Hashing

Dr. Amit Praseed

Hash Function Basics

- **Input:** Variable length data block M
- **Output:** Fixed size has value $h=H(M)$
- Ideally, generated outputs should be random
- Applications:
  - Message authentication and digital signatures
  - One way password files
  - IDS/Virus detection
  - PRFs and PRNGs

A Simple Hashing Scheme

- Let $X = X_1 \,||\, X_2 \,||\, \ldots \,||\, X_m$, where $|X_i| = n$
- *Define:* $H(X) = X_1 \oplus X_2 \oplus \ldots \oplus X_m$
- Consider an attacker observing the transmission of X and $H(X)$
- Can an attacker compute a different message Y such that $H(Y)=H(X)$?

A Simple Hashing Scheme

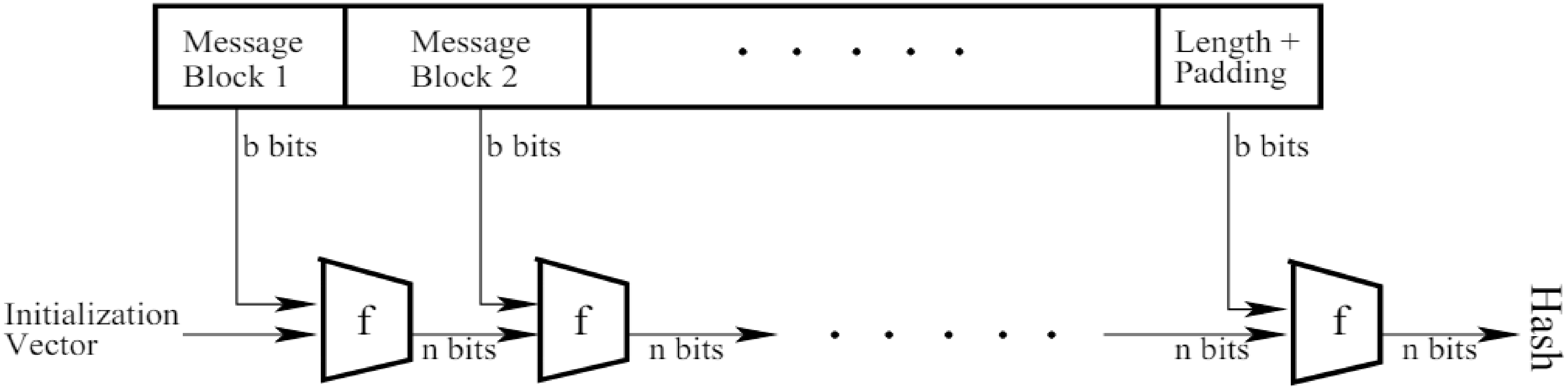
- Let $X = X_1 \,||\, X_2 \,||\, \ldots \,||\, X_m$, where $|X_i| = n$
- *Define:* $H(X) = X_1 \oplus X_2 \oplus \ldots \oplus X_m$
- Consider an attacker observing the transmission of X and $H(X)$
- Can an attacker compute a different message Y such that $H(Y)=H(X)$?
- Let $Y = Y_1 \,||\, Y_2 \,||\, \ldots\, Y_{m-1} \,||\, Y_m$
- $Y_m = Y_1 \oplus Y_2 \oplus \ldots \oplus Y_{m-1} \oplus H(X)$
- *Will this work?*

Cryptographically Secure Hash Function

- **One Way Property**
  - Given $H(m)$, it should be computationally infeasible to find m
- **Weak Collision Resistance**
  - Given $H(x)$, it is computationally infeasible to find y $(y \neq x)$ such that $H(x)=H(y)$
- **Strong Collision Resistance**
  - It is computationally infeasible to find x and y $(y \neq x)$ such that $H(x)=H(y)$

