Cryptographic Hash Functions

Reading: Chapter 5 of Katz & Lindell

Hash function

- A function mapping from a larger domain to a smaller range (thus not injective).
- Applications:
  - Fast lookup (hash tables)
  - Error detection/correction
  - Cryptography: cryptographic hash functions
  - Others
- Different applications require different properties of hash functions.

Cryptographic hash function (Informal)

- Hash functions: $h: X \to Y, \ |X| > |Y|.$
- E.g., $h: \{0,1\}^* \to \{0,1\}^n, \quad h: \{0,1\}^* \to Z_n,$
  $h: \{0,1\}^k \to \{0,1\}^l,$ with $k > l$.
- In the last case, h is also called a compression function.
- For cryptographic applications, $h(m)$ is intended to be a fingerprint or digest of m.
- A classical application is to store (username, password) as $(\text{username}, h(\text{password}))$ to protect the secrecy of passwords.
- For this application, what property is required of h?

Security requirements (Informal)

- Pre-image: if $h(m) = y,\ m$ is a pre-image of y.
- Each hash value typically has multiple pre-images.
- Collision: a pair $(m, m'),\ m \neq m',$ s.t. $h(m) = h(m')$.

- (Informal) A hash function h is said to be:
  - Pre-image resistant: given a hash value $y,$ it is computationally infeasible to find a pre-image of y.
  - Second pre-image resistant: given a message m, it is infeasible to find a second pre-image of $y = h(m)$.
  - Collision resistant: if it is infeasible to find a collision.

- Loosely speaking,

$$\text{Collision resistant} \Rightarrow \text{Second pre-image resistant}$$

$$\Rightarrow \text{Pre-image resistant}$$

- For cryptographic applications, a hash function is required to be collision resistant.

Hard to define collision-resistant hash functions

- In practice, a fixed hash function h is used.
- However, there is a technical difficulty in defining collision-resistance for a fixed hash function h.
- Try this "definition": A hash function $h:\{0,1\}^* \to \{0,1\}^n$ is collision-resistant if for every polynomial-time algorithm A,

$$\Pr\left[A^{h(\cdot)}(1^n) \text{ successfully produces a collision for } h \right] \le negl(n).$$

- Problems with this definition:
  - For any $x, x' \in \{0,1\}^*,\ x \ne x',$ let $A_{x,\,x'}$ denote the algorithm that simply prints x, x'.
  - For any h, $\exists$ an algorithm that outputs a collision with prob 1. Thus, no hash function would be collision resistant.

Hash function (formal definition)

- A hash function (with output length $l(n)$) is a pair of PPT algorithms $(Gen,\ H)$:
  - $Gen(1^n)$ outputs a key $s \in I_n$ for some index set I_n. (Assume that 1^n is implicit in s.)
  - H takes as input a key s and a string $x \in \{0,1\}^*$ and outputs a string $H^s(x) \in \{0,1\}^{l(n)}$.
- If H^s is defined only for $x \in \{0,1\}^{l'(n)}$, where $l'(n) > l(n)$, then $(Gen,\ H)$ is a fixed-length hash function (also called a compression function) for input of length $l'(n)$.

Remarks

- H is a keyed function with two inputs, and $H^s(x) = H(s, x)$.
- The key s is not necessarily a uniform string in $\{0,1\}^n$, and is not a secret; it is more like an index than a key. To emphasis this, we write H^s instead of H_s.
- For convenience, we often also refer to H^s as a hash function.
- An example compression function H: (may skip)

$$H^{(p,q,g,h)}(x,y) = g^x h^y \bmod p, \quad (x,y) \in Z_q \times Z_q, \text{ where}$$

$$I_n = \left\{ \begin{array}{c} (p,q,g,h) : p,\ q \text{ primes},\ p = 2q+1,\ |q| = n \\ g,\ h \text{ generators of } Z_q^* \end{array} \right\}$$

Collision-resistant hash function

- Let $\Pi = (Gen, H)$ be a hash function.
- Collision-finding experiment $\text{Hash-coll}_{A,\Pi}(n)$:
  - A key is generated, $s \leftarrow Gen(1^n)$.
  - The adversary A is given s and outputs $x, x' \in \{0,1\}^*$ (or $x, x' \in \{0,1\}^{l'(n)}$ if Π is fixed-length).
  - The output of the experiment is 1 if and only if $x \neq x'$ and $H^s(x) = H^s(x')$. //A finds a collision//
- Definition: A hash function $\Pi = (Gen, H)$ is collision-resistant if for all PPT adversaries A, there is a $negl(n)$ such that

$$\Pr\left[\text{Hash-coll}_{A,\Pi}(n) = 1\right] \leq negl(n).$$

Remarks

- $\Pr\left[\text{Hash-coll}_{A,\Pi}(n)=1\right]$

  $= \Pr\left[A \text{ finds a collision for } H^s : s \leftarrow Gen(1^n)\right]$

  $= \sum_{s\in I_n} \Pr[s] \cdot \Pr\left[A \text{ finds a collision for } H^s\right]$

  $=$ the probability that A finds a collision for a randomly picked hash function H^s.

