Bulletin of Artificial Intelligence
Volume 1, No 1, April 2022 Page: 25−32
ISSN 2962-3944 (media online)
https://journal.grahamitra.id/index.php/buai

Perancangan Aplikasi Duplicate Video Scanner Menerapkan Algoritma Sha-256

Rewindo Siregar

Fakultas Ilmu Komputer dan Teknologi Informasi, Prodi Teknik Informatika Universitas Budi Darma, Medan, Indonesia
Email: rewindosiregar94@gmail.com

Abstrak–Dalam zaman yang semakin maju ini teknologi baru yang belum pernah dibayangkan sebelumnya semakin banyak bermunculan, mulai dari munculnya komputer, handphone, dan masih banyak lagi. Salah satu teknologi baru yang muncul adalah MPEG4 (Moving Picture Expert Group)-4 vidio Player 3 atau yang lebih dikenal dengan MP4. MP4 Player bisa membuat seseorang mendengarkan video secara digital, tidak perlu menggunakan kaset atau CD seperti zaman dahulu dan memungkinkan jutaan orang di seluruh dunia untuk saling bertukar rekaman musik melalui komputer yang terhubung jaringan komputer. MP4 format yang biasa dimainkan oleh MP4 Player adalah salah satu format video yang paling sering digunakan dalam penyimpanan data video. Dalam ilmu kriptografi terdapat fungsi hash yang merupakan enkripsi satu arah yang menghasilkan sebuah nilai yang memiliki panjang tetap dari sebuah pesan. Dalam hal ini pesan yang dimaksudkan adalah file video. Fungsi hash dapat digunakan untuk menghasilkan identitas dari sebuah file video, di mana jika ada dua atau lebih file video yang memiliki nilai hash yang sama maka dapat di ambil kesimpulan bahwa file video tersebut ganda atau duplikat. Hal ini tentunya akan memudahkan pengguna dalam menghapus file video yang ganda atau duplikat.

Kata Kunci: Kriptografi; Algoritma SHA-256; MPEG-4; Enkripsi Dekripsi

Abstract–In this increasingly advanced era, new technologies that have never been imagined before are increasingly emerging, starting from the emergence of computers, cellphones, and many more. One of the emerging technologies is MPEG4 (Moving Picture Expert Group)-4 Video Player 3 or better known as MP4. MP4 Player can make a person listen to video digitally, no need to use cassettes or CDs like in the past and allows millions of people around the world to exchange music recordings through computers connected to computer networks. MP4 format which is commonly played by MP4 Player is one of the most frequently used video formats in video data storage. In cryptography there is a hash function which is a one-way encryption that produces a value that has a fixed length of a message. In this case the intended message is a video file. The hash function can be used to generate the identity of a video file, where if there are two or more video files that have the same hash value, it can be concluded that the video file is duplicate or duplicate. This will certainly make it easier for users to delete duplicate or duplicate video files.

Keywords: Cryptography; SHA-256 Algorithm; MPEG-4; Encryption Decryption

1. PENDAHULUAN

Dalam zaman yang semakin maju ini teknologi baru yang belum pernah dibayangkan sebelumnya semakin banyak bermunculan, mulai dari munculnya komputer, handphone, dan masih banyak lagi. Salah satu teknologi baru yang muncul adalah MPEG4 (Moving Picture Expert Group)-4 vidio Player 3 atau yang lebih dikenal dengan MP4. MP4 Player bisa membuat seseorang mendengarkan video secara digital, tidak perlu menggunakan kaset atau CD seperti zaman dahulu dan memungkinkan jutaan orang di seluruh dunia untuk saling bertukar rekaman musik melalui komputer yang terhubung jaringan komputer. MP4 format yang biasa dimainkan oleh MP4 Player adalah salah satu format video yang paling sering digunakan dalam penyimpanan data video.

MP4 menjadikan format video yang sering digunakan karena data yang disimpan menyerupai data asli pada saat direkam dan memiliki ukuran yang tidak terlalu besar dibandingkan format lainnya. Karena kemudahan untuk mendapatkan file video dengan cara mengunduh melalui jaringan internet sering sekali mengunduh file video yang sama. Hal ini tentunya akan menyita ruang penyimpanan yang tersedia karena menyimpan file video yang ganda. Untuk dapat menghapus file video yang ganda tersebut dibutuhkan ketelitian untuk melihat isi dari file video tersebut satu per satu. Tentunya hal ini akan sangat menyulitkan dan membutuhkan waktu yang lama. Untuk mengatasi permasalahan tersebut dibutuhkan sebuah cara yang dapat digunakan untuk memberikan sebuah identitas terhadap sebuah file video.

