HMAC

Larry Zeng, John Poole

CSCI 388

March 2022

Definition

MAC scheme $\mathcal{L} = (K, S, V)$.
K is the key generation algorithm, S is a tagging algorithm, and V is a verification algorithm.

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

Given key k, tag t, and message M, V returns $r \leftarrow V(k, M, t)$ *accept* or *reject*

It should satisfy the correctness property

$$\Pr\left[t = \perp \vee V(k, M, t) = \textit{accept} \mid t \xrightarrow{\$} S(k, M)\right] = 1$$

Security Definition

WUF-CMA:

Subroutine Initialize

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

Subroutine $\mathsf{Tag}(M)$

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

Subroutine $\mathtt{Vf}(M, T)$

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

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

Return v

Subroutine Finalize

Return win

SUF-CMA: S is pairs (M,T).

We can build secure MAC under SUF-CMA with a secure PRF.

But...

AES is small

We want tags for long messages

=> Big Macs?

We have seen ECBC MAC, XCBC MAC, and CMAC so far.

But but...

Block Ciphers are slow

Hash functions are faster!

hmm. What to do?

HMAC Goal: Secure PRF

Syntax:

Deterministic Function $F : K \times M \rightarrow Y$

$y := F(k, M)$

Security:

Experiment 0: $y_i \leftarrow$ random function

Experiment 1: $y_i \leftarrow F(k, m_i)$

HMAC Construction: chaining?

Hash Functions take no key :(

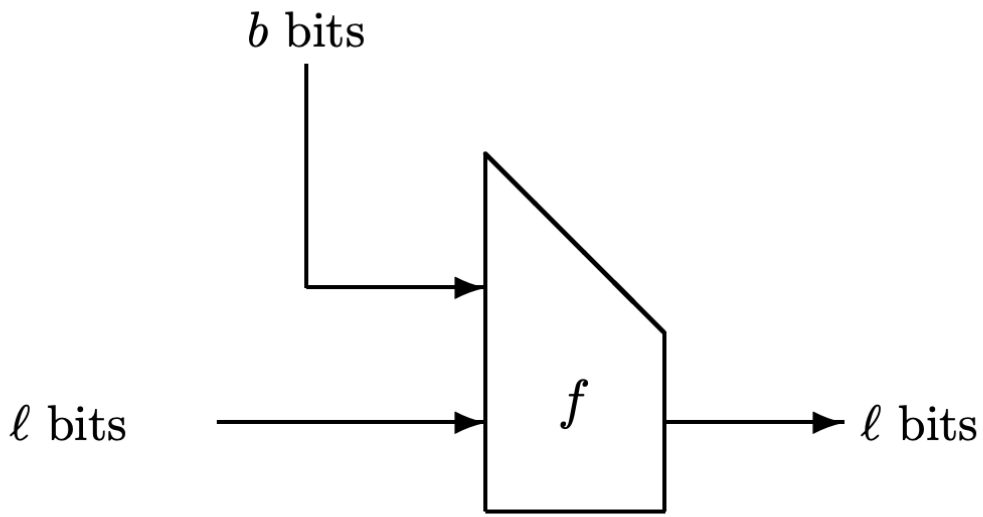
$=>$ Prepend to input?

Also, need to shrink (Compress) input to chain

Prerequisites: Compression Function

(1) Compression functions: a compression function compresses its data input. It takes in a fixed length message block to be compressed and a fixed length chaining variable whose length is shorter than the message block, and outputs data of the same length as the chaining variable. We want the compression function to be one-way and have the avalanche property (no malleability).

Compression Function (notice the ℓ)



We input a block of size b and a chaining variable of size ℓ to the compression function. The compression function outputs a block of size ℓ.

Prerequisites: Hash Function

(2) Hash function: a hash function takes in a variable length message and outputs a fixed length digest. In our presentation, the hash functions are constructed under Merkle–Damgård construction, which relies on a compression function. Such as MD5 and SHA-1.

