Computer Science 418

Hash Functions and Message Authentication Codes

Mike Jacobson

Department of Computer Science
University of Calgary

Week 7

Outline

Hash Function

Often referred to as the "work horse" of cryptography — they are ubiquitous in crypto.

Definition 1 (Hash function)

A function $H : \{0,1\}^* \rightarrow \{0,1\}^m$ ($m \in \mathbb{N}$) that is easy to compute. An image $x = H(M)$ is referred to as a *message digest* or a *digital fingerprint* or a *checksum* or simply a *hash*.

Hash functions thus satisfy two properties:

- *Compression*: H maps an input M of arbitrary bit length to an output of fixed bit length.
- *Ease of computation*: for any input M, $H(M)$ is easy to compute.

Cryptographic Requirements

Desirable properties for hash functions in the context of cryptography:

- *Pre-image resistance*: given any hash value x, it is computationally infeasible to find *any* input M for which $H(M) = x$.
- *Second pre-image resistance* or *Weak collision resistance*: given any M, it is computationally infeasible to find $M' \neq M$ with $H(M) = H(M')$.
- *Collision resistance* or *strong collision resistance*: it is computationally infeasible to find two distinct inputs M and M' such that $H(M) = H(M')$.

Note that collision resistance is the strongest of these three requirements. In other words: collision resistance $\Rightarrow$ weak collision resistance $\Rightarrow$ pre-image resistance

Uses of Cryptographically-Secure Hash Functions

Definition 2

A hash function is *cryptographic(ally secure)* if it is collision resistant.

Some example applications:

- In digital signatures to prevent impersonation (sign $H(M)$ instead of M — later)
- Data integrity without secrecy (*e.g.* downloading large files, compare checksum before and after download)
- Data integrity with secrecy (see below)
- Commitment (can verify $H(M)$ to see if M was committed to)
- Randomness (*e.g.* one-time passwords, OAEP — later)

Eg. Data Integrity with Secrecy

Using hashing plus encryption:

- Sender sends $C = E_K(M \| x)$ with $x = H(M)$
- Receiver decrypts C to obtain M', x' and checks that $H(M') = x'$.

Idea:

- Adversary cannon manipulate ciphertext blocks in such a way that $H(M') = x'$.
- May be possible if H is not cryptographically secure (eg. WEP: combination of stream cipher and checksum).

SHA-1

Secure Hash Algorithm 1: developed by NIST in 1993 (FIPS 180 and FIPS 180-1).

- Iterated round hash function with hash length 160 bits
- Can now find SHA-1 collisions in 2^{60} attempts.
- Longer versions still certified for use (up to 512 bits)

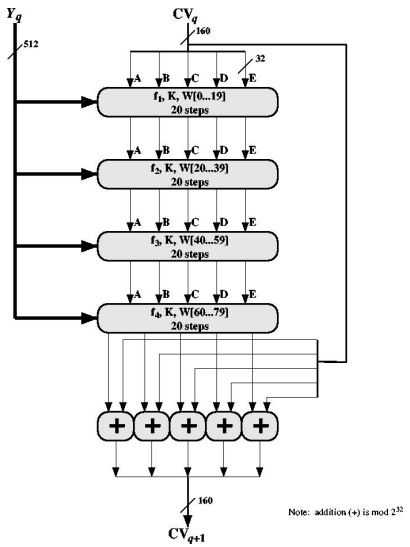
SHA-1: Description

1. Pad message so its length is 448 mod 512 (always pad with a single 1 bit followed by 0s — 1 indicates beginning of padding)
2. Append a 64-bit unsigned integer (most significant byte first). This integer represents the length of the original message before padding.
3. Initialize a 5-word (160-bit) buffer $CV_0 = (A, B, C, D, E)$ to

$$A = \texttt{67452301}, \quad B = \texttt{EFCDAB89}, \quad C = \texttt{98BADCFE},$$
$$D = \texttt{10325476}, \quad E = \texttt{C3D2E1F0}\ .$$

4. Process message in 16-word (512-bit) chunks:
   - Message block Y_q is processed with current buffer CV_q via four rounds.
   - CV_{q+1} is produced by adding wordwise (modulo 2^{32}) CV_q to the output of the fourth round.
5. Hash value is the final buffer value CV_L

