Journal of Informatics Management and Information Technology
ISSN 2774-4744 (Media Online)
Vol 1, No 2, April 2021
Hal 60-67
https://hostjournals.com/jimat

Implementasi Digital Signature Pada File Audio Menerapkan Metode SHA-256

Endelina

Program Studi Teknik informatika, Universitas Budi Darma Medan, Sumatera Utara, Indonesia
Email: enjelinamunthe21@gmail.com

Abstrak−Dalam era teknologi informasi yang berkembang sangat pesat, penggunaan tanda tangan sudah banyak diterapkan secara digital melalui tanda tangan digital. Tanda tangan digital berkembang seiring munculnya kebutuhan otentikasi suatu data atau berkas yang digunakan secara digital. Penggunaannya juga bertujuan untuk menghindari pemalsuan ataupun gangguan. Saat ini, pemanfaatan tanda tangan digital sudah banyak diterapkan pada distribusi perangkat lunak, transaksi keuangan, pengiriman berkas. Penggunaan tanda tangan digital pada file audio dapat dilakukan dengan menggunakan enkripsi terhadap pesan yang dikirim dengan kunci untuk menngunakan kombinasi fungsi hash dengan kriptografi kunci-publik. Metode ini dapat digunakan untuk sementara. Penggunaan tanda tangan digital berupa tulisan hasil enkripsi ataupun fungsi hash akan menghasilkan dokumen tanda tangan yang terlihat secara jelas karena ditulis dengan kode yang terlihat. Hal ini dapat memunculkan potensi gangguan dengan menghilangkan tanda tangan sehingga dokumen yang diterima dapat dianggap memang tidak dilengkapi tanda tangan digital.

Kata Kunci: Kriptografi; Audio; Metode Secure Hash Algoritma 256; Hasher Pro

Abstract−In an era of information technology that is growing very rapidly, the use of signatures has been implemented digitally through digital signatures. Digital signatures develop along with the emergence of the need to authenticate data or files that are used digitally. Its use also aims to avoid forgery or interference. Currently, the use of digital signatures has been widely applied to software distribution, financial transactions, and file delivery. The use of digital signatures on audio files can be accomplished by using encryption of messages sent with a key to use a combination of hash functions with public-key cryptography. This method can be used temporarily. The use of a digital signature in the form of an encrypted writing or a hash function will produce a signature document that is clearly visible because it is written in a visible code. This can lead to potential interference by removing the signature so that the document received is deemed not equipped with a digital signature.

Keywords: Cryptography; Audio; Secure Hash Algorithm 256 method; Hasher Pro

1. PENDAHULUAN

Audio (suara) adalah fenomena fisik yang dihasilkan oleh getaran suatu benda yang berupa *signal* analog dengan amplitude yang berubah secara kontiniu terhadap satuan waktu yang disebut frekuensi. Untuk dapat menjaga integritas data dari suatu *file audio*, diciptakan suatu mekanisme yang disebut *digital singnature* atau sering juga nilai *hash*, yaitu kode khusus yang dihasilkan dari fungsi penghasilan *digital signature*. Salah satu teknik keamanan yang dapat memastikan file audio adalah teknik kriptografi. Kriptografi adalah ilmu yang bersandarkan pada teknik matematikan untuk berurusan dengan keamanan informasi seperti kerahasiaan file, keutuhan data dan otentikasi entitas.

Metode yang sesuai dengan permasalahan ini dengan menggunakan. Algoritma yang biasanya dipakai untuk membuat sebuah tanda tangan digital yaitu *Algoritma* SHA-256 yang merupakan algoritma *hash* dari jenis SHA-256 yang menghasilkan *message digest* sepanjang 256 bit. Algoritma SHA-256 dapat digunakan untuk melakukan pengecekan integritas data, pembuatan *digital signature*, dan lain-lain. Sebuah tanda tangan digital dari pengamanan *file audio*, juga terdapat salah satu fungsi *hash* pada jurnal yang berjudul Analisis kecepatan dan keamanan algoritma *Secure Hash algorithm* 256 (SHA-256) untuk otentikasi pada pesan teks. Fungsi otentikasi pesan yang biasa digunakan ialah fungsi *hash*, dengan menggunakan fungsi *hash* satu arah maka akan dihasilkan *message digest* dari pesan asli [1]. Metode SHA-256 ini juga terdapat pada jurnal Implementasi aplikasi Digital Signature menggunakan fungsi *Hash.* Algoritma SHA-256 dan RSA di Badan Pertanahan Nasional kota Cimahi. Hal ini dapat digunakan sebagai mekanisme dan teknik untuk melindungi suatu yang dapat berupa data atau informasi di dalam sistem [2]. SHA-256 dapat menerima *input* pesan sehingga 2^{64}bit yang akan diproses melalui blok dengan ukuran 512 bit.

Proses yang dilakukan pada SHA-256 setelah menerima *input* pesan yaitu melakukan *padding* dengan membagi *input* yang diterima menjadi beberapa blok berukuran 512 bit. Setelah mengubah *input* pesan menjadi beberapa blok, selanjutnya yaitu menambahkan bit bernilai 1 dan 0 pada blok terakhir. Algoritma SHA-256 memproses blok pesan dan *rounds* yang berjumlah 64 sesuai penjadwalan *message schedule* dan parameter algoritma SHA-256 yang akan digunakan adalah konstanta *Initialization Vector* (1V). Variabel *hash* a sampai h akan diproses di dalam perulangan atau *rounds* sebanyak 64 kali *rounds*, Selanjutnya yaitu menjalankan fungsi *choose* atau ch yang berisi operasi *bitwise* dan XOR.

