BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 2, No 3, Agustus 2022
DOI 10.47065/bulletincsr.v2i3.140 | Hal 73−84
https://hostjournals.com/bulletincsr

Implementasi Metode Secure Hash Algorithm (SHA-1) Untuk Mendeteksi Orisinalitas File Audio

Rezky M. Nasution

Program Studi Teknik Informatika Universitas Budi Darma Medan, Indonesia
Email: rezkynst@gmail.com
Email Penulis Korespondensi: rezkynst@gmail.com

Abstrak−File audio merupakan suatu sarana informasi dari satu orang ke orang lain. File audio sangat rentan terhadap penipuan, penyadapan maupun pencurian data oleh pihak-pihak yang tidak bertanggung jawab. Demi menjaga keamanan File audio dapat dilakukan dengan pemanfaatan teknik kriptografi. Kriptografi merupakan salah satu metode pengamanan data yang dapat digunakan menjaga keaslian data, kerahasiaan data, serta keaslian pengiriman data. SHA merupakan singkatan *Secure Hash Algorithm* merupakan fungsi *hash standard* yang dipublikasi oleh NIST *(National Institut e of Standard and Technology)*. SHA diterbitkan dengan ukuran *digest* 512 bit. SHA-1 akan menghasilkan keluaran sebanyak 160 bit dari string tersebut dan *string* keluaran itu disebut *message digest*. Panjang jarak *message digest* dapat berkisar antara 160 sampai 512 bit tergantung algoritmanya. Penelitian ini menguraikan proses pengamanan untuk mendeteksi keaslian file audio dengan menggunakan Metode SHA-1 dalam bentuk pendeteksian agar audio yang bersifat rahasia yang dikirim melalui telekomunikasi umum tidak dapat dirubah atau dimodifikasi oleh orang yang tidak berhak atau orang yang tidak berkepentingan. Hal ini dilakukan sebagai upaya untuk meminimalisir tindakan-tindakan penipuan, hoax, ataupun penyalahgunaan file audio.

Kata Kunci : *Kriptografi; File Audio; SHA-1*.

Abstract−Audio file is a means of information from one person to another. Audio files are very vulnerable to fraud, eavesdropping or data theft by irresponsible parties. In order to maintain the security of audio files, this can be done by using cryptographic techniques. Cryptography is one of the data security methods that can be used to maintain data authenticity, data confidentiality, and the authenticity of data transmission. SHA stands for Secure Hash Algorithm is a standard hash function published by NIST (National Institute of Standards and Technology). SHA is published with a digest size of 512 bits. SHA-1 will output 160 bits of the string and the output string is called a message digest. The length of the message digest can range from 160 to 512 bits depending on the algorithm. This study describes the security process for detecting the authenticity of audio files using the SHA-1 method in the form of detection so that confidential audio sent via public telecommunications cannot be changed or modified by unauthorized persons or unauthorized persons. This is done as an effort to minimize acts of fraud, hoaxes, or misuse of audio files.

Keywords: Cryptography; Audio Files; SHA-1

1. PENDAHULUAN

Pada era teknologi informasi yang berkembang sangat pesat, penggunaan *file audio* sudah banyak diterapkan secara digital melalui orisinalitas *file audio*. seiring munculkannya kebutuhan *orisinalitas* suatu data atau berkas yang digunakan secara digital. Saat ini, pemanfaatan mendeteksi orisinalitas sudah banyak diterapkan pada distribusi perangkat lunak, transaksi keuangan, pengiriman berkas.

Keaslian *file audio* pada saat ini menjadi hal yang sangat penting dan terus berkembang. Karena itu *audio* dijadikan sebagai sumber utama informasi. Berbagai *software editing audio* menyulitkan seseorang untuk membedakan antara *audio* asli atau *audio* palsu. *Audio* sering kali disalah gunakan oleh pihak- pihak tertentu untuk memanipulasi sebuah rekaman dengan memalsukan *file audio* yang bertujuan untuk menghilangkan bukti-bukti yang ada didalamnya.

Untuk membedakan *audio* asli atau *audio* palsu maka diperlukan pendeteksian terhadap *audio* tersebut menggunakan kriptografi. Kriptografi merupakan salah satu metode pengamanan *file* yang dapat digunakan menjaga keaslian *file*. Metode ini bertujuan agar *file* yang bersifat rahasia yang dikirim melalui telekomunikasi umum tidak dapat dirubah atau dimodifikasi oleh orang yang tidak berhak atau orang yang tidak berkepentingan.

Pada saat ini banyak algoritma yang digunakan untuk mengamankan, salah satunya ialah *Secure Hash Algorithm* (SHA), algoritma ini merupakan fungsi *hash* yang bekerja satu arah. Sebelumnya pada tahun 2014, hasil penelitian yang dilakukan oleh Komang Aryasa dan Yesaya Tommy Paulus membuktikan bahwa SHA-1 dikatakan aman karena proses SHA-1 dihitung secara infisibel untuk mencari *string* yang sesuai menghasilkan *message digest*. SHA-1 keluaran sebanyak 160 bit. blok string mempunyai 512 bit dimana dapat dilakukan dengan 16 urutan sebesar 32 bit[1].

Pada tahun 2018 Kiki Dwi Wandani dan Sinar Sinurat melakukan sebuah penelitian, yaitu mengamankan *file vidio* menggunakan algoritma SHA-1 Untuk mengatasi masalah ini, *file video* tersebut mempunyai 20 *byte* dan 40 karakter data bagian depan. SHA-1 menggunakan variabel yang telah ditetapkan seperti bit-bit penambah, bilangan penyangga, inisialisasi MD serta putaran yang dilakukan untuk mendapat *digest* sebanyak 80 kali[2].

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

ISSN 2774-3659 (Media Online)
Vol 2, No 3, Agustus 2022
DOI 10.47065/bulletincsr.v2i3.140 | Hal 73−84
https://hostjournals.com/bulletincsr

pengirim dapat disampaikan kepada penerima dengan aman. Kriptografi bertujuan menjaga kerahasiaan informasi yang terkandung dalam data sehingga informasi tersebut tidak dapat diketahui oleh pihak yang tidak sah[3]. Kriptografi adalah ilmu yang bersandarkan pada teknik matematika untuk berurusan dengan keamanan informasi seperti kerahasiaan, keutuhan data dan otentikasi entitas[4].