SHA-1 Processing of a Single Block



SHA-1 Steps

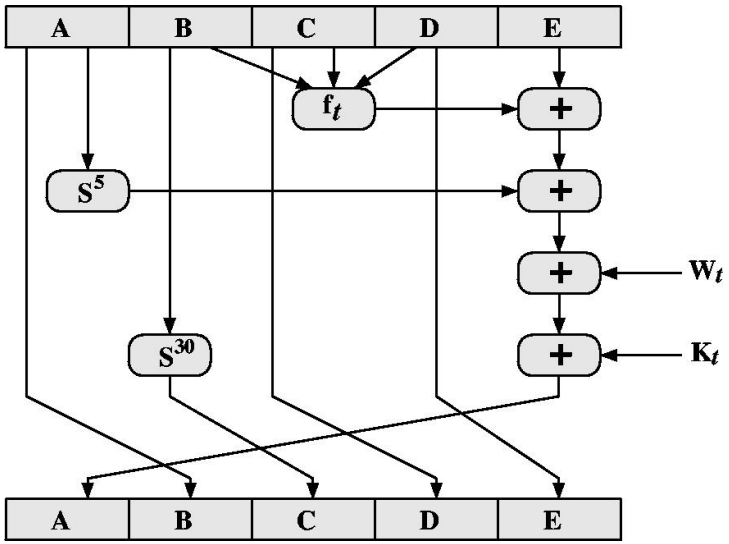
Each of the 80 steps is of the form

$$(A, B, C, D, E) \leftarrow (E + f(t, B, C, D) + S^5(A) + W_t + K_t, A, S^{30}(B), C, D)$$

where

- t — step number
- S^k — circular left shift by k bits
- addition is modulo 2^{32}

Elementary SHA-1 Operation



SHA-1: Compression Function

The compression function $f(t, B, C, D)$ is defined as

$$f(t, B, C, D) = \begin{cases} (B \wedge C) \vee (\overline{B} \wedge D) & \text{if } 0 \leq t \leq 19 \\ B \oplus C \oplus D & \text{if } 20 \leq t \leq 39 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & \text{if } 40 \leq t \leq 59 \\ B \oplus C \oplus D & \text{if } 60 \leq t \leq 79 \end{cases}$$

The words W_t, derived from the current message block Y_q, are defined as:

$$W_t = \begin{cases} \text{Word } t \text{ of } Y_q & \text{if } 0 \leq t \leq 15 \\ S^1(W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}) & \text{otherwise} \end{cases}$$

SHA-1: Compression Function, cont.

The four additive constants K_t are defined as:

$$K_t = \begin{cases} \texttt{5A827999} = \lfloor 2^{30}\sqrt{2} \rfloor & \text{if } 0 \le t \le 19 \\ \texttt{6ED9EBA1} = \lfloor 2^{30}\sqrt{3} \rfloor & \text{if } 20 \le t \le 39 \\ \texttt{8F1BBCDC} = \lfloor 2^{30}\sqrt{5} \rfloor & \text{if } 40 \le t \le 59 \\ \texttt{CA62C1D6} = \lfloor 2^{30}\sqrt{10} \rfloor & \text{if } 60 \le t \le 79 \end{cases}$$

Attacks on SHA-1

Finding collisions:

- Wang, Yin, Yu (Feb. 2005) — 2^{69} hash ops
- Wang, Yao, Yao (Aug. 2005) — 2^{63} hash ops
- Stephens (2012) — 2^{60} hash ops

Significantly less than theoretical maximum (2^{80}) — therefore, considered vulnerable.

Other Hash Functions

MD5 — 128-bit hash length, developed by Rivest.

- Essentially broken (Wang et. al., 2004). Can find MD5 collisions on a laptop in 8 hours or less (Klima, 2005).

Revised hash standard SHA-2 consisting of SHA-256, SHA-384, SHA-512

- modifications of SHA-1 to provide 128, 192, and 256 bits of security for compatibility with AES (see FIPS 180-4).
- current recommendation: use one of these in place of SHA-1.

Charles, Goren, Lauter (2009) — hash function based on expander graphs

- provable security: finding collisions reduces to computing computing isogenies between supersingular elliptic curves

See NIST's hash function page in the Cryptographic Tool Kit for more.

