Purpose: Authentication
Not Encryption

Authentication Requirements:

- □ Masquerade – Insertion of message from fraudulent source
- □ Content Modification – Changing content of message
- □ Sequence Modification – Insertion, deletion and reordering sequence
- □ Timing Modification – Replaying valid sessions

Background Theory

- Message Digest or “Fingerprint”
  - → Condensed Representation
  - → Easy to generate for a given file.
- Computationally infeasible to produce two messages with same message digest
- Impossible to recreate a message given a message digest.
- Data Integrity and Comparison Checking
  - → Message Integrity Validation

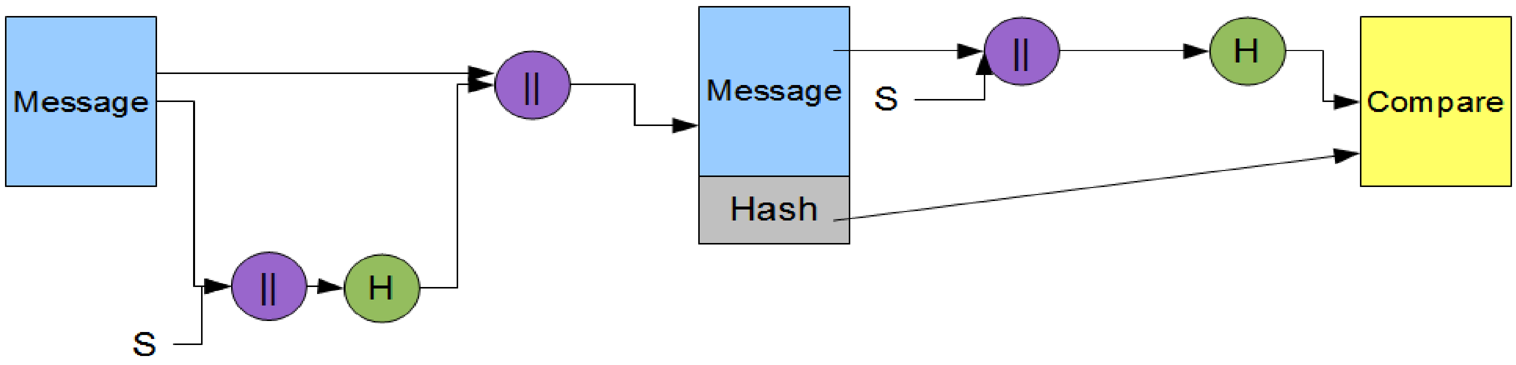
Applications:
One-way hash functions

- Public Key Algorithms
  - Password Logins
  - Encryption Key Management
  - Digital Signatures
- Integrity Checking
  - Virus and Malware Scanning
- Authentication
  - Secure Web Connections
    - (PGP, SSL, SSH, S/MIME)

Variants

- MD4 and MD5 by Ron Rivest (1990,1994)
- SHA-0, SHA-1 by NSA (1993, 1995)
- RIPEMD-160 (1996)
- SHA-2 (2002 – 224, 256, 385, 512)
- Whirlpool
- Tiger
- GOST-3411
- SHA-3
    - Winner selected from solicitations in 2012