Dalam ilmu kriptografi terdapat fungsi hash yang merupakan enkripsi satu arah yang menghasilkan sebuah nilai yang memiliki panjang tetap dari sebuah pesan. Dalam hal ini pesan yang dimaksudkan adalah file video. Fungsi hash dapat digunakan untuk menghasilkan identitas dari sebuah file video, di mana jika ada dua atau lebih file video yang memiliki nilai hash yang sama maka dapat di ambil.

Kesimpulan bahwa file video tersebut ganda atau duplikat. Hal ini tentunya akan memudahkan pengguna dalam menghapus file video yang ganda atau duplikat.

Secure Hash Algorithm (SHA) adalah salah satu jenis dari algoritma fungsi hash. SHA terdiri dari 4 macam yaitu SHA-1, SHA-256, SHA-384, SHA-512. SHA-256 adalah salah satu algoritma hash yang relatif masih baru. Algoritma ini dirancang oleh The National Institute of Standards and Technology (NIST) pada tahun 2002. SHA-256 menghasilkan message digest dengan panjang 256 bits. SHA-256 tergolong aman karena didesain sedemikian rupa sehingga tidak memungkinkan mendapatkan pesan yang perhubungan dengan message digest yang sama [1].

Bulletin of Artificial Intelligence
Volume 1, No 1, April 2022 Page: 25−32
ISSN 2962-3944 (media online)
https://journal.grahamitra.id/index.php/buai

2. METODOLOGI PENELITIAN

2.1 Kriptigrafi

Kriptografi pada awalnya dijabarkan sebagai ilmu yang mempelajari bagaimana menyembunyikan pesan. Namun pada pengertian modern *kriptografi* adalah ilmu yang bersandarkan pada teknik matematika untuk berurusan dengan keamanan informasi seperti kerahasiaan, keutuhan data dan otentikasi entitas. Jadi pengertian *kriptografi* modern adalah tidak saja berurusan hanya dengan menyembunyikan pesan, namun lebih pada sekumpulan teknik yang menyediakan keamanan informasi [3].

2.2 Algoritma SHA-256

Algoritma SHA-256 adalah salah satu algoritma hash yang relatif masih baru. Algoritma ini dirancang oleh The National Institute of Standards and Technology (NIST) pada tahun 2002. SHA-256 menghasilkan message digest dengan panjang 256 bits. SHA-256 tergolong aman karena didesain sedemikian rupa sehingga tidak memungkinkan mendapatkan pesan yang berhubungan dengan message digest yang sama. Proses untuk menghasilkan message digest pada algoritma ini meliputi lima tahapan [2].

1. *Message Padding, Input* pesan pada algoritma SHA-256 akan dibagi menjadi blok-blok yang masing-masing panjangnya adalah 512 bit. Akibat dari pembagian ini maka jumlah blok terakhir akan lebih kecil atau sama dengan 512 bit. Blok terakhir tersebut akan mengalami *message padding*. Langkah-langkah *message padding* adalah sebagai berikut :
   a. Diawali dengan masuknya *input* pesan yang memiliki kode *American Standard Code for Information Interchange* (ASCII) dan kemudian diubah ke dalam bentuk biner rangkaian bit yang akan dihitung panjangnya.
   b. Rangkaian bit tersebut dibagi menjadi blok-blok yang masing-masing mempunyai panjang 512 bit. Hasil pembagian akan menyebabkan jumlah blok terakhir lebih kecil satu atau sama dengan 512 bit.
   c. Lakukan penambahan bit-bit isian *(padding)* pada blok terakhir pesan tersebut. Bit-bit yang digunakan sebagai bit isian adalah bit 1 diikuti sejumlah bit 0 sesuai dengan kebutuhan, dengan ketentuan sebagai berikut :
      i. Jika panjang bit blok pesan terakhir lebih kecil dari 448 bit, maka ditambahkan bit '1' pada posisi bit paling akhir, diikuti dengan beberapa bit '0' sedemikian sehingga total panjang bit setelah proses tersebut adalah 448 bit.
      ii. Jika panjang bit blok pesan terakhir lebih besar atau sama dengan 448 bit, maka ditambahkan bit '1' pada posisi bit paling terakhir, diikuti dengan beberapa bit '0' sedemikian sehingga total panjang bit setelah proses tersebut 512 bit. Kemudian dibuat 448 bit baru yang isi bitnya '0'
      iii. Jika panjang bit blok pesan terakhir sama dengan 512 bit, maka harus dibuat blok baru untuk menampung proses message padding. Bit pertama dari blok baru diisi bit'1', sedangkan bit-bit berikutnya sampai dengan panjang bit 448 diisi oleh bit '0'. Jumlah total bit isian yang ditambahkan 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 terakhirnya agar total panjang blok terakhir 512 bit. Urutan byte paling kanan dari nilai representasi panjang pesan (M) dijadikan *low order*. Tambahkan representasi M tersebut pada 448 bit terakhir, sehingga jumlah panjang blok terakhir adalah 512 bit.
3. Inisialisasi Nilai Hash Awal Pada SHA – 256 untuk menyimpan nilai inisialisasi awal dan nilai *output* sementara digunakan *buffer* H0, H1, H2, H3, H4, H5, H6, H7. Di sisi lain, untuk penyimpanan proses sementatat digunakan *buffer* a, b, c, d, e, f, g, h. Nilai H0, H1, H2, H3, H4, H5, H6, H7 untuk inisialisasi awal dalam notasi heksadesimal.
4. Pemrosesan merupakan bagian inti yang terdiri atas 1 *round* 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 akan dilakukan langkah-langkah :
   a. Persiapkan penjadwalan pesan.
   b. Inisialisasi *working* variabel a, b, c, d, e, f, g dan h, untuk M(1) dengan nilai *hash* awal a = H0 (I-1) b = H1 (i-1) c = H2 (i-1) d = H3 (i-1) e = H4 (i-1) f = H5 (i-1) g = H6 (i-1) h = H7 (i-1)
   c. Hitung masing-masing penjadwalan.
   d. Menghitung nilai *hash* perantara untuk masing-masing blok pesan.
5. *Output* diperoleh setelah semua blok M (N) 512bit diproses. Setelah semua langkah pemprosesan dilakukan sejumlah N kali, maka akan didapat 256 bit *message digest*.

