Jurnal Sains Dan Teknologi Informasi
Vol 1, No 1, Desember 2021, Hal 13-18
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v1i1.963
Website https://ejurnal.seminar-id.com/index.php/jussi

Implementasi Fungsi Hash Untuk Mendeteksi Orisinalitas File Audio Menggunakan Metode Gost

Bella Nabila, Lince Tomoria Sianturi, Chandra Frenki Sianturi

Fakultas Ilmu Komputer Dan Teknologi Informasi, Tenik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: bellanabila311297@gmail.com

Abstrak− File Audio adalah file untuk menyimpan data audio digital pada sistem komputer, perkembangan teknologi umumnya pada bidang komputer sudah menjadi suatu kebutuhan, terdapat salah satu fitur digital yang banyak dimanfaatkan adalah file audio. Dengan maraknya kecanggihan digital serta teknologinya dan perangkat lunak menimbulkan keinginan bagi pengguna untuk memalsukan file audio yang bermaksud untuk melakukan pemalsuan demi keuntungannya sendiri. Saat ini ada banyak perangkat lunak yang dapat digunakan untuk memanipulasi file termasuk file audio. Dengan adanya kecanggihan perangkat lunak tersebut bisa membuat file audio yang dimanipulasi tidak memiiki kejanggalan sehingga sangat sulit untuk dideteksi. Orisinalitas atau keaslian data menjadi sangat penting, karena merupakan suatu kemurnian yang harus dijaga isi dan sebagainya, Demi keamanan sistem harus memiliki kemampuan untuk mendeteksi keaslian atau orisinalitas dari file audio tersebut. Dengan kemajuan teknologi saat ini salah satu cara yang dapat digunakan untuk mendeteksi keaslian dari suatu file adalah dengan memanfaatkan fungsi hash yang menghasilkan kode yang dapat dimanfaatkan untuk mendeteksi file audio tersebut. Salah satu fungsi hash yang dapat digunakan adalah metode Gost.

Kata Kunci: Kriptografi, Mendeteksi Orisinalitas File Audio, Metode Gost.

Abstract−Audio files are files for storing digital audio data on a computer system, technological developments in general in the computer field have become a necessity, there is one digital feature that is widely used is audio files. With the rise of digital sophistication and its technology and software, there is a desire for users to falsify audio files with the intention of falsifying for their own benefit. Today there are many software that can be used to manipulate files including audio files. With the sophistication of the software, the manipulated audio file does not have any discrepancies, making it very difficult to detect. The originality or authenticity of the data becomes very important, because it is a purity that must be maintained in the contents and so on. For the sake of security the system must have the ability to detect the authenticity or originality of the audio file. With current technological advances, one way that can be used to detect the authenticity of a file is to utilize a hash function that generates a code that can be used to detect the audio file. One hash function that can be used is the Gost method.