Iterated Hash Function (hash big inputs)

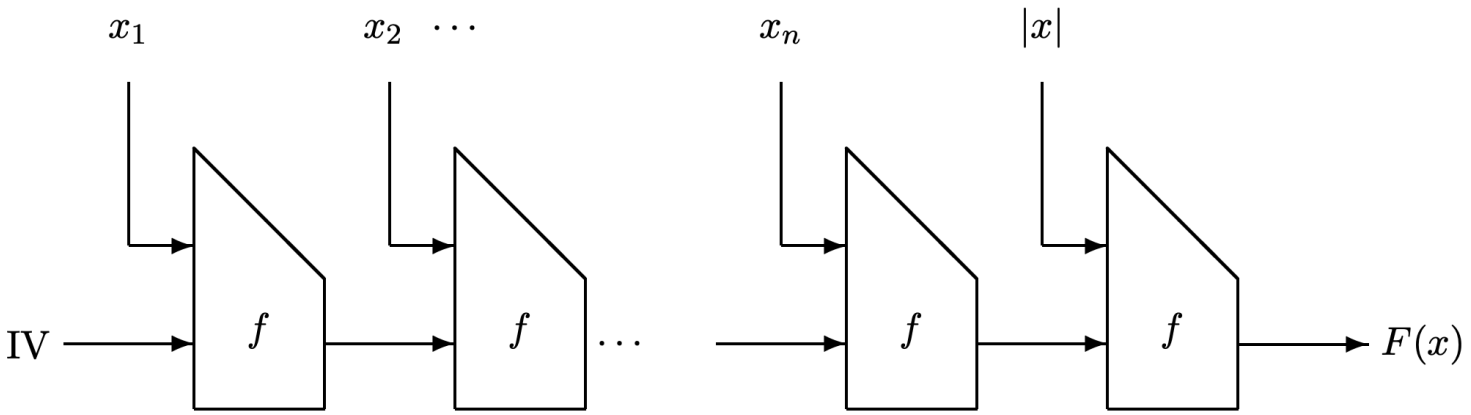


Figure 2: *The iterated construction of a hash function given a compression function f. The input $|x|$ to the last iteration illustrates the appending of the message length as in MD5 and SHA-1.*

Construction: HMAC (kinda)

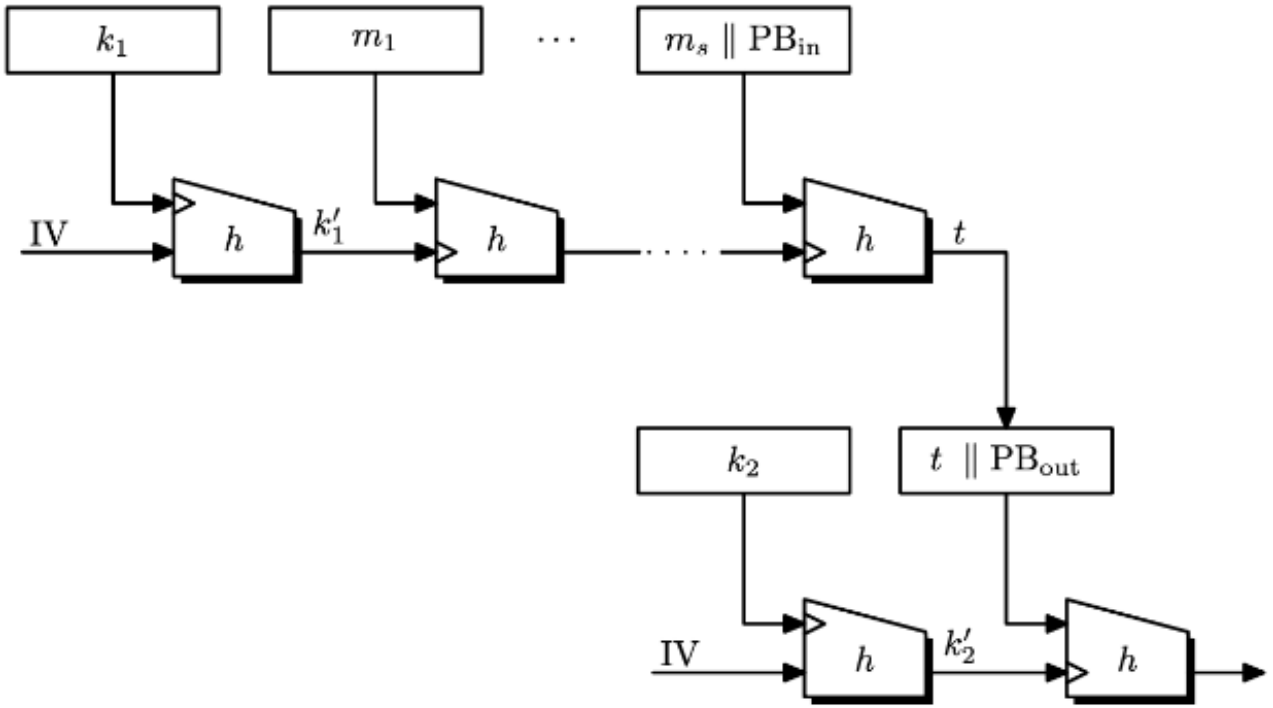


Figure 8.9: The two-key nest

Top Chain, H^*: iterated keyed hash function

Let $H^* : \{0,1\}^\ell \times \{0,1\}^{n \cdot b} \to \{0,1\}^c$ be defined as the following

Algorithm: $H^*(K, x)$

pad x so that $|x| \mod b = 0$

parse x into n blocks of length b: $x[1] \| \cdots \| x[n]$.

