Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika dan Komputer)

Vol.19. No.1, Februari 2020, pp. 53~61

P-ISSN : 1978-6603

E-ISSN : 2615-3475

Modifikasi SHA-256 dengan Algoritma *Hill Cipher* untuk Pengamanan Fungsi Hash dari Upaya Decode Hash

[*#1]Zaimah Panjaitan, [#1]Erika Fahmi Ginting, [#2]Yusnidah

[#1]Program Studi Sistem Informasi, STMIK Triguna Dharma

[#2]Akademi Maritim Indonesia

Article Info

Article history:

Received Feb [th], 2020

Revised Feb [th], 2020

Accepted Feb [th], 2020

Keyword:

Kriptografi

Hash code

SHA-256

Hill Cipher

Decode Hash

ABSTRACT

SHA-256 merupakan salah satu fungsi hash yang banyak digunakan untuk keperluan autentikasi pesan seperti otorisasi pengguna dengan *username* dan *password* pada keamanan sistem login, enkripsi tanda tangan digital, dan lain sebagainya. SHA-256 menerima ukuran variabel pesan sebagai masukan dan menghasilkan output karakter berukuran 256bit yang disebut sebagai *hash code*. Sebagaimana fungsi hash lainnya, pada SHA-256 tidak tersedia algoritma dekripsi yang dapat mengembalikan *hash code* menjadi pesan asli. Hal ini yang menyebabkan SHA-256 layak digunakan untuk keamanan autentikasi. Namun pada saat ini, banyak layanan dan *tools* yang beredar di internet yang dapat digunakan untuk mendekripsikan atau decode hash termasuk SHA-256. Oleh karena itu, SHA-256 perlu untuk dimodifikasi dengan algoritma lain agar *tools* yang beredar di internet tidak dapat dengan mudah mengembalikan fungsi hash ke dalam pesan aslinya. Upaya pengamanan fungsi hash ini dilakukan agar pihak yang tidak berkepentingan tidak dapat dengan mudah menembus sistem keamanan autentikasi yang dibuat dengan SHA-256.

Corresponding Author

Nama : Zaimah Panjaitan

Program Studi : Sistem Informasi

STMIK Triguna Dharma

Email: zaimahp09@gmail.com

1. PENDAHULUAN

SHA-256 merupakan salah satu fungsi hash. Fungsi hash banyak digunakan pada enkripsi password akun pengguna website. Dengan fungsi hash ini, password yang dimasukkan pengguna saat login akan secara otomatis berubah menjadi pesan yang panjangnya tetap dan tidak diketahui maknanya bahkan oleh pemilik pesan dalam hal ini pemilik password, dan penyedia layanan keamanan dalam hal ini pemilik website yang membubuhkan keamanan pada situsnya. Karena pada dasarnya fungsi hash satu arah tidak dapat mengembalikan pesan pada bentuk aslinya. Hal inilah yang menjadi penyebab mengapa fungsi hash selalu digunakan dalam penyandian password.

SHA-256 mengubah pesan masukan ke dalam message digest (MD) sebesar 256 bit. Jika pesan masukan kurang atau lebih dari 264 bit, maka pesan tersebut harus dioperasikan oleh 512 bit dalam kelompok dan menjadi sebuah MD yang panjangnya tetap, yaitu 256 bit.

Dewasa ini, banyak *tools* dan layannan yang tersedia diinternet yang berfungsi untuk mendekripsi atau mengembalikan MD ke dalam pesan asli. Hal ini menyebabkan tingkat keamanan SHA-256 menjadi sangat rendah. Oleh karena itu dibutuhkan modifikasi pada SHA-256.

Pada penelitian ini, akan dilakukan modifikasi SHA-256 dengan algoritma *Hill Cipher* untuk menghasilkan MD berbeda yang jika didekripsi dengan cara decode hash, hasilnya masih berupa cipherteks. Dengan demikian, pihak yang tidak berkepentingan tidak dapat mengetahui secara langsung isi pesan yang di decode dengan *tools* decode hash tersebut.

2. METODE PENELITIAN

Sebelum melakukan penelitian, diperlukan suatu metodologi atau kaidah-kaidah yang harus dilakukan untuk menunjang keabsahan penelitian yang dilakukan. Hal ini dimaksudkan untuk memaksimalkan hasil penelitian nantinya. Metodologi penelitian ini akan memuat beberapa kerangka kerja penelitian yang akan dilakukan.

