Assignment Number : E-III D1

March 3, 2016

Problem Statement

A message is to be transmitted using network resources from one machine to another calculate and demonstrate the use of a Hash value equivalent to SHA-1. Develop program in C++/Python/Scala/Java using Eclipse.

Objective

- To study SHA-1 Algorithm.
- To understand the application of SHA-1 Algorithm.

Theory

Secure Hash Algorithm 1

Introduction

SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function designed by the United States National Security Agency and is a U.S. Federal Information Processing Standard published by the United States NIST. SHA-1 produces a 160-bit (20-byte) hash value known as a message digest. A SHA-1 hash value is typically rendered as a hexadecimal number, 40 digits long.

When a message of any $length < 2^{64}$ bits is input, the SHA-1 produces a 160-bit output called a message digest. The message digest can then,

for example, be input to a signature algorithm which generates or verifies the signature for the message. Signing the message digest rather than the message often improves the efficiency of the process because the message digest is usually much smaller in size than the message. The same hash algorithm must be used by the verifier of a digital signature as was used by the creator of the digital signature. Any change to the message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify.
The SHA-1 is called secure because it is computationally infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest. Any change to a message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify.

1. **Operations on Words:**
   The following logical operators will be applied to words:

   - Bitwise logical word operations:
     X AND Y = bitwise logical "and" of X and Y.
     X OR Y = bitwise logical "inclusive-or" of X and Y.
     X XOR Y = bitwise logical "exclusive-or" of X and Y.
     NOT X = bitwise logical "complement" of X.
   - The operation X + Y is defined as follows:
     words X and Y represent integers x and y, where $0 <= x < 2^{32}$ and $0 <= y < 2^{32}$. For positive integers n and m, let n mod m be the remainder upon dividing n by m.
   - The circular left shift operation $S^n(X)$, where X is a word and n is an integer with $0 <= n < 32$, is defined by
     $S^n(X) = (X << n) OR (X >> 32 - n)$.
     In the above, $X << n$ is obtained as follows: discard the left-most n bits of X and then pad the result with n zeroes on the right (the result will still be 32 bits). $X >> n$ is obtained by discarding the right-most n bits of X and then padding the result with n zeroes on the left. Thus $S^n(X)$ is equivalent to a circular shift of X by n positions to the left.

2. **Message Padding:**
   SHA-1 is used to compute a message digest for a message or data file that is provided as input. The message or data file should be considered to be a bit string. The length of the message is the number of bits in the message (the empty message has length 0). If the number of bits in a message is a multiple of 8, for compactness we can represent the message in hex. The purpose of message padding is to make the total length of a padded message a multiple of 512. SHA-1 sequentially processes blocks of 512 bits when computing the message digest. The following specifies how this padding shall be performed. As a summary, a "1" followed by m "0"s followed by a 64- bit integer are appended to the end of the message to produce a padded message of length 512 * n. The 64-bit integer is the length of the original message. The padded message is then processed by the SHA-1 as n 512-bit blocks.

   Suppose a message has length $l < 2^{64}$. Before it is input to the SHA-1, the message is padded on the right as follows:

   (a) "1" is appended. Example: if the original message is "01010000", this is padded to "010100001".

   (b) "0"s are appended. The number of "0"s will depend on the original length of the message. The last 64 bits of the last 512-bit block are reserved for the length l of the original message.

   Example: Suppose the original message is the bit string
   01100001 01100010 01100011 01100100 01100101.
   After step (a) this gives
   01100001 01100010 01100011 01100100 01100101 1.
   Since l = 40, the number of bits in the above is 41 and 407 "0"s are appended, making the total now 448. This gives (in hex)

   61626364 65800000 00000000 00000000
   00000000 00000000 00000000 00000000
   00000000 00000000 00000000 00000000
   00000000 00000000.

(c) Obtain the 2-word representation of l, the number of bits in the original message. If $l < 2^{32}$ then the first word is all zeroes. Append these two words to the padded message.