2. METODOLOGI PENELITIAN

2.1 Kriptografi

Kriptografi (cryptography) berasal dari bahasa Yunani, yaitu dari kata crypto dan graphia yang berarti ‘penulisan rahasia’. Kriptografi adalah ilmu ataupun seni yang mempelajari bagaimana membuat suatu pesan yang dikirim oleh pengirim dapat disampaikan kepada penerima dengan aman. Kriptografi merupakan bagian dari suatu cabang ilmu

matematika yang disebut kriptologi (cryptology). Kriptografi bertujuan menjaga kerahasiaan informasi yang terkandung dalam data sehingga informasi tersebut tidak dapat diketahui oleh pihak yang tidak sah[4].

2.2 Metode (*Secure Hast Algoritma*)

SHA-256 adalah salah satu algoritma *hash* yang relatif masih baru. Algoritma ini dirancang oleh *The National Institute of Standards and Tecnology* (NIST) pada tahun 2002. SHA-256 menghasilkan *message digest* dengan panjang 256 bit. SHA-256 aman karena didesain sedemikian rupa sehingga tidak memungkinkan mendapat pesan yang berhubungan dengan *message digest*, atau untuk menemukan dua pesan yang berbeda yang menghasilkan *message digest* yang sama. Proses untuk menghasilkan *message digest* pada algoritma ini meliputi 5 tahapan yaitu [7].

1. *Message padding*
   Input pesan pada algoritma SHA-256 akan dibagi menjadi blok-blok yang masing-masing panjangnya adalah 512 bit. Akibat pembagian ini, maka jumlah blok terakhir akan lebih kecil atau sama dengan 512 bit. Blok terakhir tersebut akan mengalami *message padding.* Sebagai contoh, suatu pesan dengan panjang (dalam bit) 616 bit dinotasikan sebagai M. Setelah dibagi ke dalam blok 512 bit, menjadi blok 1=512 bit dan blok 2=104 bit. Blok terakhir (blok 2) ditambah dengan bit-bit isian. Sesuai dengan ketentuan *message padding,* pada bit terakhir blok 2 ditambahkan bit '1' dan diikuti beberapa bit '0' sedemikian sehingga total panjang bit blok 2 setelah proses *message padding* adalah 448 bit.
2. Penambahan panjang bit
   Setelah proses message padding, jumlah bit pada blok terakhir adalah 448 bit. Representasikan M ke dalam bilangan biner untuk memperoleh 64 bit terakhir, agar total panjang blok terakhir 512 bit.
   a. Urutan byte paling kanan dari nilai representasi panjang pesan (M) dijadikan *low order.*
   b. Tambahan representasi M tersebut pada 448 bit terakhir, sehingga jumlah panjang blok terakhir adalah 512 bit. Pada contoh di atas, M=616 bit dan direpresentasikan ke dalam bilangan biner 16 bit. Karena urutan byte paling kanan dijadikan *low order*, maka susunan bitnya tetap. Tambahkan representasi M ini pada blok terakhir, sehingga panjang total setelah proses ini adalah 512 bit.
3. Inisialisasi nilai *hash* awal
   Pada SHA-256 untuk menyimpan nilai inisialisasi awal dan nilai output sementara digunakan buffer H_O, H_1, H_2, H_3, H_4, H_5, H_6, H_7. Di sisi lain, untuk penyimpanan proses sementara digunakan buffer a, b, c, d, e, f, g, h. Nilai H_O, H_1, H_2, H_3, H_4, H_5, H_6, H_7 untuk inisialisasi awal dalam notasi heksadesimal :
   H_0 = 6a09e667
   H_1 = bb67ae85
   H_2 = 3c6ef372
   H_3 = a54ff53a
   H_4 = 510e527f
   H_5 = 9b05688c
   H_6 = 1f83d9ab
   H_7 = 5be0cd19
4. Pemprosesan
   Pemprosesan merupakan bagian inti yang terdiri atas 1 round yang mempunyai 64 operasi. Untuk memproses setiap satu blok pesan 512 bit diperlukan 64 operasi, setiap blok pesan $M^{(1)}$, $M^{(2)}$,…$M^{(N)}$ dengan N adalah jumlah blok pesan. Untuk setiap blok pesan $M^{(i)}$ akan dilakukan langkah-langkahnya sebagai berikut :
   a. Persiapan penjadwalan pesan, $\{W_t\}$: $M_t^{(i)}$ $0 \le t \le 15$
      $W_t = \{\sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}$ $16 \le t \le 63$
      Dengan fungsi σ_0 dan σ_0 dirumuskan sebagai berikut :
      $\sigma_0(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x)$
      $\sigma_1(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$
      $ROTR^n(x)$ adalah operasi geser kanan putar melingkar, dengan x adalah sebuah penjadwalan pesan (w) dan n adalah bilangan bulat ($0 \le n < w$), yang dapat didefinisikan $ROTR^n(x) = ((x >> n) \vee (x << w - n) = ROTL^{w-n}(x)$. Dalam hal ini, $SHR^n(x)$ adalah operasi menggeser x sebanyak n posisi ke kanan.
   b. Inisialisasi working variable a, b, c, d, e, f, g, dan h, untuk $M^{(1)}$ dengan nilai hash awal :
      $a = H_0^{(i-1)}$
      $b = H_1^{(i-1)}$
      $c = H_2^{(i-1)}$
      $d = H_3^{(i-1)}$
      $e = H_4^{(i-1)}$
      $f = H_5^{(i-1)}$
      $g = H_6^{(i-1)}$
      $h = H_7^{(i-1)}$
   c. Untuk masing-masing jadwal pesan W_t :
      $T_1 = h + \sum_1(e) + Ch(e, f, g) + K_1 + W_1$
      $T_2 = \sum_0(a) + Maj(a, b, c)$
      $h = g$

