Analysis and Design of SHA-V and RIPEMD-V with Variable Output-Length

Her, Yong-Sork
Department of Computer Science and Communication Engineering, Graduate School of Information Science and Electrical Engineering, Kyushu University : Graduate Student

Sakurai, Koichi
Department of Computer Science and Communication Engineering, Faculty of Information Science and Electrical Engineering, Kyushu University

https://doi.org/10.15017/1515810

出版情報：九州大学大学院システム情報科学紀要. 8 (1), pp.13-18, 2003-03-26. 九州大学大学院システム情報科学研究院
バージョン：
権利関係：

九州大学大学院
システム情報科学紀要
第8巻 第1号 平成15年3月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.8, No.1, March 2003

Analysis and Design of SHA-V and RIPEMD-V with Variable Output-Length

Yong-Sork HER* and Kouichi SAKURAI**

(Received December 13, 2002)

Abstract: A hash function provides services of information security, authentication, integrity and non-reputation in a branch of information security. Cryptographic hash functions had been developed since MD4 was proposed by Rivest. U. S standard of a hash function is SHA-1 with 160 bits of output length. RIPEMD was designed in 1992 by den Boer and others under the RIPE project. When we consider the improvement of computation ability and speed, it can be difficult to guarantee the security of a hash function with 160 bits of output length. It is required a hash function with variable output length that can take a suitable output length by systems. HAVAL is the first hash function with variable output length , which was proposed by Zheng et al. HAS-V based on HAVAL-1 was proposed by N. K. Park et al. In this paper, we design two hash functions with variable output length, namely SHA-V and RIPEMD-V, based on SHA-1 and RIPEMD-1, and analyze the security on two designed hash functions.

Keywords: Cryptographic hash function, SHA-1, RIPEMD-1, Authentication, Cryptography

1. Introduction

1.1 Background

Hash functions [2] take a message as input with an arbitrary and produce an output referred to as a hash-code, hash-result, hash-value, or simply hash. In 1990, Rivest proposed the cryptographic hash function MD4 [17]. MD4 has a 128-bit hash function and consists of 3 rounds. Den Boer, Bosselaers[4] and Dobbertin [8] proposed an attack method on the each round of MD4. So, it is undesirable to use MD4. There are hash functions, MD-5[19], SHA-1[15], RIPEMD-160 [7] and so on. These hash functions are called the customized hash functions based on MD4. As computer technologies progress, the output length of a hash value gradually grow for the security of a hash function. It was known to be desirable that output lengths of hash functions are more longer than 160 bits.

Generally, three potential properties are listed for an unkeyed hash function h with inputs x, x' and outputs y, y'[2] in cryptographic hash functions.

Preimage resistance: It is computationally infeasible to find any input which hashes to that output, i.e., to find any preimage x' such that $h(x') = y$ when given any y for which a corresponding input is not known.

Second preimage resistance: It is computationally infeasible to find any second input which has the same output as any specified input, i.e., given x, to find a 2nd-preimage $x' \neq x$ such that $h(x) = h(x')$.

Collision resistance: It is computationally infeasible to find any two distinct inputs x, x' which hash to the same input, *i.e.*, such that $h(x) = h(x')$.

1.2 Motivation

Many application systems using computer and network have been developed, and these systems require the high security. Almost hash functions have only fixed output length. The length of the hash value is an important factor directly connected to the security of the hash function[16]. But, the long hash values drop off system speed and efficiency. To take variable output-length via a single hash function, it is impotant technique in aspect of a hash function. We can expect that hash functions become a cost-effective and efficiency through the hash function with variable output length.

2. Algorithm techniques

In this section, we explain the design technique of a hash function. Almost all hash functions are iterative processes which hash inputs of arbitrary length by processing successive fixed-length blocks of the input. The input X is padded to a multiple of the block length and subsequently divided into t blocks X_1 through X_t. The hash function h can then be described as follows :

$H_0 = IV$; $H_i = f(H_{i-1}, X_i)$, $h(X) = H_t$

(H_i : the chaining variable between stage $i-1$ and stage i, f : Compression function of h)

Attack of Birthday paradox

* Department of Computer Science and Communication Engineering, Graduate Student

** Department of Computer Science and Communication Engineering

The collisioin of a hash function can be found by the birthday paradox. According to the attack of birthday paradox, it can find collision pairs through the operation of $2^{n/2}$ on the hash function with n bits output length. So, the minimum output length is 128 bits. In our proposal hash functions, the minimum output length is 128 bits. But, in general, it was known to be deisrable that output lengths of hash functions are more longer than 160 bits.