Example: Suppose the original message is as in (b). Then l = 40 (note that l is computed before any padding). The two-word representation of 40 is hex 00000000 00000028. Hence the final padded message is hex

61626364 65800000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000028.

The padded message will contain 16 * n words for some $n > 0$. The padded message is regarded as a sequence of n blocks M(1) , M(2), first characters (or bits) of the message.

3. **Functions and Constants Used:**
A sequence of logical functions $f(0), f(1), ..., f(79)$ is used in SHA-1. Each f(t), $0 <= t <= 79$, operates on three 32-bit words B, C, D and produces a 32-bit word as output. f(t;B,C,D) is defined as follows: for words B, C, D,

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 ($60 <= t <= 79$).

A sequence of constant words $K(0), K(1), ..., K(79)$ is used in the SHA-1. In hex these are given by

K(t) = 5A827999 ($0 <= t <= 19$)
K(t) = 6ED9EBA1 ($20 <= t <= 39$)

K(t) = 8F1BBCDC ($40 <= t <= 59$)
K(t) = CA62C1D6 ($60 <= t <= 79$).

4. **Computing the Message Digest:**
The methods given below yield the same message digest.

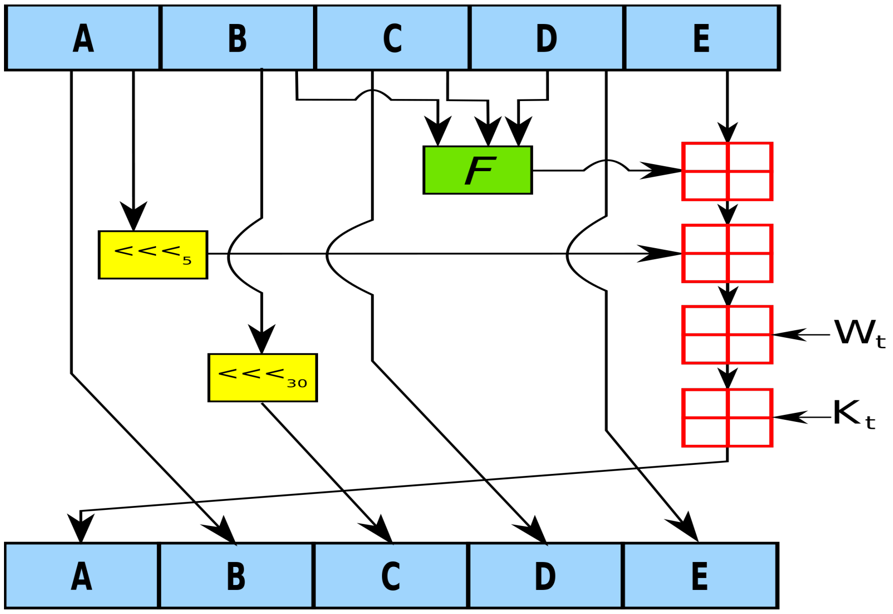

Figure 1: Round 1 of SHA-1

(a) Method 1:
The message digest is computed using the message padded as described in section 4. The computation is described using two buffers, each consisting of five 32-bit words, and a sequence of eighty 32-bit words. The words of the first 5-word buffer are labelled A,B,C,D,E. The words of the second 5-word buffer are labelled H0, H1, H2, H3, H4. The words of the 80-word sequence are labelled $W(0), W(1), ..., W(79)$. A single word buffer TEMP is also employed.
To generate the message digest, the 16-word blocks $M(1), M(2), ..., M(n)$

defined in section 4 are processed in order. The processing of each M(i) involves 80 steps.

Before processing any blocks, the H's are initialized as follows: in hex,
H0 = 67452301
H1 = EFCDAB89
H2 = 98BADCFE
H3 = 10325476
H4 = C3D2E1F0.