Langkah pertama yang dilakukan dalam penelitian ini adalah analisis kebutuhan. Masalah yang diidentifikasi pada penelitian ini adalah ancaman keamanan pada fungsi hash SHA-256 yang pada dasarnya tidak dapat didekripsi, namun karena beredar *tools* dan layaran di internet yang dapat mendekripsi atau decode hash, sehingga perlu dilakukan pengamanan pada SHA-256 dengan memodifikasinya dengan algoritma lain dalam hal ini algoritma *Hill Cipher*.

Kemudian selanjutnya langkah kedua yaitu dilakukan analisa pada modifikasi SHA-256 dengan algoritma *Hill Cipher*. Dalam hal ini akan dilakukan enkripsi pesan dengan SHA-256 untuk menghasilkan MD atau hash code sepanjang 256bit. Kemudian, MD yang sudah di dapatkan akan dienkripsi kembali dengan algoritma *Hill Cipher* yang bertujuan agar saat seseorang mendekripsi MD dengan *tools* decode hash, maka pesan yang dihasilkan berupa cipherteks yang tidak diketahui makna aslinya. Dengan demikian, keamanan MD dapat ditingkatkan.

Langkah terahir berupa pengujian. Untuk menguji hasil penelitian ini, dilakukan decode hash dengan *tools* yang tersedia di internet pada sebuah pesan yang dienkripsi dengan SHA-256. Kemudian dilakukan perbandingan dengan menguji decode hash yang dienkripsi dengan SHA-256 yang dimodifikasi dengan algoritma *Hill Cipher*. Dengan begitu dapat dibedakan hasil yang didapat dari kedua percobaan dan dibandingkan tingkat keamanannya.

3. ANALISA DAN HASIL

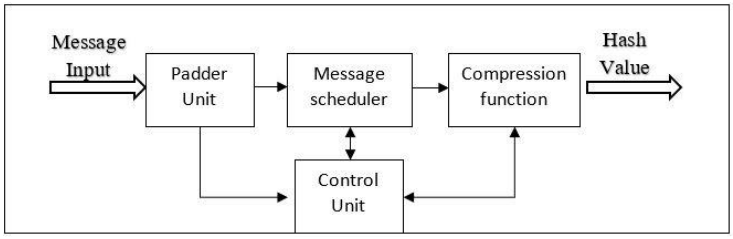
3.1. SHA-256

SHA-2 adalah sebuah kriptografi fungsi hash yang dirancang oleh National Security Agency (NSA) dan dipublikasikan oleh National Institute of Standart and Technology (NIST) sebagai sebuah Federal Information Processing Standart (FIPS) oleh U.S. Ada empat algoritma untuk keamanan fungsi hash yaitu

SHA-0, SHA-1, SHA-2, dan SHA-3. NIST memperbaharui SHA-2, dengan panjang output (256 atau 512- bit di atas 160-bit pada SHA-1) dan perbedaan-perbedaan pada SHA ini merupakan besar pesan yang ada pada proses komputasi[1].

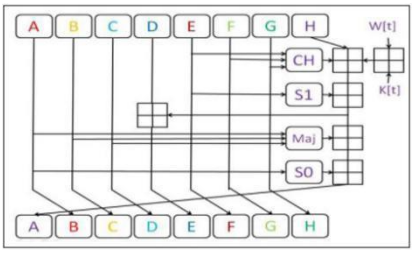
SHA (Algoritma keamanan fungsi hash) merupakan algoritma enkripsi fungsi hash yang dapat digunakan untuk menghasilkan penggambaran konsolidasi dari sebuah data teks yang disebut sebuah proses pesan. SHA-256 dan SHA-512 adalah fungsi hash dengan kapasitas terbaru dengan panjang 32-bit dan 64-bit kata secara terpisah. Kedua fungsi hash ini dalam proses matematisnya menggunakan penjumlahan karakter yang berbeda dan ditambah dengan konstanta substansi. Meski demikian, struktur keduanya pada dasarnya tidak jauh berbeda, perbedaannya hanya terletak pada jumlah putaran saja[2].

Arsitektur sederhana dari algoritma SHA-256 ditunjukkan oleh gambar berikut:



Gambar 1. Arsitektur Sederhana SHA-256.

Berikut merupakan jalur komputasi SHA-2:



Gambar 2. Jalur Komputasi SHA-256

Adapun proses atau tahapan pada algoritma SHA-256 adalah sebagai berikut :

1. Message Padding

Pada tahap pertama ini, pesan berupa binary disisipkan dengan angka 1 dan ditambahkan bit-bit pengganjal, yakni angka 0 hingga panjang pesan kongruen dengan 448 modulo 512. Panjang pesan asli ditambah sebagai angka biner 64 bit. Maka panjang pesan sekarang menjadi kelipatan 512 bit.