Gambar 1 . Area bidang kriptologi

2.2 Fungsi *Hash*

Fungsi *hash* sering disebut dengan fungsi satu arah *(one-way function), message digest*, *fingerprint*, fungsi kompresi dan *message authentication code* (MAC), merupakan suatu fungsi matematika yang mengambil masukan panjang variabel dan mengubahnya ke dalam urutan biner dengan panjang yang tetap. Sidik jari pada pesan merupakan suatu tanda bahwa pesan tersebut benar-benar berasal dari orang yang diinginkan[5].

Fungsi *hash (hash function)* merupakan salah satu teknik kriptografi untuk menghitung nilai unik dari sebuah data. Fungsi *hash* dapat diibaratkan sebagai sidik jari elektronik *(digital fingerprint)* dari informasi elektronik. Sidik jari elektronik berguna untuk menentukan orisinalitas sebuah dokumen elektronik. Dua dokumen elektronik yang berbeda akan memiliki nilai *hash* yang berbeda, itulah sebabnya apabila sebuah dokumen telah mengalami perubahan, maka nilai *hash* juga akan berubah[6].

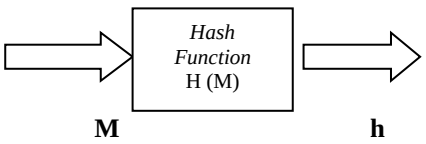
Fungsi *hash* adalah sebuah fungsi yang memasukannya adalah sebuah pesan dan keluaran sebuah sidik pesan *(message fingerprint)*. Sidik pesan sering juga disebut *message digest*, fungsi *hash* dapat digunakan untuk mewujudkan beberapa layanan keamanan jaringan misalnya untuk keutuhan data dan otentikasi pesan[4].

2.2.1 Fungsi *Hash* Satu Arah

Fungsi *hash* satu arah *(One-way hash)* adalah pesan yang sudah diubah menjadi *message digest* tidak dapat dikembalikan lagi menjadi pesan semula *(irreversible)*[3].
Fungsi *hash* satu arah mempunyai sifat sebagai berikut :

1. Fungsi *H* dapat diterapkan pada blok data berukuran beberapa saja.
2. *H* menghasilkan nilai *(h)* dengan panjang tetap *(fixed length output).*
3. *H(x)* mudah dihitung untuk nilai x yang diberikan.
4. Untuk setiap *h* yang dihasilkan, tidak mungkin dikembalikan nilai x sedemikian sehingga $H(x) = h$.
5. Untuk setiap x yang diberikan, tidak mungkin dicari $y \neq x$ sedemikian sehingga $H(y) = H(x)$.
6. Tidak mungkin dicari pasangan x dan y sedemikian sehingga $H(x) = H(y)$.

Pesamaan fungsi hash satu arah :
$hi = H(Mi, hi - 1)$
Skema fungsi *hash* ditunjukkan pada gambar 2.2 di bawah ini :



Gambar 2. Fungsi *hash* satu arah

2.2.2 *Secure Hash Algorithm* (SHA-1)

SHA merupakan singkatan *Secure Hash Algorithm* merupakan fungsi *hash standard* yang dipublikasi oleh NIST *(National Institute of Standard and Technology)*, (NIST, 1995a). SHA diterbitkan dengan beberapa versi di antaranya SHA-1 dengan ukuran *digest* 160 bit[4].

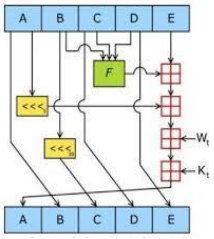
Cara kerja kriptografi algoritma SHA-1 adalah menerima input berupa pesan dengan ukuran sembarang dan menghasilkan *message digest* yang memiliki panjang 160 bit. Langkah-langkah pembuatan *message digest* dengan algoritma SHA-1 adalah sebagai berikut[7]:

1. *Input* Pesan yang akan di *hash* SHA-1.
2. Ubah pesan menjadi deretan biner.
3. Penambahan bit-bit pengganjal, yaitu dengan menambahkan pesan dengan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 mod 512.
4. Penambahan nilai panjang pesan semula, yaitu pesan ditambah lagi dengan 64 bit yang representasi *biner* dari panjang pesan asli.

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 2, No 3, Agustus 2022
DOI 10.47065/bulletincsr.v2i3.140 | Hal 73−84
https://hostjournals.com/bulletincsr

5. Inisialisasi Nilai *Hash*, pada algoritma SHA-1 nilai *hash*, H(0) terdiri dari 5 *words* dengan besar 32 bit dalam notasi *hexadecimal*.
6. *Output* nilai *hash* adalah nilai terakhir dari *buffer*.

Berdasarkan tahapan yang ada pada Fungsi *Hash* SHA-1, maka skema Fungsi *Hash* SHA-1 dapat dilihat pada gambar berikut ini :



Gambar 3. Skema Fungsi *Hash* SHA-1

SHA-1 menerima masukan berupa *string* dengan ukuran maksimum 2^{64} bit. Untuk setiap *string*, SHA-1 akan menghasilkan keluaran sebanyak 160 bit dari string tersebut dan *string* keluaran itu disebut *message digest*. Panjang jarak *message digest* dapat berkisar antara 160 sampai 512 bit tergantung algoritmanya.

SHA-1 dikatakan aman karena proses SHA-1 dihitung secara infisibel untuk mencari *string* yang sesuai untuk menghasilkan *message digest* atau dapat juga digunakan untuk mencari dua *string* yang berbeda yang akan menghasilkan *message digest* yang sama. Pada SHA-1 masing-masing *blokstring* mempunyai 512 bit dimana dapat dilakukan dengan 16 urutan sebesar 32 bit. Tujuan pengisian *string* adalah untuk menghasilkan total dari string yang diisi menjadi perkalian dari 512 *bits.* Beberapa hal yang dilakukan dalam pengisian *string*[1]:

a. Panjang dari *string*, M adalah k bit dimana panjang $k < 2^{64}$. Tambahkan bit "1" pada akhir string. Misalkan string yang asli adalah "01010000" maka setelah diisi menjadi "010100001".
b. Tambahkan bit "0", angka bit "0" tergantung dari panjang string. Misalnya : *string* asli yang merupakan bit *string* : abcde
   01100001 01100010 01100011 01100100 01100101.
   Setelah langkah (a) dilakukan
   01100001 01100010 01100011 01100100 01100101.
   Panjang k = 40 dan angka bit di atas adalah 41 dan 407 ditambah bit "0"
   (448 – (40 + 1) = 407). Kemudian diubah dalam *hex* :
   61626364 65800000 00000000 00000000
   00000000 00000000 00000000 00000000
   00000000 00000000 00000000 00000000
   00000000 00000000
c. Untuk memperoleh 2 kata dari k, angka bit dalam *string* asli yaitu jika $k < 2^{32}$ maka kata pertama adalah semua bit "0". Maka gambaran dari 2 kata dari
   k = 40 dalam *hex* adalah
   00000000 00000028
   61626364 65800000 00000000 00000000
   00000000 00000000 00000000 00000000
   00000000 00000000 00000000 00000000
   00000000 00000000 00000000 00000028

