SE-4920: Lecture 9
Hashes and message digests

- Reading
  - Chapter 5
- Today's Outcomes
  - Discuss the uses of hashes for fingerprinting and signing
  - Discuss the key properties of a cryptographic hash function contrasted with a general hash function
  - Explain why hashes need to be roughly twice as long as secret keys
  - Explain how a hash can be used for an MAC

1

Definitions and overview

- Hash (message digest)
  - One-way function
  - Arbitrary length message → fixed-length quantity
- SHA-1 = secure hash algorithm
- MD2, MD4, MD5 = message digest
- Apparent randomness
  - Pass most/all known tests (bits on for about half of messages; about half of bits on in most outputs; decorrelated outputs, even for similar inputs)
  - But, not random, because you can calculate it if you know the algorithm and inputs

2

Hash lengths and the birthday problem

- Room of N people
  - about a 50% chance that
  - 2 chosen at random will have the same birthday
  - if N=22 or 23
- d = 365 different "hashes"
- p = 50% at approximately $n \approx 1.17741\sqrt{d}$
- Because of the square root
  - Double number of bits you'd use for encryption
  - To prevent finding 2 messages with same hash
    - Why would that be a problem?
  - Typically 128 or 160 bits

3

Fingerprints

- Detect modification to a document, firewall configuration file, program, etc.
- Calculate hash
- Store securely
  - In file cabinet
  - On ROM
- Recompute and compare from time to time

4

Types of hash attacks

- Collision
  - Finding 2 messages with the same hash
- Preimage
  - Finding a message with a given hash
    - Much more difficult

5

Constructing a hash

- Like private key algorithms (*e.g.*, DES)
  - Random looking outputs
- But, no need for key
- And, not reversible
  - Maps N bits to a fixed number of bits
    - Typically 128 or 160

6

A note on usage

- "Technically"
  - Hash has no keys
  - Private key has 1 key
- More important property
  - Hash is an irreversible mapping
    - *Might* use with a key
  - Private key encryption is reversible

7

Algorithms: MD2

- 128 b hash, 128 b block
- 1989
- Optimized for 8-bit machines
- Originally for public key signatures
- Hash then sign
  - PK signing is expensive, but the hash is small
- 1997
  - Collisions found on internal component, but not exploited
- 2004
  - Preimage attack (chosen hash) found (2^{104} applications of an internal function)
    - Very serious!

8

Algorithms: MD4

- 128 b hash, 512 b block
- 1990
- Optimized for 32-bit machines
- 1991/2
  - Weaknesses found in foundation
- 2004
  - Collisions found

9

Algorithms: MD5

- 128 b hash, 512 b block
- 1991
  - Addressed potential weaknesses in MD4
- 1996
  - Flaws found, not fatal, but people started moving away
- 2004
  - Collisions announced
- March 2005
  - Practical collision (in a public key) demonstrated with X.509 certificates
- March 2006
  - Algorithm published that finds collisions within about 1 minute using a single, standard computer

10

Algorithms: SHA-1

- 160 b hash, 512 b block
- 1995: Similar to MD5, but a little stronger
- Rapid progress on breaking in 2005
  - August, 2005
    - Algorithm for finding a collision in 2^{63} operations
  - Still strong, but pace of progress and history suggest moving to a stronger algorithm
- History suggests it may be trivial to find collisions soon
  - January, 2007: NIST announces contest to replace
    - Fall, 2008: submissions due
    - End 2011: new standard chosen
  - See http://www.wired.com/politics/security/commentary/securitymatters/2007/02/72657

11

Up-and-coming algorithms: SHA-2 Family

- Published in 2001, 2004
- SHA-{224, 256, 384, 512}
  - Number indicates hash length
  - Block size is 512 b and word size is 32 b for 2 smaller algorithms
    - 1024 b and 64 b for 2 larger algorithms

12

Calculating the SHA-1 Hash: Padding

- Pad message to a multiple of 512 bits
  - Message
  - The bit '1'
  - 2-511 '0' bits such that (length % 512) is
    - (-64 % 512) = 448
  - 64 bits indicating original length
    - Bytes in little endian order
      - (but bits are in big endian order)

13

Calculating the SHA-1 Hash: Initializing

- Initialize with
  - A = 67452301
  - B = efcdab89
  - C = 98badcfe
  - D = 10325476
  - E = c3d2e1f0
- Concatenation of A, B, C, D, and E will be the hash

14

Calculating the SHA-1 Hash: Processing a message block

- For each 512 b input block, create a 512 × 5 b array
  - Fill first row with message
  - Treat as 16, 32-bit words per row
  - Words 16...79
    - w_n = (w_{n-3} XOR w_{n-8} XOR w_{n-14} XOR w_{n-16}) <<<1
      - <<< is the **rotate** left operator

15

Calculating the SHA-1 Hash: Updates for each message block

- Now, loop over the 80, 32-bit words in the block
  - W_t, where t = 0…79
- Values on right refer to the previous values
  - A = E + (A <<<5) + W_t + K_t + f(t,B,C,D)
    - Carry outs discarded
  - B = A
  - C = B <<< 30
  - D = C
  - E = D
- K_t = floor($2^{30}\sqrt{f}$), where f = {2, 3, 5, 10}
  - For t up to {19, 39, 59, 79}

16

Calculating the SHA-1 Hash: f(·) for updates

- Using C++ operator notation
- f(0…19,B,C,D) = (B & C) | (~B & D)
- f(20…39,B,C,D) = B ^ C ^ D
- f(40…59,B,C,D) = (B & C) | (B & D) | (C & D)
- f(60…79,B,C,D) = B ^ C ^ D

17

Calculating a message authentication code using a hash

- Hashing (secret|message) not a good idea
  - Due to block structure of SHA-1 and other hashes
    - Can append to message and calculate valid hash, given hash on shorter message
- HMAC is the *de facto* solution (next slide)…
  - Builds on a hash, such as SHA-1
  - Works with hashes up to 512 bits
  - Pad key out to 512 bits
    - Hash/digest key first if it is too long
    - const1 = 0x36 repeated 64 times
    - const2 = 0x5c repeated 64 times

18

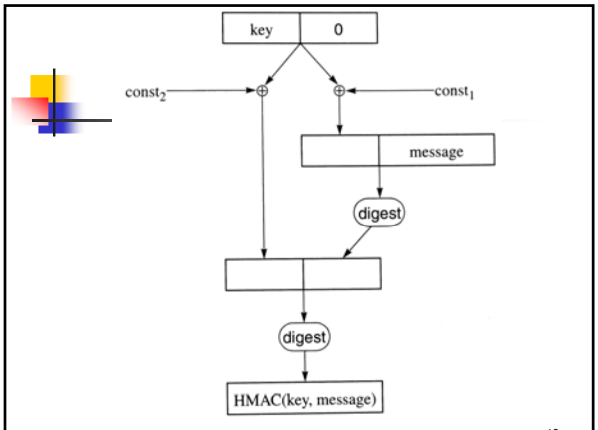

Figure 5-10. HMAC