3. HASIL DAN PEMBAHASAN

Masalah yang dihadapi ketika akan melakukan pencarian file video yang duplikat atau ganda pada sebuah media penyimpanan membutuhkan ketelitian dan ingatan yang kuat dari penggunanya karena harus melihat isi video satu persatu. Masalah ini dapat diatasi dengan cara memberikan identitas dari setiap file video, sehingga ketika didapatkan file video tersebut merupakan file video yang duplikat atau ganda. Dalam penelitian ini cara yang digunakan untuk mendapatkan identitas file video tersebut menggunakan Algoritma SHA-256.

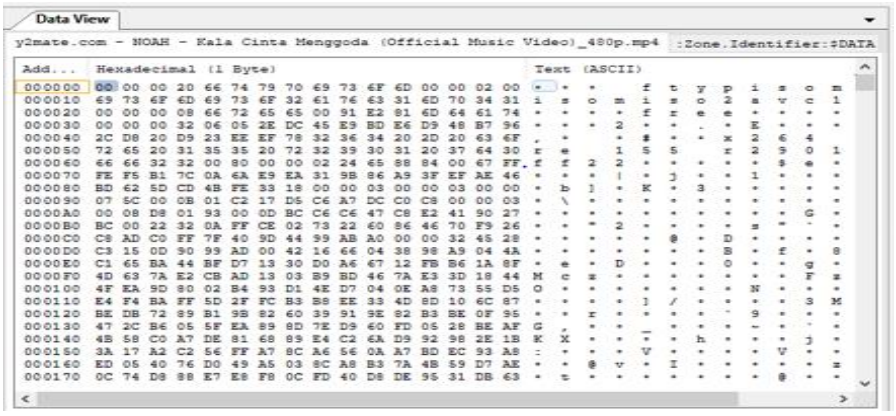
Langkah awal yang dilakukan ketika ingin melakukan pencarian file video yang ganda atau duplikat adalah melakukan scanning pada sebuah ruang penyimpanan yang di dalamnya terdapat banyak file video. Setelah melakukan proses scanning maka langkah selanjutnya adalah membangkitkan identitas dari file video hasil scanning tersebut. Setelah identitas dari file video tersebut berhasil di dapatkan maka langkah selanjutnya adalah melakukan pengelompokkan file video berdasarkan identitas file video yang di bangkitkan menggunakan fungsi hash SHA-256.

Bulletin of Artificial Intelligence
Volume 1, No 1, April 2022 Page: 25−32
ISSN 2962-3944 (media online)
https://journal.grahamitra.id/index.php/buai

3.1 Penerapan Algoritma SHA-256

Langkah awal yang dilakukan ketika ingin melakukan pencarian *file* video yang ganda atau duplikat adalah melakukan *scanning* pada sebuah ruang penyimpanan yang di dalamnya terdapat banyak *file* video. Setelah melakukan proses *scanning* maka langkah selanjutnya adalah membangkitkan identitas dari *file* video hasil *scanning* tersebut. Setelah identitas dari *file* video tersebut berhasil di dapatkan maka langkah selanjutnya adalah melakukan pengelompokkan *file* video berdasarkan identitas *file* video yang di bangkitkan menggunakan fungsi *hash* SHA-256.Objek pada penelitian ini adalah *file* video dengan ekstensi format 9,29 MB. Untuk memudahkan proses analisa maka diambil sampel dari *file* video menggunakan aplikasi Binary Viwer dan diambil sampel sebanyak 25 byte.