Merkle–Damgård construction

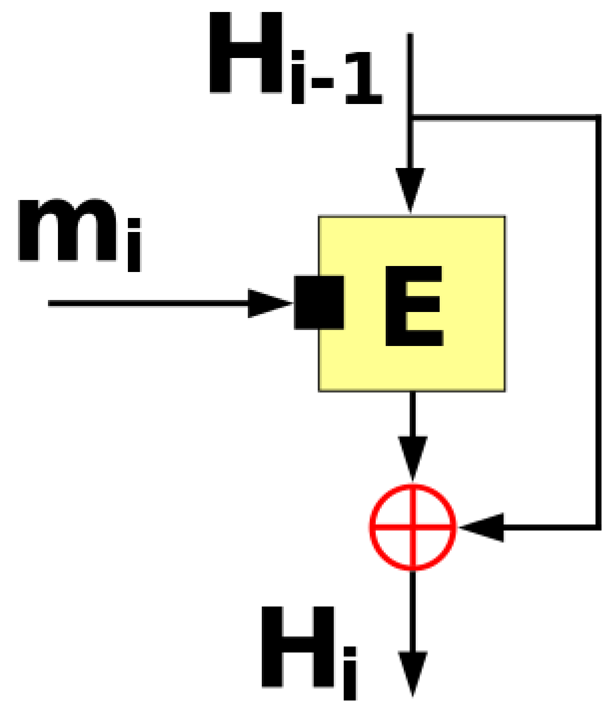
- Method of building **collision-resistant cryptographic hash functions** from **collision-resistant one-way compression functions**



If the compression function is collision resistant, then the Merkle-Damgard construction is collision resistant

Davies–Meyer Construction

- Turn a block cipher into a one-way compression function
  - H_{i-1} is the input (length n)
  - m_i is the key (length l)
  - E is an encryption function
- If E is modeled as an ideal cipher, then the Davies-Meyer construction yields a collision-resistant compression function.
- Concretely, any attacker making $q < 2^{n/2}$ queries to its ideal-cipher oracle finds a collision with probability at most $q^2/2^n$.
  - Why is this bound significant?

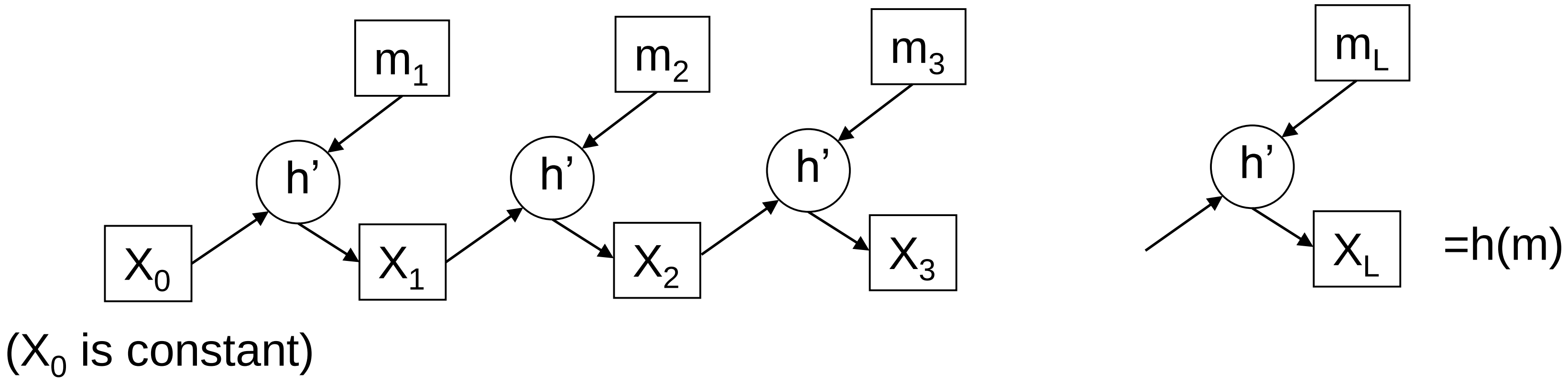


Basic Details of SHA-1

- Message Length: $<2^{64}$
- Hash value: 160 bits
- SHACAL-1 block cipher + Davies–Meyer + Merkle–Damgård
- Padding:
  - Message is made into a multiple of 512 bits
  - How?
    - Append a 1
    - Append zeroes such that message length is 64 bits short of a multiple of 512
    - Append message length (64 bits)