Keywords: Cryptography, Detecting Originality of Audio Files, Gost Method

1. PENDAHULUAN

Orisinalitas atau keaslian data menjadi sangat penting, karena merupakan suatu kemurnian yg harus dijaga isi dan sebagainya, untuk menjaga keaslian suatu data atau *file* tetap aman, sistem harus memiliki kemampuan untuk mendeteksi pemalsuan atau manipulasi data oleh pihak-pihak yang tidak bertanggung jawab. Salah satu cara yang dapat digunakan untuk mendeteksi keaslian dari suatu *file* adalah dengan memanfaatkan fungsi *hash* yang menghasilkan kode yang dapat dimanfaatkan untuk mendeteksi keaslian dari *file audio*. Salah satu fungsi *hash* yang dapat digunakan adalah metode Gost. Metode GOST merupakan suatu algoritma *block cipher* yang dikembangkan oleh seorang berkebangsaan Uni Soviet (Schneier, 1995). Metode ini dikembangkan oleh pemerintah Uni Soviet pada masa perang dingin untuk menyembunyikan data atau informasi yang bersifat rahasia pada saat komunikas[1]. Gost merupakan blok *cipher* 64 bit dengan panjang kunci 256 bit. Algoritma ini mengiterasi algoritma enkripsi sederhana sebanyak 32 putaran (*round*). Untuk mengenkripsi pertama-tama *plaintext* 64 bit dipecah menjadi 32 bit bagian kiri, L dan 32 bit bagian kanan, R Subkunci (*subkey*)[2]. Secara struktural, algoritma Gost mirip dengan algoritma DES (*Data Encryption Standart*). Algoritma DES merupakan blok *cipher* 64 bit dengan panjang kunci 56 bit. Algoritma ini mengiterasi algoritma enkripsi sebanyak 16 putaran (*round*). Karena panjang kunci yang hanya 56 bit, membuat algoritma ini sangat rawan di-*brute force* sehingga saat ini digunakan 3 buah DES secara berurutan untuk mengenkripsi sebuah *plaintext* yang disebut dengan *Triple* DES. Panjang kunci juga diperpanjang 3 kali menjadi 168 bit (56*3 = 168) [3]. Kelemahan GOST yang diketahui sampai saat ini adalah karena *key schedule*-nya yang sederhana sehingga pada keadaan tertentu menjadi titik lemahnya terhadap metoda kriptanalisis seperti *Related-key Cryptanalysis*[5]. Tetapi hal ini dapat diatasi dengan melewatkan kunci kepada fungsi *hash* yang kuat secara kriptografi seperti SHA-1, kemudian menggunakan hasil *hash* untuk input inisialisasi kunci . Kelebihan dari metoda GOST ini adalah kecepatannya yang cukup baik, walaupun tidak secepat Blowfish tetapi lebih cepat dari IDEA[5].

2. METODOLOGI PENELITIAN

2.1 Fungsi Hash

Fungsi *hash* adalah fungsi yang menerima masukan *string* yang panjangnya sembarang dan mengonversinya menjadi *string* keluaran yang panjangnya tetap (*fixed*), umumnya berukuran jauh lebih kecil dari pada ukuran *string* semula [1]. Fungsi *hash* dapat menerima masukan *string* apa saja. Jika *string* menyatakan pesan (*message*), sembarang pesan M berukuran bebas dikompresi oleh fungsi *hash* H melalui persamaan berikut :

$$H = H(M) \quad (1)$$

Jurnal Sains Dan Teknologi Informasi
Vol 1, No 1, Desember 2021, Hal 13-18
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v1i1.963
Website https://ejurnal.seminar-id.com/index.php/jussi

Keterangan :
M: Pesan dengan panjang sembarang
H: Nilai *hash* (*hash value*) atau pesan ringkas (*message digest*)

Keluaran fungsi *hash* disebut juga nilai *hash* (*hash value*) atau pesan ringkas (*message digest*). Fungsi *hash* akan mengembalikan *hash value* yang panjangnya jauh lebih pendek dibandingkan dengan panjang *string* masukan. Sebagai contoh, pesan yang ingin dicari nilai *hash*-nya hanya memiliki ukuran sebesar 1 Mb, maka *hash value* yang dihasilkan hanya 128 bit. Kebanyakan fungsi *hash* yang ada saat ini berupa fungsi *hash* satu arah. Artinya, pesan yang sudah diubah menjadi *message digest* tidak dapat dikembalikan lagi menjadi pesan semula (*irreversible*). Selain itu, fungsi *hash* satu arah mempunyai sifat sebagai berikut.

a. Fungsi H dapat diterapkan pada blok data berukuran berapa saja.
b. H menghasilkan nilai (h) dengan panjang tetap (*fixed length output*).
c. H(x) mudah dihitung untuk setiap nilai x yang diberikan.
d. Untuk setiap h yang dihasilkan, tidak mungkin dikembalikan nilai x sedemikian sehingga H(x) = h.
e. Untuk setiap x yang diberikan, tidak mungkin dicari y ≠ x sedemikian sehingga H(y) = H(x).
f. Tidak mungkin dicari pasangan x dan y sedemikian sehingga H(x) = H(y).

