Use the PRF to build a secure MAC

In the real world not all security concerns are related to the ability or inability of an adversary to learn something about messages being sent.Specifically, when two parties communicate, they have the implicit assumption that the message sent by one party is indeed the message received by the other party. This expectation of **message integrity** is the source of a critical security concern. So, we need some mechanism that will enable communicating parties to know whether a message was tampered or not. Such mechanisms are called **message authentication codes (MAC).**

Definition:

A message authentication code (MAC) is an algorithm that is applied to a message. The output of the algorithm is a MAC tag that is sent along with the message. No polynomial time adversary can generate a valid MAC tag on any message that was not sent by the legitimate communicating parties.

The MAC prevents an adversary from modifying a message sent by one party to another. Both parties will share one secret key for this process. Since both parties are sharing the same key, it belongs to private key cryptography.

A MAC consists of three algorithms **Gen, Mac, Vrfy.**

Gen algo generates a secret key k , Mac algo generates MAC tags means mapping k and message m to a tag t . Vrfy algo receives key k and message m , tag t and outputs 1 if tag t is matched with given message m and k otherwise, outputs 0.

Construction:

We use the Pseudorandom function F to construct MAC. The MAC tag t is obtained by applying F on message m and breaking MAC involves guessing the behavior of F . (Probability of guessing value of a random function is 2^{-n} when the output length is n)

Fixed length MAC:

Let F be a pseudorandom function. Define a private-key encryption scheme for messages of length n as follows:

- Gen: on input 1^n, choose $k \leftarrow \{0,1\}^n$ uniformly at random and output it as the key.
- Enc: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, choose $r \leftarrow \{0,1\}^n$ uniformly at random and output the ciphertext

$$c := \langle r, F_k(r) \oplus m \rangle.$$

- Dec: on input a key $k \in \{0,1\}^n$ and a ciphertext $c = \langle r, s \rangle$, output the plaintext message

$$m := F_k(r) \oplus s.$$

The above construction is only for fixed-length messages (message length is n) , and we use this construction to build variable-length message MAC.

Variable-length MAC:

In this design, we add some extra information along with message while computing tag.

- To prevent replay attacks, we add sequence numbers.
- To prevent reordering of message blocks we add block index in series of blocks.
- To prevent combining different signature blocks we add one random identifier (it is same for every block).
- To ensure all blocks presence we add total number of blocks.

Construction:

Let $F : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ be a function such that for every $k \in \{0,1\}^n$, $F_k(\cdot)$ maps n-bit strings to n-bit strings. Define a variable-length MAC as follows:

- $\mathsf{Gen}(1^n)$: upon input 1^n, choose $k \leftarrow \{0,1\}^n$
- $\mathsf{Mac}_k(m)$: upon input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^*$ of length at most $2^{\frac{n}{4}-1}$, first parse m into d blocks $m_1, \cdots, m_d$, each of length $n/4$. (In order to ensure unique encoding, the last block is padded with 10^*.) Next, choose a random identifier $r \leftarrow \{0,1\}^{n/4}$.

  Then, for $i = 1, \ldots, d$, compute $t_i = F_k(r\|d\|i\|m_i)$, where i and d are uniquely encoded into strings of length $n/4$ and "$\|$" denotes concatenation.[a]

  Finally, output the tag $t = (r, t_1, \ldots, t_d)$.
- $\mathsf{Vrfy}_k(m,t)$: Upon input key k, message m and tag t, run the MAC-tag generation algorithm Mac except that instead of choosing a random identifier, take the identifier r that appears in t. Output 1 if and only if the tag that is received by running Mac with this r is identical to t.

[a] Notice that i and d can be encoded in $n/4$ bits because the length of the padded m is at most $2^{n/4}$.

Using the above construction, we can construct a secure message authentication code for every arbitrary length message. But to generate tag on message length $l.n$, it is necessary to apply pseudorandom function(block cipher) $4l$ times and the final mac tag length is $4.l.n$. So, to avoid this complexity we use **CBC-MAC construction**.

CBC-MAC construction:

This is based on CBC mode of encryption. The message is divided into blocks and a block cipher is applied on each block.

Basic CBC-MAC:

The basic CBC-MAC construction is as follows:

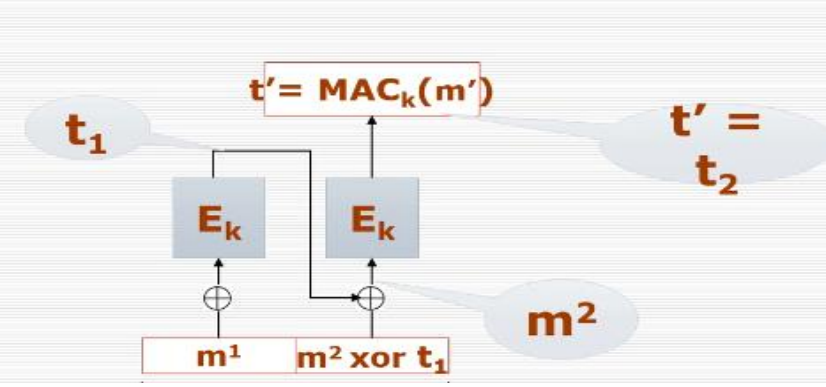
- $\mathsf{Gen}(1^n)$: upon input 1^n, choose a uniformly distributed string $k \leftarrow \{0,1\}^n$
- $\mathsf{Mac}_k(m)$: upon input key $k \in \{0,1\}^n$ and a message m of length $\ell \cdot n$, do the following:
  1. Denote $m = m_1, \ldots, m_\ell$ where each m_i is of length n, and set $t_0 = 0^n$.
  2. For $i = 1$ to ℓ, set $t_i \leftarrow F_k(t_{i-1} \oplus m_i)$ where $F : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ is a function.
  3. Output t_ℓ
- $\mathsf{Vrfy}_k(m,t)$: upon input key $k \in \{0,1\}^n$, a message m of length $\ell \cdot n$ and a tag t of length n, output 1 if and only if $t = \mathsf{Mac}_k(m)$

But the above construction is secured against fixed length messages but not against variable length messages. Let us see the reason for this.

Suppose we have two messages m^1, m^2 and their tags t^1, t^2.

Now apply block cipher on m^1 and it will gives t^1 , next apply XOR between t^1 and m^2 and apply block cipher on this and we get new tag t which is equals to t^2.

So, this kind of attack is possible here which makes this scheme not secure



A secure CBC-MAC for variable length messages:

We have the following ways to construct secure CBC-MAC for variable length messages.

1. Apply Pseudorandom function to block length l of the input message and obtain a key k_l . Now, compute CBC-MAC using the key and send the tag along with block length.
2. Prepend the message length to the message and then compute CBC-MAC.
3. Choose two different keys $k_1 \leftarrow \{0,1\}^n$ and $k_2 \leftarrow \{0,1\}^n$. Then compute CBC-MAC using the first key and using these results (t_1) Output the MAC-tag $t = F_{k_2}(t_1)$.