$a[0] \leftarrow K$

for $i = 1$ *to* n **do**

$\quad a[i] \leftarrow f(a[i-1], x[i])$

end

return $a[n]$

Let H be defined as $H(M) = H^*(IV, M)$ where IV is predefined.

Let H be a Merkle–Damgård hash function.
We define the tagging and verification algorithm for $\texttt{HMAC}(K, T, V)$ with key size b to be:

$$k \xleftarrow{\$} \{0,1\}^b$$

$$T(k, M) = H\big(k \oplus \texttt{opad} \parallel H(k \oplus \texttt{ipad} \parallel M \parallel PB_{in}) \parallel PB_{out}\big)$$

$$V(k, M, t) = \begin{cases} \textit{accept} & \text{if } T(k, M) = \perp \vee T(k, M) = t \\ \textit{reject} & \text{otherwise} \end{cases}$$

in which opad and ipad are defined as

$$\texttt{opad} = \texttt{0x5c} \text{ repeated until get } b \text{ bits}$$

$$\texttt{ipad} = \texttt{0x36} \text{ repeated until get } b \text{ bits}$$

PB_{in} and PB_{out} are just pads to make the inputs to H a valid size.

Advantages

The advantages of HMAC are

(1) It uses only one key

(2) hashing is fast

Analyzing HMAC as composed keyed functions

To analyze the tagging algorithm construction for HMAC,

$$T(k, M) = H(k \oplus \texttt{opad} \parallel H(k \oplus \texttt{ipad} \parallel M \parallel PB_{in}) \parallel PB_{out}),$$

we'll think of HMAC as the composition of two keyed functions:

$$T(k, M) = H_{out}(H_{in}).$$

note that

$$H_{in} ::= H(k \oplus \texttt{ipad} \parallel M \parallel PB_{in}) = H^*\big(f(IV, k \oplus \texttt{ipad}), M \parallel PB_{in})\big)$$

So, letting $K_{in}^k = f(IV, k \oplus \texttt{ipad})$,

$$H_{in} = H^*(K_{in}^k, M \parallel PB_{in}).$$

Analyzing HMAC as two keyed functions, cont'd

HMAC tags with $T(k, M) = H_{out}(H_{in})$.
We've shown $H_{in} = H^*(K_{in}^k, M \parallel PB_{in})$.
Now, Consider H_{out}.

$$H_{out} = H(k \oplus \text{opad} \| H_{in}) = f\left(f(IV, k \oplus \text{opad}), H_{in}\right)$$

Let $K_{out}^k = f(IV, k \oplus \text{opad})$. Substituting,

$$T(k, M) = f\left(K_{out}^k, H^*(K_{in}^k, M \parallel PB_{in}) || PB_{out}\right).$$

HMAC tags with two keyed functions, f and H^*. We can study the security of $f\left(\cdot, H^*(\cdot, M)\right)$ to learn about the security of HMAC!

Security Proof (0) 1996

Assume:

1. The underlying compression function is a PRF.
2. The underlying compression function is weakly collision resistant.

BUT! MD5/SHA-1 not weakly collision resistant!

Security Proof (1) 2006

Assume that the underlying compression function f is a PRF.

Then,

1. H^* is computationally almost universal (cAU). [H^* is from a family of Functions. It is Hard to guess a collision with no oracle]

2. If f is a PRF and H^* is cAU, their composition, $fH^*(K_{out}||K_{in}, M) = f(K_{out}, H^*(K_{in}, M))$, is also a PRF.

3. A secure PRF $\implies$ a secure MAC.

Security Proof (2)

Let f be a PRF and H^* be a cAU. Then
$fH^*(K_{out}\|K_{in}, M) = f(K_{out}\|H^*(K_{in}, M))$ is a PRF.
Note that the HMAC can be written as $f\left(K^k_{out}, H^*(K^k_{in}, M)\right)$. We are almost done!
Note that K^k_{out} and K^k_{in} are derived from a single key k. The proof for fH^* assumes that we have two different keys K_{out} and K_{in}. We need to say something about the relativity of the two keys doesn't create too much advantage.

References

(1) Keying Hash Functions for Message Authentication

(2) New Proofs for NMAC and HMAC: Security without Collision-Resistance

(3) Symmetric Cryptography Basics