Computer and Network Security: Hashes

Kameswari Chebrolu

Outline

- **Modern Cryptography**
  - Overview
  - Confidentiality
    - Background: Definition, Crypto-analysis, One Time Pads
    - Symmetric key encryption, Block modes
    - Asymmetric key encryption
  - **Integrity (includes Authentication)**
    - **Hashes**, MAC, Digital signature

Integrity and Authentication

- Focus: Communication framework

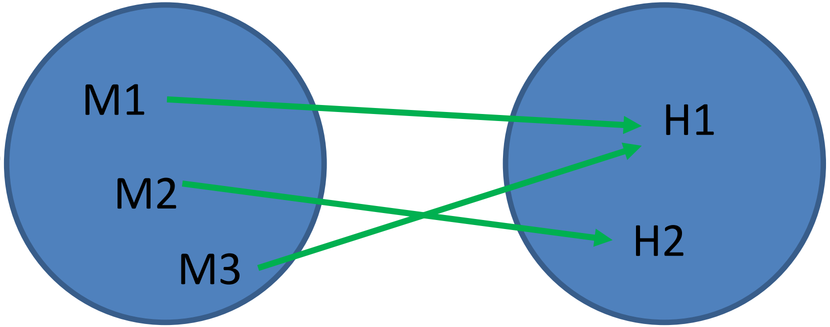
Definition:

- Integrity (data): Has the data been modified in transit?
- Authentication: Am I talking with the right person?

- Hashes/Message Digests: Integrity
- Message Authentication Codes (MACs): Integrity and Authentication
  – Based on Shared key
- Digital Signatures: Integrity and Authentication
  – Based on Asymmetric key

Hashes/Message Digest

- **One way** function used to verify message integrity
- Maps arbitrary length message (*)→ fixed length string (d)
  - h(M) hash of message M; similar to fingerprint
  - No key; public algorithm
  - Typical values of d: 128, 160, 256, 512 bits



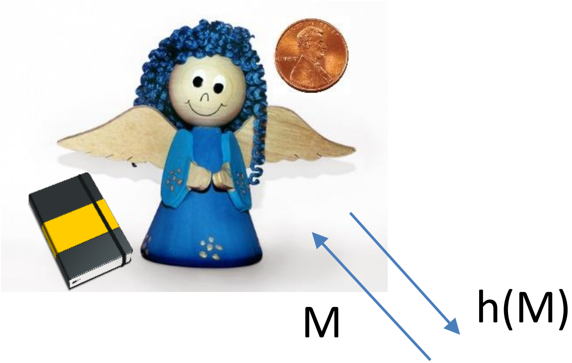
$$h : \{0,1\}^* \longrightarrow \{0,1\}^d$$

Collisions Possible

- Brute Force Attack
  - 1000 bit message and 160 bit output
    - On average 2^{840} messages map to the same output
    - Have to try approx 2^{160} possible messages to find a match
- Usage:
  - Hash of Software (e.g. Ubuntu versions); file integrity checks
  - Password verification (hash of passwords stored);
  - Commitment: Alice submits h(v|r) as sealed bid (r is random no)
    - After bidding close, to open bid Alice reveals (r, v)

Random Oracle (RO)

- Hash operation needs to be
  - Efficient (polynomial time)
  - Deterministic but should look like random string
    - For each message choose a different and independent string
- Hash functions like SHA1 (160), SHA2/3 (224, 256, 384 512), MD4(128), MD5(128) take the role of RO
  - “pseudo random” though



If M not in book
   - flip coin d times to generate h(M)
   - record (M, h(M)) in book
Else return h(M) from book

(book ensures process deterministic)

Desirable Properties

- Pre-image resistant:
  - Given y, infeasible to find any m where h(m) = y
  - m is a pre-image of y
- 2nd pre-image resistance (weak collision resistance)
  - Given m_1, infeasible to find different m_2 such that $h(m_1) = h(m_2)$
  - Adversary knows m_1 and wishes to change it to m_2 to produce same hash

- Collision Resistance (strong):
  - Infeasible to find **ANY** two different messages m_1 and m_2 such that $h(m_1) = h(m_2)$
- If h() is collision resistant, h() is also weak collision resistant
  - Converse not true
- If h() is weak collision resistant, h() is also pre-image resistant
  - Converse not true

Birthday Problem

23 or more people in a room, >50% chance two have same birthday

Proof: Calculate probability no two people have the same birthday

= 365/365 * 364/365 * 363/365 * ……… * 343/365

= $(1/365)^{23}$ * (365 * 364 * ….. * 343) =~ 0.49

→ 0.51 prob two have same birthday

Intuition: 23 people, 253 possible pairs

Each pair has to have different birthdays for no two people to have same birthday

Relevance to Collision Resistance

- b-bit hash, $n=2^b$ possible hash values
- Prob i^{th} generated message does not collide with previous i-1 messages = (n-(i-1))/n = 1 – ((i-1)/n)
- Failure after generating k messages (attacker has still not found a collision)