$g = f$
$f = e$
$e = d + T_1$
$d = c$
$c = b$
$b = a$
$a = T_1 + T_2$
dengan fungsi $\sum_0$, $\sum_1$, Ch, Maj dirumuskan sebagai berikut :
Ch (X, Y, Z) = (X & Y) Θ (X &|Z)
Maj (X, Y, Z) = (X &Y) Θ (X &|Z) Θ (Y & Z)
$\sum_0$ (x) = ROTR^2 (x) Θ ROTR^{13} (x) Θ ROTR^{22} (x)
$\sum_1$ (x) = ROTR^6 (x) Θ ROTR^{11} (x) Θ ROTR^{25} (x)
Nilai K_1 dapat dilihat pada lampiran 2.

5. Menghitung nilai hash prantara (intermediate) untuk masing-masing blok pesan :
   $H_0^{(i)} = a + H_0^{(i-1)}$
   $H_1^{(i)} = b + H_1^{(i-1)}$
   $H_2^{(i)} = c + H_2^{(i-1)}$
   $H_3^{(i)} = d + H_3^{(i-1)}$
   $H_4^{(i)} = e + H_4^{(i-1)}$
   $H_5^{(i)} = f + H_5^{(i-1)}$
   $H_6^{(i)} = g + H_6^{(i-1)}$
   $H_7^{(i)} = h + H_7^{(i-1)}$
6. Output
   Output diperoleh setelah semua blok $M^{(N)}$ 512 bit diproses. Setelah semua langkah pemprosesan dilakukan sejumlah N kali, maka akan didapat 256 bit message digest untuk pesan M yaitu :
   $H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$
7. File Audio
   MPEG-1 audio layer III atau yang lebih dikenal dengan MP3, adalah salah satu dari pengkodean dalam *digital audio* dan juga merupakan format kompresi audio yang memiliki sifat "menghilangkan". Istilah menghilangkan yang dimaksud adalah kompresi audio kedalam format MP3 menghilangkan aspek-aspek yang tidak signifikan pada pendengaran manusia untuk mengurangi besarnya *file audio*[6].

3. HASIL DAN PEMBAHASAN

Keamanan tanda tangan digital merupakan salah satu aspek terpenting yang harus diperhatikan dan diamankan. Karena sering terjadinya pemalsuan tanda tangan yang dilakukan oleh pihak-pihak yang tidak bertanggung jawab. *Audio* merupakan barang bukti digitalyang salahsatunya berasal dari *Handphone*, dalam hal kejahatan *audio* biasanya dimanipulasi untuk menghilangkan bukti-bukti yang ada di dalamnya, oleh sebab itu diperlukan analisis forensik untuk dapat mengetahui file audio tersebut. Adanya perubahan *audio*yang mengalami perubahan dari bentuk aslinya adalah berupa durasi, tambah suara, dan kasih efek suara. Perubahan tersebut dapat diklasifikasikan sebagai tindakan sengaja atau tidak sengaja. Perubahan yang disengaja memiliki tujuan yang jahat dengan memodifikasikan konten atau menghapus hak cipta. Disamping itu, perubahan yang tidak disengajamerupakan konsekuensi dari proses operasional digital, seperti memperbaiki Durasi, mengedit suara.

SHA-256 merupakan suatu metode yang digunakan untuk mengamankan suatu *file audio*, untuk menghindari dari orang-orang yang tidak berhak menerima *file* tersebut. Analisa masalah bertujuan untuk melakukan keamanan persoalan-persoalan yang muncul di sistem, hal ini dilakukan agar suatu proses tidak terjadi kesalahan-kesalahan. Dalam analisa masalah ini, masalah yang akan dianalisa yaitu keamanan tanda tangan digital pada *file audio*. Proses penerapan algoritma SHA-256 dilakukan setelah nilai *file audio* di ekstra dalam bentuk nilai heksadesimal dan selanjutnya dikonversikan dalam bentuk biner. Aplikasi *HexWorkShop* dan Aplikasi *Hasher Pro* digunakan untuk mengujian algoritma SHA-256 dalam hal menjaga *file* audio.

