Authenticated Encryption

Pierre-Alain Fouque

Introduction

- Encryption function ensures confidentiality but not integrity
- The first block of a CBC encryption can be changed without being remarked by the receiver
- The CBC padding oracle is a devasting attack against CBC in particular in TLS …
- For stream cipher, it is even easier to modify the message
- Encryption provides confidentiality but its goal is not to provide integrity
- MAC (Message Authentication Code) can be used for this task

Message Authentication Code (MAC)

Formally, a **message authentication code** (**MAC**) system is a triple of efficient[2] algorithms (G, S, V) satisfying:

- G (key-generator) gives the key k on input 1^n, where n is the security parameter.
- S (signing) outputs a tag t on the key k and the input string x.
- V (verifying) outputs *accepted* or *rejected* on inputs: the key k, the string x and the tag t.

S and V must satisfy the following:

$$\Pr[\, k \leftarrow G(1^n),\ V(\, k, x, S(k, x)\,) = \textit{accepted}\,] = 1.$$ [3]

A MAC is **unforgeable** if for every efficient adversary A

$$\Pr[\, k \leftarrow G(1^n),\ (x, t) \leftarrow A^{S(k,\,\cdot\,)}(1^n),\ x \notin \mathrm{Query}(A^{S(k,\,\cdot\,)}, 1^n),\ V(k, x, t) = \textit{accepted}] < \mathrm{negl}(n),$$

where $A^{S(k,\,\cdot\,)}$ denotes that A has access to the oracle $S(k, \cdot\,)$, and $\mathrm{Query}(A^{S(k,\,\cdot\,)}, 1^n)$ denotes the set of the queries on S made by A, which knows n. Clearly we require that any adversary cannot directly query the string x on S, since otherwise a valid tag can be easily obtained by that adversary.[4]

Security Game

Security Game for MACs (chosen message attack)

$$
\begin{array}{ccc}
\mathcal{A} & & \mathcal{C} \\
\text{choose } m_i, & & k \\
i = 1, \ldots, q \in \mathcal{M} & \xrightarrow{m_i} & \\
 & & \text{compute } S(k, m_i) = t_i \\
 & \xleftarrow{t_i} & \\
(m^*, t^*) \neq (m_i, t_i) & & \\
 & \xrightarrow{(m^*, t^*)} & \\
 & & V(k, m^*, t^*) = D
\end{array}
$$

If $D =$ "yes" then $\mathcal{A}$ wins the security game (i.e. the MAC is **not** secure against chosen message attack). If $D =$"no", $\mathcal{A}$ has lost the security game (i.e. the MAC is secure).

Constructions of MAC

- Let F_K:{0,1}* to $\{0,1\}^t$ a random function
- $M=M_1\ldots M_m$ split M into m blocks

- $MAC_1(M) = F_K(M_1) \oplus \ldots \oplus F_K(M_m)$
  - Is it a secure MAC_1 ?

- $MAC_2(M) = F_K(<1>||M_1) \oplus \ldots \oplus F_K(<m>||M_m)$
  - Is it a secure MAC_2 ?

Attack against MAC_2

- $MAC_2(M) = F_K(<1>||M_1) \oplus \ldots \oplus F_K(<m>||M_m)$
- $MAC_2(M_1,M_2) = t_1$
- $MAC_2(M_1,M_3) = t_2$
- $MAC_2(M_2,M_2) = t_3$

- $MAC_2(M_2,M_3) = t_1 \oplus t_2 \oplus t_3 = F_K(<1>||M_2) \oplus F_K(<2>||M_3)$

Keyed-hash Message Authentication Code

- Is H(K||m) a secure MAC if H is based on Merkle-Damgard ?
- Is H(m||K) a secure MAC if H is based on Merkle-Damgard ?
- The envelop technique $H(k_1||m||k_2)$

$$\mathrm{HMAC}_K(m) = h\Big((K \oplus opad) \,||\, h\big((K \oplus ipad) \,||\, m\big)\Big)$$

Extension attack on MAC(m)=H(K||m)

- $H(M)=H(M_1||M_2)=h(h(IV,M_1),M_2)$ is Merkle-Damgard construction with a 2-block message