SHA-1 menggunakan urutan fungsi logika yang dilambangkan dengan f_0, f_1,…, f_{79}. Untuk masing-masing f_t, dimana $0 \le t < 79$ akan menghasilkan *output* sebanyak 32 bit. Fungsinya adalah sebagai berikut :

$$f_t(B, C, D) = \begin{cases} (B \wedge C) \vee (\neg B \wedge D) & 0 \le t \le 19 \\ B \oplus C \oplus D & 20 \le t \le 39 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & 40 \le t \le 59 \\ B \oplus C \oplus D & 60 \le t \le 79 \end{cases}$$

$\oplus$ = **fungsi XOR**

Konstanta kata yang digunakan pada SHA-1 yang disimbolkan secara berurutan dari K(0), K(1),…, K(79) dalam bentuk hex seperti pada fungsi 02 berikut :

$$K_t = \begin{cases} 5A827999 & 0 \le t \le 19 \\ 6ED9EBA1 & 20 \le t \le 39 \\ 8F1BBCDC & 40 \le t \le 59 \\ CA62C1D6 & 60 \le t \le 79 \end{cases}$$

Algoritma SHA-1 dapat diringkas sebagai berikut:

1. Penghitungan menggunakan dua *buffer* dimana masing-masing *buffer* terdiri dari lima sebesar 32 bit kata dan urutan 80 juga sebesar 32 bit kata. Lima kata pertama pada *buffer* kata diberi nama A, B, C, D, E sedangkan lima kata

ISSN 2774-3659 (Media Online)
Vol 2, No 3, Agustus 2022
DOI 10.47065/bulletincsr.v2i3.140 | Hal 73−84
https://hostjournals.com/bulletincsr

kedua diberi nama H_0, H_1, H_2, H_3, dan H_4. Kemudian pada 80 kata yang berurutan diberi nama W0, W1,…, W79 dan pada penghitungan ini juga memakai sebuah variabel sementara.

2. Lakukan pengisian *string*, M dan kemudian kirimkan *string* tersebut ke dalam N 512 bit *blok string*, $M^{(1)}$, $M^{(2)}$, …, $M^{(n)}$, Caranya : 32 bit pertama dari blok *string* ditunjukkan ke $M_0^{(i)}$, lalu 32 bit berikutnya adalah $M_1^{(i)}$ dan selanjutnya berlaku hingga $M_{15}^{(i)}$.
3. Inisialisasi Nilai *Hash* (dalam bentuk hex) :
   H_0 = 67452301
   H_1 = EFCDAB89
   H_2 = 98BADCFE
   H_3 = 10325476
   H_4 = C3D2E1F0
4. Lakukan proses M_1, M_2, …, M_n dengan cara membagi M_i ke dalam 16 kata W_0, W_1, …, W_{15} dimana W_0 merupakan *left most*.
5. Hitung : For t = 16 to 79
   $W_t = S^1(W_{t-3} W_{t-8} W_{t-14} W_{t-16})$
6. Inisialisasi 5 variabel A, B, C, D, dan E dengan nilai *Hash* :
   A = H_0
   B = H_1
   C = H_2
   D = H_3
   E = H_4
   Hitung : For t = 0 to 79
   TEMP = S^5(A) ⊕ f_t (B,C,D) ⊕ E ⊕ Wt ⊕ Kt
   E = D
   D = C
   C = S30 (B)
   B = A
   A = TEMP

g. Hitung Nilai *Hash* :
   H0 = H0 ⊕ A
   H1 = H1 ⊕ B
   H2 = H2 ⊕ C
   H3 = H3 ⊕ D
   H4 = H4 ⊕ E

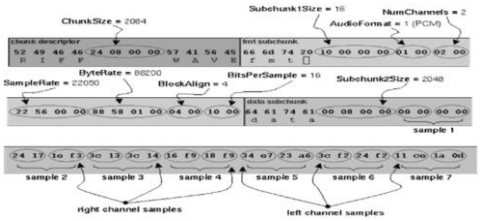
Hasil dari *message digest* sebesar 160 bit dari string, M adalah: H_0 H_1 H_2 H_3 H_4.

2.3 File Audio

Audio (Suara) adalah fenomena fisik yang dihasilkan oleh getaran suatu benda yang berupa sinyal analog dengan amplitudo yang berubah secara kontiniu terhadap satuan waktu yang disebut frekuensi[8].

2.3.1 Struktur Data Pada *File Audio*

Struktur data pada *file audio* berbeda-beda tergantung format *audio*nya, Misalnya *file audio* memiliki struktur seperti pada gambar yaitu :

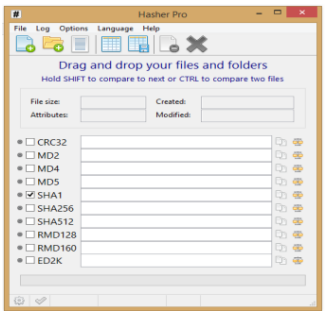


Gambar 4. Struktur *File Audio*

2.4 Aplikasi *Hasher Pro*

Hasher Pro adalah suatu aplikasi yang digunakan untuk memverifikasi memverifikasi integritas *file* menggunakan beragam algoritma seperti CRC32, MD2, MD4, MD5, SHA1, SHA256, SHA512, RipeMD128, RipeMD160, ED2K. *Hasher Pro* merupakan solusi yang tepat sebagai media untuk perbandingan yang memberikan banyak fitur dan kemudahan. Dengan fungsi perbandingan *hash* dan *file* yang mudah, seperti memegang *SHIFT* untuk membandingkan dengan *file* berikutnya atau *CTRL* untuk membandingkan dua *file* dengan membuat suatu kode berdasakan algoritma yang digunakan[9]. Adapun gambar dari aplikasi *Hasher Pro* dengan menggunakan metode SHA-1 :

