International Journal of Security and Its Application
Vol.11, No.1 (2017), pp.45-58
http://dx.doi.org/10.14257/ijsia.2017.11.1.05

128 Bit Hash of Variable Length in Short Message Service Security

Robbi Rahim

Department of Informatics Engineering, Medan Institute of Technology, JL. Gedung Arca No.52, Medan, Sumatera Utara, Indonesia
usurobbi85@zoho.com

Abstract

Hashing Variable Length is a form of cryptography that can be used to compress and secure messaging with an output of 32 characters, implementation of algorithms HAVAL in the delivery of SMS will speed up and ease the cost of SMS due to any long SMS messages will turn to 32 characters, factor in the cost and safety is extremely importance in communication, and HAVAL algorithm could be a separate solution for the users.

Keywords*: Algorithm Cryptography, Hashing, HAVAL Algorithm, Secure Message, SMS Security.*

1. Introduction

Cryptography is one technique that can be used to secure information specially the message [1] [2] [3] [4], One-way hash algorithm is an algorithm that compresses a message to any length, and generate output a value that is always the same length [5] [6]. This output value is used as a digital signature (digital signature) to verify (authentication) of the message [7]. The one-way hash algorithm used to secure and check the authenticity of a message [5]. One of the most widely used algorithms in the security network and the Internet is the Message Digest version 5 (MD5) [6] [7] [8]. Besides MD5 algorithm SHA (Secure Hash Algorithm) are also commonly used. This algorithm always produces the output value of the digest is always the same length [5]. MD5 always produces output with a length of 128-bit value and SHA produces output with 160-bit length value [6].

HAVAL is a one-way hash algorithm invented by Yuliang Zheng, Josef Pieprzky and Jennifer Seberry [9] [10]. The primary objective of the algorithm is to produce an output value of the digest length that can vary. The output length value HAVAL algorithms can be 128, 160, 192, 224 and 256 bits [9]. In this HAVAL algorithm, a message can process as much as 3, 4 or 5 times. The combination of large output (digest) and the amount of process provides 15 version HAVAL. According to the experiments of its creator, HAVAL faster, 60% compared with the MD5 in the process [6][9] [10].

This study is expected to be a solution for sending secure messages and compressed, thereby reducing the cost of sending SMS is quite expensive with cost calculations per 160 characters, and using algorithms all the characters will be compressed into a length of 32 characters so that the message sent is expected to be much cheaper than in terms of cost and also faster message is sent because only 32 characters long compared to the original message may be up to 500 characters, of course, the process of compression and hashing algorithms HAVAL not only limited to SMS messages but also can be used for almost all text messaging.

ISSN: 1738-9976 IJSIA

2. Theory

Cryptography is a field of science that studies on how to conceal an important information [1] [2] [12] [13]. The other side of cryptography is cryptanalysis which is the study of how to solve a cryptography mechanism [12] [13].

Cryptography learns about the mathematical techniques related to aspects of the information security like confidentiality, data integrity, authentication of the sender/recipient data, and authentication data [1] [13]. With the development of the field of cryptography, the division between what is and what is not cryptography have become blurred. Today, cryptography can consider as a blend of engineering studies and applications that depend on the existence of the problem [13]. The most common method for secret key cryptography is a block cipher, stream ciphers, and message authentication codes (MAC).

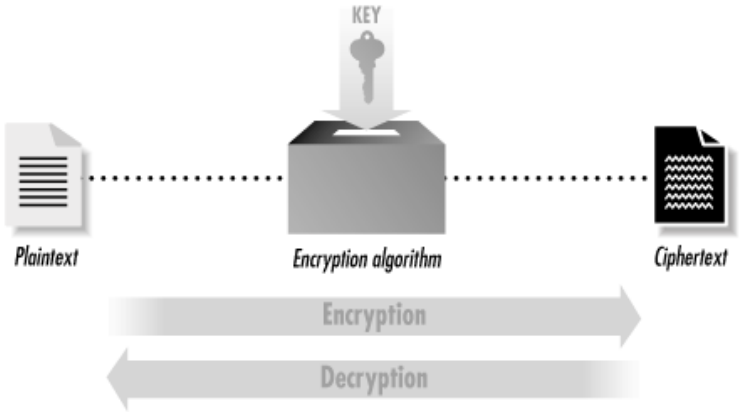

Figure 1. Secret Key Cryptography

The hash value generated by H function with form h = h(M), where M is the message to the free length, and H (M) or h is a hash value of a fixed length. This hash value will be added at the beginning of the message and then transmitted simultaneously [7][12]. Recipients of the message that will authenticate the message with the computed hash value and then compare it to the hash value contained in the initial message [12].