This is product, not addition

$$P_k = (1 - 1/n) + (1 - 2/n) + \cdots + (1 - (k-1)/n)$$

$$P_k = (1 - 1/n) + (1 - 2/n) + \cdots + (1 - (k-1)/n)$$

This is product, not addition

- For $P_k = 0.5$, $k \approx 1.17\sqrt{n}$
- 64-bit hash would only take searching 2^{32} random messages to find two with same value
  - 32 bits of security for a 64 bit output
  - Hashes >= 128 bits

Importance of Collision Resistance

- Players: Alice(Boss); Mallory (Malicious Secretary)
- Mallory: two versions of contract
  - C1: favorable to Alice; C2: favorable to self
  - Make many unnoticeable changes to C1 and C2
    - Extra space at end of line; double space between words; synonyms of words etc
    - E.g. 32 lines; change/no-change to each line → 2^{32} variations

- 2^{32} messages of type C1 and 2^{32} messages of type C2
- By birthday paradox
  - Testing ~2^{32} messages can find messages C1 and C2 where h(C1) = h(C2)
  - As against 2^{64} if original message of type C1 was fixed

Hashes vs Encryption

Hashes

- One-way
  - No un-hashing
  - Collisions
- Deterministic
  - Hash same input, get same output

Encryption

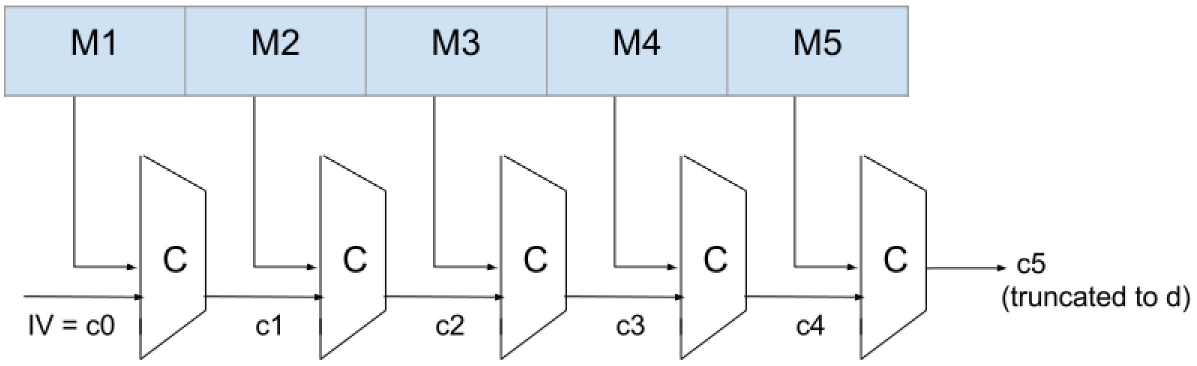
- One-way with trapdoor
  - Plaintext can be recovered from Ciphertext
  - No collisions
- Random
  - Ideally encrypt same plaintext, get different ciphertext

Common Hash Functions

- MD4, MD5 (128 bits): Developed by Ronald Rivest
  - Completely broken
- SHA-1 (160 bits): Developed by NSA
  - No longer secure against powerful attackers
- SHA-2: Developed by NSA
  - Supports 224, 256, 384 and 512 output lengths
  - No significant attack still (but good to have alternatives)
- SHA-3: Public competition, winner Keccak (2012)
  - Supports 224, 256, 384 and 512 output lengths
  - Internal structure very different from rest of SHA family

Construction

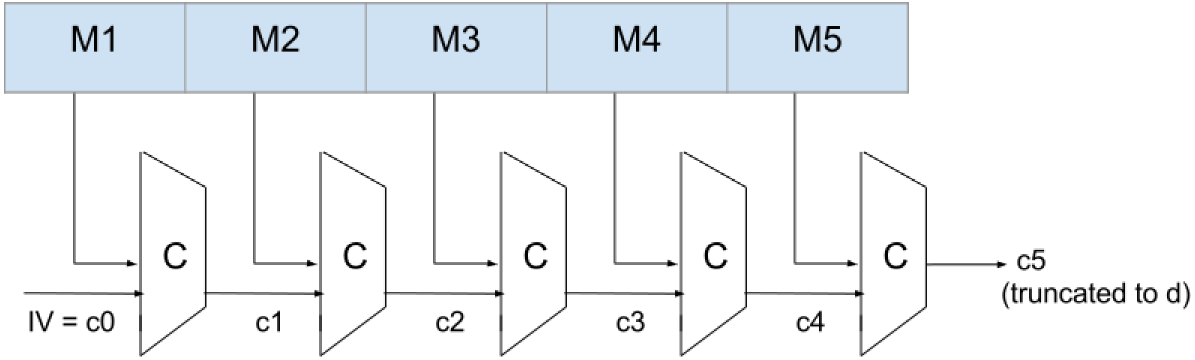
- No math; bunch of steps that increasingly mangle the message (output has to pass randomness tests)
- Common Structure (Merkle Damgard Construction)