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 2, No 3, Agustus 2022
DOI 10.47065/bulletincsr.v2i3.140 | Hal 73−84
https://hostjournals.com/bulletincsr



Gambar 5. Aplikasi *Hasher Pro*

3. HASIL DAN PEMBAHASAN

3.1 Analisa Mendeteksi Orisinalitas *File Audio*

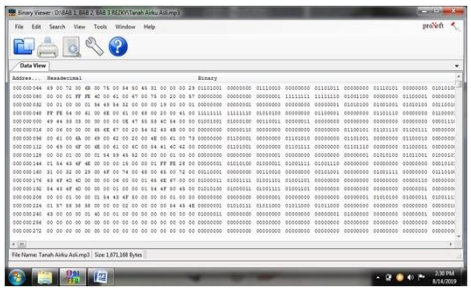
Masalah yang akan di analisa adalah bagaimana melakukan mendeteksi keaslian suatu *audio* dengan melakukan proses pengecekan atau perbandingan suatu keaslian dari *audio* asli dengan *audio* yang telah dirubah atau di edit. *Audio* merupakan barang bukti digital yang salah satunya berasal dari *Handphone*, dalam hal kejahatan *audio* biasanya dimanipulasi untuk menghilangkan bukti-bukti yang ada di dalamnya, oleh sebab itu diperlukan analisis forensik untuk dapat mendeteksi keaslian *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.

3.2 Analisa Mendeteksi Keaslian *Audio* Dengan Menerapkan Metode SHA-1

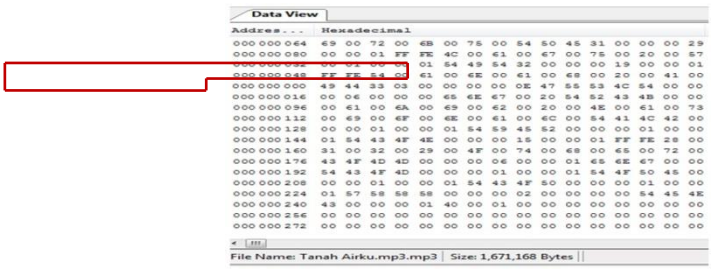
Fungsi *hash* mempunyai peroses jika ia dipanggil dua kali oleh masukan yang benar-benar sama (sebagai misal, string yang mengandung sekuen karakter yang sama), maka ia haruslah memberi hasil yang sama pula.

Contoh kasus Asli dalam proses ini yaitu *file audio* MP3 (Tanah Airku). Data yang di ambil sebanyak 24 *byte*, cara pengambilan nilai *hexadecimal file audio* menggunakan aplikasi *Binary Viewer*, seperti dibawah ini.



Gambar 6. Data *Audio Asli*

Dari data tersebut diambil sebanyak 24 *byte* atau 48 karakter heksadesimal dan dikonversi ke biner sebagai sampel metode, yang berguna untuk mengetahui nilai biner dari bilangan tersebut.



Gambar 7. Sampel Data

Dari gambar data *audio* di atas diambil sebanyak 24 *byte* untuk plainteks, yaitu :

1. Heksadesimal :
   49 44 33 03 00 00 00 00 0E 47 55 53 4C 54 00 00 00 06 00 00 00 65 6E 67
   Data dalam biner :
   01001001 01000100 00110011 00000011 00000000 00000000 00000000 00000000 00001110 01000111 01010101 01010011 01001100 01010100 00000000 00000000 00000000 00000110 00000000 00000000 00000000 01100101 01101110 01100111

ISSN 2774-3659 (Media Online)
Vol 2, No 3, Agustus 2022
DOI 10.47065/bulletincsr.v2i3.140 | Hal 73−84
https://hostjournals.com/bulletincsr

Data sebanyak 24 *byte* di atas diketahui 192 bit, untuk mencukupi 512 bit ditambah bit-bit pengganjal (*padding bits*) sebanyak 312 dan 8 bit jumlah pesan semula, karena pada SHA-1 memproses blok-blok bit yang berjumlah 64 blok atau 512 bit. Bit pengganjal yang ditambahi dimulai bit 1 diikuti bit 0 selebihnya. Untuk 8 bit terakhir menyatakan jumlah karakter dalam notasi biner yaitu 192 = 11000000.

Berikut ini adalah urutan blok-blok bit setelah ditambahkan :

01001001 01000100 00110011 00000011 00000000 00000000 00000000 00000000 00001110 01000111 01010101 01010011 01001100 01010100 00000000 00000000 00000000 00000110 00000000 00000000 00000000 01100101 01101110 01100111 10000000 00000000 **11000000**

Bit yang bercetak tebal adalah bit dari plainteks *file audio*, bit 1 pada urutan ke 193 yang bercetak tebal adalah awal dari bit pengganjal dan dikuti bit-bit 0 dan 8 bit terakhir bercetak tebal adalah bit yang menyatakan jumlah 192.

2. Inisialisasi Penyangga Nilai *Hash Message Digest* (MD)

Penyangga SHA-1 terdiri dari 5 yang setiap penyangga memiliki panjang 32 bit, berarti total 160 bit yaitu 5 x 32, dan dalam notasi HEX yaitu:

A = 67452301
B = EFCDAB89
C = 98BADCFE
D = 10325476
E = C3D2E1F0

Inisialisasi penyangga dalam biner, yaitu :

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

3. Pengolahan Pesan Dalam Blok Berukuran 512 Bit

Semua bit plainteks yang berjumlah 512 bit dibagi 16 blok yang mana setiap satu blok berisi 32 bit bagian. Berikut adalah 16 blok bit tersebut :

M_0 = 01001001 01000100 00110011 00000011
M_1 = 00000000 00000000 00000000 00000000
M_2 = 00001110 01000111 01010101 01010011
M_3 = 01001100 01010100 00000000 00000000
M_4 = 00000000 00000110 00000000 00000000
M_5 = 00000000 01100101 01101110 01100111
M_6 = 10000000 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 11000000

Pertama, blok pesan 32 bit W diturunkan untuk setiap langkah t dari t blok pesan 512 bit M_j menggunakan jadwal pesan.