SHA-3: Keccak

After the 2005 attack on SHA-1, NIST initiated a competition for new hash algorithms, similar to the AES competition; see `//csrc.nist.gov/groups/ST/hash/`. The SHA-3 winner was announced on October 2, 2012:

Keccak (pronounced "ketchuk"), invented by

- Guido Bertoni (Italy) of STMicroelectronics,
- Joan Daemen (Belgium) of STMicroelectronics (one of the AES/Rijndahl creators!),
- Michaël Peeters (Belgium) of NXP Semiconductors,
- Gilles Van Assche (Belgium) of STMicroelectronics.

Keccak: Overview

Uses *Sponge* construction whose permutation function has the following properties:

- iterated round function,
- operates on 1600-bit state; other state widths range from 25 to 800,
- uses only bitwise XOR, AND, NOT (no table look-ups, arithmetic or data-dependent rotations)

What is a sponge design?

- hash function: arbitrary input length, fixed output length
- stream cipher: fixed input length, arbitrary output length
- sponge function: arbitrary input length, variable user-supplied output length

Attacks on Hash Functions

Objectives of adversaries vs. hash functions:

- Find a pre-image: given any hash, create a corresponding message with that hash.
- Find a weak collision: given a message, modify it to another message with the same hash.
- Find a collision: find two messages with the same hash.

Brute-force Attacks

Like block ciphers, brute force should be the best attack.

For an m-bit hash function:

- Pre-images and weak collisions: 2^m attempts on average
- Strong collisions: $2^{m/2}$ attempts on average due to the *birthday paradox* — probability of having at least one duplicate out of k random numbers between 1 and n is of order $\sqrt{n}$ (see any of our textbooks).

Recommended sizes: $m = 160$, 256, 394, 512 (provide 80, 128, 192, and 256 bits of security)

Birthday Attack

Birthday attack on signature schemes with hash functions (more later):

- Attacker generates $2^{m/2}$ variations of a valid message (easy to do by adding/removing white space, replacing synonyms, etc...).
- Attacker generates $2^{m/2}$ variations of a desired fraudulent message.
- The two sets of messages are compared to find a pair with the same hash.
- Attacker has the victim sign the hash of the valid message — the signature will also be valid for the fraudulent message.

Cryptanalytic Attacks

Iterated hash functions are composed of rounds (eg. SHA-1)

- Repeated use of *compression function* f — takes n-bit input from the previous step (chaining variable) and a b-bit block from M; produces n-bit output.
- Input to H : message M consisting of L b-bit blocks $Y_0, \ldots, Y_{L-1}$ (padded to suitable length).
- $CV_0 = IV =$ initial n-bit value (e.g. all zeros).
- $CV_i = f(CV_{i-1}, Y_{i-1})$, $1 \leq i \leq L$
- $H(M) = CV_L$

Iterated hash functions can be set up in such a way so that if f is collision-resistant, so is H (Merkle 1989 and Damgard 1989).

Idea for Attacking

Exploit the structure of the hash function (similar to block ciphers):

- Analytically attack the rounds of a hash function
- Focus on collisions in function f.
- Almost all widely-used hash function have succumbed to this type of attack (due to Wang et al).

Message Authentication Codes (MACs)

A small, fixed-size, key-dependent block that is appended to a message to check data integrity.

- Similar to a hash function, but keyed.

Definition 3 (Message authentication code (MAC))

A single-parameter family $\{C_K\}_{K \in \mathcal{K}}$ of many-to-one functions $C_K : \mathcal{M} \to \{0,1\}^n$ ($n \in \mathbb{N}$) satisfying:

- *Ease of computation:* For any $M \in \mathcal{M}$ and $K \in \mathcal{K}$, $C_K(M)$ is easy to compute.
- *Computation resistance*: for any $K \in \mathcal{K}$, given zero or more message/MAC pairs $(M_i, C_K(M_i))$, it is computationally infeasible to compute any new message/MAC pair $(M, C_K(M))$, $M \neq M_i$ for all i.

Data Integrity using MACs

Computation-resistance implies data integrity (without secrecy):

- Sender and receiver share a secret key K
- Sender computes $MAC = C_K(M)$ and sends (M, MAC) (unencrypted!)
- Receiver computes $MAC' = C_K(M)$ and checks if $MAC' = MAC$. If they match and C_K is computation resistant, the integrity of M is preserved.