Gambar 1. *File* video berekstensi MP4



Gambar 2. Nilai piksel citra sampel

Tabel 1. Input Nilai Biner

00000000	00000000	00000000	01000000	01100110
01110100	01111001	01110000	01101001	01110011
01101111	01101101	00000000	00000000	00000010
00000000	01101001	01110011	01101111	01101101
01101001	01110011	01101111	00110010	01100001

a. Penambahan *Padding Bit*

Padding bit di tambahkan karena algoritma SHA-256 membutuhkan minimal satu blok data *input* dengan panjang 512 bit, sehingga jika *input* kurang dari 512 bit maka di tambahkan *padding bit* yang di mulai dengan 1 selanjutnya di tambahkan 0.

$k = l + 1 = 448 \bmod 512$

$k = 200 + 1 = 448 \bmod 512$

$k = 201 = 448 \bmod 512$

$k = 448 - 201$

$k = 247$

Maka banyaknya *padding bit* 1 dan selanjutnya nilai 0 sebanyak 247 dapat di lihat pada tabel 2 di bawah ini:

Tabel 2. Penambahan *padding* bit

00000000	00000000	00000000	01000000	01100110	01110100	01111001	01110000
01101001	01110011	01101111	01101101	00000000	00000000	00000010	00000000
01101001	01110011	01101111	01101101	01101001	01110011	01101111	00110010
01100001	**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**

Bulletin of Artificial Intelligence
Volume 1, No 1, April 2022 Page: 25−32
ISSN 2962-3944 (media online)
https://journal.grahamitra.id/index.php/buai

b. Penambahan Panjang *Append*
Penambahan Panjang *Append*
Penambahan panjang *append* dilakukan dengan penambahan panjang data sebanyak 64 bit di akhir. Panjang data adalah 200bit sehingga ditambahkan panjang *append* 11001000 di akhir sebanyak 64bit sebagai berikut :

Tabel 3. Penambahan Panjang *Append*

00000000	00000000	00000000	01000000	01100110	01110100	01111001	01110000
01101001	01110011	01101111	01101101	00000000	00000000	00000010	00000000
01101001	01110011	01101111	01101101	01101001	01110011	01101111	00110010
01100001	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**	**11001000**

c. *Parsing* Data
Pada kasus ini panjang data tidak lebih dari 512 sehingga hanya menghasilkan 1 blok 512bit yaitu $M_0^{(0)}$sampai $M_{15}^{(0)}$. Tahap selanjutnya adalah melakukan *parsing* data dengan membagi setiap blok 512bit menjadi 16 blok berukuran 32 bit.

Tabel 4. *Parsing*Data

Data		Biner	Heksadesimal
$M_0^{(0)}$	:	00000000 00000000 00000000 01000000	00000020
$M_1^{(0)}$	:	01100110 01110100 01111001 01110000	66747970
$M_2^{(0)}$	:	01101001 01110011 01101111 01101101	69736F6D
$M_3^{(0)}$	:	00000000 00000000 00000010 00000000	00000200
$M_4^{(0)}$	:	01101001 01110011 01101111 01101101	69736F6D
$M_5^{(0)}$	:	01101001 01110011 01101111 00110010	69736F32
$M_6^{(0)}$	:	01100001 10000000 00000000 00000000	61800000
$M_7^{(0)}$	:	00000000 00000000 00000000 00000000	00000000
$M_8^{(0)}$	:	00000000 00000000 00000000 00000000	00000000
$M_9^{(0)}$	:	00000000 00000000 00000000 00000000	00000000
$M_{10}^{(0)}$	:	00000000 00000000 00000000 00000000	00000000
$M_{11}^{(0)}$	:	00000000 00000000 00000000 00000000	00000000
$M_{12}^{(0)}$	:	00000000 00000000 00000000 00000000	00000000
$M_{13}^{(0)}$	:	00000000 00000000 00000000 00000000	00000000
$M_{14}^{(0)}$	:	00000000 00000000 00000000 00000000	00000000
$M_{15}^{(0)}$	:	00000000 00000000 00000000 11001000	000000C8

d. Inisialisasi Nilai *Hash*

Setelah proses *parsing* data maka langkah selanjutnya adalah inisialisasi nilai *hash* di mana nilai ini merupakan sebuah ketentuan yaitu :