W_0 = 01001001 01000100 00110011 00000011
W_1 = 00000000 00000000 00000000 00000000
W_2 = 00001110 01000111 01010101 01010011
W_3 = 01001100 01010100 00000000 00000000
W_4 = 00000000 00000110 00000000 00000000
W_5 = 00000000 01100101 01101110 01100111
W_6 = 10000000 00000000 00000000 00000000

ISSN 2774-3659 (Media Online)
Vol 2, No 3, Agustus 2022
DOI 10.47065/bulletincsr.v2i3.140 | Hal 73−84
https://hostjournals.com/bulletincsr

W_7 = 00000000 00000000 00000000 00000000
W_8 =00000000 00000000 00000000 00000000
W_9 = 00000000 00000000 00000000 00000000
W_{10} = 00000000 00000000 00000000 00000000
W_{11} = 00000000 00000000 00000000 00000000
W_{12} = 00000000 00000000 00000000 00000000
W_{13} = 00000000 00000000 00000000 00000000
W_{14} = 00000000 00000000 00000000 00000000
W_{15} = 00000000 00000000 00000000 11000000

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

$$W_t = (W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}) \text{ ROT } 1$$

Untuk mencari W_{16} :

$$W_{16} = (W_{16-16} \oplus W_{16-14} \oplus W_{16-8} \oplus W_{16-3}) \text{ ROT } 1$$

Berarti :

$W_t = W_{16-16} = W_0$
$W_t = W_{16-14} = W_2$
$W_t = W_{16-8} = W_8$
$W_t = W_{16-3} = W_{13}$
$W_t = (W_0 \oplus W_2 \oplus W_8 \oplus W_{13})$ ROT 1
W_0 = 01001001 01000100 00110011 00000011
W_2 = 00001110 01000111 01010101 01010011
W_8 = 00000000 00000000 00000000 00000000
W_{13} = 00000000 00000000 00000000 00000000
W_{16} = 01000111 00000011 01100110 01010000
W_{16} = 01000111 00000011 01100110 01010000 ROT 1
W_{16} = 10001110 00000110 11001100 10100000

Jika sudah mencapai nilai terkecil yaitu angka satu, maka nilainya kembali keawal atau yang sudah di tetapkan W. Karena t sebanyak 80 iterasi maka selanjutnya adalah hasil iterasi dan telah mengalami rotasi 1 kali keseluruhannya dari 16 hingga 79 untuk W_t.

W_{16} =	10001110	00000110	11001100	10100000
W_{17} =	00011100	10000010	10101011	00100110
W_{18} =	00111001	00001001	01010110	01001101
W_{19} =	00011100	00001101	10011001	01000001
W_{20} =	01101110	00011111	00110101	11011001
W_{21} =	10101110	00101100	11000111	00101010
W_{22} =	11100100	00100101	01011001	00110000
W_{23} =	10001000	00011110	10100101	01110101
W_{24} =	01100010	00100111	11100110	01110011
W_{25} =	00010100	01110100	11011001	01010011
W_{26} =	00001100	00000101	01111110	10000111
W_{27} =	11011100	01000101	00110001	11101000
W_{28} =	00110000	11100011	01001111	10101000
W_{29} =	11011001	01001100	11111100	10000001
W_{30} =	10101110	10010100	01100000	01000010
W_{31} =	00010011	11010100	00110011	11001000
W_{32} =	01011001	01010100	11111110	10010000
W_{33} =	00010000	11111011	00011110	11010111
W_{34} =	01011110	00011111	01011111	10010100
W_{35} =	10101100	10011110	10110010	01000100
W_{36} =	01001110	00011101	11001010	00011111
W_{37} =	01000101	00100010	00111101	10001011
W_{38} =	11010011	01101111	01100010	01110000
W_{39} =	11100001	00010110	01111111	10101101
W_{40} =	00010110	01111111	11101111	00101000
W_{41} =	01100100	11110010	00111011	10111010
W_{42} =	11100101	00011011	10101001	00100100
W_{43} =	11100110	11001000	10001100	00011001
W_{44} =	10010111	00011111	10010010	11001100
W_{45} =	10011100	10010110	11011000	10111001
W_{46} =	01110001	10101111	10100111	00011011
W_{47} =	10100101	00010110	00100101	01001101

ISSN 2774-3659 (Media Online)
Vol 2, No 3, Agustus 2022
DOI 10.47065/bulletincsr.v2i3.140 | Hal 73−84
https://hostjournals.com/bulletincsr

W_{48} = 01110010 10100100 10101101 00001010
W_{49} = 10111001 10101001 01111101 00000111
W_{50} = 10011101 01111010 10111001 11111010
W_{51} = 01010001 01001011 11011100 10111011
W_{52} = 01011001 10111011 00110000 01111101
W_{53} = 10110111 00011100 11000010 11011110
W_{54} = 11001001 10000101 10110011 00101011
W_{55} = 01110000 01010100 00101000 01100001
W_{56} = 00000000 00000000 00000000 00000000
W_{57} = 00000000 00000000 00000000 00000000
W_{58} = 11100111 01001011 10111010 10101101
W_{59} = 11100011 01011111 01001110 00110110
W_{60} = 00100011 11110111 11000110 01111101
W_{61} = 11110100 01100010 00000110 10101110
W_{62} = 11110100 10011001 11101100 00000000
W_{63} = 01111010 00111000 10111110 01010110
W_{64} = 00001101 10001101 01010111 01001001
W_{65} = 11001110 10000100 00001111 01011001
W_{66} = 00101110 01111110 01111110 01001001
W_{67} = 10001000 00010001 11001000 11000101
W_{68} = 01011100 11111100 11111100 10010010
W_{69} = 01110111 01101110 10001100 01111111
W_{70} = 01010111 11001100 00011101 01110001
W_{71} = 00000111 11101110 11000000 11111111
W_{72} = 11000001 01001010 11110100 10100101
W_{73} = 11101000 01110111 10110110 00011000
W_{74} = 01101011 10100110 00110001 01001010
W_{75} = 11001100 01010110 11001100 00000110
W_{76} = 10001100 00010011 10010000 10100011
W_{77} = 00100001 01010100 01100001 00101001
W_{78} = 11101101 00110000 11111000 10110000
W_{79} = 10011010 10110010 11001000 00001100

