SHA-1

1 Preprocessing

1.1 Padding the Message

The message M is padded before hash computation begins. The purpose of this padding is to ensure that the padded message is a multiple of 512 bits.

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[1] to the equation $l + 1 + k \equiv 448 \ mod\ 512$. 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 1, then with $(448 - 24 - 1)\ mod\ 512 = 423$ zero bits, and then the message length, to become the 512-bit padded message

$$\underbrace{01100001}_{a}\ \underbrace{01100010}_{b}\ \underbrace{01100011}_{c}\ 1\ \overbrace{00\ldots00}^{423}\ \overbrace{00\ldots0\underbrace{11000}_{l=24}}^{64}$$

The length of the padded message will be a multiple of 512 bits.

1.2 Parsing the Padded Message

The padded message is parsed into[2] 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 M^i are denoted M_0^i, the next 32 bits are M_1^i, and so on up to M_{15}^i.

[1] $k = (448 - l - 1)\ mod\ 512$

[2] $N = \frac{l+1+k+64}{512}$

1.3 Setting the Initial Hash Value (H^0)

The initial hash value, H^0, consists of the following five 32-bit words (in hex): $H_0^0 = 67452301$, $H_1^0 = efcdab89$, $H_2^0 = 98badcfe$, $H_3^0 = 10325476$, $H_4^0 = c3d2e1f0$.

2 Hash Computation

The SHA-1 hash computation uses some functions and constants which will be defined below. After preprocessing is completed, the message blocks $M^1, M^2, \ldots, M^N$, are processed in order, using the following steps:

`for` i=1 `to` N `do`

{

1. Prepare the message schedule, $\{W_t\}$:
   $$W_t = \begin{cases} M_t^i, & \text{if } 0 \leq t \leq 15; \\ ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}), & \text{if } 16 \leq t \leq 79. \end{cases}$$
2. Initialize the five working variables, a, b, c, d, e with the $(i-1)^{st}$ hash value:
   $a = H_0^{i-1}$
   $b = H_1^{i-1}$
   $c = H_2^{i-1}$
   $d = H_3^{i-1}$
   $e = H_4^{i-1}$
3. `for` t=0 `to` 79 `do`

   {

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

   }
4. Compute the i^{th} intermediate hash value H^i:
   $H_0^i = a + H_0^{i-1}$
   $H_1^i = b + H_1^{i-1}$
   $H_2^i = c + H_2^{i-1}$
   $H_3^i = d + H_3^{i-1}$
   $H_4^i = e + H_4^{i-1}$

}

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** $H_0^N H_1^N H_2^N H_3^N H_4^N$.

3 SHA–1 functions and constants

$ROTL^n(x)$	the circular shift (rotation) of x by n positions to the left
$\oplus$	the bitwise XOR operation
$+$	addition modulo 2^{32}
$\wedge$	the bitwise AND operation
$\neg$	the bitwise complement operation
$f_t(x,y,z)$	$f_t(x,y,z) = \begin{cases} (x \wedge y) \oplus (\neg x \wedge z), & \text{if } 0 \le t \le 19; \\ x \oplus y \oplus z, & \text{if } 20 \le t \le 39; \\ (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z), & \text{if } 40 \le t \le 59; \\ x \oplus y \oplus z, & \text{if } 60 \le t \le 79. \end{cases}$
K_t	$K_t = \begin{cases} 5a827999, & \text{if } 0 \le t \le 19; \\ 6ed9eba1, & \text{if } 20 \le t \le 39; \\ 8f1bbcdc, & \text{if } 40 \le t \le 59; \\ ca62c1d6, & \text{if } 60 \le t \le 79. \end{cases}$

4 SHA-1 Examples

See `http://csrc.nist.gov/groups/ST/toolkit/documents/Examples/SHA1.pdf`