A collision free

Damagard[10] proved that if a compression function f is a collision free on a hash function h, it can keep a collision free. Our proposal hash functions are parallelizing hash functions for variable output lengh.

< *Theorem* >: Let F be a collision free function family mapping m bits to $t(m)$ bits. Then, there exists a collision free hash function family H mapping arbitrary strings to $t(m)$-bit strings with the following property : Let h be an instance in H of size m. Put $t = t(m)$. Then evaluating h on input of length n can be done in $O(\frac{\log_2(n/t)t}{(m-t)})$ steps using $n/2t$ processors. Generally, the compression function of the proposed hash function satisfied this theorm. Compression functions of SHA-V and RIPEMD-V should be satisfied the upper theorem.

3. Description of SHA-V

We propose SHA-V based on SHA-1 and HAVAL. The structure of SHA-V is two parallel lines, denoted as Left-line and Right-line, consisting of 80 steps each. The input length is 1024 bits, and the output length is from 128 bits to 320 bits by 32-bit. SHA-V has the most advantage of SHA-1. That is, new messages is created in combination with input message and step calculations. These new messages provide the resistance against most of attacks that search the collision resistance by the fabricating of input messages. We explain the detailed description of SHA-V in appendix A.

3.1 Initial values

The initial values of the chaining are used in SHA-V are given in appendix A. SHA-V is divided into two parallel lines: left-line and right-line. The initial values of left line, $H_0(i), H_1(i), H_2(i), H_3(i), H_4(i)$, are the same SHA-1, ($H_j(i)$: chaining variables). The initial values of right line are based on HAS-V[15]. In order to increase the output length, SHA-V has ten 32-bit words.

3.2 Constants

The constants of left-line are based on SHA-1 except the third constant. That is, the third constant of SHA-1 is Oxca62c1d6 ($2^{30}\sqrt{10}$), but the third constant of SHA-V is Oxa953fd4e ($2^{30}\sqrt{7}$). The constants of right-line are based on HAS-V, but the constant sequence is different from HAS-V. Also, the constants of HAS-V consist of five 32 bits for 5 rounds. SHA-V consists of 4 rounds and the constants of four 32 bits.

3.3 Boolean functions

The boolean functions to be used each step operations are descriped in appendix A. The boolean functions of left-line are based on SHA-1 and the boolean functions of right-line have the reverse order with left-line.

3.4 Append padding bits

The message words to be used in the compression function are 32 words (or 1024 bits block). The padding bits of SHA-V are based on SHA-1 and HAS-V. The message is padded so that its length is congruent to 952 modulo 1024. Padding is performed by appending a single "1"bit and necessary "0"bit to satisfy the above constraints. The remaining 72 bits, in order to be a multiple of 1024 bits, are filled by appending the desired length of the hash-code represented in bytes.

3.5 Output tailoring

In order to create the output length with variables, the hash-code is computed as appendix A. The output length is $128 + 32(t-3)$. The character of SHA-V is fairly to be computed in each line. For the computation speed, it uses only shift and addition operation. The arrange of output length is from 128 bits to 320 bits by 32-bit. For example, in order to create the 128-bit output, it adds to the each two by two in 10 chaining variables as **Table 1**. In the case of 320-bit hash-code, the output length is the same which is given as the contents of the 10 chaining variables concatenated, i.e.
$H_0(i) \| H_1(i) \| H_2(i) \| H_3(i) \| H_4(i) \| H_5(i) \| H_6(i)$
$\| H_7(i) \| H_8(i) \| H_9(i)$, ($H_j(i)$: chaining variables).

4. Description of RIPEMD-V

RIPEMD[3] was designed in 1992 by den Boer and others under the European RACE Integrity Primitives Evaluation (RIPE) project[2] [3]. RIPEMD has the strong resistance on known attacks of MD4; its

compression function has two parallel computation lines of three 16 step rounds. The output lengths of RIPEMD hash functions are 128, 160, 256 and 320-bit; RIPEMD-128, RIPEMD-160, RIPEMD-256 and RIPEMD-320, namely RIPEMD-family. Early in 1995, Dobbertin[6] found that the reduced versions of RIPEMD, where the first or the last round of the compression function is omitted, are not collision-free. When we design RIPEMD-V, we should be considered this point. RIPEMD-V consists of two parallel lines, denoted as left-line and right-line, consists of 80 steps by each line. Each line is composed of 5 rounds, where each round consists of 16 steps, and maintains 5 words of chaining variables, total of 10 chaining variables for the entire compression function.