Untuk semua W_t di atas akan digunakan untuk proses dengan penyangga dan penambah yang telah ditetapkan dalam SHA-1. Selanjutnya memproses dari persamaan operasi dasar SHA-1, yaitu:

a, b, c, d, e ← (CLS_5(a) + f_t(b, c, d) + e + W_t + K_t), a, CLS_{30}(b), c, d

K_t 19 = 5A827999 = 0101 1010 1000 0010 0111 1001 1001 1001
a = A = 67452301 = 0110 0111 0100 0101 0010 0011 0000 0001
b = B = EFCDAB89 = 1110 1111 1100 1101 1010 1011 1000 1001
c = C = 98BADCFE = 1001 1000 1011 1010 1101 1100 1111 1110
d = D = 10325476 = 0001 0000 0011 0010 0101 0100 0111 0101
e = E = C3D2E1F0 = 1100 0011 1101 0010 1110 0001 1111 0000

CLS_S yaitu *Circular Left Shift* dengan maksud pergeseran atau rotasi bit ke kiri sebanyak s kali, untuk a CLS_5 berarti sebanyak 5 kali dan untuk b CLS_{30} berarti 30 kali. Berikut bentuk CLS :

(a) = 0110 0111 0100 0101 0010 0011 0000 0001
CLS_5 (a) = 1110 1000 1010 0100 0110 0000 0010 1100

Tabel 1. Lima penyangga a, b, c, d, dan e

A	B	C	D	E
67452301	EFCDAB89	98BADCFE	10325476	C3D2E1F0

Dikerjakan terlebih dahulu f_t(b,c,d) dengan fungsi logika AND (∧), NAND (~), OR (∨), f_t (b ∧ c) ∨ (~b ∧ d) :

(b ∧ c)
B= 1110 1111 1100 1101 1010 1011 1000 1001
C= 1001 1000 1011 1010 1101 1100 1111 1110
AND = 1000 1000 1000 1000 1000 1000 1000 1000
(~b ∧ d)
~B = 1110 1111 1100 1101 1010 1011 1000 1001
D = 0001 0000 0011 0010 0101 0100 0111 0101
NAND=1111 1111 1111 1111 1111 1111 1111 1111
OR (b ∧ c) ∨ (~b ∧ d)
(b ∧ c) = 1000 1000 1000 1000 1000 1000 1000 1000

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 2, No 3, Agustus 2022
DOI 10.47065/bulletincsr.v2i3.140 | Hal 73−84
https://hostjournals.com/bulletincsr

$(\sim b \wedge d)$ = 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111

Hasil f_t = 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111

Selanjutnya untuk $(CLS_5(a) + f_t(b, c, d) + e + W_t + K_t)$ diurutkan bit-bit untuk putaran 0 tersebut :

CLS_5 (a) = 11101000 10100100 01100000 00101100
ft_0 = 11111111 11111111 11111111 11111111
e_0 = 11000011 11010010 11100001 11110000
$W_t = W_0$ = 01001001 01000100 00110011 00000011
K_t 19 = 01011010 10000010 01111001 10011001
XOR = 11000111 01001111 00110100 10111001
Maka a' = 11000111 01001111 00110100 10111001

= C74F34B9

Hasil di atas dijadikan nilai a', dan diurutkan dengan b', c', d', dan e'untuk putaran ke 0 dengan ketentuan :

a_0 jadi b' atau b_1
b_0 rotasi 30 kali (CLS_{30}) menjadi c' atau c_1
c_0 menjadi d' atau d_1
d_0 menjadi e' atau e_1
e_0 yang mengalami operasi dasar SHA-1 menjadi a' atau a_1

hasil tersebut dijadikan sebagai t_0 dan untuk $CLS_{30}(b)$ yaitu rotasi b sebanyak 30 kali bit kanan ke kiri berurutan :

B = E F C D A B 8 9
= 1110 1111 1100 1101 1010 1011 1000 1001
$CLS_{30}(b)$ = 0111 1011 1111 0011 0110 1010 1110 0010
= 7 B F 3 6 A E 2

Tabel 2. Hasil a, b, c, d, dan e untuk t_0 hingga t_{79}

Round	A	B	C	D	E
t_0	C74F34B9	67452301	7BF36AE2	98BADCFE	10325476
t_1	5CA94528	C74F34B9	59D148C0	7BF36AE2	98BADCFE
t_2	A4AA8AE0	5CA94528	71D3CD2E	59D148C0	7BF36AE2
t_3	0F8BB090	A4AA8AE0	172A514A	71D3CD2E	59D148C0
t_4	2D5C5487	0F8BB090	292AA2B8	172A514A	71D3CD2E
t_5	7941A5CA	2D5C5487	03E2EC24	292AA2B8	172A514A
t_6	B26B6EE3	7941A5CA	CB571521	03E2EC24	292AA2B8
t_7	C13A58A8	B26B6EE3	9E506972	CB571521	03E2EC24
t_8	81D57B5A	C13A58A8	EC9ADBB8	9E506972	CB571521
t_9	5485F817	81D57B5A	304E962A	EC9ADBB8	9E506972
t_{10}	2B02A4EE	5485F817	A0755ED6	304E962A	EC9ADBB8
t_{11}	39B3401A	2B02A4EE	D5217E05	A0755ED6	304E962A
t_{12}	835B13CD	39B3401A	8AC0A93B	D5217E05	A0755ED6
t_{13}	7F4AE100	835B13CD	8E6CD006	8AC0A93B	D5217E05
t_{14}	4C20A760	7F4AE100	60D6C4F3	8E6CD006	8AC0A93B
t_{15}	A5A1C394	4C20A760	1FD2B840	60D6C4F3	8E6CD006
t_{16}	512FEC74	A5A1C394	130829D8	1FD2B840	60D6C4F3
t_{17}	F855A73B	512FEC74	296870E5	130829D8	1FD2B840
t_{18}	99128705	F855A73B	144BFB1D	296870E5	130829D8
t_{19}	A028D66C	99128705	FE1569CE	144BFB1D	296870E5
t_{20}	E712CCCF	A028D66C	6644A1C1	FE1569CE	144BFB1D
t_{21}	7D4363A1	E712CCCF	280A359B	6644A1C1	FE1569CE
t_{22}	512142F7	7D4363A1	F9C4B333	280A359B	6644A1C1
t_{23}	6F0DDCB8	512142F7	5F50D8E8	F9C4B333	280A359B
t_{24}	0EEBC090	6F0DDCB8	D44850BD	5F50D8E8	F9C4B333
t_{25}	9EB57607	0EEBC090	1BC3772E	D44850BD	5F50D8E8
t_{26}	248EE06A	9EB57607	03BAF024	1BC3772E	D44850BD
t_{27}	24A5EC85	248EE06A	E7AD5D81	03BAF024	1BC3772E
t_{28}	1AE28E64	24A5EC85	8923B81A	E7AD5D81	03BAF024
t_{29}	075E02C1	1AE28E64	49297B21	8923B81A	E7AD5D81
t_{30}	07866385	075E02C1	06B8A399	49297B21	8923B81A
t_{31}	CE467914	07866385	41D780B0	06B8A399	49297B21
t_{32}	7BDF024F	CE467914	41E198E1	41D780B0	06B8A399
t_{33}	D8D872D7	7BDF024F	33919E45	41E198E1	41D780B0