Dalam proses ini seperti dijelaskan dalam analisa yaitu file audio MP3 (via vallen meraih bintang.mp3). Data yang diambil hanya sebanyak 16 byte untuk *hexsa*, cara pengambilan nilai *hexsa*, data audio menggunakan aplikasi *HexWorkShop* dan langkah-langkahnya sebagai berikut:

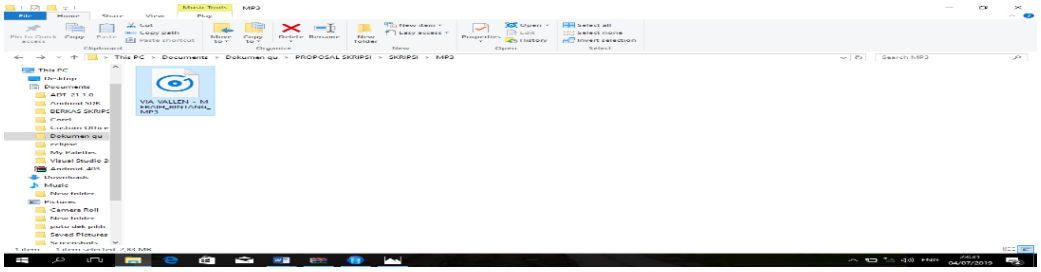
a. Jalankan Aplikasi *HexworkShop*
b. Klik menu *File* pilih *Open*
c. Telusuri dan pilih *file* video di *drive harddisk* dan klik *OK*

Gambar di bawah adalah pemilihan file audio yang akan digunakan untuk contoh kasus yaitu audio via vallen meraih bintang.mp3. Pada contoh kasus penerapan SHA-256 pada penelitan ini menggunakan objek *Audio* yang berformat MP3 dengan spesifikasi, sebagai berikut:

Nama Audio : via vallen meraih bintang
Ekstensi/Type : *.MP3

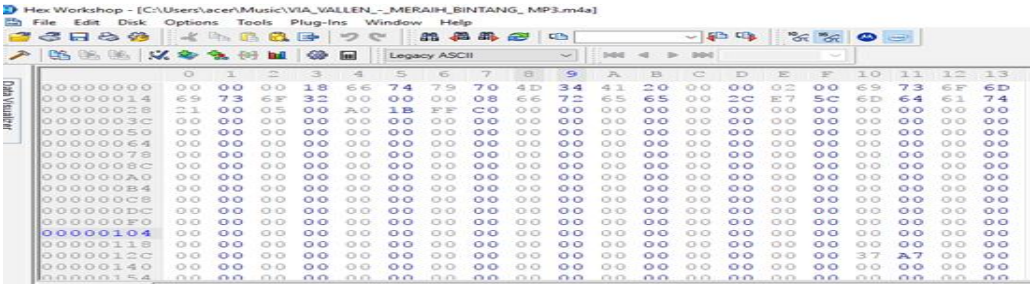
ISSN 2774-4744 (Media Online)
Vol 1, No 2, April 2021
Hal 60-67
https://hostjournals.com/jimat

Kapasitas : 2,83 Megabyte = 2.975.272 bytes
Durasi : 00:03:03
Bit rate : 127 kbps



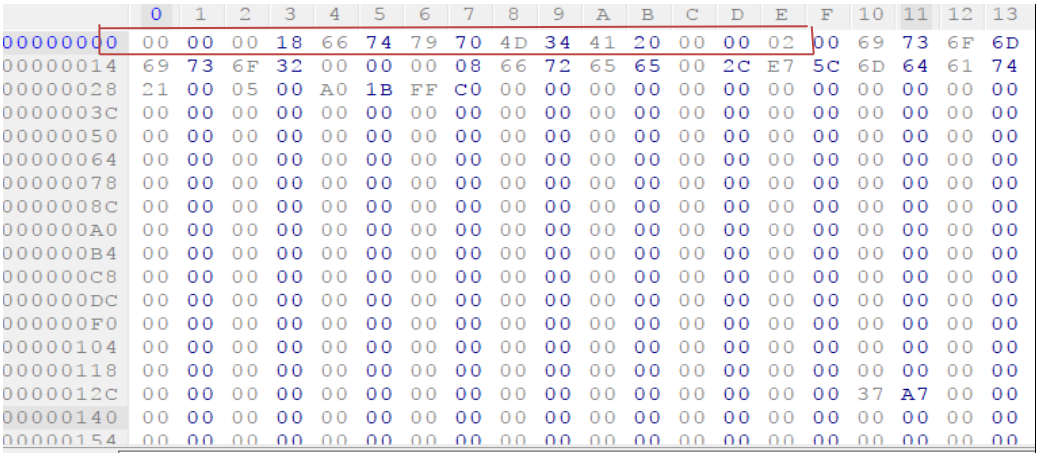
Gambar 1. File Audio di Drive Penyimpanan

Gambar di dibawah ini adalah data heksadesimal dari *file* audio Via Vallen Meraih Bintang mp3 menggunakan aplikasi *HexWorkShop*.



Gambar 2. Data audio *MP3*

Dari data tersebut di ambil sebanyak 16*byte* atau 32 karakter heksadesimal dan dikonversi ke biner, yang berguna untuk mengetahui nilai biner dari bilangan tersebut.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
00000000	00	00	00	18	66	74	79	70	4D	34	41	20	00	00	02	00	69	73	6F	6D
00000014	69	73	6F	32	00	00	00	08	66	72	65	65	00	2C	E7	5C	6D	64	61	74
00000028	21	00	05	00	A0	1B	FF	C0	00	00	00	00	00	00	00	00	00	00	00	00
0000003C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000064	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000078	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000008C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000B4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000C8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000DC	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000104	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000118	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000012C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	37	A7	00	00
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000154	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Gambar 3. Data *byte* audio

Dari gambar data audio di atas diambil sebanyak 16 byte untuk plainteks, yaitu: 000000186674797 04D34412000000200.