- For different H^s, A may succeed with different probabilities.

How to construct a collision-resistant hash function

$$H^s : \{0,1\}^* \rightarrow \{0,1\}^n \ ?$$

- Provably collision-resistant hash functions can be constructed from claw-free pairs of one-way permutations. (Section 10.2 of Delfs & Knebl)
- In practice, hash functions are constructed from compression functions

$$h^s : \{0,1\}^{n+r} \rightarrow \{0,1\}^n$$

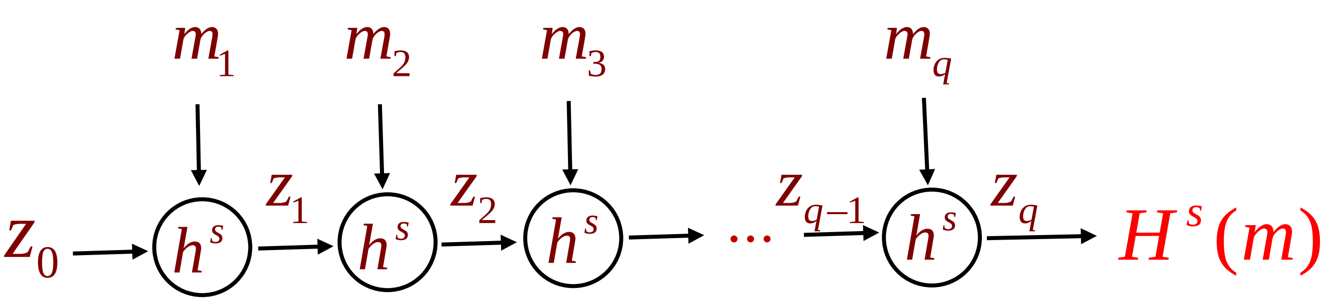
by a process called Merkle-Damgard's construction.

Merkle-Damgard construction

Construct a hash function $H^s : \{0,1\}^* \to \{0,1\}^n$

from a compression function $h^s : \{0,1\}^{n+r} \to \{0,1\}^n$.   // $r = r(n)$ //

1. For $m \in \{0,1\}^*$ of length less than 2^r, add a padding so that the length of the result is a multiple of r.
   - padding = 10...0 $|m|$, where $|m|$ is the original length of m.
2. Let padded $m = m_1 m_2 \ldots m_q$, where $|m_i| = r$.
3. Let $z_0 = 0^n$ and $z_i = h^s(z_{i-1} \| m_i)$ for $1 \le i \le q$.   // z_0 is the IV //
4. The hash value is $H^s(m) = z_q$.



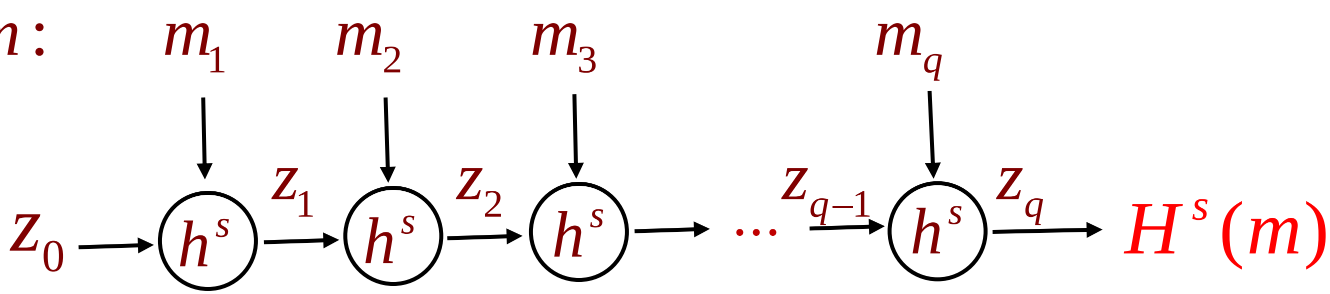
Theorem: If (Gen, h) is collision resistant, then so is (Gen, H).