ISSN 2774-3659 (Media Online)
Vol 2, No 3, Agustus 2022
DOI 10.47065/bulletincsr.v2i3.140 | Hal 73−84
https://hostjournals.com/bulletincsr

Round	A	B	C	D	E
t_{34}	E1FB03B9	D8D872D7	DEF7C093	33919E45	41E198E1
t_{35}	7762C9BB	E1FB03B9	F6361CB5	DEF7C093	33919E45
t_{36}	3469E552	7762C9BB	787EC0EE	F6361CB5	DEF7C093
t_{37}	B594D929	3469E552	DDD8B26E	787EC0EE	F6361CB5
t_{38}	129F9D95	B594D929	8D1A7954	DDD8B26E	787EC0EE
t_{39}	7F660BAB	129F9D95	6D65364A	8D1A7954	DDD8B26E
t_{40}	8201A75B	7F660BAB	44A7E765	6D65364A	8D1A7954
t_{41}	25E16FF6	8201A75B	DFD982EA	44A7E765	6D65364A
t_{42}	912EC34D	25E16FF6	E08069D6	DFD982EA	44A7E765
t_{43}	258B8680	912EC34D	89785BFE	E08069D6	DFD982EA
t_{44}	5CCB4644	258B8680	644BB0D3	89785BFE	E08069D6
t_{45}	4103FD82	5CCB4644	0962E1A0	644BB0D3	89785BFE
t_{46}	39DCCA8B	4103FD82	1732D191	0962E1A0	644BB0D3
t_{47}	5FB9629F	39DCCA8B	9040FF60	1732D191	0962E1A0
t_{48}	288799A6	5FB9629F	CE7732A2	9040FF60	1732D191
t_{49}	0B155E35	288799A6	D7EE58A7	CE7732A2	9040FF60
t_{50}	C2ED065D	0B155E35	8A21E669	D7EE58A7	CE7732A2
t_{51}	66E5B9C3	C2ED065D	42C5578D	8A21E669	D7EE58A7
t_{52}	77BFD69D	66E5B9C3	70BB4197	42C5578D	8A21E669
t_{53}	6DFE60DF	77BFD69D	D9B96E70	70BB4197	42C5578D
t_{54}	B1F379AD	6DFE60DF	1DEFF5A4	D9B96E70	70BB4197
t_{55}	DBECDA76	B1F379AD	DB7F9837	1DEFF5A4	D9B96E70
t_{56}	01DFC74D	DBECDA76	6C7CDE6B	DB7F9837	1DEFF5A4
t_{57}	106A9A76	01DFC74D	B6FB369D	6C7CDE6B	DB7F9837
t_{58}	941E2A7E	106A9A76	4077F1D3	B6FB369D	6C7CDE6B
t_{59}	B9935BED	941E2A7E	841AA69D	4077F1D3	B6FB369D
t_{60}	021E0B73	B9935BED	A5078A9F	841AA69D	4077F1D3
t_{61}	52991F7B	021E0B73	6E64D6FB	A5078A9F	841AA69D
t_{62}	86DF2B91	52991F7B	C08782DC	6E64D6FB	A5078A9F
t_{63}	A3A7D4BC	86DF2B91	D4A647DE	C08782DC	6E64D6FB
t_{64}	B26F9040	A3A7D4BC	61B7CAE4	D4A647DE	C08782DC
t_{65}	668CD7ED	B26F9040	28E9F52F	61B7CAE4	D4A647DE
t_{66}	8E23421D	668CD7ED	2C9BE410	28E9F52F	61B7CAE4
t_{67}	88B386DF	8E23421D	59A301FB	2C9BE410	28E9F52F
t_{68}	C318142A	88B386DF	6388D087	59A301FB	2C9BE410
t_{69}	95A96B09	C318142A	E22CE1B7	6388D087	59A301FB
t_{70}	3FBBE1E5	95A96B09	B0C6050A	E22CE1B7	6388D087
t71	36CFCAB8	3FBBE1E5	656A5AC2	B0C6050A	E22CE1B7
t72	A054D8CC	36CFCAB8	4FEEF879	656A5AC2	B0C6050A
t73	D7572E60	A054D8CC	0DB3F2AE	4FEEF879	656A5AC2
t74	411029B4	D7572E60	28153633	0DB3F2AE	4FEEF879
t75	01C29491	411029B4	35D5CB98	28153633	0DB3F2AE
t76	1C8F566B	01C29491	10440A6D	35D5CB98	28153633
t77	3C569C8F	1C8F566B	B731BC6B	10440A6D	35D5CB98
t78	ED883F8A	3C569C8F	5E41FC32	B731BC6B	10440A6D
t79	0AF751C3	ED883F8A	584B951C	5E41FC32	B731BC6B

Selanjutnya setelah didapatkan a, b, c, d, dan e untuk t_{79}, maka nilai t_{79} digunakan untuk mendapatkan disebut *digest* dalam SHA-1 dengan cara di XOR dengan nilai a, b, c, d, dan e awal (penyangga).