Tabel 5. Inisial *hash value*

Variabel	Hash Value
$H_0^{(0)}$	6A09E667
$H_1^{(0)}$	BB67EA85
$H_2^{(0)}$	3C6EF372
$H_3^{(0)}$	A54FF53A
$H_4^{(0)}$	510E527F
$H_5^{(0)}$	9B05688C
$H_6^{(0)}$	1F83D9AB
$H_7^{(0)}$	5BE0CD19

e. Penjadwalan Data
Kemudian dilakukan proses penjadwalan data, langkah ini diawali dengan mengubah setiap blok data menjadi bilangan heksadesimal dengan ketentuan sebagai berikut:

$$Wt = \begin{cases} M_t^{(t)} & 0 \le t \le 15 \\ \sigma_1^{(256)}(W_{i-2}) + W_{i-7} + \sigma_0^{(256)}(W_{i-15}) + W_{i-16}, & 16 \le t \le 63 \end{cases}$$

Bulletin of Artificial Intelligence
Volume 1, No 1, April 2022 Page: 25−32
ISSN 2962-3944 (media online)
https://journal.grahamitra.id/index.php/buai

Tabel 6. Penjadwalan Data

00000020	66747970	69736F6D	00000200	69736F6D	69736F32	61800000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	000000C8
F25E7E61	741FE75D	78AA7859	FDE95854	AE430D39	A114B7B3	45074A79	252E841D
C093B681	E61D5F0A	7F846B02	A9DED709	6879D2E3	93A19283	888E357F	C821696D
4A283965	E135983A	3E7B5C9B	AA976450	C64FBE21	F3B58E5A	843C9647	0EA3B334
86AB2E38	E6645E22	8AEAB1B5	1F9B7E24	DF36F59C	6C13F555	6B8331F5	A0551511
1F3B6B33	2A8AE118	C42AB6E7	0B4AADB5	4CBBF86A	BCC2FAC0	2CCE0171	7CB469FE
24CE7E01	9A758470	1B3B5078	78E0BF24	B83C0861	60B697EE	F5094FB5	1D440C8B

Untuk menjadwalkan data ke 16 sampai 63 dilakukan perhitungan sebagai berikut :
Data ke 16

$$\sigma_1^{(256)}(W_{i-2}) = ((W_{i-2})ROTR\ 17) \oplus ((W_{i-2})ROTR\ 19) \oplus ((W_{i-2})SHR10)$$

$((W_{16-2})ROTR\ 17)$ $= ((W_{14})ROTR\ 17)$
$= ((00000000)ROTR\ 17)$
$= (00000000)$

$((W_{16-2})ROTR\ 19)$ $= ((W_{14})ROTR\ 19)$
$= ((00000000)ROTR\ 19)$
$= (00000000)$

$((W_{16-2})SHR10)$ $= ((W_{14})SHR10)$
$= ((00000000)SHR10)$
$= (00000000)$

$\sigma_1^{(256)}(W_{i-2}) = (00000000) \oplus (00000000) \oplus (00000000)$
$= (\mathbf{00000000})$

W_{16-7} $= W_9$
$= \mathbf{00000000}$

$\sigma_0^{(256)}(W_{i-15}) = ((W_{i-15})ROTR\ 7) \oplus ((W_{i-15})ROTR\ 18) \oplus ((W_{i-15})SHR3)$

$((W_{16-15})ROTR\ 7)$ $= ((W_1)ROTR\ 7)$
$= ((66747970)ROTR\ 7)$
$= (\boldsymbol{E0CCE8F2})$

$((W_{16-15})ROTR\ 18)$ $= ((W_1)ROTR\ 18)$
$= ((66747970)ROTR\ 18)$
$= (\boldsymbol{1E5C199D})$

$((W_{16-15})SHR\ 3)$ $= ((W_1)SHR\ 3)$
$= ((66747970)SHR\ 3)$
$= (\boldsymbol{0CCE8F2E})$

$\sigma_0^{(256)}(W_{i-15}) = (E0CCE8F2) \oplus (1E5C199D) \oplus (0CCE8F2E)$
$= (\boldsymbol{F25E7E41})$

W_{16-16} $= W_0$
$= 00000020$

$$Wt = \sigma_1^{(256)}(W_{i-2}) + W_{i-7} + \sigma_0^{(256)}(W_{i-15}) + W_{i-16}$$

$Wt = 00000000 + 00000000 + F25E7E41 + 00000020$
$Wt = \boldsymbol{F25E7E61}$

Demikian seterusnya hingga W_{63}(i-1), dimana i adalah jumlah blok 512 bit.

f. Inisialisasi Variabel kerja

Selanjutnya melakukan inisialisasi variabel kerja *a, b, c, d, e, f, g* dan *h* di mana setiap variabel diambil dari *initial hash value a=H0(0), b=H1(0), c=H2(0) , d=H3(0), e=H4(0), f=H5(0), g=H6(0), h=H7(0).* Selanjutnya dilakukan proses komputasi fungsi *hash* SHA-256 dari t=0 sampai t=63.