Proof. We will show that if H is not collision resistant, then h is not collision resistant. Specifically, whenever the adversary can find a collision (m, m') for H^s, then it can find a collision for h^s. Consider two cases:

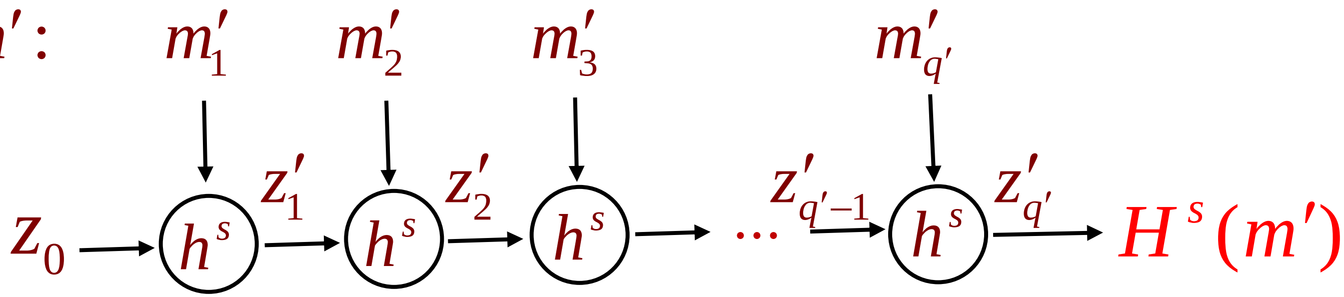
- $|m| \neq |m'|$. In this case, $m_q \neq m_{q'}$, and so $(m_q, z_{q-1}) \neq (m'_{q'}, z'_{q'-1})$. But $h^s(m_q, z_{q-1}) = H^s(m) = H^s(m') = h^s(m'_{q'}, z'_{q'-1})$, a collision for h^s.

- $|m| = |m'|$. In this case, $q = q'$. Since $H^s(m) = H^s(m')$, there exists an i such that $(m_i, z_{i-1}) \neq (m'_i, z'_{i-1})$ but $h^s(m_i, z_{i-1}) = h^s(m'_i, z'_{i-1})$, which is a collision for h^s.

padded m:



padded m':



If the adversary can find a collision (m, m') for H^s, then it can find a collision for h^s.

- We have shown that for every key s,

  Whenever A can find a collision for H^s, it can find a collision for h^s.

- So, for every key s,

$$\Pr\left[A \text{ finds a collision for } h^s\right] \geq \Pr\left[A \text{ finds a collision for } H^s\right]$$

- So,

$$\Pr\left[A \text{ finds a collision for } h^s : s \leftarrow Gen(1^n)\right]$$
$$\geq \Pr\left[A \text{ finds a collision for } H^s : s \leftarrow Gen(1^n)\right]$$

How large should ℓ be?

$$H^s : \{0,1\}^* \rightarrow \{0,1\}^\ell$$

- Minimum requirement: ℓ must be large enough for H^s to resist the birthday attack.
- Birthday attack: randomly generate a set of messages $\{m_1, m_2, \ldots, m_k\}$, and check if $H^s(m_i) = H^s(m_j)$ for some $i \neq j$.
- Why is it called a birthday attack?
- Birthday problem: In a group of k people, what is the probability that at least two of them have a same birthday?
  - Having a same birthday $=$ a collision.

Birthday attack's success rate

- If k objects are each assigned a random value in $\{1,\ 2,\ ...,\ N\}$, the probability of a collision is

$$p = 1 - 1 \cdot \frac{N-1}{N} \cdot \frac{N-2}{N} \cdot \cdots \cdot \frac{N-k+1}{N} \quad \text{(i.e., } 1 - \Pr[\text{no collision}]\text{)}$$

$$= 1 - \prod_{i=1}^{k-1} \left(1 - \frac{i}{N}\right) \qquad \text{(note: } 1 - x \le e^{-x} \text{ if } 0 < x < 1\text{)}$$

$$\ge 1 - \prod_{i=1}^{k-1} e^{-i/N} = 1 - e^{-\sum_{1 \le i \le k-1} i/N} = 1 - e^{-k(k-1)/2N}$$

- $p \ge 1/2 \quad \text{if} \quad k \ge 1.17\sqrt{N}.$

- Birthday paradox: with $N = 365$, $p \ge 1/2$ for k as small as 23.