2. Parsing

Pada proses ini, pesan yang telah dipadding kemudian dibagi menjadi N buah blok 512 bit : $M^{(1)}, M^{(2)}, \dots M^{(n)}$

3. Message Expansion

Masing-masing blok 512 bit tadi dipecah menjadi 16 word 32 bit : $M_0^{(i)}, M_1^{(i)}, \dots M_{15}^{(i)}$

Kemudian akan diperluas menjadi 64 word yang diberli label W0, W1,...W63.

4. Message Compression

Masing-masing dari 64 word yang diberi label W0, W1,...W64 tadi diproses denga algoritma fungsi hash SHA-256.

Dalam proses tersebut, inti utama dari algoritma SHA-256 adalah membuat 8 variabel yang diberikan nilai awal L0-L7. Nilai awal tersebut adalah sebagai berikut :

Tabel 1. Nilai Awal Variabel SHA-256

L0	a	6A09E667	L4	e	510E527F
L1	b	BB67AE85	L5	f	9B05688C
L2	c	3C6EF372	L6	g	IF83D9AB
L3	d	A54FF53A	L7	h	5BE0CD19

5. Kemudian dilakukan perhitungan sebanyak 64 kali pitaran untuk setiap blok. Delapan variabel yang diberikan pada nilai awal berupa L0 sampai dengan L7 asumsikan menjadi nilai A,B,C,D,E,F,G, dan H nilainya terus berganti selama perputaran dengan rumus sebagai berikut :

$T1 = h + s1 + CH + K[t] + W[t]$

$T2 = s0 + MAJ$

$h = g$

$g = f$

$f = e$

$e = d + T1$

$d = c$

$c = b$

$b = a$

$a = T1 + T2$

Keterangan :

$s0 = (a \ggg 2) \oplus (a \ggg 13) \oplus (a \ggg 22)$

$s1 = (e \ggg 6) \oplus (e \ggg 11) \oplus (e \ggg 25)$

$CH = (e \,\&f) \oplus \big((\neg\, e)\&\, g)\big)$

$MAJ = (a \,\&\, b) \oplus (a \,\&c) \oplus (b \,\&c)$

Nilai ahir hash adalah sebagai berikut :

$L0 = L0 + a$

$$L1 = L1 + b$$

$$L2 = L2 + c$$

$$L3 = L3 + d$$

$$L4 = L4 + e$$

$$L5 = L5 + f$$

$$L6 = L6 + g$$

$$L7 = L7 + h$$

Maka, MD yang didapatkan adalah hasil ahir penjumlahan yang disusun secara memanjang [3].

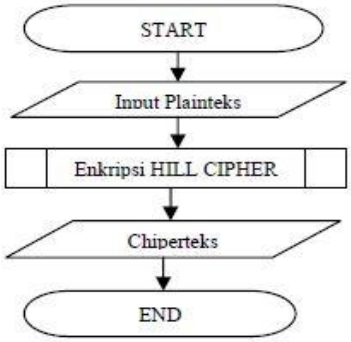
3.2. *Hill Cipher*

Hill Cipher adalah teknik kriptografi cipher block dengan kunci simetris yang dikembangkan oleh matematikawan Lester pada tahun 1929[4]. Algoritma *Hill Cipher* menggunakan matriks berukuran m x m sebagai kunci untuk proses enkripsi dan dekripsi [5]. Pada algoritma *Hill Cipher* pengirim dan penerima harus berbagi dan menggunakan kunci berupa matriks yang sama untuk proses enkripsi dan dekripsinya [6]. Teknik kriptografi ini diciptakan dengan maksud untuk dapat menciptakan Cipher (kode) yang tidak dapat dipecahkan menggunakan teknik analisis frekuensi [7].

Algoritma *Hill Cipher* adalah salah satu algoritma klasik, namun dibandingkan dengan algoritma kriptografi klasik lainnya *Hill Cipher* dianggap cukup kuat karena menggunakan matriks sebagai kuncinya. Hal ini menyebabkan cipher teks yang dihasilkan oleh algoritma ini lebih sulit untuk dipecahkan oleh kriptanalis.

