Soumodipta Bose| 2021201086

Using Collision resistant hash function to build H-MACs(Theory)

HMAC or Hash-Based Message Authentication code is the industry standard since CBC-MAC is considered to be very slow. We will discuss how to create it using the already available api's in our crypto library.

Sketches/Ideas:

MAC using Hashing

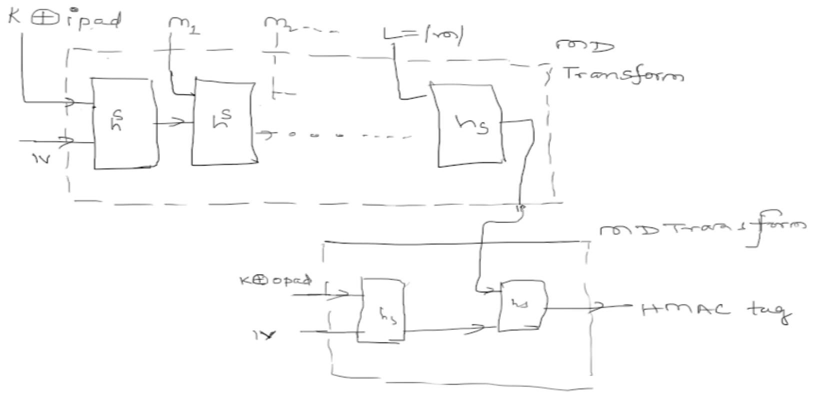
HMAC

(Gen, h) → Fixed length Hash function

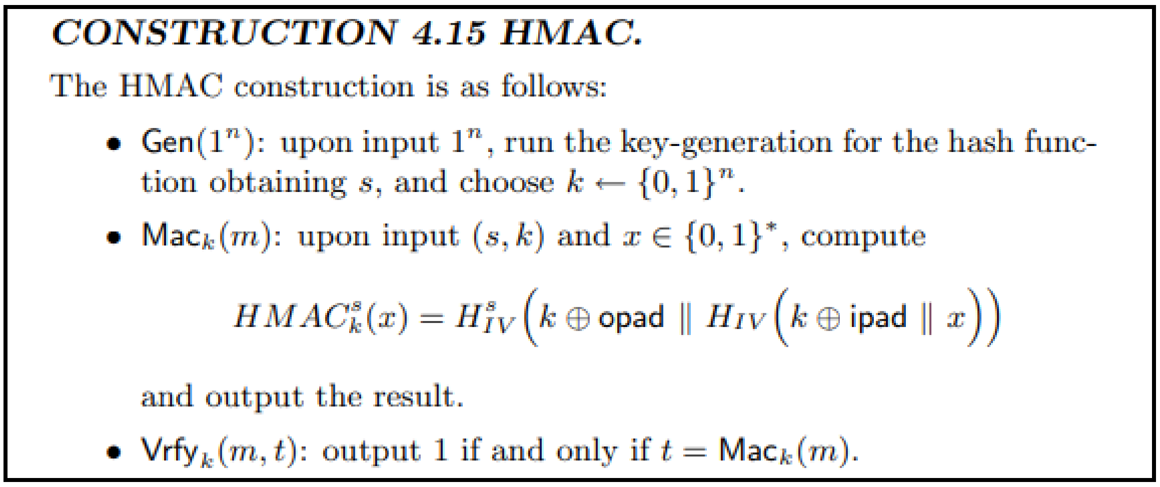
(Gen, H) → Applying MD Transform to Hash function

Constants : IV, opad ipad

HMAC tag for m = $H^s_{IV}((K \oplus opad) \| (H^s_{IV}(K \oplus ipad) \| m)$



Construction:



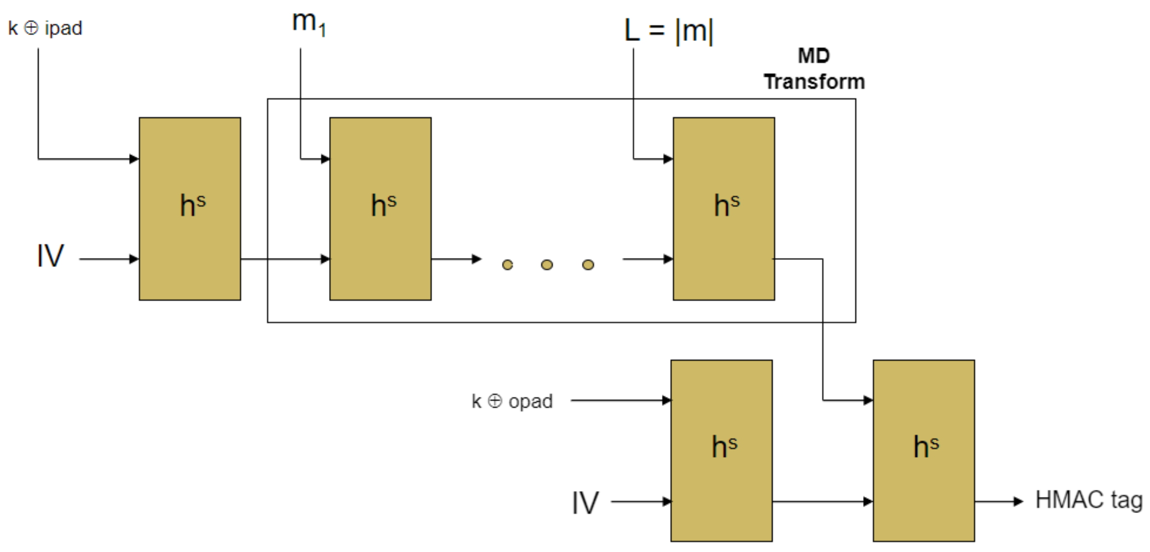
CONSTRUCTION 4.15 HMAC.

The HMAC construction is as follows:

- $\mathsf{Gen}(1^n)$: upon input 1^n, run the key-generation for the hash function obtaining s, and choose $k \leftarrow \{0,1\}^n$.
- $\mathsf{Mac}_k(m)$: upon input (s,k) and $x \in \{0,1\}^*$, compute

$$HMAC^s_k(x) = H^s_{IV}\Big(k \oplus \mathsf{opad} \parallel H_{IV}\big(k \oplus \mathsf{ipad} \parallel x\big)\Big)$$

  and output the result.
- $\mathsf{Vrfy}_k(m,t)$: output 1 if and only if $t = \mathsf{Mac}_k(m)$.



Constructing HMAC is relatively easy.

Firstly we have two binary strings ipad and opad equal to 0x36H and 0x5cH. Our HMAC takes 3 inputs k, iv, message. k is the key, iv is the initialization vector and message is our message on which we will apply HMAC to find the tag.

Here we will extrapolate our ipad and opad strings by repeating their sequences such that their length is equal to the key size.

The initial hash will be calculated using the iv and k xored with ipad. Lets call this h1. We will also calculate the same with k xored with opad and then hash using iv and lets call it h2.

My implementation uses the merkle damgard api in our crypto library to abstract and simplify some of our working. We will feed h1 and the message to merkle damgard. The result will then go with the calculated hash h2 in the last stage. This will give us our final hash, which will be our message authentication code tag.

We can then simply use the verification algorithm at the receiver end to verify our generated hash.