1. *Message Padding*

   Dari data yang digunakan sebagai plainteks diubah ke biner.
   Heksadesimal : 00 00 00 18 66 74 79 70 4D 34 41 20 00 00 02 00.
   Data dalam biner :
   00000000 00000000 00000000 00011000 01100110 01110100 01111001 01110000 01001101 00110100 01000001 00100000 00000000 00000000 00000010 00000000

2. Penambahan Panjang Bit

   Data sebanyak 16 byte di atas diketahui 128 bit, untuk mencukupi 512 bit ditambah bit-bit pengganjal (*padding bits*) sebanyak 376 dan 8 bit jumlah pesan semula, karena pada SHA-256 memproses blok-blok bit yang berjumlah 64 blok atau 512 bit. Bit pengganjal yang ditambahi dimulai bit 1 diikuti bit 0 selebihnya hingga urutan bit 464. Untuk 8 bit terakhir menyatakan jumlah karakter dalam notasi biner yaitu 128 =**10000000**

ISSN 2774-4744 (Media Online)
Vol 1, No 2, April 2021
Hal 60-67
https://hostjournals.com/jimat

Tabel 1. Posisi 8 bit terakhir

00000000	00000000	00000000	00011000	01100110	01110100	01111001
01110000	01001101	00110100	01000001	00100000	00000000	00000000
00000010	00000000	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	**10000000**

Bit yang bercetak tebal adalah bit dari plainteks file audio, bit 1 pada urutan ke 128 yang bercetak tebal adalah awal dari bit pengganjal dan dikuti bit-bit 0 dan 8 bit terakhir bercetak tebal adalah bit yang menyatakan jumlah 128.

3. Inisialisasi nilai *hash* awal

   Penyangga SHA-256 terdiri dari delapan yang setiap penyangga memiliki panjang 32 bit, berarti total 256 bit yaitu 8 x 32, dan dalam notasi HEX yaitu :

   $H_0^{(0)}$ = 6a09e667
   $H_1^{(0)}$ = bb67ae85
   $H_2^{(0)}$ = 3c6ef372
   $H_3^{(0)}$ = a54ff53a
   $H_4^{(0)}$ = 510e527f
   $H_5^{(0)}$ = 9b05688c
   $H_6^{(0)}$ = 1f83d9ab
   $H_7^{(0)}$ = 5be0cd19

   Inisialisasi penyangga dalam biner, yaitu :

   A = 6 a 0 9 e 6 6 7
   0110 1010 0000 1001 1110 0110 0110 0111
   B = b b 6 7 a e 8 5
   1011 1011 0110 0111 1010 1110 1000 0101
   C = 3 c 6 e f 3 7 2
   0011 1100 0110 1110 1111 0011 0111 0010
   D = a 5 4 f f 5 3 a
   1010 0101 0100 1111 1111 0101 0011 1010
   E = 5 1 0 e 5 2 7 f
   0101 0001 0000 1110 0101 0010 0111 1111
   F = 9 b 0 5 6 8 8 c
   1001 1011 0000 0101 0110 1000 1000 1100
   G = 1 f 8 3 d 9 a b
   0001 1111 1000 0011 1101 1001 1010 1011
   H = 5 b e 0 C d 1 9
   0101 1011 1110 0000 1100 1101 0001 1001

4. Pemprosesan

   Pemprosesan merupakan bagian inti yang terdiri atas 1 *round* yang mempunyai 64 operasi untuk memproses setiap satu blok pesan 512 bit diperlukan 64 operasi, setiap blok pesan $M^{(0)}$, $M^{(1)}$, $M^{(2)}$,…$M^{(N)}$ dengan N adalah jumlah blok pesan.

Tabel 2. Penambahan Bit

$M^{(0)}$	00000000	00000000	00000000	00011000
$M^{(1)}$	01100110	01110100	01111001	01110000
$M^{(2)}$	01001101	00110100	01000001	00100000
$M^{(3)}$	00000000	00000000	00000010	00000000
$M^{(4)}$	10000000	00000000	00000000	00000000
$M^{(5)}$	00000000	00000000	00000000	00000000
$M^{(6)}$	00000000	00000000	00000000	00000000
$M^{(7)}$	00000000	00000000	00000000	00000000
$M^{(8)}$	00000000	00000000	00000000	00000000
$M^{(9)}$	00000000	00000000	00000000	00000000
$M^{(10)}$	00000000	00000000	00000000	00000000
$M^{(11)}$	00000000	00000000	00000000	00000000
$M^{(12)}$	00000000	00000000	00000000	00000000
$M^{(13)}$	00000000	00000000	00000000	00000000