Hill Cipher termasuk kepada algoritma kriptografi klasik yang sangat sulit dipecahkan oleh kriptanalis apabila dilakukan hanya dengan mengetahui berkas *ciphertext* saja. Karena *Hill Cipher* tidak mengganti setiap abjad yang sama pada *plaintext* dengan abjad lainnya yang sama pada *ciphertext* karena menggunakan perkalian matriks pada dasar enkripsi dan dekripsinya [8]

Proses enkripsi pada *Hill Cipher* dilakukan dengan mengalikan setiap blok plainteks dengan kunci yang telah disediakan terlebih dahulu. Kunci yang dibuat adalah berupa matriks yang invertible atau memiliki invers. Berikut adalah flowchart enkripsi dari *Hill Cipher* :



Gambar 3. Flowchart Enkripsi *Hill Cipher*

Secara matematis, proses enkripsi pada *Hill Cipher* adalah [9]:

$C = K.P$

Dimana :

C = Cipherteks

K = Kunci

P = Plainteks

Adapun pembentukan kunci *Hill Cipher* adalah dengan membentuk matriks m x m yang invertible yaitu sesuai dengan persamaan :

$K . K^{-1} = I$

Dimana :

K = Matriks kunci

K^{-1} = Invers matriks kunci

I = Matriks identitas

Sebelum dilakukan proses enkripsi pada plainteks, terlebih dahulu plainteks diubah ke dalam angka 0-25 dengan sekema sebagai berikut :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Gambar 4. Skema Angka Pada *Hill Cipher*

Kemudian proses dekripsi pada *Hill Cipher* adalah kebalikan dari proses enkripsi, secara matematis :

$P = K^{-1} . C$

Dimana :

P = Plainteks

K^{-1} = Invers matriks kunci, untuk menentukannya berlaku rumus 1/det K mod 26.

C = Cipherteks

3.3. Modifikasi SHA-256 dengan Algoritma *Hill Cipher*

Pada kasus ini plainteks dienkripsi dengan SHA-256 untuk menghasilkan string hash code, kemudian untuk memperkuat hash tersebut agar tidak dapat didekripsi oleh layanan *tools* decode hash, maka akan dilakukan enkrispsi hash dengan algoritma *Hill Cipher*.

Pesan yang dienkripsi dalam hal ini berupa angka misalnya : 347 (angka ini hanya contoh untuk mempermudah proses perhitungan secara manual). Adapun tahapannya sebagai berikut :

1. Pesan diubah ke bentuk biner :

3 = 00000011

4 = 00000100

7 = 00000111

Maka pesan, M = 00000011 00000100 00000111.

Panjang pesan, l = 24 bit

2. *Message Padding*

Padding pesan dilakukan dengan cara menambahkan bit 1 dan sisanya adalah bit 0 hingga pesan sepanjang 512 bit. Untuk mencari jumlah nol yang ditambahkan (k) digunakan rumus berikut:

$$l + 1 + k \equiv 448 \; mod \; 512$$

$$24 + 1 + k \equiv 448 \; mod \; 512$$

$$k \equiv 448 - 25 \; mod \; 512$$

$$k \equiv 423$$

Karena k = 423 maka banyaknya bit 0 yang akan ditambahkan adalah sebanyak 423 bit. Setelah itu ditambahkan jumlah panjang pesan pada akhir pesan yang dipadding sebanyak 8 bit dengan nilai l = 24 = 00011000

Tabel 2. Hasil Padding Pesan

00000011	00000100	00000111	10000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00011000

3. *Parsing*

Pesan yang telah dipadding akan menghasilkan blok pesan 512 bit M(0), M(1),... M(n-1). Namun, dalam contah kasus ini karena panjang pesan yang dipadding tidak melebihi 512 bit, maka hanya menghasilkan satu blok 512 bit yaitu M(0).

Tahapan selanjutnya adalah membagi setiap blok 512 bit menjadi 16 buah word 32 bit sebagai berikut :

Tabel 3. Hasil Parsing Pesan

$M_0^{(0)}$	0000	0011	0000	0100	0000	0111	1000	0000
$M_0^{(1)}$	0000	0000	0000	0000	0000	0000	0000	0000
$M_0^{(2)}$	0000	0000	0000	0000	0000	0000	0000	0000
$M_0^{(3)}$	0000	0000	0000	0000	0000	0000	0000	0000
$M_0^{(4)}$	0000	0000	0000	0000	0000	0000	0000	0000
$M_0^{(5)}$	0000	0000	0000	0000	0000	0000	0000	0000
$M_0^{(6)}$	0000	0000	0000	0000	0000	0000	0000	0000