Tabel 7. Inisialisasi Variabel Kerja

	a	***b***	***c***	***d***	***e***	***f***	***g***	***h***
Init	6A09E667	BB67AE85	3C6EF372	A54FF53A	510E527F	9B05688C	1F83D9AB	5BE0CD19
t=0	FC08886D	6A09E667	BB67AE85	3C6EF372	98C7E2C2	510E527F	9B05688C	1F83D9AB
t=1	892FA611	FC08886D	6A09E667	3C6EF372	8EEE4446	98C7E2C2	510E527F	9B05688C
t=2	E7F041A6	FC08886D	6A09E667	6A09E667	892FA611	8EEE4446	98C7E2C2	510E527F
t=3	C34B1F69	E7F041A6	FC08886D	6A09E667	C97E90F5	892FA611	8EEE4446	98C7E2C2
t=4	2E9B3788	C34B1F69	E7F041A6	FC08886D	AC1F6E79	C97E90F5	892FA611	8EEE4446
t=5	A88777A1	C4D8B985	2E9B3788	C34B1F69	E99E5A07	FDCC90C9	AC1F6E79	C97E90F5
t=6	295BCCD3	A88777A1	C4D8B985	2E9B3788	B5E98D03	E99E5A07	FDCC90C9	AC1F6E79
t=7	54EEB5AF	295BCCD3	A88777A1	C4D8B985	F6B60443	B5E98D03	E99E5A07	FDCC90C9
t=8	4459039B	54EEB5AF	295BCCD3	A88777A1	B1E2AD56	F6B60443	B5E98D03	E99E5A07

Bulletin of Artificial Intelligence
Volume 1, No 1, April 2022 Page: 25−32
ISSN 2962-3944 (media online)
https://journal.grahamitra.id/index.php/buai

