Winter 2018 CS 485/585 Introduction to Cryptography

Lecture 9

Portland State University
Lecturer: Fang Song

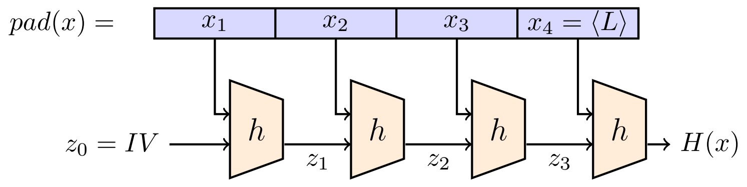
Feb. 6, 2018

Draft note. Version: February 11, 2018. Email fang.song@pdx.edu for comments and corrections.

Agenda

- (Last time) Hash functions, collision resistance, generic security, hash-and-mac
- HMAC
- authenticated encryption
- Quiz 2

Review: Merkle-Damgård



On input string x of length L

1. (**Padding**) Set $B := \lceil L/n \rceil$ i.e. number of blocks in x. Pad the last block with 0 to make it a full block. Denote the padded input $x_1, \ldots, x_B$, and let $x_{B+1} := \langle L \rangle$, i.e. the length represented as an n-bit string.
2. (IV) Set $z_0 := \mathsf{IV} = 0^n$.
3. (**Cascading**) For $i = 1, \ldots, B+1$, compute $z_i = h(z_{i-1} \| x_i)$.
4. Output z_{B+1}.

- $H(x) := h(x_1\|x_2)\|h(x_3\|h_4)\| \cdots$. Ignore the issue of variable-length output, is H collision resistant (assuming h is)?
- Picking a random IV? Hash function needs to be deterministic: the same message better produces the same digest no matter who and when hashes it. In SHA family, some peculiar IV rather than 0^n is used.
- Without encoding message length in last block? Explicit attack is possible depending on the compression function. Including the length makes the proof simple and universal. Read more at `https://eprint.iacr.org/2009/325`.

Application: MAC from hash functions

Hash and MAC paradigm $S'_k(m) := S_k(H(m))$: a generic solution; but don't use it literally because of two concerns.

1. In practice, hash functions have fixed small output length. Once one finds a collision offline, breaking any MAC scheme of this kind is trivial.

2. It relies on two primitives, a collision resistant hash and a secure MAC. It is preferable, from the implementation point of view, to rely on one primitive only.

$$\boxed{\text{How to make a keyed function } S_k \text{ to authenticate messages out of an unkeyed hash function } H\text{?}}$$

How about $S_k(m) := H(k\|m)$? Insecure if H is Merkle-Damgård type.[KL: Exercise 5.10]

HMAC. Following the hash-and-mac paradigm, can we instantiate both hash and MAC by a hash function? This two-layer approach leads us to HMAC, a popular scheme widely used on the Internet, constructed from MD hash functions[1].

[1] Commonly used: HMAC-SHA1 (should probably be replaced?) and HMAC-SHA-256.

$$H := \mathsf{MD}(h) \quad \text{Merkle-Damgård on } h\,;$$
$$k_1 := k \oplus \mathsf{ipad}, \quad k_2 := k \oplus \mathsf{opad}\,;$$
$$\mathsf{HMAC}_k(m) := H(k_2 \| H(k_1 \| m))\,.$$

Draw HMAC diagram. [KL: Fig. 5.2]

ipad := byte `0x36` repeated multiple times

opad := byte `0x5C` repeated multiple times

Connection to Hash-and-MAC paradigm.

$$\tilde{H}(x) := H(k_1 \| x)\,,$$
$$S_k(m) := h(k\|m)\,;$$
$$\Rightarrow \mathsf{HMAC}_k(m) = S_{k_{out}}(\tilde{H}(m))$$

Theorem 1 (Informal. [KL: Thm. 5.8]). *If k_{in} and k_{out} are pseudorandom, $\tilde{S}$ is a secure fixed-length MAC, then HMAC is secure.*

Connection to NMAC (encrypted Cascade). Main distinction:

- derive two keys from one uniform key k;

  $$k_1 = k \oplus \mathsf{ipad}, \quad k_2 := k \oplus \mathsf{opad}\,.$$

  and then define

  $$k_{in} := h(\mathsf{IV}\|k_1), \quad k_{out} := h(\mathsf{IV}\|k_2)\,.$$

- use a hash function instead of block ciphers.