To be able to act as a hash function message authentication, a hash function H must qualify - requisites below [14]:

a. H can apply to a block of data of any size.
b. H produces output with a fixed length.
c. It is relatively easy to compute H (x) for any x.
d. By knowing the hash value y, it can be computational to find x that satisfies H (x) = y.
e. Let it x; it can be computational to find $y \neq x$ with H (y) = H (x).
f. There can be computational to locate the pair (x, y) satisfying H (x) = H (y).

The first three conditions to function as a condition for the practical application of the hash function to authenticate the message [11], the fourth requirement is a requirement of "one-way" (one-way). Easy to generate the code with the hash value is given a message but cannot to produce encrypted message with the hash value.

The fifth requirement ensures that the original message will not granted can be found some other messages with the same hash value with the hash value of the original message. This provision is to prevent fraud if used encrypted hash code. For this case, the opposing party can read the message and also generates its hash code. However, because the opposition does not have the key to decrypt the hash code [6], then the opposition will not be able to change the message without being noticed. If this requirement is not available then, the opposition can do it - in the following way:

a. Intercept a message with the hash code.
b. Generate an unencrypted hash code of the message.
c. Create a new message that others with the same hash value code and then send it back.

If all five requirements above met, then the hash function called a hash function is weak. If the sixth requirement also met, then the hash function is considered a strong hash function [5] [14]. Also, to authenticate the message, the hash function can also be applied to generate keys based on a passphrase. The way it works is as follows:

a. By using a hash function hash value produced from the passphrase.
b. The hash value of the passphrase used as a key for encryption/decryption.

Values represent the hash function message is shorter than the document from which the value measured; this value is often called a message digest. The message digests considered as a "digital fingerprint" of a longer document. Examples of well-known hash functions are MD2, MD5, SHA and HAVAL [12] [14].

The role of the cryptography hash function is regarding checking the condition of the integrity of the message and the digital signature. A digest can be made public without showing the contents of the document from which the summary derived. This particularly important in the digital time stamping which by using a hash function, one can obtain a record with a timestamp (document time-stamped) without showing the contents of the documents to the time stamping service provider [5] [6] [12] [14].

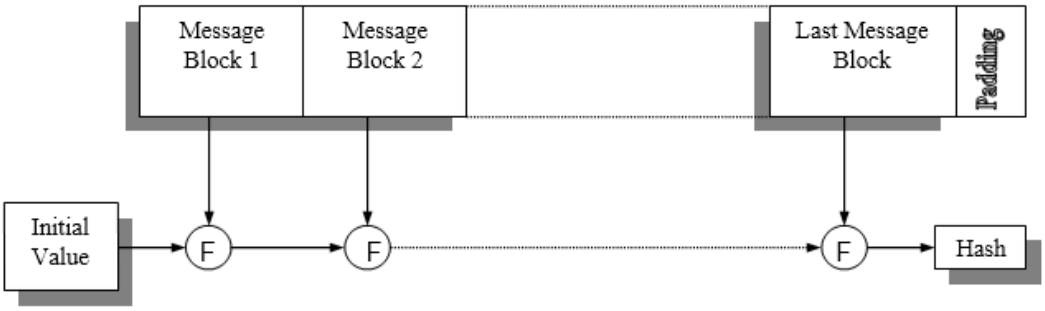

Figure 2. The Structure of an Iterative Hash Function Damgard/Merkle; F = the Compression Function

This article discusses the algorithms HAVAL; HAVAL algorithm may work as follows:

a. Input in the form of a message (message) with a length of 128 characters = 1024 bits = 256 hexadecimal digits.
b. Value constants used to be set as the initial value of each variable x (the value of this constant is fixed and in accordance with the method HAVAL 128)
   1) K_0 = 243F6A88
   2) K_1 = 85A308D3
   3) K_2 = 13198A2E
   4) K_3 = 03707344
   5) K_4 = A4093822
   6) K_5 = 299F31D0
   7) K_6 = 082EFA98
   8) K_7 = EC4E6C89

Function used:

a. $F(A_6, A_5, A_4, A_3, A_2, A_1, A_0) = ((A_1 \wedge A_4) \oplus (A_2 \wedge A_5) \oplus (A_3 \wedge A_6) \oplus (A_0 \wedge A_1) \oplus A_0)$
b. $G(A_6, A_5, A_4, A_3, A_2, A_1, A_0) = ((A_1 \wedge A_2 \wedge A_3) \oplus (A_2 \wedge A_4 \wedge A_5) \oplus (A_1 \wedge A_2) \oplus (A_1 \wedge A_4) \oplus (A_2 \wedge A_6) \oplus (A_3 \wedge A_5) \oplus (A_4 \wedge A_5) \oplus (A_0 \wedge A_2) \oplus A_0)$