2.2 Algoritma *Government Standard* (GOST)

Algoritma Gost merupakan blok *cipher* 64 bit dengan panjang kunci 256 bit (Saarinen, 1998). Algoritma ini mengiterasi algoritma enkripsi sederhana sebanyak 32 putaran (*round*) (Saarinen, 1998). Untuk mengenkripsi pertama-tama *plaintext* 64 bit dipecah menjadi 32 bit bagian kiri, L dan 32 bit bagian kanan, R. Subkunci (*subkey*) untuk putaran *i* adalah *Ki*. Pada satu putaran ke-*i* operasinya adalah sebagai berikut:

$$Li = Ri\text{-}1$$
$$Ri = Li\text{-}1 \oplus f(Ri\text{-}1, Ki) \quad (2)$$

Secara struktural, algoritma Gost mirip dengan algoritma DES (*Data Encryption Standart*) (Kelsey, 1996). Algoritma DES merupakan blok *cipher* 64 bit dengan panjang kunci 56 bit (Kelsey, 1996). Algoritma ini mengiterasi algoritma enkripsi sebanyak 16 putaran (*round*) (Kelsey, 1996). Karena panjang kunci yang hanya 56 bit, membuat algoritma ini sangat rawan di-*brute force* sehingga saat ini digunakan 3 buah DES secara berurutan untuk mengenkripsi sebuah *plaintext* yang disebut dengan *Triple DES* (Kelsey, 1996). Panjang kunci juga diperpanjang 3 kali menjadi 168 bit (56*3 = 168) [1].

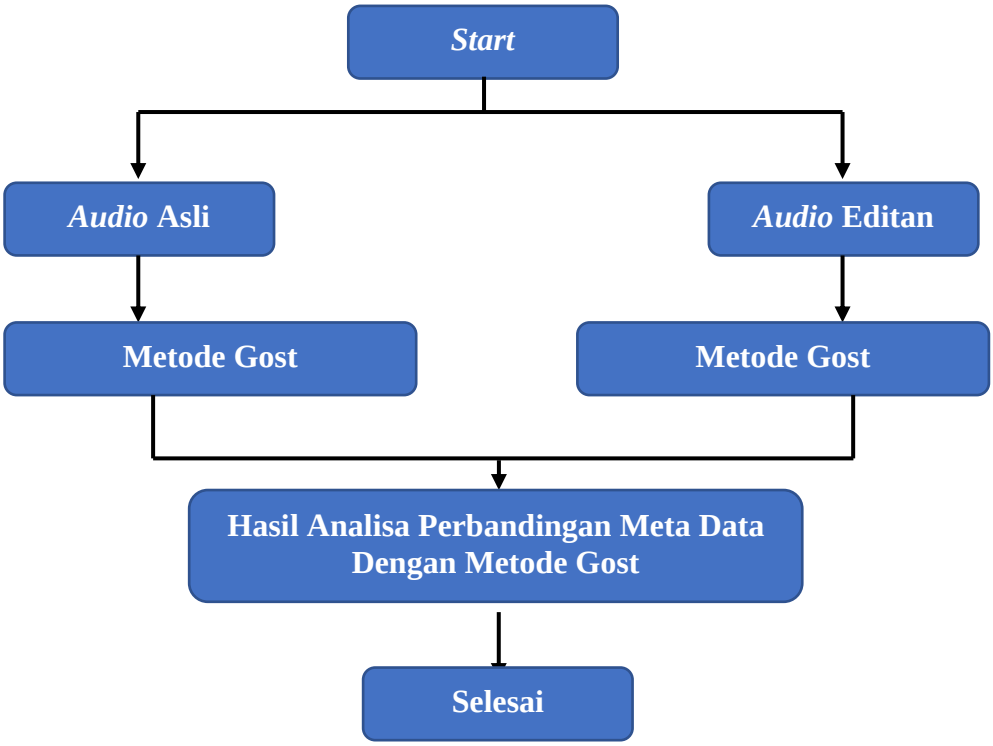
2.3 File Audio

Audio (Suara) adalah fenomena fisik yang dihasilkan oleh getaran suatu benda yang berupa sinyal analog dengan amplitude yang berubah secara continue terhadap satuan waktu yang disebut frekuensi [7].

