the authenticity of the plaintext x but also the authenticity of a string **authenticated tag**.

GCM ciphertexts are identical to counter (CTR) mode ciphertexts except the authentication tag. In particular, the last ciphertext block is truncated if the plaintext length is not an integral number of blocks.

Authentication tags are computed in $GF(2^{128}) = \frac{\mathbb{F}_2[x]}{x^{128}+x^7+x^2+x+1}$. Hence, GCM requires a 128-bit block size.

Auth Data 1 is a 128-bit block of authenticated unencrypted data, viewed as an element of $GF(2^{128})$. More than one block is supported, but only one is shown.



Let

- $e()$: a block cipher of block size 128 bit.
- x: the plaintext consisting of the clocks $x_1, ..., x_2$
- AAD: the authenticated tag
- Encryption
  - Derive a counter value CTR_0 from the IV and compute $CTR_1 = CTR_0 + 1$.
  - Compute ciphertext: $y_0 = e_k(CTR_i) \oplus x_i$, $i \geq 1$.
- Authentication
  - Generate authentication subkey $H = e_k(0)$
  - $$GaloisFrieldMultiplication$$ Compute $g_0 = AAD \times H$
  - $$GaloisFrieldMultiplication$$ Compute $g_i = (g_{i-1} \oplus y_i) \times H$, $1 \leq i \leq n$.
  - Final authentication tag: $T = (g_n \times H) \oplus e_k(CTR_0)$

4 Password Hashing

4.1 Basic Ideas

Definition: Passwords are human-memorizable strings that are used for authentication.

Authenticators can be **categorized** as:

- Knowledge-Based: Something you know.
- Object-Based: Something you have.
- ID-Based: Something you are.
- Location-Based: Somewhere you are.

Multi-factor authentication uses combinations from multiple different categories of authenticators.

4.2 Entropy: a Measure of Password Strength

Entropy measures the uncertainty in values generated from a random process.