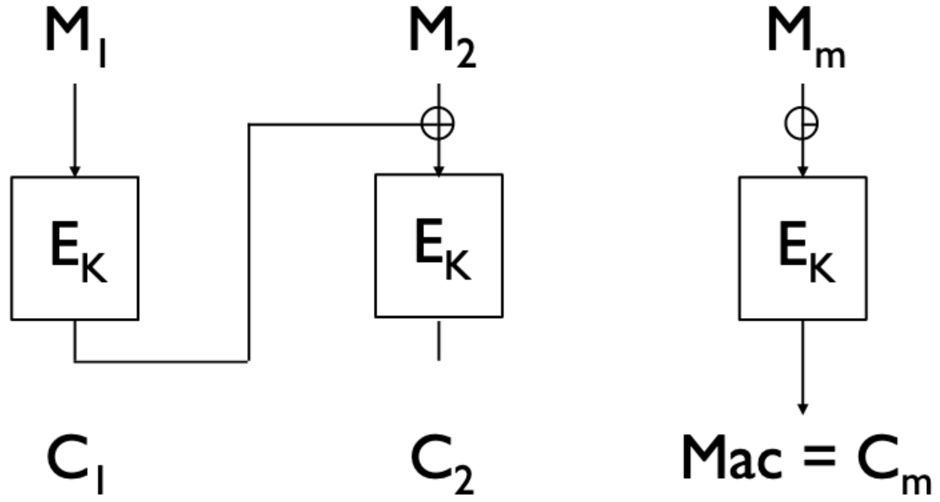
- Assume we know the MAC of message m: MAC(m)= H(K||m)=t
- We can compute the MAC of message m||N from t, without knowing the key K, MAC(m||N)=h(t,N)

- For MAC(m)= H(m||K) if we can compute a collusion for H: m ≠m' s.t. H(m)=H(m'), then we can forge MAC(m'||K) once we have a MAC for MAC(m||K)

- For SHA-3 hash function, SHA3(K||m) is a valid MAC since MD is not used

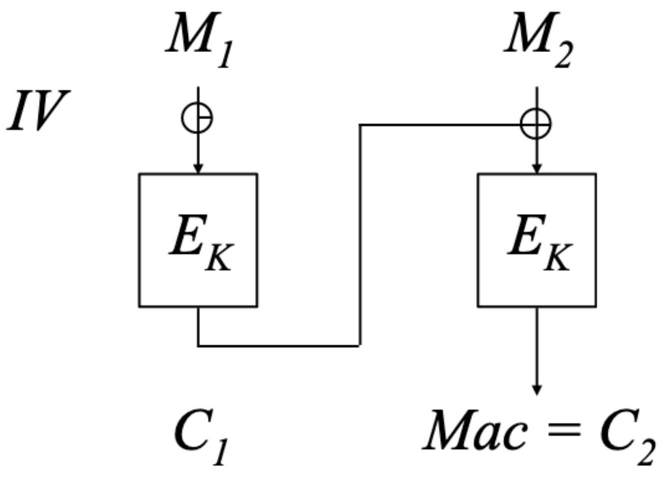
Block-cipher based MAC

- Unencrypted CBC-MAC
- There is no IV (initialization vector)
- Secure only for message M of the same length