3. HASIL DAN PEMBAHASAN

Analisa masalah dalam melakukan pendeteksian keaslian suatu *audio* dengan melakukan proses pengecekan atau perbandingan suatu keaslian dari *audio* asli dengan *audio* yang telah dirubah. *audio* merupakan barang bukti digital yang salah satunya berasal dari perekam suara, 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 penambahan dan pemotongan 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 disengaja merupakan konsekuensi dari proses operasional digital, seperti pengurangan ukuran. Untuk membedakan *audio* asli atau *audio* yang sudah dirubah maka diperlukan pendeteksian terhadap *audio* tersebut. Metode yang digunakan untuk mendeteksi *audio* dengan deteksi yang hanya mengecek integritas dari *audio* tanpa menunjukkan bagian mana pada *audio* yang telah dimanipulasi. Namun tidak dapat memberikan informasi lokasi mana yang telah dimanipulasi pada *audio*. Oleh sebab itu maka di perlukan cara mengecek keaslian suatu *audio* dengan memberikan suatu kode atau metadata dari *file audio* asli. Adapun prosedur mendeteksi keaslian *audio* dapat dilihat seperti bagan dibawah ini:

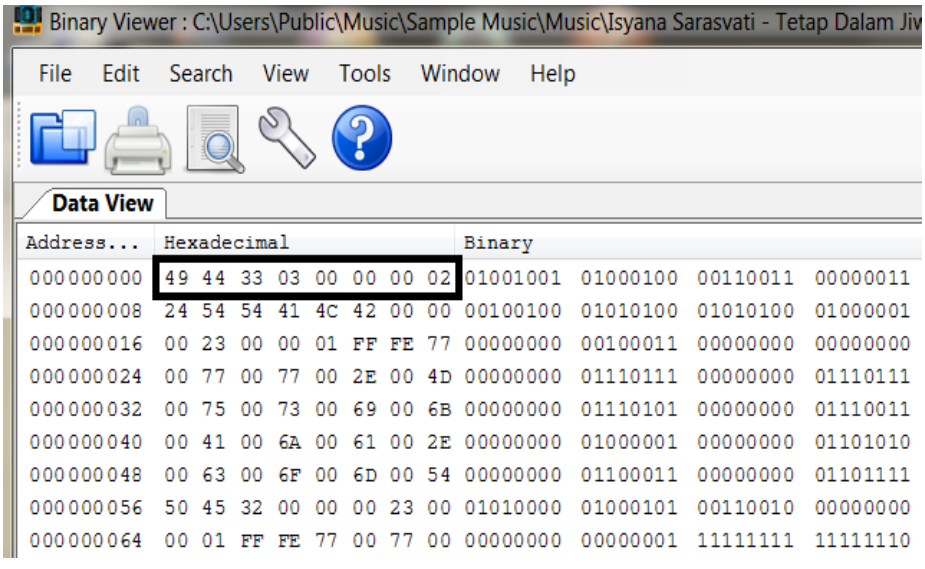
Jurnal Sains Dan Teknologi Informasi
Vol 1, No 1, Desember 2021, Hal 13-18
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v1i1.963
Website https://ejurnal.seminar-id.com/index.php/jussi