- Define $\Pr[C_i] = \Pr[\text{object } i \text{ collides with some object } j < i]$.
- The birthday attack's success probability p satisfies:

$$\begin{aligned} p &= \Pr[C_1 \vee C_2 \vee \cdots \vee C_k] \\ &\le \Pr[C_1] + \Pr[C_2] + \cdots + \Pr[C_k] \\ &\le \frac{0}{N} + \frac{1}{N} + \frac{2}{N} + \cdots + \frac{k-1}{N} \\ &= \frac{k(k-1)}{2N} \quad \Rightarrow \quad k \ge \sqrt{2pN} \end{aligned}$$

- For a hash function $H : \{0,1\}^* \to \{0,1\}^\ell$, $N = 2^\ell$.
- To resist the birthday attack, N should be large enough that generating $k \ge \sqrt{2pN}$ messages is practically infeasible.
- Currently, a minimum of $\ell \ge 128$ is recommended.
- For $\ell = 128$, it will take $k \ge 2^{50}$ to have a successful rate of $p = 2^{-29}$.

The Secure Hash Algorithm (SHA-1)

- an NIST standard.
- using Merkle-Damgard construction.
- input message m is divided into blocks with padding.
- padding = $10...0|m|$, where $|m| \in \{0,1\}^{64}$.
- thus, message length is limited to $|m| \leq 2^{64} - 1$.
- block = 512 bits = 16 words = $W_0 \| \ldots \| W_{15}$.
- IV = a constant of 160 bits = 5 words = $H_0 \| \ldots \| H_4$.
- resulting hash value: 160 bits.
- underlying compression function $h: \{0,1\}^{160+512} \rightarrow \{0,1\}^{160}$, a series (80 rounds) of $\wedge$, $\vee$, $\oplus$, $\neg$, +, and Rotate on words W_i's & H_i's.

Is SHA-1 secure?

- $\ell = 160$ is big enough to resist birthday attacks for now.
- There is no mathematical proof for its collision resistance.
- In 2004, a collision for a 58-round SHA-1 was found.
- Newer SHA's have been included in the standard:
  - SHA-256, SHA-384, SHA-512.
  - These are called the SHA-2 family.
- SHA-3 is currently undergoing standardization.
- On 2/23/2017, Google researchers announced the first SHA-1 collision.

  News article

  https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html?m=1

Application of hash functions to MACs

K&L Section 5.3

Hash-then-MAC: basic idea

- A general MAC scheme with $M = \{0,1\}^*$ can be constructed using the hash-then-MAC paradigm. To compute a tag t for $m \in \{0,1\}^*$,
  - We first hash m to a block $\tilde{m} \in \{0,1\}^{l(n)}$, using a collision-resistant hash function.
  - Then compute a tag t from $\tilde{m}$, using a secure $l(n)$-bit fixed-length MAC scheme.

$$m \in \{0,1\}^* \xrightarrow{\text{hash } H^s} \tilde{m} \in \{0,1\}^{l(n)} \xrightarrow{l(n)\text{-bit MAC}_k} t$$

Hash-then-MAC: Formal Definition

- (Gen_H, H): a collision-resistant hash function with output length $l(n)$.
- $(Gen_M, Mac, Vrfy)$: a fixed-length MAC for messages of length $l(n)$.
- Construct a general MAC scheme $\Pi' = (Gen', Mac', Vrfy')$:
  - Gen': On input 1^n, output a hash key $s \leftarrow Gen_H(1^n)$ and a MAC key $k \leftarrow_u \{0,1\}^n$. The key is $k' = (k, s)$
  - Mac': On input a key (k, s) and a message $m \in \{0,1\}^*$, output $t \leftarrow Mac_k\left(H^s(m)\right)$.
  - $Vrfy'$: On input a key (k, s), a message $m \in \{0,1\}^*$, a tag t, output $Vrfy_k\left(H^s(m),\ t\right)$.

Hash-then-MAC: Security

- Theorem: If (Gen_H, H) is a collision resistant and $(Gen_M,\ Mac,\ Vrfy)$ is secure , then the MAC scheme $\Pi' = (Gen',\ Mac',\ Vrfy')$ constructed above is secure.