c. $H(A_6, A_5, A_4, A_3, A_2, A_1, A_0) = ((A_1 \wedge A_2 \wedge A_3) \oplus (A_1 \wedge A_4) \oplus (A_2 \wedge A_5) \oplus (A_3 \wedge A_6) \oplus (A_0 \wedge A_3) \oplus A_0)$
d. $F\_phi(A_6, A_5, A_4, A_3, A_2, A_1, A_0) = F(A_1, A_0, A_3, A_5, A_6, A_2, A_4)$
e. $G\_phi(A_6, A_5, A_4, A_3, A_2, A_1, A_0) = G(A_4, A_2, A_1, A_0, A_5, A_3, A_6)$
f. $H\_phi(A_6, A_5, A_4, A_3, A_2, A_1, A_0) = H(A_6, A_1, A_2, A_3, A_4, A_5, A_0)$
g. $FF(A_7, A_6, A_5, A_4, A_3, A_2, A_1, A_0, w)$
   $temp = F\_phi(A_6, A_5, A_4, A_3, A_2, A_1, A_0)$
   $A_7 = (temp >>> 7) + (A_7 >>> 11) + w$
h. $GG(A_7, A_6, A_5, A_4, A_3, A_2, A_1, A_0, w, c)$
   $temp = G\_phi(A_6, A_5, A_4, A_3, A_2, A_1, A_0)$
   $A_7 = (temp >>> 7) + (A_7 >>> 11) + w + c$
i. $HH(A_7, A_6, A_5, A_4, A_3, A_2, A_1, A_0, w, c)$
   $temp = H\_phi(A_6, A_5, A_4, A_3, A_2, A_1, A_0)$
   $A_7 = (temp >>> 7) + (A_7 >>> 11) + w + c$

The next is the process of permutations for every function that exist, for FF, GG and HH as below:

a. Round 1
   $FF(x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0, w_0)$
   $FF(x_6, x_5, x_4, x_3, x_2, x_1, x_0, x_7, w_1)$
   $FF(x_5, x_4, x_3, x_2, x_1, x_0, x_7, x_6, w_2)$
   $FF(x_4, x_3, x_2, x_1, x_0, x_7, x_6, x_5, w_3)$
   the permutation 1, each loop + 1 then the value x = x-1 and w = w + 1, the process is conducted up to w = w31, the results are as follows:
   $FF(x_0, x_7, x_6, x_5, x_4, x_3, x_2, x_1, w_{31})$
b. Round 2
   by function G and G_phi already described above, then the second round for the G-value
   $GG(x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0, w_5, 452821E6)$
   $GG(x_6, x_5, x_4, x_3, x_2, x_1, x_0, x_7, w_{14}, 38D01377)$
   $GG(x_5, x_4, x_3, x_2, x_1, x_0, x_7, x_6, w_{26}, BE5466CF)$
   $GG(x_4, x_3, x_2, x_1, x_0, x_7, x_6, x_5, w_{18}, 34E90C6C)$
   This process will do until end of looping and the last result will be
   $GG(x_0, x_7, x_6, x_5, x_4, x_3, x_2, x_1, w_{27}, C25A59B5)$
c. Round 3
   by function H and H_phi already described above, then the second round for the H-value
   $HH(x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0, w_{19}, 9C30D539)$
   $HH(x_6, x_5, x_4, x_3, x_2, x_1, x_0, x_7, w_9, 2AF26013)$
   $HH(x_5, x_4, x_3, x_2, x_1, x_0, x_7, x_6, w_4, C5D1B023)$
   $HH(x_4, x_3, x_2, x_1, x_0, x_7, x_6, x_5, w_{20}, 286085F0)$
   This process will do until end of looping and the last result will be
   $HH(x_0, x_7, x_6, x_5, x_4, x_3, x_2, x_1, w_2, 6C24CF5C)$

Output processes by adding each value of the variable x from the process with the corresponding constants

$x_0 = x_0 + K_0$
$x_1 = x_1 + K_1$
$x_2 = x_2 + K_2$
$x_3 = x_3 + K_3$
$x_4 = x_4 + K_4$
$x_5 = x_5 + K_5$
$x_6 = x_6 + K_6$
$x_7 = x_7 + K_7$

Tailor output process is done by using the following functions:
temp = ($x_7 \wedge$ 000000FF) $\vee$ ($x_6 \wedge$ FF000000) $\vee$ ($x_5 \wedge$ 00FF0000) $\vee$ ($x_4 \wedge$ 0000FF00)
$x_0 = x_0$ + (temp >>> 8)
temp = ($x_7 \wedge$ 0000FF00) $\vee$ ($x_6 \wedge$ 000000FF) $\vee$ ($x_5 \wedge$ FF000000) $\vee$ ($x_4 \wedge$ 00FF0000)
$x_1 = x_1$ + (temp >>> 16)
temp = ($x_7 \wedge$ 00FF0000) $\vee$ ($x_6 \wedge$ 0000FF00) $\vee$ ($x_5 \wedge$ 000000FF) $\vee$ ($x_4 \wedge$ FF000000)
$x_2 = x_2$ + (temp >>> 24)
temp = ($x_7 \wedge$ FF000000) $\vee$ ($x_6 \wedge$ 00FF0000) $\vee$ ($x_5 \wedge$ 0000FF00) $\vee$ ($x_4 \wedge$ 000000FF)
$x_3 = x_3$ + temp
Output = $x_0 \| x_1 \| x_2 \| x_3$ ($\|$ = *concatenation*)

as the information of the above functions are as follows:
a. xi (i = 0 .. 7); length = 32 bits = 8 digit hexadecimal
b. wi (i = 0 .. 31); a sub-word of the message;
c. length = 4 characters = 32 bit = 8 digit hexadecimal