4.1 Initial values

The initiall values are divided into two parallel lines like SHA-V: left-line and right-line. Almost cryptographic hash functions with long output length have a parallel structure exclusive of SHA-384 and SHA-512. SHA-384 and 512 have the initial values of 384 and 512-bit. RIPEMD-V uses ten 32-bit words like RIPEMD-320.

4.2 Constants

The constants consist of parallel structure and ten 32-bit words as appendix B. When we compare with the constants of SHA-V, it is increased two 32-bit words in RIPEMD-V. It means to be increased the step operations. RIPEMD-V uses ten 32-bit words.

4.3 Boolean functions

RIPEMD-family uses five (four) boolean functions without being changed these boolean functions because RIPEMD-family has a parallel structure. The boolean functions of RIPEMD-V are used as the reverse order of RIPEMD-320 in each line.

4.4 Shifts and selection of message word

In appendix B, we introduce shifts and selection of message word.

4.5 Append padding bits and length

The message words to be used in the compression function are 32 words (or a 1024 bits block). The message is padded so that its length is congruent to 952 modulo 1024. Padding is performed by appending a single "1" bit and necessary "0" bits to satisfy the above constraints. The remaining 72 bits are filled by appending the desired length of the hash-code represented in bytes. In RIPEMD-V, the calculation method of variable output length **Table 1** is same like SHA-V.

5. Estimations of HAS-V, SHA-V and RIPEMD-V

In this paper, we analyze HAS-V[16], SHA-V and RIPEMD-V through step computations.

5.1 Endianess

Basic structures of compression functions of HAS-V, SHA-V and RIPEMD-V are two parallel lines, and input messages are 1024 bits, too. HAS-V and RIPEMD-V favor 'little-endian' architecture such like MD4-family, whereas SHA-V favors 'big-endian' architecture such like SHA-family. In the result of implementation, both methods can be unified for fairness. Because little-endian architecture is suitable to the pentium processors, and big-endian architecture is suitable to the SUN system.

5.2 Step operations and comparison of speed

The step operation of HAS-V consists of 3 additions, 2 circular shifts and a boolean function. The boolean function consists of 4 unit operations and a single step operation consists of 9 unit operations. The total number of unit operations for generating the extra messages is

$2(lines) \times 5(rounds) \times 4(messages) \times 3(unit\ operations) = 120(unit\ operations)$.

Therefore, the number of unit operations to hash 1024 -bit block is given as follows[16].

$1(block) \times 200(steps) \times 9(step\ operations) + 120\ (message\ expansion) = 1920(unit\ operations)$.

In the case of RIPEMD-V, the total number of unit operation to hash 1024 bit block is given below.

$1(block) \times 160(steps) \times 9(step\ operations) = 1440(unit\ operations)$.

In the case of SHA-V, step operations of SHA-V consist of 80 steps each line, a single step operation consists of 10 unit operations.1024 bits block is given as follows.

$1(block) \times 160(steps) \times 10(step\ operations) = 1600(unit\ operations)$.

HAS-V is about 25% more operation than RIPEMD-V, and SHA-V is about 10% faster than HAS-V. Also, RIPEMD-V is about 10% faster that SHA-V. These values are not practical value not the computation value.

Table 1 The calculation methods of a variable output-length.

	128-bit	160-bit	192-bit	224-bit	256-bit	288-bit	320-bit
O_0	$a + v^{<<2}$	$a + f$	$a + j^{<<2}$	$a + j^{<<2}$	$a + j^{<<2}$	$a + j^{<<2}$	a
O_1	$b + v^{<<3}$	$b + g$	$b + j^{<<3}$	$b + j^{<<3}$	$b + j^{<<3}$	$b + j^{<<3}$	b
O_2		$c + h$	$c + j^{<<5}$	$c + j^{<<5}$	$c + j^{<<5}$	$c + j^{<<5}$	c
O_3		$d + i$		$d + j^{<<7}$	$d + j^{<<7}$	$d + j^{<<7}$	d
O_4		$e + j$		$e + j^{<<11}$		$e + j^{<<11}$	e
O_5	$f + w^{<<2}$		$f + j^{<<2}$	$f + j^{<<13}$	$f + e^{<<2}$	$f + j^{<<13}$	f
O_6	$g + w^{<<3}$		$g + j^{<<3}$	$g + j^{<<17}$	$g + e^{<<3}$	$g + j^{<<17}$	g
O_7			$h + j^{<<5}$		$h + e^{<<5}$	$h + j^{<<19}$	h
O_8					$i + e^{<<7}$	$i + j^{<<23}$	i
O_9							j
$v = c \oplus d \oplus e$, $w = h \oplus i \oplus j$							

