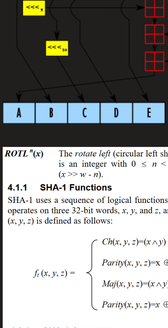
SHA-1

Thursday, February 25, 2021 13:48

How Does SHA-1 Work - Intro to Cryptographic Hash Functions and SHA-1

SHA-1 attack (SHAttered): http://shattered.io/

SHA-1 attack (SHAmbles): https://sha-mbles.github.io/



ROTL ^{n}**(x)** The *rotate left* (circular left shift) operation, where x is a w-bit word and n is an integer with $0 \le n < w$, is defined by $ROTL^{n}(x)=(x << n) \vee (x >> w - n)$.

4.1.1 SHA-1 Functions

SHA-1 uses a sequence of logical functions, $f_0, f_1, \ldots, f_{79}$. Each function f_t, where $0 \le t \le 79$, operates on three 32-bit words, x, y, and z, and produces a 32-bit word as output. The function $f_t(x, y, z)$ is defined as follows:

$$f_t(x,y,z) = \begin{cases} Ch(x,y,z)=(x \wedge y) \oplus (\neg x \wedge z) & 0 \le t \le 19 \\ Parity(x,y,z)=x \oplus y \oplus z & 20 \le t \le 39 \\ Maj(x,y,z)=(x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \le t \le 59 \\ Parity(x,y,z)=x \oplus y \oplus z & 60 \le t \le 79. \end{cases} \quad (4.1)$$

4.2.1 SHA-1 Constants

SHA-1 uses a sequence of eighty constant 32-bit words, $K_0, K_1, \ldots, K_{79}$, which are given by

$$K_t = \begin{cases} \texttt{5a827999} & 0 \le t \le 19 \\ \texttt{6ed9eba1} & 20 \le t \le 39 \\ \texttt{8f1bbcdc} & 40 \le t \le 59 \\ \texttt{ca62c1d6} & 60 \le t \le 79 \end{cases} \quad (4.14)$$

5.1.1 SHA-1, SHA-224 and SHA-256

Suppose that the length of the message, M, is ℓ bits. Append the bit "1" to the end of the message, followed by k zero bits, where k is the smallest, non-negative solution to the equation $\ell + 1 + k \equiv 448 \bmod 512$. Then append the 64-bit block that is equal to the number ℓ expressed using a binary representation. For example, the (8-bit ASCII) message "**abc**" has length $8 \times 3 = 24$, so the message is padded with a one bit, then $448 - (24 + 1) = 423$ zero bits, and then the message length, to become the 512-bit padded message

$$\underbrace{\texttt{01100001}}_{\text{"a"}}\ \underbrace{\texttt{01100010}}_{\text{"b"}}\ \underbrace{\texttt{01100011}}_{\text{"c"}}\ \ 1\ \ \overbrace{\texttt{00...00}}^{423}\ \ \overbrace{\underbrace{\texttt{00...011000}}_{\ell = 24}}^{64}$$

5.2.1 SHA-1, SHA-224 and SHA-256

For SHA-1, SHA-224 and SHA-256, the message and its padding are parsed into N 512-bit blocks, $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$. Since the 512 bits of the input block may be expressed as sixteen 32-bit words, the first 32 bits of message block i are denoted $M_0^{(i)}$, the next 32 bits are $M_1^{(i)}$, and so on up to $M_{15}^{(i)}$.

5.3.1 SHA-1

For SHA-1, the initial hash value, $H^{(0)}$, shall consist of the following five 32-bit words, in hex:

$$\begin{aligned} H_0^{(0)} &= \texttt{67452301} \\ H_1^{(0)} &= \texttt{efcdab89} \\ H_2^{(0)} &= \texttt{98badcfe} \\ H_3^{(0)} &= \texttt{10325476} \\ H_4^{(0)} &= \texttt{c3d2e1f0} \end{aligned}$$

6.1 SHA-1

SHA-1 may be used to hash a message, M, having a length of ℓ bits, where $0 \le \ell < 2^{64}$. The algorithm uses 1) a message schedule of eighty 32-bit words, 2) five working variables of 32 bits each, and 3) a hash value of five 32-bit words. The final result of SHA-1 is a 160-bit message digest.

The words of the message schedule are labeled $W_0, W_1, \ldots, W_{79}$. The five working variables are labeled $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}$, and $\boldsymbol{e}$. The words of the hash value are labeled $H_0^{(i)}, H_1^{(i)}, \ldots, H_4^{(i)}$, which will hold the initial hash value, $H^{(0)}$, replaced by each successive intermediate hash value (after each message block is processed), $H^{(i)}$, and ending with the final hash value, $H^{(N)}$. SHA-1 also uses a single temporary word, T.