3. Result And Discussion

For example, testing of algorithms HAVAL uses plaintext as follows:
“This is only message for testing how HAVAL algorithm work and show how the result were getting after hashing, this were made iam”

The next step to change the SMS message into hexadecimal, the results(w) are as follows:
'This' -> w(0) = 54686973
' is ' -> w(1) = 20697320
'only' -> w(2) = 6F6E6C79
' mes' -> w(3) = 206D6573
'sage' -> w(4) = 73616765
' for' -> w(5) = 20666F72
' tes' -> w(6) = 20746573
'ting' -> w(7) = 74696E67
' how' -> w(8) = 20686F77
' HAV' -> w(9) = 20484156
'AL a' -> w(10) = 414C2061
'lgor' -> w(11) = 6C676F72
'ithm' -> w(12) = 6974686D
' wor' -> w(13) = 20776F72
'k an' -> w(14) = 6B20616E
'd sh' -> w(15) = 64207368
'ow h' -> w(16) = 6F772068
'ow t' -> w(17) = 6F772074
'he r' -> w(18) = 68652072
'esul' -> w(19) = 6573756C
't we' -> w(20) = 74207765
're g' -> w(21) = 72652067
'etti' -> w(22) = 65747469
'ng a' -> w(23) = 6E672061
'fter' -> w(24) = 66746572
' has' -> w(25) = 20686173
'hing' -> w(26) = 68696E67
', th' -> w(27) = 2C207468
'is w' -> w(28) = 69732077
'ere ' -> w(29) = 65726520

'made' -> w(30) = 6D616465
' iam' -> w(31) = 2069616D
process performed to convert the message into hexadecimal, can be seen in the following figure:

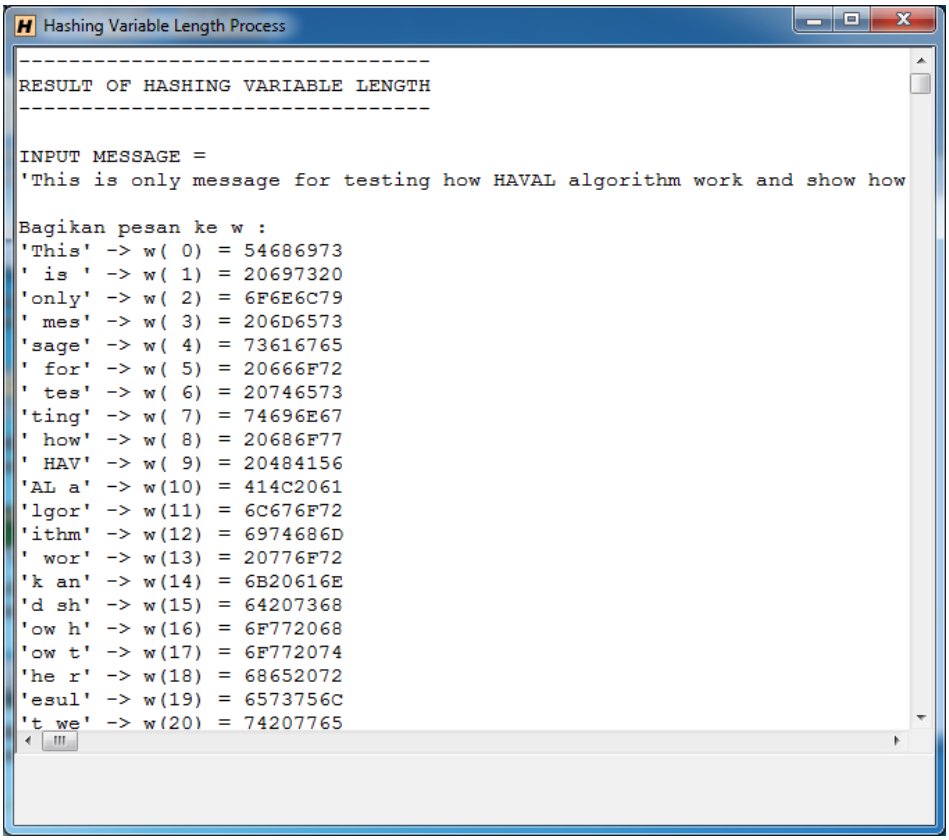

Figure 3. Process Hexadecimal Message

Next process is to perform early initiation value (k):
K0 = X0 = 243F6A88
K1 = X1 = 85A308D3
K2 = X2 = 13198A2E
K3 = X3 = 03707344
K4 = X4 = A4093822
K5 = X5 = 299F31D0
K6 = X6 = 082EFA98
K7 = X7 = EC4E6C89