- Remarks:
  - The MAC scheme Π' is secure, even if the hash key s is known to the adversary.
  - The MAC key k must be kept secret.

$$m \in \{0,1\}^* \xrightarrow{\text{hash } H^s} \tilde{m} \in \{0,1\}^{l(n)} \xrightarrow{l(n)\text{-bit MAC}_k} t$$

MACs in practice

- In the hash-then-MAC paradigm, we need a collision-resistant hash function and a fixed-length MAC/pseudorandom function.

$$m \in \{0,1\}^* \xrightarrow{\text{hash } H^s} \tilde{m} \in \{0,1\}^{l(n)} \xrightarrow[l(n)\text{-bit } F_k]{l(n)\text{-bit MAC}_k} t$$

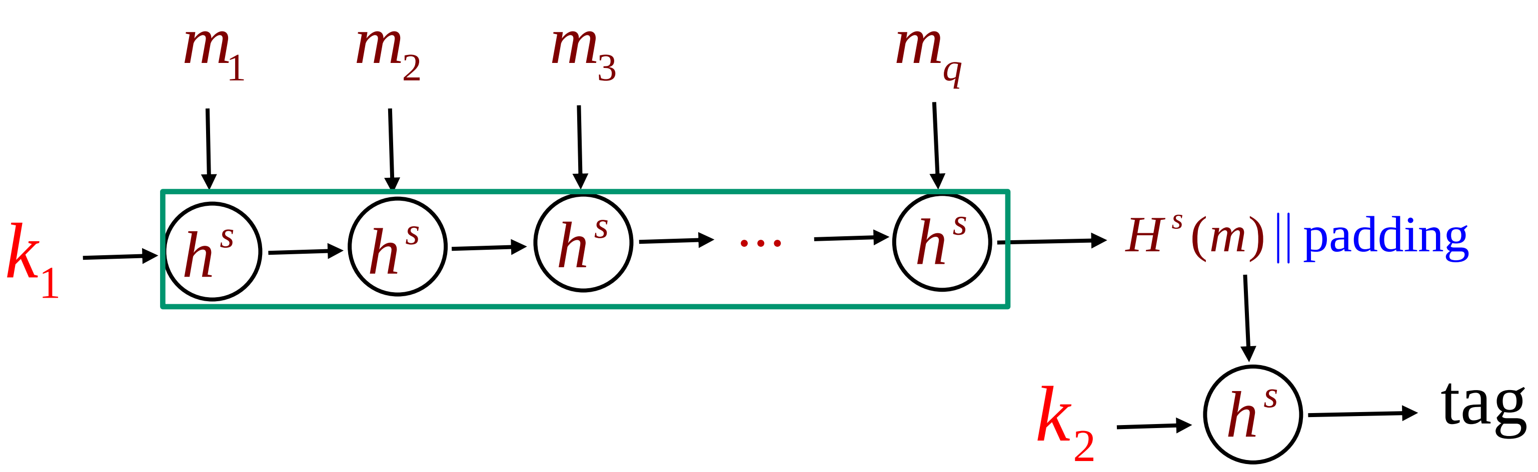
- In practice, people like to use just a hash function or just a pseudorandom function:
  - HMAC (hash-based MAC)
  - CBC-MAC (pseudorandom function based MAC)