6. Comparison of a cryptographic hash function with same output-length

6.1 The hash functions with 160 bits output-length

We compare SHA-1 to SHA-V(160 bits) and RIPEMD-160 to RIPEMD-V(160 bits) with 160 bits of output length. **Table 2** shows features of SHA-1, SHA-V(160 bits), RIPEMD-160 and RIPEMD-V(160 bits).

A Number of computation and speed

The step operation of SHA-1 consists of 4 additions, 1 circular shifts and a boolean function. The boolean function consists of 3 unit operations, and the step operations consist of 80 steps. That is,

$1(block) \times 80(steps) \times 10(step\ operations)$
$= 800(unit\ operations)$.

SHA-V (160-bit) consists of 4 additions, 1 circular shift and a boolean function. The Boolean functions consist of 3 unit operations of 2 lines and the step operations consist of 80 steps of 2 lines.

$1(block) \times 160(steps) \times 10(step\ operations)+$
$5(output\ message\ expansion)$
$= 1605(unit\ operations)$.

RIPEMD-160 consists of 2 circular shifts, 4 additions and a boolean function. Boolean function consists of 3 unit operations.

$2(blocks) \times 160(steps) \times 9(step\ operations)$
$= 2880(unit\ operations)$.

The step operations of RIPEMD-V(160 bits) are same with RIPEMD-160, but message block is one because the message size is 1024-bit. That is,

$1(block) \times 160(steps) \times 9(step\ operations)$
$= 1440(unit\ operations)$.

Therefore, when we compare with step operations, we can know that SHA-1 is faster than others.

6.2 The hash functions with 256-bit output-length

In this section, we compare SHA-256 to SHA-V(256 bits) and RIPEMD-256 to RIPEMD-V(256 bits) with 256 bits of output length. **Table 3** shows features of SHA-256, SHA-V(256 bits), RIPEMD-256 and RIPEMD-V(256 bits).

A Number of computation and speed

The step operation of SHA-256 consists of 7 additions and 2 summations, and the step operations consist of 64 steps. That is

$1(block) \times 64(steps) \times 15(step\ operations)$
$= 960(unit\ operations)$.

SHA-V (256 bits) consists of 4 additions, 1 circular shift and a boolean function. Boolean functions consist of 3 unit operations of 2 lines and the step operations consist of 80 steps of 2 lines.

$1(block) \times 160(steps) \times 10(step\ operations)+$
$16(output\ message\ expansion)$
$= 1616(unit\ operations)$.

RIPEMD-256 consists of 2 circular shifts, 4 additions and a boolean function. Boolean function consists of 3 unit operations.

$2(blocks) \times 128(steps) \times 9(step\ operations)$
$= 2304(unit\ operations)$.

The step operations of RIPEMD-V(160 bits) are same with RIPEMD-160, but message block is one because the message size is 1024 bits. That is,

$1(block) \times 160(steps) \times 9(stepoperations)$
$= 1440(unit\ operations)$.

When we compare with step operations, we can know that SHA-256 is faster than others.

7. Conclusions

We proposed two hash functions with variable output length. We can select output length by programs. Therefore, we can expect a lot of the practice. In the future, we must verify more security

Table 2 Features of SHA-1, SHA-V (160 bits), RIPEMD-160 and RIPEMD-V (160 bits).

	SHA-1	SHA-V(160 bits)	RIPEMD-160	RIPEMD-V (160 bits)
Endianess	Big	Big	Little	Little
Message block size	512	1024	512	1024
Digest size (bits)	160	160	160	160
Word size (bits)	32	64	32	64
The number of steps	80	2×80	2×80	2×80

Table 3 Features of SHA-256, SHA-V (256 bits), RIPEMD-256 and RIPEMD-V (256 bits).