Next is the result for round 1 processing:

FF(X7, X6, X5, X4, X3, X2, X1, X0, W0)
FF(EC4E6C89,082EFA98,299F31D0,A4093822,03707344,13198A2E,85A308D3,243F6A88,54686973)

1. Temp = F_Phi(082EFA98,299F31D0,A4093822,03707344,13198A2E,85A308D3,243F6A88)

   Temp = F(85A308D3,243F6A88,03707344,299F31D0,082EFA98,13198A2E,A4093822)

Temp = (13198A2E AND 03707344) XOR (082EFA98 AND 243F6A88) XOR (299F31D0 AND 85A308D3) XOR (A4093822 AND 13198A2E) XOR A4093822

Temp = A6BD585C

2. A7 = (Temp >>> 7) + (A7 >>> 11) + w
   A7 = (A6BD585C >>> 7) + (EC4E6C89 >>> 11) + 54686973
   A7 = 9EF36DF0

FF(X6, X5, X4, X3, X2, X1, X0, X7, W1)
FF(082EFA98,299F31D0,A4093822,03707344,13198A2E,85A308D3,243F6A88,9EF36DF0,20697320)

1. Temp = F_Phi(299F31D0,A4093822,03707344,13198A2E,85A308D3,243F6A88,9EF36DF0)

   Temp = F(243F6A88,9EF36DF0,13198A2E,A4093822,299F31D0,85A308D3,03707344)

   Temp = (85A308D3 AND 13198A2E) XOR (299F31D0 AND 9EF36DF0) XOR (A4093822 AND 243F6A88) XOR (03707344 AND 85A308D3) XOR 03707344

   Temp = 2FCB72D6

2. A7 = (Temp >>> 7) + (A7 >>> 11) + w
   A7 = (2FCB72D6 >>> 7) + (082EFA98 >>> 11) + 20697320
   A7 = 1FCA0FE4

This FF Process will do until w31 and the value end of the first round are as follows:
FF(X0, X7, X6, X5, X4, X3, X2, X1, W31)
FF(4EEB9EC0,D40359D4,EC5250E3,46CEDB9A,8AB988F2,99E73EF2,636806D3,D4D6AD26,2069616D)

1. Temp =
2. F_Phi(D40359D4,EC5250E3,46CEDB9A,8AB988F2,99E73EF2,636806D3,D4D6AD26)

   Temp = F(636806D3,D4D6AD26,8AB988F2,EC5250E3,D40359D4,99E73EF2,46CEDB9A)

   Temp = (99E73EF2 AND 8AB988F2) XOR (D40359D4 AND D4D6AD26) XOR (EC5250E3 AND 636806D3) XOR (46CEDB9A AND 99E73EF2) XOR 46CEDB9A

   Temp = 7AEBC03D

3. A7 = (Temp >>> 7) + (A7 >>> 11) + w
   A7 = (7AEBC03D >>> 7) + (4EEB9EC0 >>> 11) + 2069616D
   A7 = 73691660

process for round 2 to G are not much different from the process F, here are the results
GG(X7, X6, X5, X4, X3, X2, X1, X0, W5, 452821E6)
GG(D40359D4,EC5250E3,46CEDB9A,8AB988F2,99E73EF2,636806D3,D4D6AD26,73691660,20666F72,452821E6)

1. Temp = G_Phi(EC5250E3,46CEDB9A,8AB988F2,99E73EF2,636806D3,D4D6AD26,73691660)

   Temp = G(8AB988F2,636806D3,D4D6AD26,73691660,46CEDB9A,99E73EF2,EC5250E3)

   Temp = (99E73EF2 AND 46CEDB9A AND 73691660) XOR (46CEDB9A AND D4D6AD26 AND 636806D3) XOR (99E73EF2 AND 46CEDB9A) XOR (99E73EF2 AND D4D6AD26) XOR (46CEDB9A AND 8AB988F2) XOR (73691660 AND 636806D3) XOR (D4D6AD26 AND 636806D3) XOR (EC5250E3 AND 46CEDB9A) XOR EC5250E3

   Temp = 59B0AE03
2. A7 = (Temp >>> 7) + (A7 >>> 11) + w + c
   A7 = (59B0AE03 >>> 7) + (D40359D4 >>> 11) + 20666F72 + 452821E6
   A7 = A6DC731F

