Author: Harrison Nicholls

HMAC

1 Intro

These scribe notes are for Larry Zheng and John Poole's presentation on the topic of HMAC. Their talk was in two main sections plus a portion for references. They began with a review of MAC, and followed by presenting HMAC.

2 Review of MAC

Larry and John started by giving the definitions for a general MAC scheme:
A MAC scheme consists of a key generation algorithm KG, tagging algorithm S, and verification algorithm V.

S: Given key k and message M, S produces tag t of message M such that $t \xleftarrow{\$} S(k, M)$.

V: Given key k, tag t, and message M, V returns $r \leftarrow V(k, M, t)$ where r denotes accepting or rejecting.

Correctness: The following correctness property should be satisfied by a MAC scheme:

$$Pr[t = \perp \vee V(k, M, t) = \mathsf{accept} \mid t \xleftarrow{\$} S(k, M)] = 1$$

The two definitions we use for MAC schemes are WUF-CMA (Weak unforgeability under chosen message attack) and SUF-CMA (strong unforgeability under chosen message attack). Each has a different associated game. Both the WUF-CMA and SUF-CMA games have the same functions: Tag, Vf, and Finalize. The WUF-CMA game is as seen below.

Subroutine `Initialize`

$K \xleftarrow{\$} \mathsf{KG}$; $S \leftarrow \emptyset$; $\mathsf{win} \leftarrow \mathsf{false}$

Subroutine `Tag`(M)

$S \leftarrow S \cup \{M\}$; Return $\mathsf{Tag}(K, M)$

Subroutine `Vf`(M, T)

$v \leftarrow \mathsf{Vf}(M, T)$

If $v = 1$ and $M \notin S$ then $\mathsf{win} \leftarrow \mathsf{true}$

Return v

Subroutine `Finalize`

Return win

The only difference in the strong setting (SUF-CMA) is that the game instead stores pairs (M, T) in S. To summarize: in the weak setting an adversary forges on a brand new message, and in the strong setting they forge on a new message tag pair.

A SUF-CMA secure MAC scheme can be built with a secure PRF. John and Larry then mentioned that there is a desire to make tags on long messages, but without block ciphers. This is because a sufficiently long message could be slow to generate a tag for using only block ciphers. The proposed solution that John and Larry focused on for this is using the faster option of hash functions, resulting in HMAC.

3 HMAC

The goal of HMAC is to create a secure PRF.

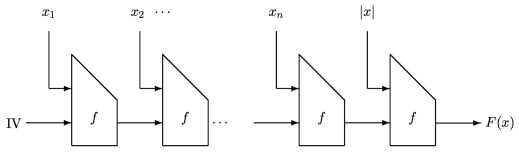
3.1 Syntax

Deterministic Function $F : K \times M \to Y$
$y \leftarrow F(k, M)$

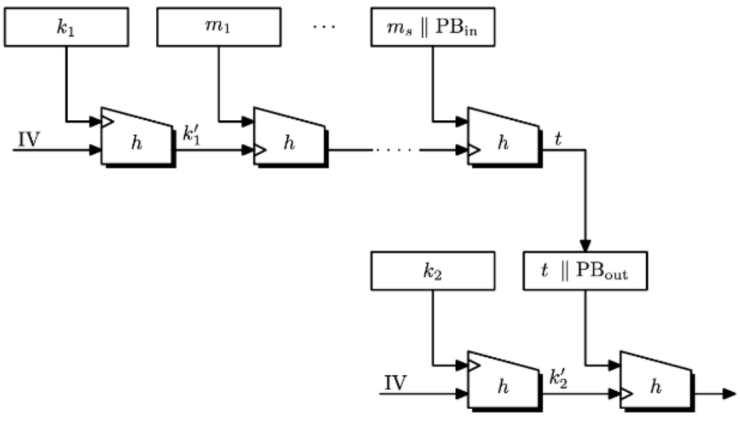
3.2 Security

Larry and John presented the two cases for a security experiment as follows:
Experiment where $b = 0 : y_i \leftarrow$ random function
Experiment where $b = 1 : y_i \leftarrow F(k, m_i)$

Larry and John then explained that we need to find a way to input a key, but hash functions don't take keys by default. They discussed compression functions, and how we can prepend a key to their inputs to incorporate the key as a secret bitstring. The other prerequisite, of course, is a hash function. They explained that for this presentation of HMAC, the hash functions are constructed under Merkle–Damgard construction, which relies on a compression function. Examples include MD5 and SHA-1. They showed the following as an example of a chained (iterated) hash function construction.



They then explained the 'two key nest' case, which comes closer to the true construction of an HMAC.



Finally, they gave a formal definition for a chained hash function, which I will not include here (see slide 14). It takes an arbitrary length string, and returns a fixed length digest, using padding and chaining.

4 Conclusion

To conclude, they explained that HMAC provides two main advantages. It requires only one key, and it is faster than any system that would use block ciphers. They discussed a security proof from 1996 with some flaws (a false assumption about MD5/SHA-1), and a later one from 2006 which establishes that a secure PRF implies a secure MAC. Overall, HMAC provides a reliable way to authenticate messages in a secure manner.