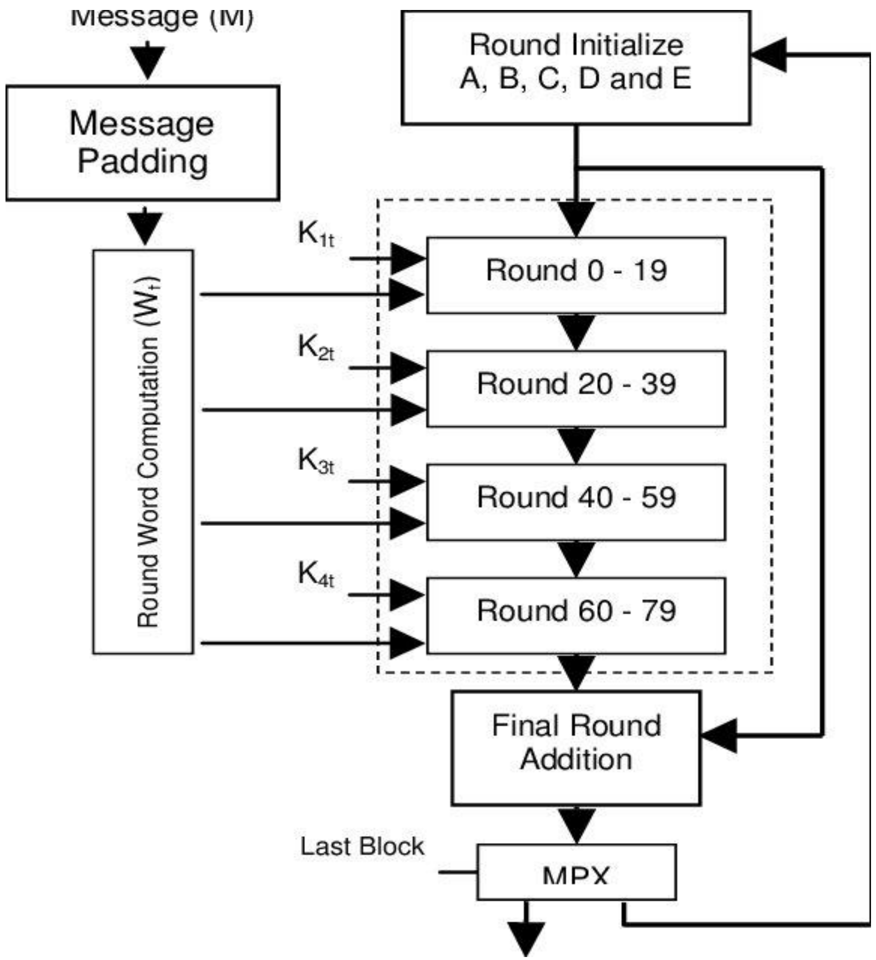
Basic SHA-1 Architecture

- Iterate over all of the blocks, outputting a value that is a function of the previous output and the current block:



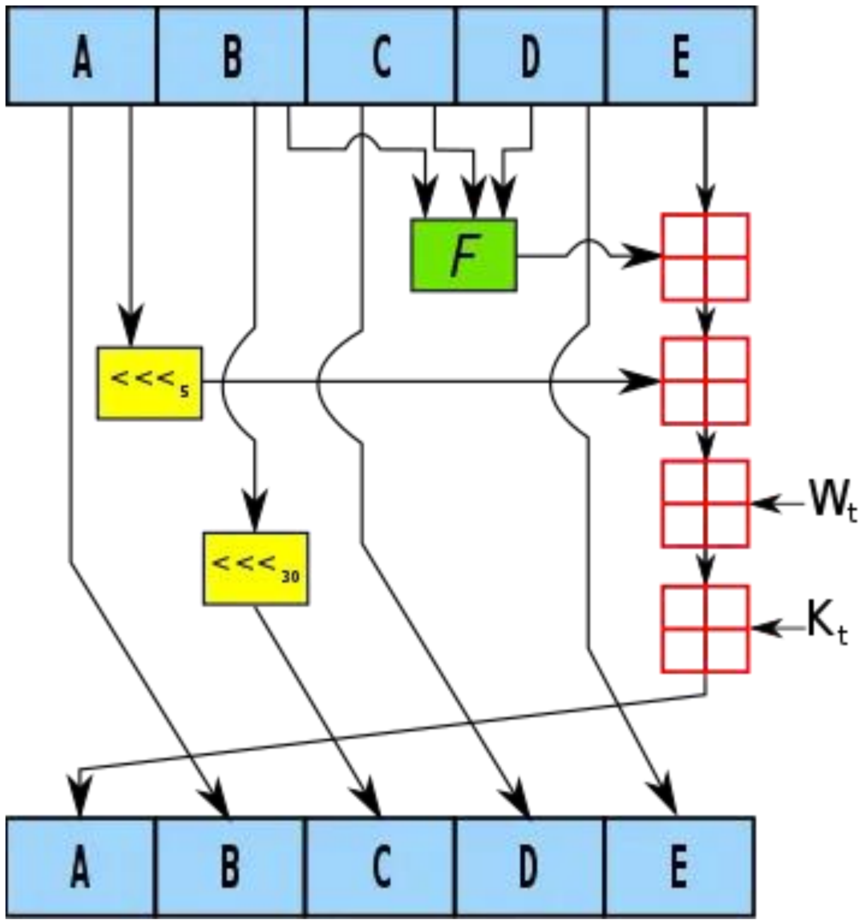
The h' function

- Each of the 512 bit blocks are passed through the compression function h'
  - Total 80 rounds – 4 stages of 20 operations each
- For each function h', there are two inputs
  - 512 bit message block
  - 160 bit hash value output from previous h' function
    - For the first stage, an IV is used
    - IV is initialized as:
      - A = 0x67452301
      - B = 0xEFCDAB89
      - C = 0x98BADCFE
      - D = 0x10325476
      - E = 0xC3D2E1F0



One SHA Operation

- Each SHA operation does the following:
  - A non-linear function on three words
  - Shifting
  - Modular addition
- Two additional inputs:
  - W_t: Generated from input block [Message Schedule]
  - K_t: Constants



Message Schedule

- Original message block is 512 bits (16 32-bit words)
- Transformed into 80 32-bit blocks

$$W_t = M_t \text{ for } t=0\text{-}15$$

$$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) <<< 1, \text{ for } t=16\text{-}79$$

Functions and Constants

- F function

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

- Constants
    - $K_{0..19}$=0x5A827999
    - $K_{20..39}$=0x6ED9EBA1
    - $K_{40..59}$=0x8F1BBCDC
    - $K_{60..79}$=0xCA62C1D6

SHA-1 Security

- SHA-1 hash length is 160 bits
  - For a pool of 80 randomly chosen messages, there exists two messages with the same hash value with 50% probability
- Theoretical attacks against SHA-1 were developed in 2005 with a collision using only 2^{69} messages
- In 2015, a practical attack against SHA-1 was demonstrated using only 2^{57} evaluations
- In 2022, NIST recommended that SHA-1 be phased out in favour of newer versions
  - ”SHA-1 should be phased out by Dec. 31, 2030”

Newer Variants

- **SHA-2:**
  - SHA-256 and SHA-512
  - SHA-256 uses 32-bit words where SHA-512 uses 64-bit words
- **SHA-3:**
  - Formerly called Keccak
  - Supports the same hash lengths as SHA-2
  - Internal structure differs significantly from the rest of the SHA family.

Message Authentication Codes (MACs)

- Symmetric Encryption primitives to ensure authenticated messages
- MACs can be constructed using block ciphers or hashes
- Consider a block cipher $F{:}\{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$
  - Is this secure?
  - Is this usable?

Message Authentication Codes (MACs)