GG(X6, X5, X4, X3, X2, X1, X0, X7, W14, 38D01377)
GG(EC5250E3,46CEDB9A,8AB988F2,99E73EF2,636806D3,D4D6AD26,73691660,A6DC731F,6B20616E,38D01377)

1. Temp = G_Phi(46CEDB9A,8AB988F2,99E73EF2,636806D3,D4D6AD26,73691660,A6DC731F)

   Temp = G(99E73EF2,D4D6AD26,73691660,A6DC731F,8AB988F2,636806D3,46CEDB9A)

   Temp = (636806D3 AND 8AB988F2 AND A6DC731F) XOR (8AB988F2 AND 73691660 AND D4D6AD26) XOR (636806D3 AND 8AB988F2) XOR (636806D3 AND 73691660) XOR (8AB988F2 AND 99E73EF2) XOR (A6DC731F AND D4D6AD26) XOR (73691660 AND D4D6AD26) XOR (46CEDB9A AND 8AB988F2) XOR 46CEDB9A

   Temp = 7B3B787C
2. A7 = (Temp >>> 7) + (A7 >>> 11) + w + c
   A7 = (7B3B787C >>> 7) + (EC5250E3 >>> 11) + 6B20616E + 38D01377
   A7 = B964761F

The final value of the process is as follows GG:

GG(X0, X7, X6, X5, X4, X3, X2, X1, W27, C25A59B5)
GG(E201D6A3,38BF1E66,28318DC4,07688657,918C6349,70CA9575,29A73219,743C4741,2C207468,C25A59B5)

1. Temp = G_Phi(38BF1E66,28318DC4,07688657,918C6349,70CA9575,29A73219,743C4741)

   Temp = G(07688657,70CA9575,29A73219,743C4741,28318DC4,918C6349,38BF1E66)

   Temp = (918C6349 AND 28318DC4 AND 743C4741) XOR (28318DC4 AND 29A73219 AND 70CA9575) XOR (918C6349 AND 28318DC4) XOR (918C6349 AND 29A73219) XOR (28318DC4 AND 07688657) XOR (743C4741 AND

70CA9575) XOR (29A73219 AND 70CA9575) XOR (38BF1E66 AND 28318DC4) XOR 38BF1E66

Temp = 61A0A13F

2. A7 = (Temp >>> 7) + (A7 >>> 11) + w + c
   A7 = (61A0A13F >>> 7) + (E201D6A3 >>> 11) + 2C207468 + C25A59B5
   A7 = 41BA4F99

The next is for the calculation process HH:

HH(X7, X6, X5, X4, X3, X2, X1, X0, W19, 9C30D539)

HH(38BF1E66,28318DC4,07688657,918C6349,70CA9575,29A73219,743C4741,41BA4F99,6573756C,9C30D539)

1. Temp = H_Phi(28318DC4,07688657,918C6349,70CA9575,29A73219,743C4741,41BA4F99)

   Temp = H(28318DC4,743C4741,29A73219,70CA9575,918C6349,07688657,41BA4F99)

   Temp = (07688657 AND 918C6349 AND 70CA9575) XOR (07688657 AND 29A73219) XOR (918C6349 AND 743C4741) XOR (70CA9575 AND 28318DC4) XOR (41BA4F99 AND 70CA9575) XOR 41BA4F99

   Temp = 30148EDD

2. A7 = (Temp >>> 7) + (A7 >>> 11) + w + c
   A7 = (30148EDD >>> 7) + (38BF1E66 >>> 11) + 6573756C + 9C30D539
   A7 = 88CB8BA5

HH(X6, X5, X4, X3, X2, X1, X0, X7, W9, 2AF26013)

HH(28318DC4,07688657,918C6349,70CA9575,29A73219,743C4741,41BA4F99,88CB8BA5,20484156,2AF26013)

1. Temp = H_Phi(07688657,918C6349,70CA9575,29A73219,743C4741,41BA4F99,88CB8BA5)

   Temp = H(07688657,41BA4F99,743C4741,29A73219,70CA9575,918C6349,88CB8BA5)

   Temp = (918C6349 AND 70CA9575 AND 29A73219) XOR (918C6349 AND 743C4741) XOR (70CA9575 AND 41BA4F99) XOR (29A73219 AND 07688657) XOR (88CB8BA5 AND 29A73219) XOR 88CB8BA5

   Temp = D16ECDE4

2. A7 = (Temp >>> 7) + (A7 >>> 11) + w + c
   A7 = (D16ECDE4 >>> 7) + (28318DC4 >>> 11) + 20484156 + 2AF26013
   A7 = CD628535

The final value of the process is as follows GG:

HH(X0, X7, X6, X5, X4, X3, X2, X1, W2, 6C24CF5C)

HH(57EC4146,78E0D445,BB75C79E,986F7457,B4801471,FD1269E6,22559C16,B01C5D89,6F6E6C79,6C24CF5C)