$M^{(14)}$	00000000	00000000	00000000	00000000
$M^{(15)}$	00000000	00000000	00000000	10000000

Tabel 3. Penambahan Panjang Pesan

$W^{(0)}$	00000000	00000000	00000000	00011000	00000000
$W^{(1)}$	01100110	01110100	01111001	01110000	01100110
$W^{(2)}$	01001101	00110100	01000001	00100000	01001101
$W^{(3)}$	00000000	00000000	00000010	00000000	00000000
$W^{(4)}$	10000000	00000000	00000000	00000000	10000000
$W^{(5)}$	00000000	00000000	00000000	00000000	00000000
$W^{(6)}$	00000000	00000000	00000000	00000000	00000000
$W^{(7)}$	00000000	00000000	00000000	00000000	00000000
$W^{(8)}$	00000000	00000000	00000000	00000000	00000000
$W^{(9)}$	00000000	00000000	00000000	00000000	00000000
$W^{(10)}$	00000000	00000000	00000000	00000000	00000000
$W^{(11)}$	00000000	00000000	00000000	00000000	00000000
$W^{(12)}$	00000000	00000000	00000000	00000000	00000000
$W^{(13)}$	00000000	00000000	00000000	00000000	00000000
$W^{(14)}$	00000000	00000000	00000000	00000000	00000000
$W^{(15)}$	00000000	00000000	00000000	00000000	10000000

$$W_t \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \{\sigma_1(W_t\text{-}2) + (W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16} & 16 \le t \le 63 \end{cases}$$

Dengan fungsi σ_0 dan σ_0 dirumuskan sebagai berikut:

$\sigma_0(x) = ROTR^7(X) \oplus ROTR^{18}(X) \oplus SHR^3(X)$

$\sigma_1(x) = ROTR^{17}(X)) \oplus ROTR^{19}(X) \oplus SHR^{10}(X)$

Untuk t < 16, W adalah t hanya 32 word di atas. Pada saat t ≥ 16 atau untuk W_{16}, W diturunkan secara rekursif dengan rumus berikut :

$W_t = (W_{t-2} \oplus W_{t-7} \oplus W_{t-15} \oplus W_{t-16})ROT\ 1$

$W_{16} = (W_{16-2} \oplus W_{16-7} \oplus W_{16-15} \oplus W_{16-16})ROT\ 1$

Berarti :

$W_t = W_{16-16} = W0$

$W_t = W_{16-15} = W1$

$W_t = W_{16-7} = W9$

$W_t = W_{16-2} = W14$

$W_t = (W_0 \oplus W_1 \oplus W_9 \oplus W_{14})ROT\ 1$

W_0 = 00000000 00000000 00000000 00011000

W_1 = 01100110 01110100 01111001 01110000

W_9 = 00000000 00000000 00000000 00000000

W_{14} = 00000000 00000000 00000000 00000000 ⊕

W_{16} = 01100110 01110100 01111001 01101000

W_{16}= 01100110 01110100 01111001 01101000ROT 1

W_{16} = 11001100111010001111001011010000

Untuk $W_t = W_{17}$ atau selanjutnya, yaitu :

$W_{17} = (W_{17-15} \oplus W_{17-14} \oplus W_{17-6} \oplus W_{17-1})ROT\ 1$

dan untuk seperti itu seterusnya.

Untuk semua W_t di atas akan digunakan untuk proses dengan penyangga dan penambah yang telah ditetapkan dalam SHA-256. Selanjutnya memproses dari persamaan operasi dasar SHA-256, yaitu :

Tabel 4. Hasil dari nilai T_1 sampai T_{63} keseluruhan

Round	A	B	C	D	E	F	G	H
t_0	6a09e667	Bb67ae85	3c6ef372	A54ff53a	510e527f	9b05688c	1f8309ab	5be0cd19
t_1	58e00881	6a09e667	Bb67ae85	3c6ef372	D443949f	510e527f	9b05688c	1f8309ab
t_2	0f820820	58e00881	6a09e667	Bb67ae85	F510e527	D443949f	510e527f	9b05688c
t_3	4b012889	0f820820	58e00881	6a09e667	Fd443949	F510e527	D443949f	510e527f
t_4	10024118	4b012889	0f820820	58e00881	7f510e52	Fd443949	F510e527	D443949f
t_5	500223599	10024118	4b012889	0f820820	9fd44394	7f510e52	Fd443949	F510e527
t_6	D0165109	500223599	10024118	4b012889	27f510e5	9fd44394	7f510e52	Fd443949
t_7	7a821851	D0165109	500223599	10024118	49fd4439	27f510e5	9fd44394	7f510e52

ISSN 2774-4744 (Media Online)
Vol 1, No 2, April 2021
Hal 60-67
https://hostjournals.com/jimat