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*
t=9	6DCE9582	4459039B	295BCCD3	2E9B3788	6C497961	B1E2AD56	F6B60443	B5E98D03
t=10	C59E9CBA	6DCE9582	4459039B	295BCCD3	79A78C0C	6C497961	B1E2AD56	F6B60443
t=11	2A072E6C	C59E9CBA	6DCE9582	4459039B	DEC104D9	79A78C0C	6C497961	B1E2AD56
t=12	1AF5550C	2A072E6C	C59E9CBA	6DCE9582	8C6690F3	DEC104D9	79A78C0C	6C497961
t=13	33883B7C	1AF5550C	2A072E6C	C59E9CBA	13F695B4	8C6690F3	DEC104D9	79A78C0C
t=14	D7A62F8A	33883B7C	1AF5550C	2A072E6C	6AD12A88	13F695B4	8C6690F3	DEC104D9
t=15	2EF9737F	9D715D26	708CB16D	81405D99	A3DD4CC9	3BA4CDD5	13F695B4	4ED11808
t=16	29F7D849	2EF9737F	9D715D26	708CB16D	B8B3ECC4	A3DD4CC9	3BA4CDD5	13F695B4
t=17	98C2ABB0	29F7D849	2EF9737F	9D715D26	B8B3ECC4	A3DD4CC9	A3DD4CC9	3BA4CDD5
t=18	6F0A7A63	98C2ABB0	29F7D849	2EF9737F	726D2D76	B8B3ECC4	A3DD4CC9	A3DD4CC9
t=19	BE3F9B3F	6F0A7A63	98C2ABB0	29F7D849	A245A8E6	726D2D76	B8B3ECC4	A3DD4CC9
t=20	F510C7D1	BE3F9B3F	6F0A7A63	98C2ABB0	98DEFB1C	A245A8E6	726D2D76	B8B3ECC4
t=21	BB4F3617	F510C7D1	BE3F9B3F	6F0A7A63	542227AE	98DEFB1C	A245A8E6	726D2D76
t=22	403CA1D1	BB4F3617	F510C7D1	F510C7D1	8D713C0C	542227AE	98DEFB1C	A245A8E6
t=23	C09018C8	403CA1D1	BB4F3617	F510C7D1	1881CC37	8D713C0C	542227AE	98DEFB1C
t=24	761D5281	C09018C8	403CA1D1	BB4F3617	82C92475	1881CC37	8D713C0C	542227AE
t=25	CA67107C	761D5281	C09018C8	403CA1D1	88D35440	82C92475	1881CC37	8D713C0C
t=26	5A570BA8	CA67107C	761D5281	C09018C8	AB4038A3	88D35440	82C92475	1881CC37
t=27	7DB9BC74	5A570BA8	CA67107C	761D5281	CBF91159	AB4038A3	88D35440	82C92475
t=28	9CA63F64	7DB9BC74	5A570BA8	CA67107C	9E102643	CBF91159	AB4038A3	88D35440
t=29	333CE141	9CA63F64	7DB9BC74	5A570BA8	5BFBB9BF	9E102643	CBF91159	AB4038A3
t=30	4BF39CD5	333CE141	9CA63F64	7DB9BC74	B34A459E	5BFBB9BF	9E102643	CBF91159
t=31	5F48B2F4	4BF39CD5	333CE141	9CA63F64	4725C49D	B34A459E	5BFBB9BF	9E102643
t=32	3BC552D6	5F48B2F4	4BF39CD5	333CE141	DA37D9E0	4725C49D	B34A459E	5BFBB9BF
t=33	587E1829	3BC552D6	5F48B2F4	4BF39CD5	22ED9426	DA37D9E0	4725C49D	B34A459E
t=34	2D47743C	587E1829	3BC552D6	5F48B2F4	AEB91D82	22ED9426	DA37D9E0	4725C49D
t=35	BC395433	2D47743C	587E1829	3BC552D6	2ADA6F6A	AEB91D82	22ED9426	DA37D9E0
t=36	8031664F	BC395433	2D47743C	587E1829	D3B3EAB6	2ADA6F6A	AEB91D82	22ED9426
t=37	70042FD2	8031664F	BC395433	2D47743C	04676DA4	D3B3EAB6	2ADA6F6A	AEB91D82
t=38	C330E0D2	70042FD2	8031664F	BC395433	BEF99FBE	04676DA4	D3B3EAB6	2ADA6F6A
t=39	5F74E86A	C330E0D2	70042FD2	8031664F	E5A46B0C	BEF99FBE	04676DA4	D3B3EAB6
t=40	E1B8B3B4	5F74E86A	C330E0D2	70042FD2	0786C871	E5A46B0C	BEF99FBE	04676DA4
t=41	232EB172	E1B8B3B4	5F74E86A	C330E0D2	88FA0DA4	0786C871	E5A46B0C	BEF99FBE
t=42	84182DF7	232EB172	E1B8B3B4	5F74E86A	2A6CDFB2	88FA0DA4	0786C871	E5A46B0C
t=43	3C222634	84182DF7	232EB172	E1B8B3B4	0C5066FC	2A6CDFB2	88FA0DA4	0786C871
t=44	B4FDA70D	3C222634	84182DF7	232EB172	BC5A7CDF	0C5066FC	2A6CDFB2	88FA0DA4
t=45	17400AAE	B4FDA70D	3C222634	84182DF7	E2679BEE	BC5A7CDF	0C5066FC	2A6CDFB2
t=46	E423CA3E	17400AAE	B4FDA70D	3C222634	84182DF7	E2679BEE	BC5A7CDF	0C5066FC
t=47	BEF76151	E423CA3E	17400AAE	B4FDA70D	DEE0D556	84182DF7	E2679BEE	BC5A7CDF
t=48	9A07816F	BEF76151	E423CA3E	17400AAE	DFEC752A	DEE0D556	84182DF7	E2679BEE
t=49	55260CF5	9A07816F	BEF76151	E423CA3E	BA45C815	DFEC752A	DEE0D556	84182DF7
t=50	62AA43B5	55260CF5	9A07816F	BEF76151	FDD60D25	BA45C815	DFEC752A	DEE0D556
t=51	04CDB8A6	62AA43B5	55260CF5	9A07816F	617EB8F4	FDD60D25	BA45C815	DFEC752A
t=53	E0E09B9A	04CDB8A6	62AA43B5	55260CF5	F13C92CC	617EB8F4	FDD60D25	BA45C815
t=54	EC2CCB72	E0E09B9A	04CDB8A6	62AA43B5	F9E8F250	F13C92CC	617EB8F4	FDD60D25
t=55	0A176C61	EC2CCB72	E0E09B9A	04CDB8A6	34237C5A	F9E8F250	F13C92CC	617EB8F4
t=56	42A2AC38	0A176C61	EC2CCB72	E0E09B9A	E30F89E1	34237C5A	F9E8F250	F13C92CC
t=57	BFB0EFA3	42A2AC38	0A176C61	EC2CCB72	5A903FBC	E30F89E1	34237C5A	F9E8F250
t=58	AB110644	BFB0EFA3	42A2AC38	0A176C61	3B3B9D04	5A903FBC	E30F89E1	34237C5A
t=59	4A2D205A	AB110644	BFB0EFA3	42A2AC38	4B9BD2E6	3B3B9D04	5A903FBC	E30F89E1
t=60	1F3E5505	4A2D205A	AB110644	BFB0EFA3	91D76AA4	4B9BD2E6	3B3B9D04	5A903FBC
t=61	DAA3CE5B	1F3E5505	4A2D205A	AB110644	786440EB	91D76AA4	4B9BD2E6	3B3B9D04
t=62	EB4BD9C8	DAA3CE5B	1F3E5505	4A2D205A	30DE51CF	786440EB	91D76AA4	4B9BD2E6
t=63	01C2B52D	EB4BD9C8	DAA3CE5B	1F3E5505	94D168BD	30DE51CF	786440EB	91D76AA4