	SHA-256	SHA-V (256 bits)	RIPEMD-256	RIPEMD-V (256 bits)
Endianess	Big	Big	Little	Little
Message block size	512	1024	512	1024
Digest size (bits)	256	256	256	256
Word size (bits)	32	64	32	64
The number of steps	64	2×80	2×64	2×80

test of SHA-V/RIPEMD-V and speed test in the hardware/software.

Acknowledgements

The first author has been supported by the Grant-in-Aid for Creative Scientific Research No.14GS0218 (Research on System LSI Design Methodology for Social Infrastructure, Head of Researchers : Prof. Hiroto Yasuura, System LSI Research center, Kyushu University) of the Ministry of Education, Science, Sports and Culture(MEXT) from 2002 to 2006. He is grateful for their support.

References

1) A.Bosselaers, R.Govaerts and J.Vandewalle *"SHA: A Design for Parallel Architecture?"* Advances in Cryptology-Eurocrypt'97, pp348-362, 1997.
2) A.J.Menzs, P.C.Van Oorshot and S.A.Vanstone *"Handbook of Applied Cryptography"* CRC Press, 1997.
3) Antoon Bosselaers *"The Hash Function RIPEMD 160"* http://www.esat.kuleuven.ac.be/ bosselae
4) B.den Boer and A.Bosselaers *"An attack on the last two rounds of MD4"* Advances in Cryptology-Crypto'91, LNCS576, Springer-Verlag, pp 194-203, 1992.
5) C.H. Lim, N.K.Park, E.J. Lee, P.J.Lee *"The proposal of the new hash function possible to select the output length"* preprint, 1997 (Korean).
6) H.Dobbertin *"RIPEMD with two-round compress function is not collision-free"* Journal of Cryptology, to appear; announced at rump session, Eurocrypt'95.
7) H.Dobbertin, A. Bosselaers, B. Preneel *"RIPEMD-160 : A strengened Version of RIPEMD"* Fast Software Encryption, LNCS 1039, Springer-Verlag, pp71-82, 1996.
8) H.Dobbertin *"Cryptanalysis of MD4"* Fast Software Encryption, LNCS 1039, Springer-Verlag, pp53-69, 1996.
9) H.Kuwakado, H. Tanaka *"New Algorithm for Finding Preimage in a Reduced Version of the MD4 Compression Function "* IEICE TRANS, Fundamentals, Vol.E83-A, No.1, Jan,2000.
10) I.B.Damagard *"A Design Principle for Hash Functions"* CRYPT'89, LNCS 435, pp416-427, 1990
11) J.Nakajima, M.Matsui *"Performance Analysis and Parallel Implementation of Dedicated Hash Functions"* EUROCRYPT 2002, LNCS 2332, pp165-180, 2002.
12) M.J.B.Robshaw *"On Recent Results for MD2, MD4, MD5 "* April, 1996.
13) National Security Research Institute *"Descriptions of SHA-256, SHA-384, and SHA-512"* NIST 2001. http://csrc.nist.gov/cryptval/shs.html
14) NIST *"Descriptions of SHA256, 384 and SHA-512"* http://csrc.nist.gov/cryptval/shs.html, 2001
15) NIST *"Secure Hash Standard"* FIPS PUB180-1, May, 1993.
16) N.K.Park, J.H.Hwang, P.J.Lee *"HAS-V : A New Hash Function with Variable Output Length"* SAC2000, LNCS2012, Springer-Verlag, pp.202-216, 2001.
17) P.Sarkar, P,J, Schellenberg *"A Parallelizable Design Principle for Cryptography Hash Functions"* Indocrypt 2001, LNCS 2247, pp40-49, 2001.
18) R.Rivest *"The MD4 Message Digest Algorithm"* Advances in Cryptology-Crypto'90, LNCS 537, Springer-Verlag, pp303-311, 1991.
19) Telecommunication Technology Association *"Hash Function Standard - Part 2 : Hash Function Algorithm Standard (HAS-160) "* TTA, KO-12.0011/R1, Dec 2000.
20) Y.Zheng, J.Pieprzyk and J.Sebbery *"HAVAL-A - One-Way Hashing Algorithm with Variable Length of Output"* Advances in Cryptography-AUSCRYPT'92,LNCS718, Springer-Verlag, pp83-104, 1993.