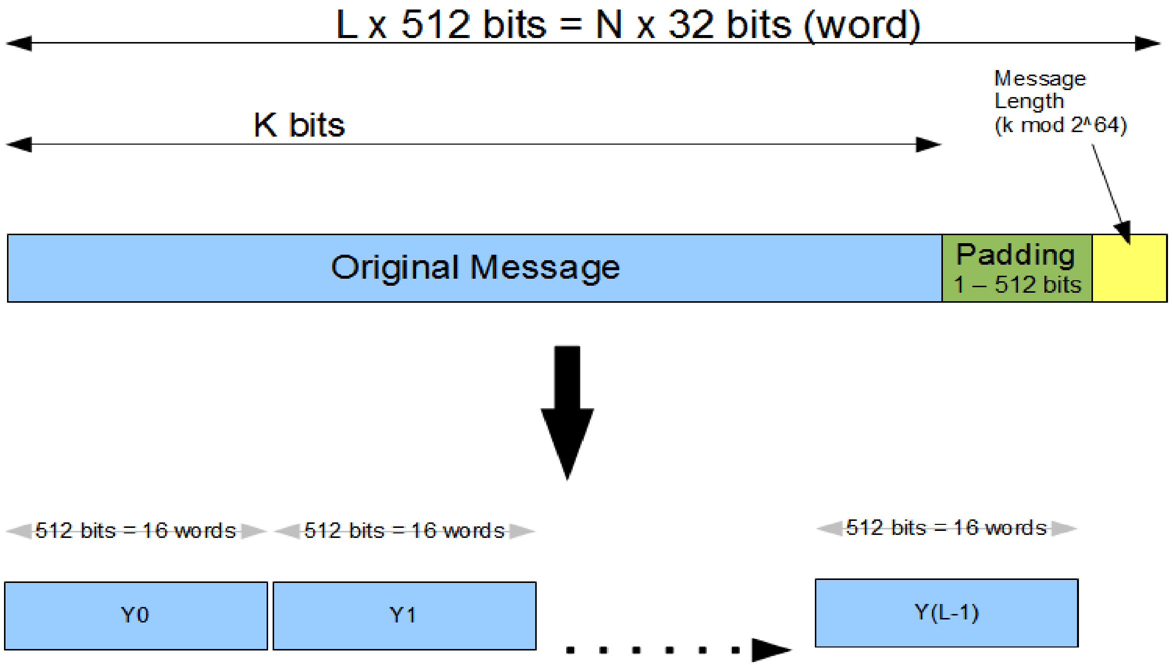
Basic Hash Function Diagram



NOTE: No Encryption with sender and destination holding single security key (S).

Cryptography and Network Security by Stalling

Message Diagram



Cryptography and Network Security by Stalling

SHA-1 (160 bit message) Algorithm Framework

- Step 1: Append Padding Bits....

  Message is “padded” with a 1 and as many 0’s as necessary to bring the message length to 64 bits fewer than an even multiple of 512.

- Step 2: Append Length....

  64 bits are appended to the end of the padded message. These bits hold the binary format of 64 bits indicating the length of the original message.

- http://www.herongyang.com

SHA-1 Framework Continued

- Step 3: Prepare Processing Functions....

SHA1 requires 80 processing functions defined as:

f(t;B,C,D) = (B AND C) OR ((NOT B) AND D)	(0 <= t <= 19)
f(t;B,C,D) = B XOR C XOR D	(20 <= t <= 39)
f(t;B,C,D) = (B AND C) OR (B AND D) OR (C AND D)	(40 <= t <=59)
f(t;B,C,D) = B XOR C XOR D	(60 <= t <= 79)

- Step 4: Prepare Processing Constants....

SHA1 requires 80 processing constant words defined as:

K(t) = 0x5A827999	(0 <= t <= 19)
K(t) = 0x6ED9EBA1	(20 <= t <= 39)
K(t) = 0x8F1BBCDC	(40 <= t <= 59)
K(t) = 0xCA62C1D6	(60 <= t <= 79)

- http://www.herongyang.com

SHA-1 Framework Continued

- Step 5: Initialize Buffers….

SHA1 requires 160 bits or 5 buffers of words (32 bits):

H0 = 0x67452301

H1 = 0xEFCDAB89

H2 = 0x98BADCFE

H3 = 0x10325476

H4 = 0xC3D2E1F0

- http://www.herongyang.com

SHA-1 Framework Final Step

- Step 6: Processing Message in 512-bit blocks (L blocks in total message)….

This is the main task of SHA1 algorithm which loops through the padded and appended message in 512-bit blocks.