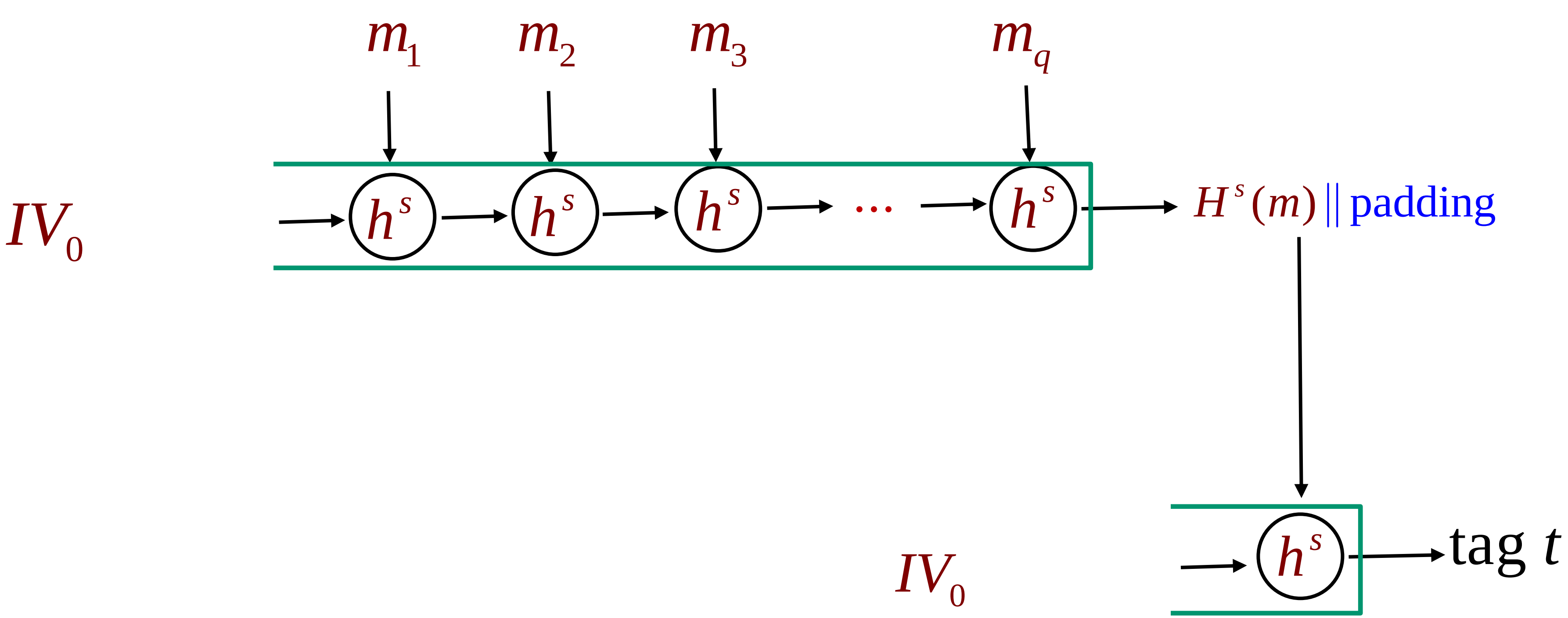
HMAC: basic idea

- HMAC is based on the idea:

$$m \in \{0,1\}^* \xrightarrow{H^s_{k_1}} H^s_{k_{\text{in}}}(m) \xrightarrow{h^s_{k_2}} t := h^s_{k_2}\left(H^s_{k_1}(m) \,\|\, \text{padding}\right)$$

- Two keys are used as IVs: k_1 and k_2, each of length n.
- Unfortunately, a standard hash function (e.g., SHA-1) usually has a fixed IV, say IV_0, which cannot be changed by users.





- Then we have HMAC with keys $(k_{\text{in}}, k_{\text{out}})$:

$$t := H^s\left(k_{\text{out}} \,\|\, H^s(k_{\text{in}} \,\|\, m)\right)$$

HMAC

- A FIPS standard for constructing MAC from a hash function H^s. Conceptually,

$$\mathrm{HMAC}_k(m) = H^s\left(k_{\text{out}} \,\|\, H^s(k_{\text{in}} \,\|\, m)\right)$$

 where k_{in} and k_{out} are two keys generated from a main key k.

- Various hash functions (e.g., SHA-1, MD5) may be used for H^s.

- If we use SHA-1, then HMAC is as follows:

$$\mathrm{HMAC}_k(m) = \text{SHA-1}\left(k \oplus opad \,\|\, \text{SHA-1}(k \oplus ipad \,\|\, m)\right)$$

 where

 - k is padded with 0's to 512 bits
 - $ipad = 3636\cdots36$ (x036 repeated 64 times)
 - $opad = 5\text{c}5\text{c}\cdots5\text{c}$ (x05c repeated 64 times)

Security of HMAC

- Loosely speaking, HMAC is secure if
  - the underlying compression function h is collision-resistant (and hence the hash function H is collision-resitant)
  - and h^s behaves like a pseudorandom function.
- In the hash-then-MAC paradigm, the hash H^s does not need a secret key. In HMAC, the key k_{in} is introduced to enhance the security.

Toss a coin by email

- Problem: Alice and Bob want to toss a coin by email to decide who is going to pay for dinner.
- A proposed solution:
  - Use a collision resistant hash function h.
  - Alice chooses a string x_1 and compute $y_1 := h(x_1)$.
  - Bob chooses a string x_2 and compute $y_2 := h(x_2)$.
  - Alice and Bob exchange y_1 and y_2. //commit but hide x_1 and x_2//
  - Alice and Bob exchange x_1 and x_2. //reveal x_1 and x_2//
  - Alice and Bob check if $y_2 := h(x_2), y_1 := h(x_1),$ respectively.
  - Alice and Bob compute a boolean value from x_1 and x_2 (e.g., take the XOR of the last bits).
- Is the proposed scheme "secure/fair"?