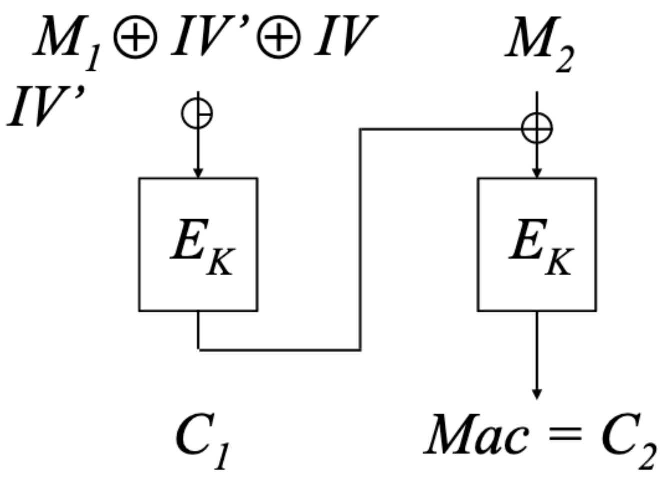


No IV in CBC-MAC

The integrity of the first block is not ensured if an IV is used



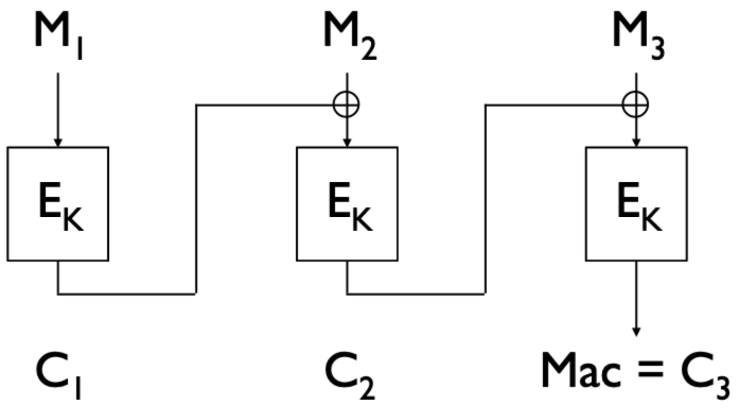
(M, IV, Mac)



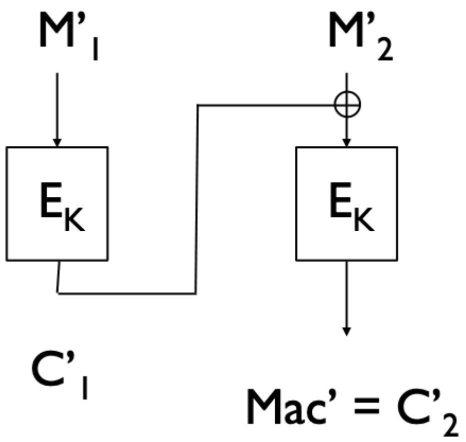
(M', IV', Mac)

Security against messages of different lengths

Let 2 arbitrary messages M and M'



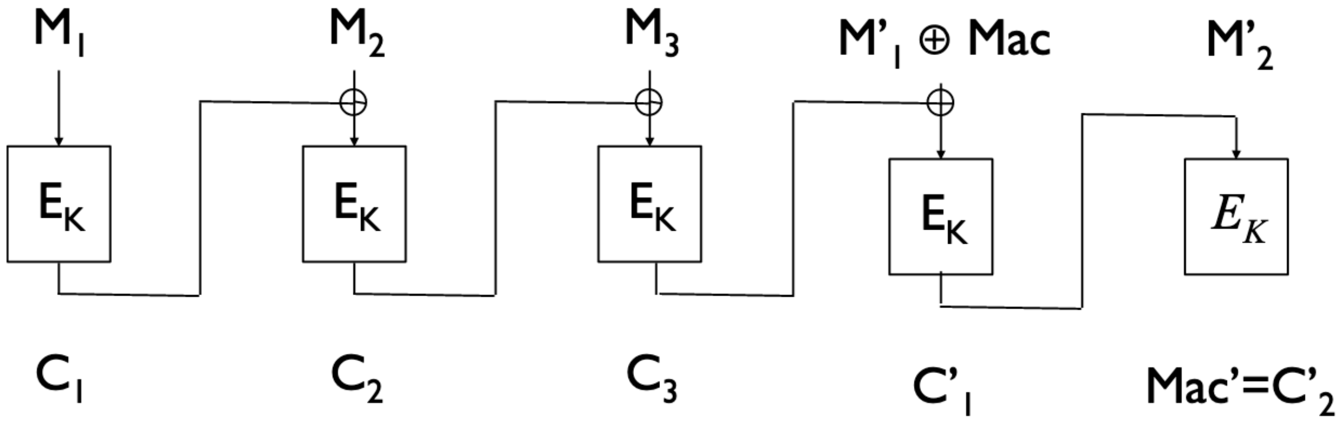
MAC(M) is C_3 = Mac



MAC(M') is C'_2 = Mac'

Security

Given MACs of M and M', it is possible to forge MAC of another message



Recovering the secret key is in 2^k MAC computation where k is the bit length of the used key (exhaustive search)

Better solution : Encrypted CBC-MAC