Now $M(1), M(2), ..., M(n)$ are processed. To process M(i), we proceed as follows:

- Divide M(i) into 16 words $W(0), W(1), ..., W(15)$, where W(0) is the left-most word.
- For t = 16 to 79 let
  $W(t) = S^1(W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16))$.
- Let A = H0, B = H1, C = H2, D = H3, E = H4
- For t = 0 to 79 do
  $TEMP = S^5(A) + f(t; B, C, D) + E + W(t) + K(t);$
  $E = D; D = C; C = S^{30}(B); B = A; A = TEMP;$
- Let H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E.

After processing M(n), the message digest is the 160-bit string represented by the 5 words
H0 H1 H2 H3 H4.

(b) Method 2:
The method above assumes that the sequence $W(0), ..., W(79)$ is implemented as an array of eighty 32-bit words. This is efficient from the standpoint of minimization of execution time, since the addresses of $W(t-3), ..., W(t-16)$ in step (b) are easily computed. If space is at a premium, an alternative is to regard W(t)

as a circular queue, which may be implemented using an array of sixteen 32-bit words $W[0], ...W[15]$.
In this case, in hex let $MASK = 0000000F$.
Then processing of M(i) is as follows:

- Divide M(i) into 16 words $W[0], ..., W[15]$, where W[0] is the left-most word.
- Let A = H0, B = H1, C = H2, D = H3, E = H4.
- For t = 0 to 79 do
  $s = tANDMASK;$
  $if(t >= 16)$
  $W[s] = S^1(W[(s+13) \quad AND \quad MASK] \quad XOR \quad W[(s+8) \quad AND \quad MASK] \quad XOR \quad W[(s+2) \quad AND \quad MASK] \quad XOR \quad W[s]);$
  $TEMP = S^5(A) + f(t; B, C, D) + E + W[s] + K(t);$
  $E = D; \quad D = C; \quad C = S^{30}(B); \quad B = A; \quad A = TEMP;$
- Let H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E.

After processing M(n), the message digest is the 160-bit string represented by the 5 words
H0 H1 H2 H3 H4.

Algorithm (Steps to follow)

Server Side:

1. Start.
2. Create socket.
3. Listen to the port.
4. Accept the the Message and Digest from the Client.
5. Compute Digest of the received message.
6. Check if the computed and received Digests are same.

7. Print the Message, Digest and the result whether the message is altered or not.
8. Close socket connection.
9. End.

Client Side:

1. Start.
2. Create socket.
3. Connect to the Server.
4. Accept the the Message to be sent from the User.
5. Compute Digest of the message.
6. Send the Message and Digest to the Server.
7. Print the Message and Digest.
8. Close socket connection.
9. End.

Mathematical Model

Let S be the system that represents the SHA-1 Algorithm.

Initially,

$$\mathbf{S} = \{\phi\}$$

Let,

$$\mathbf{S} = \{\mathbf{I}, \mathbf{O}, \mathbf{F}\}$$

Where,

- I - Represents Input set
- O - Represents Output set
- F - Represents Function set

Input set - I:

$$\mathbf{I} = \{M\}$$

Where,

- M - Represents the input message.

Output set - O:

$$\mathbf{O} = \{H\}$$

Where,

- H - Represents the hash digest.

Function set :

$$\mathbf{F} = \{F_1, F_2, F_3\}$$

Where,

- F_1 - Represents the function for generating chunks of data.
  $F_1(L) -> \{B\}$
  - L - Represents the length of data chunks.
  - B - Represents the input data chunks in bytes.
- F_2 - Represens the function for shifting bits.
  $F_2(N, B) -> \{B'\}$
  - N - Represents the bits to be rotated.
  - B - Represents the input data.
  - B' - Represents the shifted data.
- F_3 - Represents the function for digest calculation.
  $F_3(B') -> \{H\}$
  - B' - Represents the chunk of data.
  - H - Represents the hash digest.

Finally,

$$\mathbf{S} = \{I, O, F\}$$

Conclusion

Thus, studied how to implement SHA-1 algorithm.