Gambar 1. Bagan Proses Pendeteksian Orisinalitas *Audio*

Mendeteksi orisinalitas *audio* dapat dilakukan dengan menggunakan metode yang digunakan untuk mendeteksi *audio* dengan deteksi yang hanya mengecek integritas dari *audio* tanpa menunjukkan bagian mana pada *audio* yang telah dimanipulasi. Salah satu metode yang dapat digunakan adalah dengan menerapkan fungsi *hash*. Fungsi *hash* mempunyai proses 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. Ini adalah sebuah kontrak dalam banyak bahasa pemrograman yang membolehkan pengguna melakukan *override* pada kesamaan morfologi dan fungsi *hash* bagi sebuah objek. Jika dua objek adalah sama, maka kode *hash*-nya pun sama. menjadi hal yang sangat penting untuk menemukan sebuah elemen di dalam tabel *hash* dengan cepat, juga karena dua elemen yang sama akan sama-sama meng-*hash* ke slot yang sama. Adapun metode yang digunakan untuk mengecek orisinalitas *audio* yaitu metode Gost. Adapun proses pendeteksian orisinalitas *audio* adalah dengan cara membandingkan hasil kunci yang didapatkan oleh metode Gost.

Contoh kasus dalam proses ini seperti dijelaskan dalam analisa yaitu *file audio* MP3 (Isyana Sarasvati- Tetap dalam jiwa). data yang diambil hanya sebanyak 8 byte untuk plainteks, cara pengambilan nilai hex data *audio* menggunakan aplikasi *Binary Viewer*, dari data tersebut diambil sebanyak 8 byte atau 16 karakter heksadesimal dan dikonversi ke biner sebagai *sample* metode, yang berguna untuk mengetahui nilai biner dari bilangan tersebut.



Gambar 2. *Sample* data

Jurnal Sains Dan Teknologi Informasi
Vol 1, No 1, Desember 2021, Hal 13-18
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v1i1.963
Website https://ejurnal.seminar-id.com/index.php/jussi

Dari gambar data *audio* di atas diambil sebanyak 8 byte untuk plainteks, yaitu :

Hex	49	44	33	03	00	00	00	02
Biner	01001001	01000100	00110011	00000011	00000000	00000000	00000000	00000010

Gabungan Biner Plainteks :
01001001010001000011001100000011→R0
00000000000000000000000000000010→L0
L[0] = 01000000000000000000000000000000
R[0] = 11000000110011000010001010010010
Kunci :

Tabel 1. Konversi Kunci

Char	Hex	Biner	Char	Hex	Biner
B	42	01000010	K	4B	01001011
E	45	01000101	-	2D	00101101
L	4C	01001100	B	42	01000010
L	4C	01001100	U	55	01010101
A	41	01000001	D	44	01000100
-	2D	00101101	I	49	01001001
N	4E	01001110	D	44	01000100
A	41	01000001	A	41	01000001
B	42	01000010	R	52	01010010
I	49	01001001	M	4D	01001101
L	4C	01001100	A	41	01000001
A	41	01000001	M	4D	01001101
S	53	01010011	E	45	01000101
T	54	01010100	D	44	01000100
M	4D	01001101	A	41	01000001
I	49	01001001	N	4E	01001110

a. Kelompokkan Biner Kunci menjadi 8 Kelompok Jumlah bit kunci setiap kelompok adalah 32 bit. dimulai dari
K[0] = bit ke 32..1,
K[1] = bit ke 64..33,
.....dst, sampai
K[7] = bit ke 256..225
01000010010001010100110001001100010000010010110101001110010000010100001001001001010011000100001010100110101010001001101010010010100101100101110101000001001010101010001000100100101000100010000010101001001001110101000001010011010100010101000100010000010100111 0