Input and predefined functions:

M[1, 2, ..., L]: Blocks of the padded and appended message
f(0;B,C,D), f(1,B,C,D), ..., f(79,B,C,D): 80 Processing Functions K(0), K(1), ..., K(79): 80 Processing Constant Words

H0, H1, H2, H3, H4, H5: 5 Word buffers with initial values

- http://www.herongyang.com

SHA-1 Framework Continued

- Step 6: Pseudo Code….

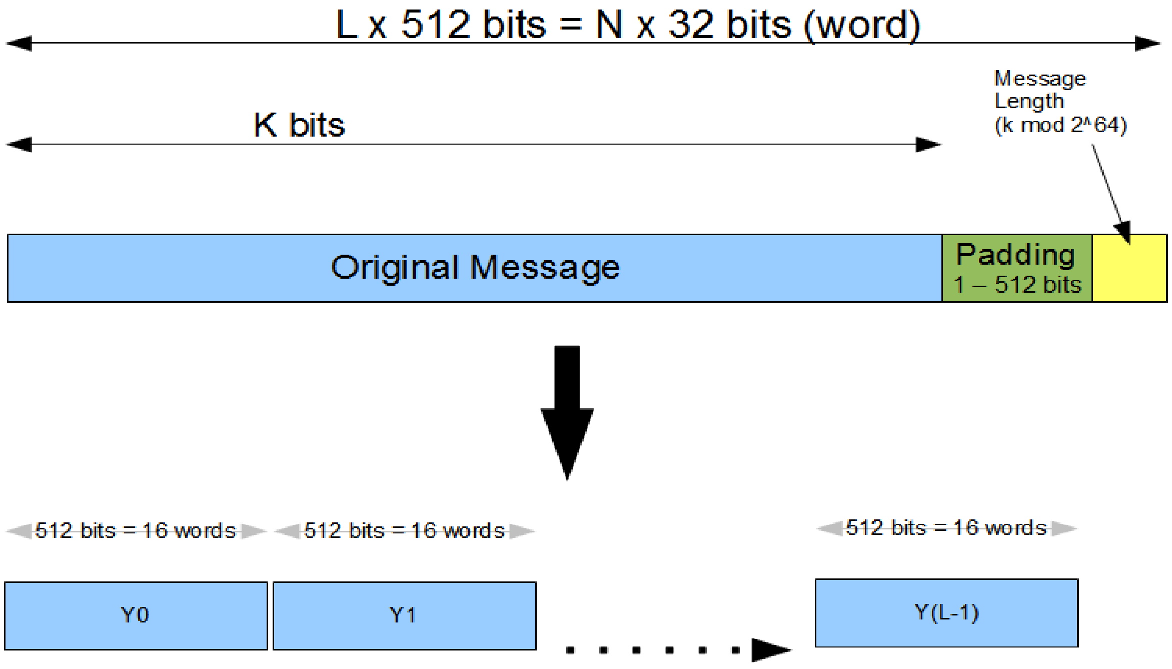
```
For loop on k = 1 to L
        (W(0),W(1),...,W(15)) = M[k] /* Divide M[k] into 16 words */
        For t = 16 to 79 do:
            W(t) = (W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16)) <<< 1
        A = H0, B = H1, C = H2, D = H3, E = H4
        For t = 0 to 79 do:
            TEMP = A<<<5 + f(t;B,C,D) + E + W(t) + K(t) E = D, D = C,
                                C = B<<<30, B = A, A = TEMP
        End of for loop
      H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E
 End of for loop
```

Output:

H0, H1, H2, H3, H4, H5: Word buffers with final message digest

- http://www.herongyang.com

Message Diagram



Cryptography and Network Security by Stalling

SHA-1 Message Digest

The message digest of the string:

“This is a test for theory of computation”

4480afca4407400b035d9debeb88bfc402db514f

Cryptanalysis and Limitation

- Key Premises for Hash Functions:

  1. Impossible to re-create a message given a fingerprint

  2. Collision Free

- SHA-1 failure using brute force attack in 2^{80} operations
- Collision failure found in 2005 in 2^{33} operations