Similar to encryption, but (a) no secrecy, (b) MACs need not be reversible, (c) there are many messages with the same MAC.

Sender Authentication using MACs

MACs also provide sender authentication in a similar manner to encryption

- only sender or receiver, who knows K could generate the MAC.

Note: Non-repudiation of data origin not provided

- *either* party possessing K can generate MACs.

More on MACs

Note 1

MAC should depend equally on all bits of the message. Given valid message/MAC pair, it should still be hard to find another valid pair even if only one bit of the message is modified.

Note 2

Apply first MAC, then encryption to message with MAC appended, rather than vice versa

- $C = E_{K_1}(M \| MAC_{K_2}(M))$ — if encryption is defeated, message integrity is still preserved.
- $(C \| MAC_{K_2}(C))$ with $C = E_{K_1}(M)$ — preserves only integrity of ciphertext which is useless if encryption is defeated

Why MACs?

Why use MACs (instead of encrypting message plus checksum/hash)?

- Sometimes only integrity is needed (no secrecy).
- Sometimes need integrity to persist longer than the encryption (eg. archival use)

CMAC

A secure block cipher (satisfying additional statistical properties) can be used to generate MACs. Two methods are:

1. CBC-MAC:
   - Encrypt the message (zero IV, last block padded with 0s) using CBC mode.
   - The last cipher block (whose bits are dependent on all the key bits and all message bits) is the MAC.
2. CFB-MAC: Same idea as CBC-MAC

A CBC-MAC using DES appears in both FIPS 113 and the ANSI X9.17 standard.

Problem with CBC-MAC

Problem: only secure if messages of *one* fixed length are processed (Bellare, Killian, Rogaway 2000) — see Assignment 3.

Solution (CMAC):

- Use *three* keys, one at each step of the chaining, two for the last block (Black, Rogaway 2000).
- Second two keys may be derived from the encryption key (Iwata, Kurosawa 2003).

Properties of CMAC

Cipher-based Message Authentication Code (CMAC)

- Specified for use with AES and 3DES in NIST Special Pub. 800-38B
- Can be proven secure as long as the underlying block cipher's output is indistinguishable from a random permutation.
- No known weaknesses.

Operation of CMAC

Message M is padded so its length is a multiple of the cipher's block length n (128 for AES, 64 for 3DES) by appending a 1 and as many 0s as necessary, then divided into blocks $M_1, \ldots, M_m$.

Let K be the block cipher key. Two additional keys K_1 and K_2 are computed as follows:

$$\begin{aligned} L &= E_K(0^n) \\ K_1 &= L \cdot x \\ K_2 &= L \cdot x^2 = K_1 \cdot x \end{aligned}$$

where $\cdot$ denotes multiplication of polynomials with bit coefficients modulo $x^{64} + x^4 + x^3 + x + 1$ or $x^{128} + x^7 + x^2 + x + 1$ (i.e., mult. in $GF(2^n)$).

Operation of CMAC, cont.

To compute the MAC of message M, process blocks $M_1, \ldots, M_{m-1}$ using CBC with $IV = 0$:

$$\begin{aligned} C_0 &= 0^n \\ C_i &= E_K(M_i \oplus C_{i-1}) \quad 1 \leq i \leq m-1 \ . \end{aligned}$$

Compute

$$C_m = E_K(M_m \oplus C_{m-1} \oplus K_i)$$

where $i = 1$ if M was not padded and $i = 2$ if M was padded.

MAC is the s leftmost (most significant) bits of C_m (s is determined by the desired level of security).

HMAC

Basic idea: $MAC = H(M\|K)$ where H is a cryptographically secure hash function and K is a secret key.

Advantage over CMAC: hash functions are faster than block ciphers.