Tabel 2. Pengelompokan Kunci

Kunci	Posisi Bit	Biner yang diambil
K[0]	32...1	01000010010001010100110001001100
K[1]	64...33	10000010011100101011010010000010
K[2]	96...65	10000010001100100011000101000010
K[3]	128...97	10010010101100100010101011001010
K[4]	160...129	10101010010000101011010011010010
K[5]	192...161	10000010001000101001001000100010
K[6]	224...193	10110010100000101011001001000101
K[7]	256...225	01110010100000100010001010100010

b. Kelompokkan plainteks menjadi 2 bagian yaitu R[0] dan L[0] dengan jumlah tiap kelompok adalah 32 bit. 32 bit bagian kiri menjadi menjadi R[0] dan mulai bit ke 33 sampai bit ke 64 (sebelah kanan) menjadi L[0]. Proses penulisan bitnya dilakukan secara terbalik.
R[0] = bit[32], bit[31].....bit[1]
L[0] = bit[64], bit[63].....bit[33]
L[0] = 01000000000000000000000000000000
R[0] = 11000000110011000010001010010010

Proses Enkripsi

Plainteks : 49 44 33 03 00 00 00 02
Kunci : BELLA-NABILASTMIK-BUDIDARMAMEDAN
Putaran 0→ i=0

a. L(0) = 01000000000000000000000000000000
R(0) = 11000000110011000010001010010010

Jurnal Sains Dan Teknologi Informasi
Vol 1, No 1, Desember 2021, Hal 13-18
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v1i1.963
Website https://ejurnal.seminar-id.com/index.php/jussi

b. (R[0]+K[0]) mod 2^{32}
R[0] = 3234603666
K[0] = 1111837772 +
= 4346441438 mod 2^{32}
= 51474142
= 00000011000100010110111011011110
c. Pengelompokan

Tabel 3. Hasil Pengelompokan Putran 1

Biner Kelompok	Dec Nilai Bit	SBOX	Hasil Permutasi dengan SBOX	Biner
0000	0	→S-Box(0)	4	0100
0011	3	→S-Box(1)	12	1100
0001	1	→S-Box(2)	8	1000
0001	1	→S-Box(3)	13	1101
0110	6	→S-Box(4)	13	1101
1110	14	→S-Box(5)	15	1111
1101	13	→S-Box(6)	8	1000
1110	14	→S-Box(7)	8	1000

Lakukan sampai putran ke 31
Putaran 31→ i=31
a. L[31] = 00101011010001001111001101000111
R[31] = 01001101110010010011101010011100
b. (R[31]+K[0]) mod 2^{32}
R[31] = 1305033372
K[0] = 1111837772 +
= 2416871144 mod 2^{32}
= 2416871144
= 10010000000011101000011011101000
c. Pengelompokan

Tabel 4. Hasil Pengelompokan Putran 31

Biner Kelompok	Dec Nilai Bit	SBOX	Hasil Permutasi dengan SBOX	Biner
1001	9	→S-Box(0)	11	1011
0000	0	→S-Box(1)	14	1110
0000	0	→S-Box(2)	5	0101
1110	14	→S-Box(3)	5	0101
1000	8	→S-Box(4)	4	0100
0110	6	→S-Box(5)	1	0001
1110	14	→S-Box(6)	2	0010
1000	8	→S-Box(7)	9	1001

d. Gabungan = 10111110010101010100000100101001
RLS 11 bit =10101010000010010100110111110010
e. RLS[31] XOR L[31]
= 10101010000010010100110111110010
= 00101011010001001111001101000111 ⊕
R[32]= 10000001010011011011111010110101
f. Pembalikan penulisan biner biner L[32] dan R[32]
L[32] = 01001101110010010011101010011100
R[32] = 10000001010011011011111010110101
g. Gabungan R[32] dan L[32]
10000001010011011011111010110101010011011100100100111010100111 00
h. Cipherteks

Tabel 5. Hasil Cipherteks

10000001	01001101	10111110	10110101	01001101	11001001	00111010	10011100
Ü	M	╛	╡	M	╔	:	£

Pengujian dalam mendeteksi orisinalitas *audio* sangat diperlukan karena dengan adanya pengujian kita dapat mengetahui hasil dari perbedaan dari *audio* asli dan *audio* yang telah dirubah. Pengujian mendeteksi keaslian *audio* yang akan diuji merupakan hasil dari pembentukan kode fungsi *Hash* dengan menerapkan metode Gost yang digunakan. Sehingga kita akan dapat melihat kekurangan dan perubahan dari *file audio* asli dan yang telah dirubah. Berikut ini merupakan data

