Hash Algorithms

CSCI 454/554

Overview

- MD5, SHA-1, SHA-512
  - All based on MD4
  - MD5: 128-bit
  - SHA-1: 160-bit
  - SHA-512: 512-bit
- HMAC
  - MAC derived from a hash function
  - Integrate a secret key with a hash function
Hash Algorithms

- see similarities in the evolution of hash functions & block ciphers
  - increasing power of brute-force attacks
  - leading to evolution in algorithms
  - from MD4 & MD5 to SHA-1 & SHA-512 in hash algorithms
- likewise basic structure does not change
  - tend to use common iterative structure

Common Steps

- Step1: Append padding bits (last block)
  - Padded message is 64 bit less than multiple 512 bits
- Step2: Append length
  - 64-bit represents the length of original message
- Step3: Initialize MD buffer
  - Holding intermediate results (Iterative)
- Step4: Process message in 512-bit blocks
- Step5: Output
MD5

- designed by Ronald Rivest (the R in RSA)
- latest in a series of MD2, MD4
- produces a 128-bit hash value
- until recently was the most widely used hash algorithm
  - in recent times have both brute-force & cryptanalytic concerns
- specified as Internet standard RFC1321

MD5 Overview

1. pad message so its length is 448 mod 512
2. append a 64-bit length value to message
3. initialise 4-word (128-bit) MD buffer (A,B,C,D)
4. process message in 16-word (512-bit) blocks:
   - using 4 rounds of 16 steps (bit operations on message block & buffer)
   - add output to buffer input to form new buffer value
5. output hash value is the final buffer value
MD5 Overview

Diagram of MD5 message processing with steps and variables.
MD5 Compression Function

- each round has 16 steps of the form:
  \[ a = b + ((a + g(b,c,d) + X[k] + T[i]) << s) \]
- \(a, b, c, d\) refer to the 4 words of the buffer, but used in varying permutations
  - note this updates 1 word only of the buffer
  - after 16 steps each word is updated 4 times
- where \(g(b,c,d)\) is a different nonlinear function in each round (F,G,H,I)
- \(T[i]\) is a constant value derived from sin
MD4

- precursor to MD5
- also produces a 128-bit hash of message
- has 3 rounds of 16 steps vs 4 in MD5
- design goals:
  - collision resistant (hard to find collisions)
  - direct security (no dependence on "hard" problems)
  - fast, simple, compact
  - favours little-endian systems (eg PCs)

Weakness of MD5

- MD5 hash is dependent on all message bits
- known attacks are:
  - Berson 92 attacked any 1 round using differential cryptanalysis (but can’t extend)
  - Boer & Bosselaers 93 found a pseudo collision (again unable to extend)
  - Dobbertin 96 created collisions on MD compression function (but initial constants prevent exploit)

Real MD5 collision attack posted on Aug. 17, 2004
Secure Hash Algorithm (SHA-1)

- SHA was designed by NIST & NSA in 1993, revised 1995 as SHA-1
- US standard for use with DSA signature scheme
  - standard is FIPS 180-1 1995, also Internet RFC3174
- produces 160-bit hash values
- based on design of MD4 with key differences

SHA Overview

1. pad message so its length is 448 mod 512
2. append a 64-bit length value to message
3. initialise 5-word (160-bit) buffer (A,B,C,D,E) to (67452301,efcdab89,98badcfe,10325476,c3d2e1f0)
4. process message in 16-word (512-bit) chunks:
   - expand 16 words into 80 words by mixing & shifting
   - use 4 rounds of 20 bit operations on message block & buffer
   - add output to input to form new buffer value
5. output hash value is the final buffer value
SHA-1 Compression Function

- each round has 20 steps which replaces the 5 buffer words thus:
  
  \[(A, B, C, D, E) \leftarrow (E + f(t, B, C, D) + (A \ll 5) + W_t + K_t), A, (B \ll 30), C, D)\]

- \(a, b, c, d\) refer to the 4 words of the buffer
- \(t\) is the step number
- \(f(t, B, C, D)\) is nonlinear function for round
- \(W_t\) is derived from the message block
- \(K_t\) is a constant value derived from sin

Note: addition (+) is mod \(2^{32}\)
SHA-1 verses MD5

- brute force attack is harder (160 vs 128 bits for MD5)
- a little slower than MD5 (80 vs 64 steps)
- both designed as simple and compact
- optimised for big endian CPU’s (vs MD5 which is optimised for little endian CPU’s)
- found a collision in SHA-1 in $2^{63}$, instead of $2^{80}$, operations.

Revised Secure Hash Standard

- NIST have issued a revision FIPS 180-2
- adds 3 additional hash algorithms
- SHA-256, SHA-384, SHA-512
- designed for compatibility with increased security provided by the AES cipher
- structure & detail is similar to SHA-1
- hence analysis should be similar
SHA-512 Overview

- heart of the algorithm
- processing message in 1024-bit blocks
- consists of 80 rounds
  - updating a 512-bit buffer
  - using a 64-bit value Wt derived from the current message block
  - and a round constant based on cube root of first 80 prime numbers
Keyed Hash Functions as MACs

- have desire to create a MAC using a hash function rather than a block cipher
  - because hash functions are generally faster
  - not limited by export controls unlike block ciphers
- hash includes a key along with the message
- original proposal:
  \[\text{KeyedHash} = \text{Hash}(\text{Key} \| \text{Message})\]
  - some weaknesses were found with this
- eventually led to development of HMAC
HMAC (RFC 2104)

- **Prepend key twice, and digest twice**
- uses hash function on the message:
  \[
  \text{HMAC}_K = \text{Hash}[(K^+ \text{ XOR opad}) || \text{Hash}[(K^+ \text{ XOR ipad}) || M]]
  \]
- where $K^+$ is the key padded with 0 to 512-bit *(not public key)*
- and opad, ipad are specified padding constants
- any of MD5, SHA-1, SHA-512 can be used

---

HMAC Overview
HMAC Security

- know that the security of HMAC relates to that of the underlying hash algorithm
- attacking HMAC requires either:
  - brute force attack on key used
  - birthday attack (but since keyed would need to observe a very large number of messages)
- choose hash function used based on speed verses security constraints

Summary

- have considered:
  - some current hash algorithms: MD5, SHA
  - HMAC authentication using hash function