Round	A	B	C	D	E	F	G	H
t_8	02025809	7a821851	D0165109	500223599	527f510e	49fd4439	27f510e5	9fd44394
t_9	58825094	02025809	7a821851	D0165109	949fd443	527f510e	49fd4439	27f510e5
t_{10}	12060009	58825094	02025809	7a821851	E527f510	949fd443	527f510e	49fd4439
t_{11}	48204119	12060009	58825094	02025809	3949FD44	E527F510	949FD443	527F510E
t_{12}	02027001	48204119	12060009	58825094	0E527F51	3949FD44	E527F510	949FD443
t_{13}	50085001	02027001	48204119	12060009	43949FD4	0E527F51	3949FD44	E527F510
t_{14}	1104F119	50085001	02027001	48204119	10E527F5	43949FD4	0E527F51	3949FD44
t_{15}	40807011	1104F119	50085001	02027001	443949FD	10E527F5	43949FD4	0E527F51
t_{16}	16580349	40807011	1104F119	50085001	510E527F	443949FD	10E527F5	43949FD4
t_{17}	12048E4B	16580349	40807011	1104F119	D443949F	510E527F	443949FD	10E527F5
t_{18}	15E08AC0	12048E4B	16580349	40807011	F510E527	D443949F	510E527F	443949FD
t_{19}	14284068	15E08AC0	12048E4B	16580349	FD443949	F510E527	D443949F	510E527F
t_{20}	004A1F40	14284068	15E08AC0	12048E4B	7F510E52	FD443949	F510E527	D443949F
t_{21}	84408802	004A1F40	14284068	15E08AC0	9FD44394	7F510E52	FD443949	F510E527
t_{22}	84008500	84408802	004A1F40	14284068	27F510E5	9FD44394	7F510E52	FD443949
t_{23}	A4402900	84008500	84408802	004A1F40	49FD4439	27F510E5	9FD44394	7F510E52
t_{24}	4A242810	A4402900	84008500	84408802	527F510E	49FD4439	27F510E5	9FD44394
t_{25}	4A00718D	4A242810	A4402900	84008500	949FD443	527F510E	49FD4439	27F510E5
t_{26}	420221E1	4A00718D	4A242810	A4402900	E527F510	949FD443	527F510E	49FD4439
t_{27}	48A07A11	420221E1	4A00718D	4A242810	3949FD44	E527F510	949FD443	527F510E
t_{28}	50001040	48A07A11	420221E1	4A00718D	0E527F51	3949FD44	E527F510	949FD443
t_{29}	14825840	50001040	48A07A11	420221E1	43949FD4	0E527F51	3949FD44	E527F510
t_{30}	5100FD00	14825840	50001040	48A07A11	10E527F5	43949FD4	0E527F51	3949FD44
t_{31}	18427000	5100FD00	14825840	50001040	443949FD	10E527F5	43949FD4	0E527F51
t_{32}	4F105B10	18427000	5100FD00	14825840	510E527F	443949FD	10E527F5	43949FD4
t_{33}	02867AC9	4F105B10	18427000	5100FD00	D443949F	510E527F	443949FD	10E527F5
t_{34}	54027D20	02867AC9	4F105B10	18427000	F510E527	D443949F	510E527F	443949FD
t_{35}	042A4958	54027D20	02867AC9	4F105B10	FD443949	F510E527	D443949F	510E527F
t_{36}	44007938	042A4958	54027D20	02867AC9	7F510E52	FD443949	F510E527	D443949F
t_{37}	C040858B	44007938	042A4958	54027D20	9FD44394	7F510E52	FD443949	F510E527
t_{38}	58001560	C040858B	44007938	042A4958	27F510E5	9FD44394	7F510E52	FD443949
t_{39}	00006011	58001560	C040858B	44007938	49FD4439	27F510E5	9FD44394	7F510E52
t_{40}	4E024840	00006011	58001560	C040858B	527F510E	49FD4439	27F510E5	9FD44394
t_{41}	88480905	4E024840	00006011	58001560	949FD443	527F510E	49FD4439	27F510E5
t_{42}	0AE004E1	88480905	4E024840	00006011	E527F510	949FD443	527F510E	49FD4439
t_{43}	58480D09	0AE004E1	88480905	4E024840	3949FD44	E527F510	949FD443	527F510E
t_{44}	520A7108	58480D09	0AE004E1	88480905	0E527F51	3949FD44	E527F510	949FD443
t_{45}	54827440	520A7108	58480D09	0AE004E1	43949FD4	0E527F51	3949FD44	E527F510
t_{46}	81288C08	54827440	520A7108	58480D09	10E527F5	43949FD4	0E527F51	3949FD44
t_{47}	0C107D00	81288C08	54827440	520A7108	443949ED	10E527F5	43949FD4	0E527F51
t_{48}	44105241	0C107D00	81288C08	54827440	510E527F	443949ED	10E527F5	43949FD4
t_{49}	50865201	44105241	0C107D00	81288C08	D443949F	510E527F	443949ED	10E527F5
t_{50}	000363B1	50865201	44105241	0C107D00	F510E527	D443949F	510E527F	443949ED
t_{51}	440078F0	000363B1	50865201	44105241	FD443949	F510E527	D443949F	510E527F
t_{52}	40481C09	440078F0	000363B1	50865201	7F510E52	FD443949	F510E527	D443949F
t_{53}	4840858B	40481C09	440078F0	000363B1	9FD44394	7F510E52	FD443949	F510E527
t_{54}	1800A429	4840858B	40481009	440078F0	27F510E5	9FD44394	7F510E52	FD443949
t_{55}	AC003849	1800A429	4840858B	40481009	49FD4439	27F510E5	9FD44394	7F510E52
t_{56}	42420111	AC003849	1800A429	4840858B	527F510E	49FD4439	27F510E5	9FD44394
t_{57}	96400014	42420111	AC003849	1800A429	949FD443	527F510E	49FD4439	27F510E5
t_{58}	42E00071	96400014	42420111	AC003849	E527F510	949FD443	527F510E	49FD4439
t_{59}	48484D29	42E00071	96400014	42420111	3949FD44	E527F510	949FD443	527F510E
t_{60}	526A0048	48484D29	42E00071	96400014	0E527F51	3949FD44	E527F510	949FD443
t_{61}	9C188D01	526A0048	48484D29	42E00071	43949FD4	0E527F51	3949FD44	E527F510
t_{62}	910A8410	9C188D01	526A0048	48484D29	10E527F5	43949FD4	0E527F51	3949FD44
t_{63}	0848A950	910A8410	9C188D01	526A0048	443949FD	10E527F5	43949FD4	0E527F51

Menghitung nilai *hash* inisialisasi prantara (intermediate) untuk masing-masing blok pesan. Selanjutnya setelah didapatkan a, b, c, d, e, f, g dan h untuk t_{63}, maka nilai t_{63}digunakan untuk mendapatkan *cipher* atau biasa disebut *digest* dalam SHA-256 dengan cara di XOR dengan nilai a, b, c, d, e, f, g dan h awal (penyangga).