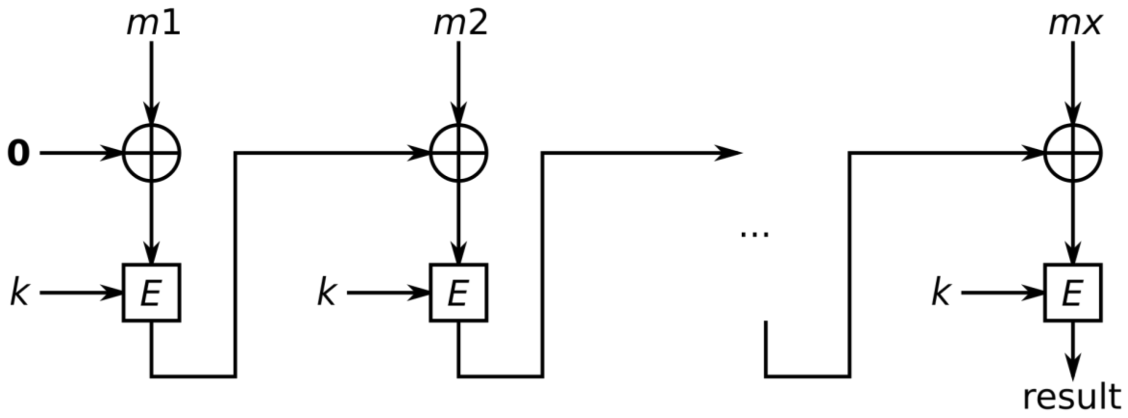
- Symmetric Encryption primitives to ensure authenticated messages
- MACs can be constructed using block ciphers or hashes
- Consider a block cipher $F{:}\{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$
  - Is this secure?
  - Is this usable?
- To extend it to multiple blocks, can we calculate the tag for each block separately and then authenticate each block separately?
  - Is this secure?
  - Is this usable?

Message Authentication Codes (MACs)

- To extend it to multiple blocks, can we calculate the tag for each block separately and then authenticate each block separately?
  - Is this secure?
  - Is this usable?
- Let us consider these modifications to the scheme (in sequence):
  - Adding a block number to each block
  - Adding the message length to each block
  - Adding a random number to each block
    - And sending the random number along with the tag as well!!!
  - Think about the security and usability of each of these schemes!!!

Message Authentication Codes (MACs)

- Use a block cipher in CBC mode – hash is the last encrypted block



  - Is this a cryptographically secure hash function?

Brainstorming Session

- Assume that you have message-tag pairs (m,t) and (m',t').
- *Can we come up with a new message m'' that has the tag t'?*
- *Hint:*
  - *Consider the message* $m_1 \,||\, m_2 \,||_{\dots} ||\, m_n \,||\, m'_1 \,||\, m'_{2\dots} \,||\, m'_k$
  - *Can we modify any one block so that the tag becomes t'?*

Brainstorming Session

- Assume that you have message-tag pairs *(m,t)* and *(m',t')*.
- *Can we come up with a new message m'' that has the tag t'?*
- *Hint:*
    - *Consider the message* $m_1 \,||\, m_2 \,||_{\ldots}||\, m_n \,||\, m'_1 \,||\, m'_{2\ldots} \,||\, m'_k$
    - *Can we modify any one block so that the tag becomes t'?*
- *Can we prevent this?*

Hashes as MDCs

- Hashes can be used as Manipulation Detection Codes (MDC)
- Using Asymmetric Cryptographic Primitives:
  - A→ B: $M \,||\, PR_A(H(M))$
  - *How can B verify whether M is modified?*
- Using Symmetric Cryptographic Primitives:
  - A→ B: $M \,||\, H(K \,||\, M)$
  - *An example of MAC!!!*
  - *How can B verify whether M is modified?*

Length Extension Attacks

- Merkle-Damgard construction allows hashes to be extended.
- Assume that Eve intercepts the message $M \mid\mid H(K \mid\mid M)$ sent from A to B
- Can Eve add a message M’ to the original message, and force B to accept it?
  - i.e Eve wants to send a message $M \mid\mid$ *<potential garbage>* $\mid\mid M'$
  - Can Eve do this, considering the extension properties of the Merkle-Damgard construction?

Nested MAC (NMAC)

- Define an NMAC as follows:
  - $NMAC_{K1,K2}(M) = H(K_2 \,||\, H(K_1 \,||\, M))$
- *Is Length Extension possible here?*

Nested MAC (NMAC)

- Define an NMAC as follows:
  - $NMAC_{K1,K2}(M) = H(K_2 \,||\, H(K_1 \,||\, M))$
- *Is Length Extension possible here?*
- Keys K1 and K2 should preferably be independent
- HMAC or Hashed MAC is a single-key implementation of NMAC
  - $K1 = K \oplus ipad$, $K2 = K \oplus opad$