  We can identify $\mathsf{HMAC}_k[h]$ with $\mathsf{NMAC}_{k_{in},k_{out}}[h]$.

Authenticated encryption

We have addressed two major security concerns: data secrecy and integrity. We introduced two primitives, (symmetric-key) encryption and MAC, to achieve them.

We have demonstrated that an encryption scheme does not necessarily provide authentication, and MAC doesn't need to hide the message at all. Can we achieve both simultaneously?

Informal goal: An encryption scheme such that no one can forge a valid ciphertext. We call it an *authenticated encryption.*

Given $\Pi = (G, E, D)$ CPA-secure, $\Sigma = (G', S, V)$ a secure MAC, how to construct $\tilde{\Pi} = (\tilde{G}, \tilde{E}, \tilde{D})$ that protects both secrecy and integrity? Let k_e and k_m be an encryption key and an signing key from Π and Σ respectively.

Encrypt-and-authenticate.

- G': $k_e \leftarrow G(1^n)$ and $k_m \leftarrow G'(1^n)$.
- $\tilde{E}_{k_e,k_m}(m)$: compute $c_1 = E_{k_e}(m)$ and $c_2 = S_{k_m}(m)$; output $c := (c_1, c_2)$.
- $\tilde{D}_{k_e,k_m}(c)$: let $c = (c_1, c_2)$. Compute $m \leftarrow D_{k_e}(c_1)$, and output m iff. $V_{k_m}(m, c_2) = 1$.

The tag c_2 may reveal information about m. More generally, most MACs are deterministic. $\tilde{\Pi}$ will not be CPA-secure.

Authenticate-then-encrypt.

- G': $k_e \leftarrow G(1^n)$ and $k_m \leftarrow G'(1^n)$.
- $\tilde{E}_{k_e,k_m}(m)$: compute $t := S_{k_m}(m)$, and output $c := E_{k_e}(m\|t)$.
- $\tilde{D}_{k_e,k_m}(c)$: compute $m\|t \leftarrow D_{k_e}(c)$, and output m iff. $V_{k_m}(m, t) = 1$.

Some instantiation OK (e.g., MAC then randomized counter mode). Not secure in general (e.g., broken with CBC-ENC [KL: P134]).

Encrypt-then-authenticate.

- G': $k_e \leftarrow G(1^n)$ and $k_m \leftarrow G'(1^n)$.
- $\tilde{E}_{k_e,k_m}(m)$: compute $c_1 := E_{k_m}(m\|t)$, $c_1 := S_{k_m}(c_1)$; output $c := (c_1, c_2)$.
- $\tilde{D}_{k_e,k_m}(c)$: compute $V_{k_m}(c_1, c_2)$; if accepting, output $m \leftarrow D_{k_e}(c_1)$.

Example. Galois Counter Mode (GCM): random counter mode + Carter-Wegman MAC.

Theorem 2 ([KL: Thm. 4.19])**.** *If* Π *CPA-secure and* Σ *strongly-secure MAC, then* $\tilde{\Pi}$ *is an authenticated encryption scheme.*

Intuition: Signing the ciphertext doesn't leak additional information. To forge a valid ciphertext, one has to forge on Σ. Here we need a strongly-secure MAC, since different messages could lead to the same c_1.
It is crucial to use independent keys. Some scheme will be broken under the same key $k_e = k_m$.

What do we mean exactly by authenticated encryption? We give a formal definition integrating a strong notion of secrecy (against chosen-ciphertext-attacks) and a notion of unforgeability.