6.1.1 SHA-1 Preprocessing

1. Set the initial hash value, $H^{(0)}$, as specified in Sec. 5.3.1.
2. The message is padded and parsed as specified in Section 5.

6.1.2 SHA-1 Hash Computation

The SHA-1 hash computation uses functions and constants previously defined in Sec. 4.1.1 and Sec. 4.2.1, respectively. Addition (+) is performed modulo 2^{32}.

Each message block, $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$, is processed in order, using the following steps:

For i=1 to N:
{

1. Prepare the message schedule, $\{W_t\}$:

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ ROTL^{1}(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & 16 \le t \le 79 \end{cases}$$

2. Initialize the five working variables, $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}$, and $\boldsymbol{e}$, with the (i-1)th hash value:

$$\begin{aligned} a &= H_0^{(i-1)} \\ b &= H_1^{(i-1)} \\ c &= H_2^{(i-1)} \\ d &= H_3^{(i-1)} \\ e &= H_4^{(i-1)} \end{aligned}$$

3. For t=0 to 79:
{

$$\begin{aligned} T &= ROTL^{5}(a) + f_t(b,c,d) + e + K_t + W_t \\ e &= d \\ d &= c \\ c &= ROTL^{30}(b) \\ b &= a \\ a &= T \end{aligned}$$

}

4. Compute the ith intermediate hash value $H^{(i)}$:

$$\begin{aligned} H_0^{(i)} &= a + H_0^{(i-1)} \\ H_1^{(i)} &= b + H_1^{(i-1)} \\ H_2^{(i)} &= c + H_2^{(i-1)} \\ H_3^{(i)} &= d + H_3^{(i-1)} \\ H_4^{(i)} &= e + H_4^{(i-1)} \end{aligned}$$

}

After repeating steps one through four a total of N times (i.e., after processing $M^{(N)}$), the resulting 160-bit message digest of the message, M, is

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)}$$

6.1.3 Alternate Method for Computing a SHA-1 Message Digest

The SHA-1 hash computation method described in Sec. 6.1.2 assumes that the message schedule $W_0, W_1, \ldots, W_{79}$ is implemented as an array of eighty 32-bit words. This is efficient from the standpoint of the minimization of execution time, since the addresses of $W_{t-3}, \ldots, W_{t-16}$ in step (2) of Sec. 6.1.2 are easily computed.

However, if memory is limited, an alternative is to regard $\{W_t\}$ as a circular queue that may be implemented using an array of sixteen 32-bit words, $W_0, W_1, \ldots, W_{15}$. The alternate method that is described in this section yields the same message digest as the SHA-1 computation method described in Sec. 6.1.2. Although this alternate method saves sixty-four 32-bit words of storage, it is likely to lengthen the execution time due to the increased complexity of the address computations for the $\{W_t\}$ in step (3).

For this alternate SHA-1 method, let $MASK$=`0000000f` (in hex). As in Sec. 6.1.1, addition is performed modulo 2^{32}. Assuming that the preprocessing as described in Sec. 6.1.1 has been performed, the processing of $M^{(i)}$ is as follows:

For i=1 to N:
{

1. For t=0 to 15:
{

$$W_t = M_t^{(i)}$$

}

2. Initialize the five working variables, $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}$, and $\boldsymbol{e}$, with the (i-1)th hash value:

$$\begin{aligned} a &= H_0^{(i-1)} \\ b &= H_1^{(i-1)} \\ c &= H_2^{(i-1)} \\ d &= H_3^{(i-1)} \\ e &= H_4^{(i-1)} \end{aligned}$$

3. For t=0 to 79:
{

$$s = t \wedge MASK$$

If $t \ge 16$ then
{

$$W_s = ROTL^{1}(W_{(s+13)\text{-}MASK} \oplus W_{(s+8)\text{-}MASK} \oplus W_{(s+2)\text{-}MASK} \oplus W_s)$$

}

$$\begin{aligned} T &= ROTL^{5}(a) + f_t(b,c,d) + e + K_t + W_s \\ e &= d \\ d &= c \\ c &= ROTL^{30}(b) \\ b &= a \\ a &= T \end{aligned}$$

}

4. Compute the ith intermediate hash value $H^{(i)}$:

$$\begin{aligned} H_0^{(i)} &= a + H_0^{(i-1)} \\ H_1^{(i)} &= b + H_1^{(i-1)} \\ H_2^{(i)} &= c + H_2^{(i-1)} \\ H_3^{(i)} &= d + H_3^{(i-1)} \\ H_4^{(i)} &= e + H_4^{(i-1)} \end{aligned}$$

}

After repeating steps one through four a total of N times (i.e., after processing $M^{(N)}$), the resulting 160-bit message digest of the message, M, is

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)}$$