Appendix

A. Pseudo-code of SHA-V

Initial values

- Left line

$h_0^{(i)} = 67452301$, $h_1^{(i)} = efcdab89$, $h_2^{(i)} = 98badcfe$,
$h_3^{(i)} = 10325476$, $h_4^{(i)} = c3d2e1f0$

- Right line

$h_5^{(i)} = 8796a5b4$, $h_6^{(i)} = 4b5a6978$, $h_7^{(i)} = 0f1e2d3c$,
$h_8^{(i)} = a0b1c2d3$, $h_9^{(i)} = 68794e5f$

Constants

- Left line

$K[0] = 5a827999$, $K[1] = 6ed9eba1$,
$K[2] = 8f1bbcdc$, $K[3] = a953fd4e$

- Right line

$K'[0] = 7a6d76e9$, $K'[1] = 6d703ef3$,
$K'[2] = 5c4dd124$, $K'[3] = 50a28be6$

Boolean functions

- Left line

$F_t(x,y,z) = (x \wedge y) \vee (\neg x \wedge z)$ $(0 \le t \le 19)$
$F_t(x,y,z) = x \oplus y \oplus z$ $(20 \le t \le 39)$
$F_t(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$ $(40 \le t \le 59)$
$F_t(x,y,z) = x \oplus y \oplus z$ $(60 \le t \le 79)$

- Right line

$G_t(x,y,z) = x \oplus y \oplus z$ $(0 \le t \le 19)$
$G_t(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$ $(20 \le t \le 39)$
$G_t(x,y,z) = x \oplus y \oplus z$ $(40 \le t \le 59)$
$G_t(x,y,z) = (x \wedge y) \vee (\neg x \wedge z)$ $(60 \le t \le 79)$

for $i = 0$ to $t-1$ {
$A = h_0^{(i)}$, $B = h_1^{(i)}$, $C = h_2^{(i)}$, $D = h_3^{(i)}$, $E = h_4^{(i)}$,
$F = h_5^{(i)}$, $G = h_6^{(i)}$, $H = h_7^{(i)}$, $I = h_8^{(i)}$, $J = h_9^{(i)}$ }
for $t = 16$ to 79 {
$W_t = ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$; Left line
$W'_t = ROTL^1(W'_{t-3} \oplus W'_{t-8} \oplus W'_{t-14} \oplus W'_{t-16})$; Right line
}
for $t = 0$ to 79 {
$\mathrm{T} := \mathrm{ROTL}^5(a) + F_t(b,c,d) + e + K_t + W_t$
$e := d$, $d := c$, $c := ROTL^{30}(b)$, $b := a$,$a := T$
$T' := ROTL^5(f) + G_t(g,h,i) + j + K'_t + W'_t$
$j := i$, $i := h$, $h := ROTL^{30}(g)$, $g := f$, $f := T'$ }
$h_0^{(i)} += A$, $h_1^{(i)} += B$, $h_2^{(i)} += C$, $h_3^{(i)} += D$, $h_4^{(i)} += E$,
$h_5^{(i)} += F$, $h_6^{(i)} += G$, $h_7^{(i)} += H$, $h_8^{(i)} += I$, $h_9^{(i)} += J$ }

B. Pseudo-code of RIPEMD-V

Initial values

- Left line

$h_0^{(i)} = 67452301$, $h_1^{(i)} = efcdab89$, $h_2^{(i)} = 98badcfe$,
$h_3^{(i)} = 10325476$, $h_4^{(i)} = c3d2e1f0$

- Right line

$h_5^{(i)} = 76542210$, $h_6^{(i)} = fedcba98$, $h_7^{(i)} = 89abcdef$,
$h_8^{(i)} = 01234567$, $h_9^{(i)} = 3c2d1e0f$

Constants

- Left line

$K[0] = 00000000$, $K[1] = 5a827999$, $K[2] = 6ed9eba1$,
$K[3] = 8f1bbcdc$, $K[4] = a953fd4e$

- Right line

$K'[0] = 50a288e6$, $K'[1] = 5c4dd124$, $K'[2] = 6d703ef3$,
$K'[3] = 7a6d76e9$, $K'[4] = 00000000$