1. Temp = H_Phi(78E0D445,BB75C79E,986F7457,B4801471,FD1269E6,22559C16,B01C5D89)

   Temp = H(78E0D445,22559C16,FD1269E6,B4801471,986F7457,BB75C79E,B01C5D89)

   Temp = (BB75C79E AND 986F7457 AND B4801471) XOR (BB75C79E AND FD1269E6) XOR (986F7457 AND 22559C16) XOR (B4801471 AND 78E0D445) XOR (B01C5D89 AND B4801471) XOR B01C5D89

   Temp = 19C90C49
2. A7 = (Temp >>> 7) + (A7 >>> 11) + w + c
   A7 = (19C90C49 >>> 7) + (57EC4146 >>> 11) + 6F6E6C79 + 6C24CF5C
   A7 = 9691CB75

The next process is the process of summing each value there is to get the output value, the value will be processed back to tailor production, the following are the results:

1) X0 = X0 + K0
   = 9691CB75 + 243F6A88
   = BAD135FD

2) X1 = X1 + K1
   = B01C5D89 + 85A308D3
   = 35BF665C

3) X2 = X2 + K2
   = 22559C16 + 13198A2E
   = 356F2644

4) X3 = X3 + K3
   = FD1269E6 + 03707344
   = 0082DD2A

5) X4 = X4 + K4
   = B4801471 + A4093822
   = 58894C93

6) X5 = X5 + K5
   = 986F7457 + 299F31D0
   = C20EA627

7) X6 = X6 + K6
   = BB75C79E + 082EFA98
   = C3A4C236

8) X7 = X7 + K7
   = 78E0D445 + EC4E6C89
   = 652F40CE

The FF, GG, HH already explained the authors visualized in the form of a simulation as follows:

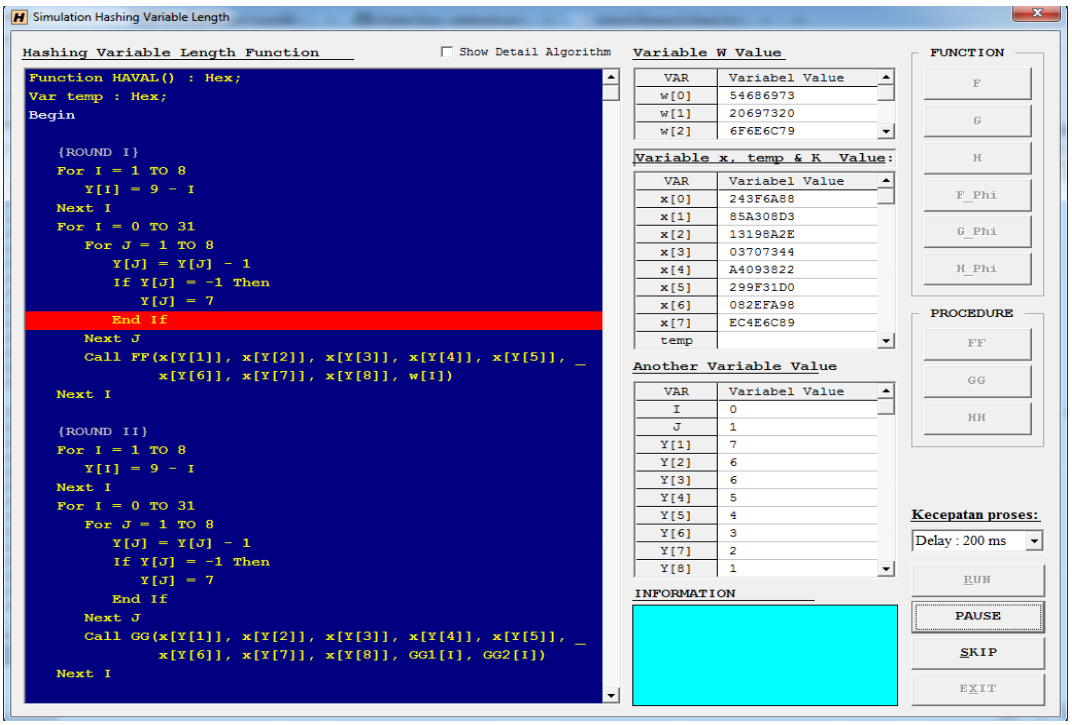

Figure 4. Simulation Hashing Variable Length

For tailor output shown below:

temp = (X7 AND 000000FF) OR (X6 AND FF000000) OR (X5 AND 00FF0000) OR (X4 AND 0000FF00)
temp = (652F40CE AND 000000FF) OR (C3A4C236 AND FF000000) OR (C20EA627 AND 00FF0000) OR (58894C93 AND 0000FF00)
temp = C30E4CCE