Idea:

- $MAC = H(M\|K)$: insecure if H is iterated (see Assignment 3)
- $MAC = H(K\|M)$: similar problem (see Assignment 3)
- $MAC = H(K_1\|M\|K_2)$: better, but potentially also vulnerable
- $MAC = H(K_1\|H(K_2\|M))$: Bellare, Canetti, Krawczyk (CRYPTO 1996) — HMAC

Operation of HMAC

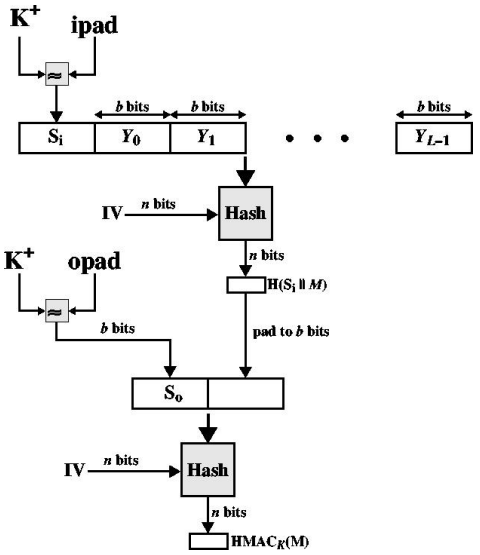
HMAC (FIPS 198):

$$HMAC_K(M) = H\left((K^+ \oplus \text{opad}) \parallel H\left((K^+ \oplus \text{ipad}) \mid M\right)\right)$$

Description (assume H operates on b-bit blocks, eg. for SHA-1, $b = 512$):

1. $K^+ = 0 \ldots 0K$ (0-bits prepended so K^+ has b bits)
2. $S_i = K^+ \oplus \text{ipad}$, with $\text{ipad} = (00110110)_{b/8}$
3. $T = H(S_i\|M)$ (note that $f(IV, S_i)$, compression function of H applied to S_i, can be precomputed)
4. $S_o = K^+ \oplus \text{opad}$, with $\text{opad} = (01011100)_{b/8}$
5. $HMAC_K(M) = H(S_o\|T)$ (note that $f(IV, S_o)$ can be precomputed)

Diagram of HMAC



Properties of HMAC

$K^+ \oplus \text{ipad}$ and $K^+ \oplus \text{opad}$ — two pseudorandom keys generated from K.

- XORing with ipad and opad each cause $1/2$ of the bits of K to be flipped.
- Helps ensure that the Hamming distance between S_i and S_o is fairly high, maximizing the statistical independence of $f(IV, S_i)$ and $f(IV, S_o)$.

Only three additional executions of the compression function of H compared with only hashing M

- only one if key-dependent precomputation is used as mentioned above

More Properties

Provable security, equivalent to one of:

- computing an output of the compression function of H assuming the IV is unknown,
- finding collisions of the hash function assuming the IV is unknown.

Note that a birthday attack based on the second case is possible.

- significantly more difficult than on a hash function
- requires a MAC-generating oracle to compute valid message/MAC pairs due to the fact that IV is secret

Attacks on MACs

Objectives of adversaries vs. MACs (without prior knowledge of K):

- Compute a new message/MAC pair $(M, C_K(M))$ for some message $M \neq M_i$, given one or more pairs $(M_i, C_K(M_i))$.
- Known-text, chosen-text, and adaptive-chosen-text variations are possible.

MAC Space Attack

Assume n-bit MACs, m-bit keys.

Attack:

- Pick a message, guess the MAC value (probability 2^{-n} of being correct)
- Requires "black-box" MAC generator to verify guesses.
- Expected number of attempts is 2^n.

Key Space Attack

Assumes $m > n$ (longer keys than MACs, reasonable). This is a KTA:

- Given $MAC_1 = C_{K_1}(M_1)$, compute $MAC_i = C_{K_i}(M_1)$ for all possible keys K_i $(1 \leq i \leq 2^m)$
- Expect 2^{m-n} keys to produce a match (2^m MACs produced, only 2^n possible MACs).
- Repeat with (MAC_2, M_2), reducing the number of possible keys to 2^{m-2n}. Iterate with (MAC_i, M_i), $i = 3, 4, \ldots$

Requirements:

- $\lceil m/n \rceil$ message/MAC pairs
- $\lceil m/n \rceil \cdot 2^m$ MAC computations, but these can be conducted off-line.

Summary

Brute-force attack requires effort $\min\{\lceil \frac{m}{n} \rceil 2^m, 2^n\}$.

As usual, this should be best possible.

Cryptanalytic attacks also possible:

- For CMAC, one can try to attack the underlying block cipher.
- For HMAC, one can try to attack the underlying hash function.