$M_0^{(7)}$	0000	0000	0000	0000	0000	0000	0000	0000
$M_0^{(8)}$	0000	0000	0000	0000	0000	0000	0000	0000
$M_0^{(9)}$	0000	0000	0000	0000	0000	0000	0000	0000
$M_0^{(10)}$	0000	0000	0000	0000	0000	0000	0000	0000
$M_0^{(11)}$	0000	0000	0000	0000	0000	0000	0000	0000
$M_0^{(12)}$	0000	0000	0000	0000	0000	0000	0000	0000
$M_0^{(13)}$	0000	0000	0000	0000	0000	0000	0000	0000
$M_0^{(14)}$	0000	0000	0000	0000	0000	0000	0000	0000
$M_0^{(15)}$	0000	0000	0000	0000	0000	0000	0001	1000

4. *Message Schadule*

Selanjutnya, masing-masing 16 word di atas diperluas menjadi 64 buah 32 bit word sebagai berikut:

Tabel 4. Hasil Message Schadule

W_0	03040780	W_{16}	00000000	W_{32}	00000000	W_{48}	00000000
W_1	00000000	W_{17}	00000000	W_{33}	00000000	W_{49}	00000000
W_2	00000000	W_{18}	00000000	W_{34}	00000000	W_{50}	00000000
W_3	00000000	W_{19}	00000000	W_{35}	00000000	W_{51}	00000000
W_4	00000000	W_{20}	00000000	W_{36}	00000000	W_{52}	00000000
W_5	00000000	W_{21}	00000000	W_{37}	00000000	W_{53}	00000000
W_6	00000000	W_{22}	00000000	W_{38}	00000000	W_{54}	00000000
W_7	00000000	W_{23}	00000000	W_{39}	00000000	W_{55}	00000000
W_8	00000000	W_{24}	00000000	W_{40}	00000000	W_{56}	00000000
W_9	00000000	W_{25}	00000000	W_{41}	00000000	W_{57}	00000000
W_{10}	00000000	W_{26}	00000000	W_{42}	00000000	W_{58}	00000000
W_{11}	00000000	W_{27}	00000000	W_{43}	00000000	W_{59}	00000000
W_{12}	00000000	W_{28}	00000000	W_{44}	00000000	W_{60}	00000000
W_{13}	00000000	W_{29}	00000000	W_{45}	00000000	W_{61}	00000000
W_{14}	00000000	W_{30}	00000000	W_{46}	00000000	W_{62}	00000000
W_{15}	00000018	W_{31}	00000000	W_{47}	00000000	W_{63}	00000000

5. Inisialisasi Variabel dan Konstanta

Variabel awal pada fungsi hash SHA-256 adalah sebagai berikut :

a = H0(0) = 6A09E667

b = H0(1) = BB67AE85

c = H0(2) = 3C6EF372

d = H0(3) = A54FF53A

e = H0(4) = 510E527F

f = H0(5) = 9B05688C

g = H0(6) = 1F83D9AB

h = H0(7) = 5BE0CD19

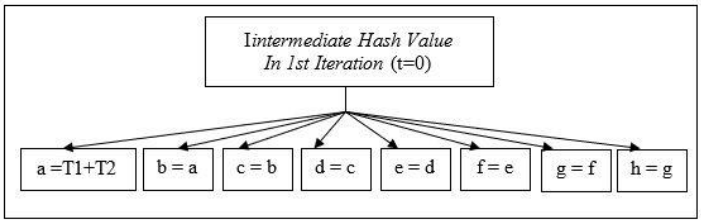
Nilai konstanta pada fungsi hash SHA-256 (K0{256}, K1{256},.... K63{256}) adalah sebagai berikut :

Tabel 5. Nilai Konstanta SHA-256

428A2F98	71374491	B5C0FBCF	E9B5DBA5	3956C25B	59F111F1	923F82A4	AB1C5ED5
D807AA98	12835B01	243185BE	550C7DC3	72BE5D74	80DEB1FE	9BDC06A7	C19BF174
E49B69C1	EFBE4786	0FC19DC6	240CA1CC	2DE92C6F	4A7484AA	5CB0A9DC	76F988DA
983E5152	A831C66D	B00327C8	BF597FC7	C6E00BF3	D5A79147	06CA6351	14292967
27B70A85	2E1B2138	4D2C6DFC	53380D13	650A7354	766A0ABB	81C2C92E	92722C85
A2BFE8A1	A81A664B	C24B8B70	C76C51A3	D192E819	D6990624	F40E3585	106AA070
19A4C116	1E376C08	2748774C	34B0BCB5	391C0CB3	4ED8AA4A	5B9CCA4F	682E6FF3
748F82EE	78A5636F	84C87814	8CC70208	90BEFFFA	A4506CEB	BEF9A3F7	C67178F2