Jurnal Sains Dan Teknologi Informasi
Vol 1, No 1, Desember 2021, Hal 13-18
ISSN 2809-610X (Media Online)
DOI 10.47065/jussi.v1i1.963
Website https://ejurnal.seminar-id.com/index.php/jussi

hasil perbandingan pendeteksian orisinalitas *file audio* yang diperoleh dari data *audio* asli dan *audio* editan pada tabel di bawah ini:

Tabel 6. Hasil deteksi Orisinalitas *Audio*

Audio Asli	GOST	Audio Editan	GOST	Kesimpulan
Isyana Sarasvati-Tetap Dalam Jiwa	Ü M ┘ ┤ M ╔ : £	Isyana Sarasvati-Tetap Dalam Jiwa	Ü W ┘ ¥ N ╔ Ü £	Dari Hasil Perbandingan metadata *file audio* Asli dan editan dinyatakan berbeda berdasarkan kode dari metode yang di dapatkan

Berdasarkan data hasil pengujian mendeteksi orisinalitas *file audio* menggunakan metode Gost menunjukkan bahwa perubahan sekecil apapun sangat mempengaruhi hasil dari pendeteksi atau keaslian dari *file* tersebut sehingga tingkat akurat dari perbedaan *file audio* asli dan yang telah dimanipulasi sangat besar perbedaannya.

4. KESIMPULAN

Berdasarkan penelitian yang telah dilakukan dapat diambil beberapa kesimpulan dimana pendeteksian orisinalitas suatu file audio dapat dilakukan dengan cara membandingkan nilai hash dari file audio yang asli dengan yang telah dirubah. Pada pengujian yang dilakukan untuk penelitian ini yaitu dengan menerapkan algoritma GOST pada fungi hash yang menghasilkan kode, dengan memanfaatkan fungsi hash tersebut maka file audio tersebut tidak dapat dimanipulasi. Dengan menerapkan algoritma GOST telah membuktikan bahwa file audio yang telah dimanipulasi dapat terdeteksi keorisinalitasannya dengan akurat.

REFERENCES

[1] R.A and M.S ., REKAYASA PERANGKAT LUNAK TERSTRUKTUR dan BERORIENTASI OBJEK, BANDUNG: INFORMATIKA BANDUNG, 2013.
[2] N. Usman, Konteks Implementasi Berbasis Kurikulum, 2002, p. 70.
[3] R. Munir, Kriptografi, Bandung, 2006.
[4] Dony Ariyus, Pengantar Ilmu Kriptogarfi Teori Analisis dan Implementasi, FI. Sigit Suyantoro, Ed. Yogyakarta, Indonesia: Andi, 2008.
[5] H. Mukhtar, Kriptografi Untuk Keamanan Data, Yogyakarta: Deepublish, 2018.
[6] S.si., M.Kom Emy Setyaningsih, Kriptografi & Implementasinya menggunakan MATLAB, Yogyakarta: ANDI, 2015.
[7] F. M. Santoso H, PERANCANGAN APLIKASI KEAMANAN FILE AUDIO FORMAT WAV (WAVEFROM).
[8] F. R. Muhammad, S. Muhammad dan A. Soeb, "Implementasi Pisanc Chiper Untuk Autentikasi Voice Chat," Journal of Computer System and Informatics (JoSYC), vol. 2, no. 4, pp. 288-294, 2021.
[9] A. Soeb dan S. Muhammad, "Pengaman File Video Menggunakan Algoritma Merkle Hellman Knapsack," JURNAL MEDIA INFORMATIKA BUDIDARMA, vol. 4, no. 2, pp. 461-465, 2020.