X0 = X0 + (temp >>> 8)
= BAD135FD + (C30E4CCE >>> 8)
= 89944449

temp = (X7 AND 0000FF00) OR (X6 AND 000000FF) OR (X5 AND FF000000) OR (X4 AND 00FF0000)
temp = (652F40CE AND 0000FF00) OR (C3A4C236 AND 000000FF) OR (C20EA627 AND FF000000) OR (58894C93 AND 00FF0000)
temp = C2894036

X1 = X1 + (temp >>> 16)
= 35BF665C + (C2894036 >>> 16)
= 75F628E5

temp = (X7 AND 00FF0000) OR (X6 AND 0000FF00) OR (X5 AND 000000FF) OR (X4 AND FF000000)
temp = (652F40CE AND 00FF0000) OR (C3A4C236 AND 0000FF00) OR (C20EA627 AND 000000FF) OR (58894C93 AND FF000000)
temp = 582FC227

X2 = X2 + (temp >>> 24)
= 356F2644 + (582FC227 >>> 24)
= 65314D9C

temp = (X7 AND FF000000) OR (X6 AND 00FF0000) OR (X5 AND 0000FF00) OR (X4 AND 000000FF)
temp = (652F40CE AND FF000000) OR (C3A4C236 AND 00FF0000) OR (C20EA627 AND 0000FF00) OR (58894C93 AND 000000FF)
temp = 65A4A693

X3 = X3 + temp
= 0082DD2A + 65A4A693
= 662783BD

for the final result of the hashing process of SMS message that will be sent is as follows:

X0 || X1 || X2 || X3, (where || = concatenation)
Result Hash = 8994444975F628E565314D9C662783BD

based on the results of the process of hashing is there that the results of cryptographic algorithms HAVAL produce a different message length and message safer than without the cryptographic process

Plaintext :
“This is only message for testing how HAVAL algorithm work and show how the result were getting after hashing, this were made iam”

Hashing Ciphertext:
8994444975F628E565314D9C662783BD

3. Conclusion

Conclusions that could obtain from the use of Hashing Variable Length algorithms in SMS is; an SMS will be safe from the process of tapping is due to be hashing and all character even up to 1000 character from SMS will be compressed to 32 characters, so that HAVAL algorithm can be used as separate solutions for security and compression of the message, the function HAVAL, only one bit change in the message will change the overall value of the hash function, or in other words the hash function HAVAL generate hash values very random and then HAVAL hash function can be used in a cryptographic method that requires a hash value, such as a digital signature E-Sign and interlock protocol.

References

[1] R. Rahim and A. Ikhwan, "Cryptography Technique with Modular Multiplication Block Cipher and Playfair Cipher," *IJSRST*, vol. II, no. 6, pp. 71-78, 2016.

[2] R. Rahim and A. Ikhwan, "Study of Three-Pass Protocol on Data Security," *International Journal of Science and Research (IJSR)*, vol. 5, no. 11, pp. 102-104, 2016.

[3] E. Haryanto and R. Rahim, "Arnold’s Cat Map Algorithm in Digital Image," *International Journal of Science and Research (IJSR)*, vol. V, no. 10, pp. 1363-1365, 2016.

[4] M. Iqbal, M. A. Syahbana Pane and A. P. Utama Siahaan, "SMS Encryption Using One-Time Pad Cipher," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 18, no. 6, pp. 54-58, 2016.

[5] M. A. Taha, M. Farajalla and R. Tahboub, "A Practical One Way Hash Algorithm based on Matrix Multiplication," *International Journal of Computer Applications*, vol. 23, no. 2, pp. 34-38, 2011.

[6] S. Mishra, S. Mishra and N. Kumar, "Hashing Algorithm: MD5," *International Journal of Scientific Research & Development*, vol. I, no. 9, pp. 1931-1933, 2013.

[7] R. Sobti and G. Geetha, "Cryptographic Hash Functions: A Review," *International Journal of Computer Science*, vol. IX, no. 2, pp. 461-479, 2012.

[8] V. R. Kulkarni, S. Mujawar and S. Apte, "Hash Function Implementation Using Artificial Neural Network," *International Journal on Soft Computing*, vol. I, no. 1, pp. 1-8, 2010.

[9] Y. Zheng, J. Pieprzyk and J. Seberry, "HAVAL - A one-way hashing algorithm with variable length output," in *Research Online*, Australia, 1993.

[10] P. K. Pearson, "Fast Hashing of Variable-Length Text String," *Communication of ACM*, vol. 33, no. 6, pp. 677-680, 1990.

[11] J. Majumder, "Dictionary Attack on MD5 Hash," *International Journal of Engineering Research and Applications*, vol. II, no. 3, pp. 721-724, 2012.

[12] B. Schneier, Applied Cryptography, California: John Wiley & Sons, 1996.

[13] K. H. Rosen, An Introduction to Cryptography Second Edition, USA: Chapman & Hall/CRC, 2007.

[14] W. Stallings, Cryptography and Network Security, USA: Prentice Hall, 2005.