6. *Hash Computation*

Dalam proses ini dilakukan perhitungan nilai a sampai nilai h sebanyak 64 kali putaran. Adapun bentuk perhitungannya adalah sebagai berikut :



Gambar 5. *Intermediate Hash Value In 1st Iteration*

Rumus untuk menghasilkan nilai T1 adalah :

$T1 = h + S1(e) + ch(e, f, g) + kt + wt$

Di mana,

h = Nilai h pada iterasi sebelumnya

S1(e) = $(e \ggg 6) \oplus (e \ggg 11) \oplus (e \ggg 25)$

Ch (e, f,g) = $(e \,\&f) \oplus \big((\neg\, e)\& \, g)\big)$

Kt = Nilai konstanta

Wt = Nilai W pada Message Schadule

Maka nilai T1 adalah sebagai berikut :

$h = 5BE0CD19$

$Si(e) = (510E527F \ggg 6) \oplus (510E527F \ggg 11) \oplus (510E527F \ggg 11)$

$Si(e) = 3587272B$

$ch(e, f, g) = (510E527F \,\&\, 9B05688C \oplus \big((\neg\, 510E527F)\& \, 1F83D9AB)\big)$

$ch(e, f, g) = 1F85C98C$

$T1 = 5BE0CD19 + 3587272B + 1F85C98C + 428A2F98 + 03040780$

$T1 = F67BF4E8$

Rumus untuk menghasilkan nilai T2 adalah sebagai berikut :

$T2 = S0(a) + Maj(a, b, c)$

Di mana,

S0(a) = $(a \ggg 2) \oplus (a \ggg 13) \oplus (a \ggg 22)$

Maj (a,b,c) = $(a \,\&\, b) \oplus (a \,\&\, c) \oplus (b \,\&\, c)$

Maka nilai T2 adalah sebagai berikut :

$S0(a) = (6A09E667 \ggg 2) \oplus (6A09E667 \ggg 13) \oplus (6A09E667 \ggg 22)$

$S0(a) = CE20B47E$

$Maj(a, b, c) = (6A09E667 \,\&\, BB67AE85) \oplus (6A09E667 \,\&\, 3C6EF372) \oplus (BB67AE85 \,\&\, 3C6EF372)$

$Maj(a, b, c) = 3A6FE667$

$T2 = CE20B47E + 3A6FE667$

$T2 = 08909AE5$

Karena nilai T1 dan T2 telah didapatkan, maka nilai a adalah sebagai berikut :

$a = T1 + T2$

$a = F67BF4E8 + 08909AE5$

$a = FF0C8FCD$

Maka putaran pertama (t = 0) hash code dari pesan "347" adalah sebagai berikut:

Tabel 6. Nilai putaran pertama

	a	b	c	d	e	f	g	h
init	6a09e667	bb67ae85	3cbef372	a54ff53a	510e527f	9b05688c	1f83d9ab	5be0c019
t=0	ff0c8fcd	6a09e667	bb67ae85	3cbef372	a54ff53a	510e527f	9b05688c	1f83d9ab

Selanjutnya dilakukan lagi pencarian nilai a untuk putaran kedua dan seterusnya hingga putaran 64 (t = 63).

7. *Compute intermediate hash value + initial hash value*

Setelah didapat ke 64 putaran dari hash computation, kemudian pada tahap ini dilakukan proses penjumlahan hasil putaran yang ke-64 dengan initial hash value. Maka hasil yang didapatkan adalah sebagai berikut:

H0(0) = b87fcbba + 6a09e667 = 2289b221

H0(1) = f82e573e + bb67ae85 = b39605c3

H0(2) = 0cdf7f1e + 3c6ef372 = 494e7290

H0(3) = e01223af + a54ff53a = 856218e9

H0(4) = e0b1b876 + 510e527f = 31c00af5

H0(5) = bbca117b + 9b05688c = 56cf7a07

H0(6) = 62ed2e6e + 1f83d9ab = 82710819

H0(7) = df4697f8 + 5be0cd19 = 3b276511

8. penggaabungan H0 - H7

H0|| H1|| H2|| H3|| H4|| H5|| H6|| H7

2289b221||b39605c3||494e7290||856218e9||31c00af5||56cf7a07||82710819||3b276511

9. Nilai *hash*

Maka dengan seluruh proses yang dilalui didapatlah nilai hash sebagai berikut:

2289b221b39605c3494e7290856218e931c00af556cf7a07827108193b276511

Hash code yang telah didapatkan di atas kemudian dienkripsi dengan algoritma *Hill Cipher* :

1. Menentukan kunci.

Dalam hal ini karena hash code terdiri dari 64 karakter. Matriks kunci kita bentuk berupa matriks 2 x 2 sehingga perkalian matriks dilakukan dengan setiap 2 blok pesan. Adapun matriks kunci adalah :

$$K = \begin{bmatrix}5 & 3\\ 3 & 2\end{bmatrix} \text{ x } K^{-1} = \begin{bmatrix}5 & 3\\ 3 & 2\end{bmatrix} = \begin{bmatrix}79 & 130\\ 52 & 79\end{bmatrix} mod\ 26 = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

2. Mengkonversi plainteks.

Karena plainteks terdiri dari angka dan huruf, maka konversinya sebagai berikut :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

S	T	U	V	W	X	Y	Z	0	1	2	3	4	5	6	7	8	9
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Jika plainteks berupa hash di atas dikonversi, maka kasilnya sebagai berikut :

Tabel 7. Nilai konversi pesan

2 = 28	b = 1	4 = 30	8 = 34	3 = 29	5 = 31	8 = 34	3 = 29
2 = 28	3 = 29	9 = 35	5 = 31	1 = 27	6 = 32	2 = 28	b = 1
8 = 34	9 = 35	4 = 30	6 = 32	c = 2	c = 2	7 = 33	2 = 28
9 = 35	6 = 32	e = 4	2 = 28	0 = 26	f = 5	1 = 27	7 = 33
b = 1	0 = 26	7 = 33	1 = 27	0 = 26	7 = 33	0 = 26	6 = 32
2 = 28	5 = 31	2 = 28	8 = 34	a = 0	a = 0	8 = 34	5 = 31
2 = 28	c = 2	9 = 35	e = 4	f = 5	0 = 26	1 = 27	1 = 27
1 = 27	3 = 29	0 = 26	9 = 35	5 = 31	7 = 33	9 = 35	1 = 27

3. Mengalikan matriks kunci dengan blok plainteks, dan hasilnya di mod 26 :

Blok 1 :

$$\begin{bmatrix}5 & 3\\3 & 2\end{bmatrix} x \begin{bmatrix}28\\28\end{bmatrix} = \begin{bmatrix}5*28 + & 3*28\\3*28 + & 2*28\end{bmatrix} = \begin{bmatrix}224\ mod\ 26 = 16\\140\ mod\ 26 = 10\end{bmatrix}$$

Hasilnya, karakter pertama 16 = q dan karakter kedua 10 = k.

Kemudian lakukan perhitungan pada seluruh hash code di atas sehingga mendapatkan hasil berupa hash code baru yaitu : qkpqihniojlnpktmvegupztbdikwdtveslkgaaozrbzqjvvoucmxyqgvslfutcif