$H_i = a_0 \oplus a_n$

H_0 = 67452301 ⊕ 0AF751C3
= 0110 0111 0100 0101 0010 0011 0000 0001
0000 1010 1111 0111 0101 0001 1100 0011
= 0110 1101 1011 0010 0111 0010 1100 0010
= 6DB272C2

H_1 = EFCDAB89 ⊕ ED883F8A
= 1110 1111 1100 1101 1010 1011 1000 1001
1110 1101 1000 1000 0011 1111 1000 1010
= 0000 0010 0100 0101 1001 0100 0000 0011
= 02459403

H_2 = 98BADCFE ⊕ 584B951C

ISSN 2774-3659 (Media Online)
Vol 2, No 3, Agustus 2022
DOI 10.47065/bulletincsr.v2i3.140 | Hal 73−84
https://hostjournals.com/bulletincsr

= 1001 1000 1011 1010 1101 1100 1111 1110
0101 1000 0100 1011 1001 0101 0001 1100
= 1100 0000 1111 0001 0100 1001 1110 0010
= C0F149E2

H_3 = 10325476 ⊕ 5E41FC32
= 0001 0000 0011 0010 0101 0100 0111 0101
0101 1110 0100 0001 1111 1100 0011 0010
= 0100 1110 0111 0011 1010 1000 0100 0111
= 4E73A847

H_4 = C3D2E1F0 ⊕ B731BC6B
= 1100 0011 1101 0010 1110 0001 1111 0000
1011 0111 0011 0001 1011 1100 0110 1011
= 1000 0011 1010 0010 0100 0100 1101 0100
= 74E34D9B

Concate H_0, H_1, H_2, H_3, dan H_4. Hasil *concate* sebanyak 20 *byte* 40 karakter tersebut menjadi *message digest* adalah :
6DB272C2 02459403 C0F149E2 4E73A847 74E34D9B

3.3 Hasil Pengujian

Dengan menggunakan aplikasi *Hasher Pro* pada pengujian implementasi metode SHA-1 untuk mendeteksi *file audio* maka didapat sebuah hasil berikut ini:

Tabel 3. Hasil *Hasher Pro*

No	Nama Audio Asli	Hasil SHA-1	Nama Audio Editan	Hasil SHA-1	Kesimpulan
1.	Tanah Airku	b272c2 459403 f149e2 73a847 e34d9b	Tanah Airku	4904ce4a f7751ad8 baf06f79 0ff41fa36 6278aa3	Dari Hasil Perbandingan meta data *file audio* Asli dan editan nyatakan berbeda berdasarkan kode dari metode yang di dapatkan
2.	Menyimpan Rasa	67e99d a61119 9dab41 3cb33b 6d375b	Menyimpan Rasa	d7a4eb78 0a03c324 a1995140 5feef1a9f 424ea93	Dari Hasil Perbandingan meta data *file audio* Asli dan editan nyatakan berbeda berdasarkan kode dari metode yang di dapatkan
3.	Cinta Luar Biasa	98e3c0 7e6418 e63134 0004e8 daa769	Cinta Luar Biasa	147ddfb7 b8d0e341 c1cdd4c2 54344fb4 112588a6	Dari Hasil Perbandingan meta data *file audio* Asli dan editan nyatakan berbeda berdasarkan kode dari metode yang di dapatkan
4.	Laskar Pelagi	bc751a 0c67db 73744f 438c7f 314d40	Laskar Pelagi	5690e9f8 800bd5df 244f1940 3a381f54 938fbdba	Dari Hasil Perbandingan meta data *file audio* Asli dan editan nyatakan berbeda berdasarkan kode dari metode yang di dapatkan
5	Aku Pasti Kembali	36dc40 5bbca9 30c845 273e6e 8d77ad	Aku Pasti Kembali	fe4cc9be 066c50fa 8d74a873 eb8b004a d12cf172	Dari Hasil Perbandingan meta data *file audio* Asli dan editan nyatakan berbeda berdasarkan kode dari metode yang di dapatkan

Berdasarkan data hasil pengujian mendeteksi keaslian *file audio* menggunakan metode SHA-1 menunjukkan bahwa perubahan sekecil apapun sangat mempengaruhi hasil dari pendeteksian atau keaslian dari *file* tersebut sehing tingkat akurat dari perbedaan *file audio* asli dan yang telah diedit sangat besar perbedaanya.

4. KESIMPULAN

Berdasarkan cara kerja dengan menggunakan algoritma SHA-1 telah berhasil melakukan proses mendeteksi keaslian *file audio* yang berformat MP3 sehingga proses mendeteksi keaslian *file audio* dapat berjalan sesuai dengan teknik mendet eksi keasliannya. Berdasarkan penerapan Algoritma SHA-1 telah berhasil membedakan *file audio* asli dengan

ISSN 2774-3659 (Media Online)
Vol 2, No 3, Agustus 2022
DOI 10.47065/bulletincsr.v2i3.140 | Hal 73−84
https://hostjournals.com/bulletincsr

file audio yang palsu. *File audio* berhasil diuji menggunakan aplikasi *Hasher Pro* dengan menerapkan algoritma SHA-1 sehingga memudahkan penulis untuk mendeteksi keaslian *file audio*.

REFERENCES

[1] K. Aryasa and Y. T. Paulus, "Implementasi Secure Hash Algorithm-1 Untuk Pengamanan Data Dalam Library Pada Pemrograman Java," *Creat. Inf. Technol. J.*, vol. 1, no. 1, p. 57, 2018.
[2] K. D. Wandani and S. Sinurat, "IMPLEMENTASI SECURE HASH ALGORITMA UNTUK PENGAMANAN PADA FILE VIDEO," vol. 13, pp. 165–168, 2018.
[3] M. K. Emy Setyaningsih, S.Si., *Kriptografi & Implementasinya menggunakan MATLAB*. Yogyakarta: ANDI, 2015.
[4] Rifki Sadikin, *Kriptografi untuk keamanan jaringan dan implementasi dalam Bahasa Java*. Yogyakarta: ANDI, 2012.
[5] Dony Ariyus, *PENGANTAR ILMU KRIPTOGRAFI Teori Analisis & Implementasi*. Yogyakarta, 2008.
[6] Dimaz A Wijaya, *MENGENAL BITCOIN & CRYPTOCURRENCY*. Medan, 2016.
[7] R. Prasetyo and A. Suryana, "Aplikasi Pengamanan Data dengan Teknik Algoritma Kriptografi AES dan Fungsi Hash SHA-1 Berbasis Desktop," *J. Sisfokom (Sistem Inf. dan Komputer)*, vol. 5, no. 2, p. 61, 2018.
[8] H. Santoso and M. Fakhriza, "Perancangan Aplikasi File Audio Format Wav (Waveform) Menggunakan Algoritma Rsa," *Algoritm. J. Ilmu Komput. dan Inform.*, vol. 2, no. 1, pp. 47–54, 2018.
[9] http://www.den4b.com/products/hasher/