Boolean functions

- Left line

$f_j(x,y,z) = x \oplus y \oplus z$ $(0 \le j \le 15)$
$f_j(x,y,z) = (x \wedge y) \vee (x \wedge z)$ $(16 \le t \le 31)$
$f_j(x,y,z) = (x \wedge y) \oplus z$ $(32 \le t \le 47)$
$f_j(x,y,z) = (x \wedge z) \vee (y \wedge z)$ $(48 \le t \le 63)$
$f_j(x,y,z) = x \oplus (y \vee z)$ $(64 \le t \le 79)$

- Right line

$g_j(x,y,z) = x \oplus (y \vee z)$ $(0 \le t \le 15)$
$g_j(x,y,z) = (x \wedge z) \vee (y \wedge z)$ $(16 \le t \le 31)$
$g_j(x,y,z) = (x \wedge y) \oplus z$ $(32 \le t \le 47)$
$g_j(x,y,z) = (x \wedge y) \vee (x \wedge z)$ $(48 \le t \le 63)$
$g_j(x,y,z) = x \oplus y \oplus z (64 \le j \le 79)$

Amount for rotate left (rol)

$r(j) = j (0 \le t \le 15)$
$r(16,...,31) = 7,4,13,1,10,6,15,3,12,0,9,5,2,14,11,8$
$r(32,..,47) = 3,10,14,4,9,15,8,1,2,7,0,6,13,11,5,12$
$r(48,...,63) = 1,9,11,10,0,8,12,4,13,3,7,15,14,5,6,2$
$r(64,...,79) = 4,0,5,9,7,12,2,10,14,1,3,8,11,6,15,13$
$r'(0,...,15) = 5,14,7,0,9,2,11,4,13,6,15,8,1,10,3,12$
$r'(16,...,31) = 6,11,3,7,0,13,5,10,14,15,8,12,4,9,1,2$
$r'(32,...,47) = 15,5,1,3,7,14,6,9,11,8,12,2,10,0,4,13$
$r'(48,...,63) = 8,6,4,1,3,11,15,0,5,12,2,13,9,7,10,14$
$r'(64,...,79) = 12,15,10,4,1,5,8,7,6,2,13,14,0,3,9,11$

Selection of message word

$s(0,...,15) = 11,14,15,12,5,8,7,9,11,13,14,15,6,7,9,8$
$s(16,...,31) = 7,6,8,13,11,9,7,15,7,12,15,9,11,7,13,12$
$s(32,..,47) = 11,13,6,7,14,9,13,15,14,8,13,6,5,12,7,5$
$s(48,...,63) = 11,12,14,15,14,15,9,8,9,14,5,6,8,6,5,12$
$s(64,...,79) = 9,15,5,11,6,8,13,12,5,12,13,14,11,8,5,6$
$s'(0,...,15) = 8,9,9,11,13,15,15,5,7,7,8,11,14,14,12,6$
$s'(16,...,31) = 9,13,15,7,12,8,9,11,7,7,12,7,6,15,13,11$
$s'(32,...,47) = 9,7,15,11,8,6,6,14,12,13,5,14,13,13,7,5$
$s'(48,...,63) = 15,5,8,11,14,14,6,14,6,9,12,9,12,5,15,8$
$s'(64,...,79) = 8,5,12,9,12,5,14,6,8,13,6,5,15,13,11,11$

for $i = 0,...,t-1$ {
$A = h_0^{(i)}$, $B = h_1^{(i)}$,$C = h_2^{(i)}$, $D = h_3^{(i)}$, $E = h_4^{(i)}$,
$F = h_5^{(i)}$, $G = h_6^{(i)}$,$H = h_7^{(i)}$, $I = h_8^{(i)}$, $J = h_9^{(i)}$, }
for $t = 0$ to 79 {
$\mathrm{T} := \mathrm{rol}_{s(t)}(A + f_t(t,B,C,D) + X_i[r(t)] + K(t)) + E$
$A := E$, $E := D$, $D := rol_{10}(C)$, $C := B$, $B := T$
$T' = rol_{s'(t)}(F + g_t(79-t,G,H,I) + X_i[r'(t)] + K'(t) + J$
$J := I$, $I := H$, $H := rol_{10}(G)$, $G := F$, $F := T'$ }
In case of 320 bits output :
$h_0^{(i)} += A$, $h_1^{(i)} += B$, $h_2^{(i)} += C$, $h_3^{(i)} += D$,
$h_4^{(i)} += E$,
$h_5^{(i)} += F$, $h_6^{(i)} += G$, $h_7^{(i)} += H$, $h_8^{(i)} += I$,
$h_9^{(i)} += J$ }