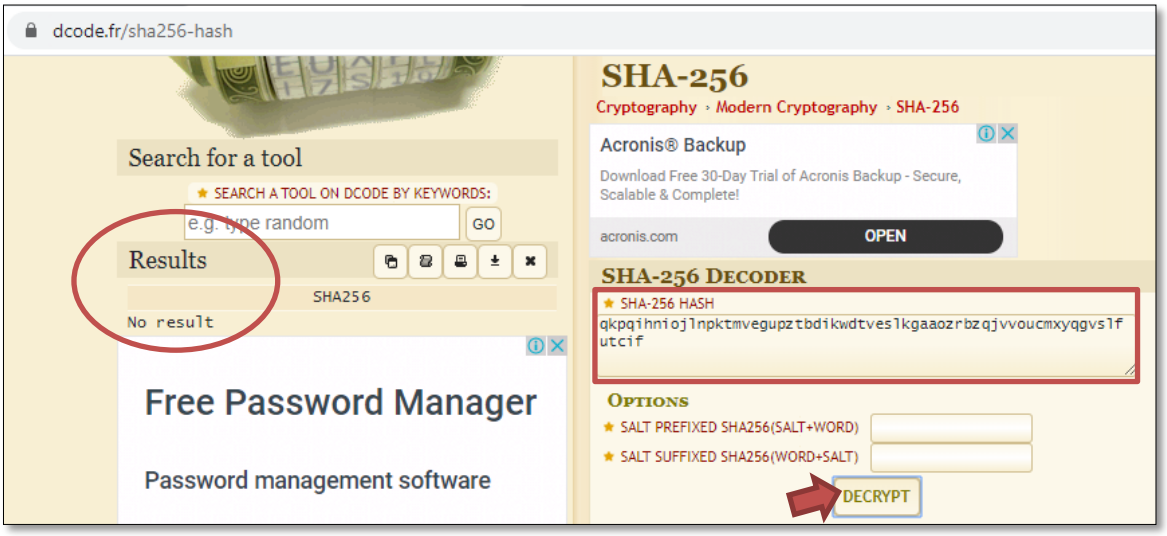
3.4. Pengujian

Pada kasus ini akan dilihat bagaimana *tools* yang tersedia di internet dapat mendekripsi hash code yang tidak dimodifikasi dengan algoritma lain.

Gambar 6. Hasil Decode SHA-256

Ketika dimasukkan kode hash pada kolom yang disediakan, kemudian klik Decrypt maka hasilnya akan terlihat plainteks.

Jika dimasukkan hash code yang telah dimodifikasi dengan *Hill Cipher*, maka hasilnya akan tampak sebagai berikut :



Gambar 7. Hasil Decode SHA-256 Modifikasi *Hill Cipher*

Program yang ada pada *tools* tersebut tidak dapat membaca pesan asli karena sudah dimodifikasi.

4. KESIMPULAN

Dari penelitian yang dilakukan dapat diambil kesimpulan :

1. Modifikasi SHA-256 dengan algoritma *Hill Cipher* dapat dilakukan untuk meningkatkan keamanan *hash code* agar tidak dapat didekripsi oleh *tools* decode hash yang beredar di internet.

2. algoritma *Hill Cipher* layak digunakan untuk memodifkasi sebuah *hash code* sebab tingkat keamanannya yang tergolong kuat. Hal ini dikarenakan kunci pada algoritma *Hill Cipher* tidak mudah untuk ditebak karena berbentuk matriks yang hanya pembuatnya yang mengetahui kunci tersebut.

3. Fungsi hash dewasa ini perlu untuk dijaga keamanannya agar tidak dapat di dekripsi oleh *tools* decode hash yang beredar di internet

REFERENSI

[1] R. K. Ibrahim, *et al.*, "Incorporating Sha-2 256 With OFB To Relize A Novel Encryption Method" *2015 World Symposium on Computer Network and Information Security (WSCNIS) Hammamet, Tunisia, IEEE Transactions on Industrial Electronics",* Sep 2015, pp. 19-21.

[2] C. P. Sukhbir, *et al.*, "Crypto Currencies for Digital Currency Using Cipher Text and SHA256" *Jour of Adv Research in Dynamical & Control Systems", IEEE* pp. 530-534, 2017

[3] S. S. Omran, *et al.*, "Design Of SHA-1 & SHA-2 MIPS Processor Using FPGA" *Annual Conference on New Trends in Informations Technology Applications (NTICT 2017) : Hold at University of Information Technology and Communication*", *IEEE communication society, Iraq Chapeter,* 7-9 Mar 2017, pp. 268-273.

[4] M. Eisenberg, "Hill Ciphers and Modular Linear Alebra," *1999 Mimegrophed Notes, University of Massachusetts,* pp. 1-19.

[5] D. Novriansyah, *et al.,* "A New Image Encryption Technique Combining Hill Cipher Method, Morse Code and Least Significant Bit Algorithm", *Journal of Physics: Conference Series*, 2018 vol 954.

[6] Z. E. Dewadeh, *et al*., "A New Image Encryption Technique Combining Elliptic Curve Cryptosystem with Hill Cipher," *Journal of King Saud University-Computer and Information Sciences,* pp.349-355

[7] J. I. Sari, *et al.,* "Implementasi Penyembunyian Pesan pada Citra Digital dengan Menggabungkan Algoritma Hill Cipher dan Metode Least Significant Bit (LSB)", *Jurnal Manajement dan Informatika Pelita Nusantara STMIK Pelita Nusantara Medan,* 2015 vol:1.

[8] A. H. Hasugian, "Implementasi Algoritma *Hill Cipher* dalam Penyandian Data", ISSN 2301-9425.