T_{63}	0848A950 910A8410 9C188D01 526A0048 443949FD 10E527F5 43949FD4 0E527F51
a, b, c, d, e, f, g, h	6A09E6C7 BB67AE85 3C6EF372 A54FF53A 510E527F 9B05688C 1F83D9AB 5BE0CD19
Hasil Akhirnya	EA284C0F E84AC318 7C10FA00 50E26A22 2206E088 0A316CF7 7EC92825 C9CFAD0A

ISSN 2774-4744 (Media Online)
Vol 1, No 2, April 2021
Hal 60-67
https://hostjournals.com/jimat

Tabel 5. Hasil Nilai SHA-256

EA284C0F	2206E088
E84AC318	0A316CF7
7C10FA00	7EC92825
50E26A22	C9CFAD0A

Concate h0, h1, h2, h3, h4, h5, h6 dan h7. Hasil *concate* sebanyak 64 karakter tersebut menjadi cipherteks atau disebut *message digest*.
Output diperoleh setelah semua blok $M^{(N)}$ 512 bit diproses. Setelah semua langkah pemprosesan dilakukan sejumlah N kali, maka akan didapat 256 bit *message digest* untuk pesan M yaitu :
EA284C0F E84AC318 7C10FA00 50E26A22 2206E088 0A316CF7 7EC92825 C9CFAD0A

Tabel 6. Hasil Perbandingan audio

Parameter	Audio Asli	Audio Manipulasi	Nilai *Hash* Asli	NIlai *Hash* Manipulasi	Ket
Melakukan perubahan pada durasi audio	 Size : 2.83 MB Durasi : 03:03 Bit Rate : 127kbps	 Size : 869 KB Durasi : 00:55 Bit Rate : 128kbps	27131476 2a675185 342cf372 A55fb132 1b1368cc Db1368cc 01d6cece 51e0fd19	8433B8A4 61F6da70 dd7698A5 09596B61 fd6ECB91 A334d28b B6ae648c 8dd04e7b	Terdeteksi

Dari hasil operasi yang dilakukan dengan menggunakan metode SHA-256 untuk mendeteksi file audio maka dapat disimpulkan bahwa metode SHA 256 dapat mendeteksi keaslian dari file audio tersebut. Perbandingan yang diperoleh terdapat pada nilai *hash* yang diperoleh dari audio asli dengan audio yang dimanipulasi perubahan telah dilakukan pada *file audio* baik dari durasi maupun pergantian ektensi audio. Algoritma SHA 256 dapat mendeteksi dan menampilkan nilai *hash* masing-masing audio.

4. KESIMPULAN

Berdasarkan penelitian yang dilakukan dapat disimpulkan penerapan metode SHA-256 dapat menguji file audio pada aplikasi *Hasher Pro*. SHA-256 salah satu proses perhitungan nilai-nilai *pixe*l dari pada file audio 16 *byte* 32 karakter. Nilai *hash* dari SHA-256 sebuah file yang dihasilkan dari nilai 16 *byte* terakhir dan dari nilai file audio dalam bentuk heksadesimal. Aplikasi Hasher Pro digunakan untuk melakukan pembuktian atau pengujian untuk keamanan tanda tangan digital pada file audio tersebut.

REFERENCES

[1] A. G. Tammam, "Fungsi Hash dan Algoritma SHA-256," Keamanan Komputer, p. 5, 2016.
[2] S. M. Egi Cahyo Prabowo Irawan Afrianto, "Implementasi Aplikasi Digital Signature Menggunakan Fungsi Hash Algoritma SHA-256 dan RSA di Badan Pertanahan Nasional Kota Cimahi," Tekni Informatika, pp. 1-8, 2014.
[3] M. Arinda Firdianti, Arinda Firdianti, Implementasi manajemen Berbasis Sekolah dalam Meningkatkan Prestasi Belajar Siswa., Kota Yogyakarta 55241: CV. GRE PUBLISHING Jl. Kelurahan Karang Waru Lor TR II/22IE , Oktober 2018.
[4] S. Emy Setyaningsih, Kriptografi dan implementasinya menggunakan Matlab, Yogyakarta: CV. ANDI , 2015.
[5] A. Rifki Sadikin, Kriptografi untuk keamanan jaringan dalam implementasinya Bahasa Java, Yogyakarta: Rifki Sadikin, Andi, 2012.
[6] I. J. Kusuma, "Analisis Teknik Steganografi pada Audio MP3 menggunakan Metode Parity coding dan Enkripsi Chiper Transposition," Jurnal Elekronik Sistem Informasi dan Komputer Sekolah Tinggi Manajemen Informatika dan Komputer (STMIK) Bina Mulia, vol. 3, p. 4, 2017.
[7] M. Syafriadi, "AnalisisKecepatan dan Keamanan Algoritma Secure Hash Algorithm 256 (SHA-256) Untuk Otentikasi Pesan Teks," pp. 2-5, 2016.