Construction

Message M divided into blocks M1, M2 etc

- Message block size b (e.g. 512 bits)
- Pad message to make it multiple of b



Output size: d

Chaining variable c

- c_0 is fixed and public
- c >= d

C: Compression function

SHA-1: m = 512 bits and final hash (and c's) 160 bits

Compression function collision resistant → Above construction collision resistant

Example: SHA-1

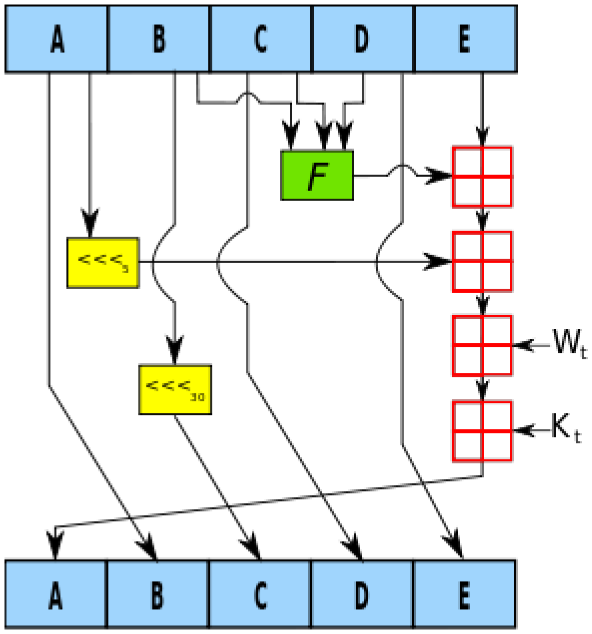
Processing one message block of size 512 bits

- Each message block split into 16 words of 32 bits (W_1, W_2 …. W_{16}), further

$$W_i = W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \quad 16 < i \leq 80$$

(for a total of 80 words)

- Shift Register: A, B, C, D and E (32 bit words)
  - Initialized to h1,h2,h3,h4 and h5
  - At the very beginning (first message chunk)
    - h1 = 0x67452301, h2 = 0xEFCDAB89, h3 = 0x98BADCFE, h4 = 0x10325476, h5 = 0xC3D2E1F0



<<< : rotation to left

: modulo addition 2^{32}

Example: SHA-1

$$W_i = W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \quad 16 < i \leq 80$$

Procedure:

For i = 1 to 80 {

Temp ← E + (A << 5) + F_i(B, C, D) + K_i + W_i

E ← D

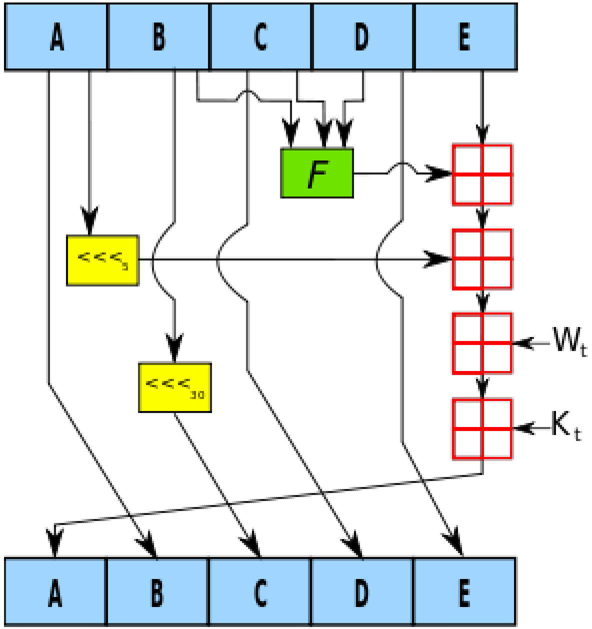
D ← C

C ← B >> 2

B ← A

A ← temp

}



K_i : known constant

F_i: Function

<<< : rotation to left

: modulo addition 2^{32}

- Function F:

$F_i(B,C,D)$ = (B AND C) OR ((NOT B) AND D) (1 <= i <= 20)

$F_i(B,C,D)$ = B XOR C XOR D (21 <= i <= 40)

$F_i(B,C,D)$ = (B AND C) OR (B AND D) OR (C AND D) (41 <= i <= 60)

$F_i(B,C,D)$ = B XOR C XOR D (61 <= i <= 80)

- *Add this message chunk's hash to result so far:* h1 = h1 + A; h2 = h2 + B; h3 = h3 + C; h4 = h4 + D; h5 = h5 + E
- After all message blocks processed, content of shift register is the h1|h2|h3|h4|h5
- SHA1("The quick brown fox jumps over the lazy dog") gives hexadecimal: 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12

Summary

- Integrity/Authentication: A very important aspect of security
- Hashes provide integrity (not authentication)
  - Desirable properties
  - Construction
  - Example: SHA-1