Untuk *t=0* lakukan perhitungan sebagai berikut :

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*
6A09E667	BB67AE85	3C6EF372	A54FF53A	510E527F	9B05688C	1F83D9AB	5BE0CD19

$$T_1 = h + \sum_1^{(256)}(e) + Ch(e,f,g) + K_t^{(256)} + W_t$$

$$h = \text{5BE0CD19}$$

$$\sum_1^{(256)}(e) = (e\ ROTR\ 6) \oplus (e\ ROTR\ 11) \oplus (e\ ROTR\ 25)$$

$$\textstyle\sum_1^{(256)}(e) = ((510E527F) ROTR\ 6) \oplus ((510E527F)\ ROTR\ 11) \oplus ((510E527F)\ ROTR\ 25)$$

$$\sum_1^{(256)}(e) = (FD443949) \oplus (4FEA21CA) \oplus (87293FA8)$$

$$\sum_1^{(256)}(e) = 3587272B$$

$$Ch(e,f,g) \quad = (e\ \wedge f) \oplus (\sim e\ \wedge g)$$

Bulletin of Artificial Intelligence
Volume 1, No 1, April 2022 Page: 25−32
ISSN 2962-3944 (media online)
https://journal.grahamitra.id/index.php/buai

$Ch(e,f,g)$ = $(510E527F \text{ ^ } 9B05688C) \oplus (\sim 510E527F \text{ ^ } 1F83D9AB)$
$Ch(e,f,g)$ = $(1104400C) \oplus (0E818980)$
$Ch(e,f,g)$ = $1F85C98C$
$K_0^{(256)}$ = 428A2F98
W_t = 00000020
T_1 = $5BE0CD19 + 3587272B + 1F85C98C + 428A2F98 + 00000020$
T_1 = **F377ED88**

$T_2 = \sum_0^{(256)}(a) + Maj(a,b,c)$
$\sum_0^{(256)}(a) = ((6A09E667)ROTR\ 2) \oplus ((6A09E667)ROTR\ 13) \oplus ((6A09E667)ROTR\ 22)$
$\sum_0^{(256)}(a) = (DA827999) \oplus (333B504F) \oplus (27999DA8)$
$\sum_0^{(256)}(a) = CE20B47E$

$Maj(a,b,c)$ = $(a \text{ ^ } b) \oplus (a \text{ ^ } c) \oplus (b \text{ ^ } c)$
$Maj(a,b,c)$ = $(6A09E667 \text{^} BB67AE85) \oplus (6A09E667 \text{^} 3C6EF372) \oplus (BB67AE85 \text{^} 3C6EF372)$
$Maj(a,b,c)$ = $(2A01A605) \oplus (2808E262) \oplus (3866A200)$
$Maj(a,b,c)$ = 3A6FE667
$T_2 = CE20B47E + 3A6FE667$
$T_2 = \mathbf{08909AE5}$
h =g
=1F83D9AB
g =f
= 9B05688C
f =e
= 510E527F
e =$d + T_1$
=A54FF53A + F377ED88
= 98C7E2C2
d =c
=3C6EF372
c =b
=BB67AE85
b =a
=6A09E667
a =$T_1 + T_2$
= F377ED88 + 08909AE5
= FC08886D

g. Menghitung *IntermediateHashValue*

Tabel 8. Penjumlahan dengan *initial hash value*

Variabel	*Initial Hash Value*		*Variabel Kerja*	*Hasil*
$H_0^{(0)}$	01C2B52D	+	6A09E667	6BCC9B94
$H_1^{(0)}$	EB4BD9CB	+	BB67AE85	A6B3884D
$H_2^{(0)}$	DAA3CE5B	+	3C6EF372	1712C1CD
$H_3^{(0)}$	1F3E5505	+	A54FF53A	C48E4A3F
$H_4^{(0)}$	94D168BD	+	510E527F	E5DFBB3C
$H_5^{(0)}$	30DE51CF	+	9B05688C	CBE3BA5B
$H_6^{(0)}$	786440EB	+	1F83D9AB	97E81A96
$H_7^{(0)}$	91D76AA4	+	5BE0CD19	EDB837BD

h. Output

Output dari SHA-256 merupakan penggabungan dari *H0(0)* sampai *H7(0)* sebagai berikut :

6BCC9B94 || A6B3884D || 1712C1CD || C48E4A3F || E5DFBB3C || CBE3BA5B 9 || 97E81A96 || EDB837BD

Bulletin of Artificial Intelligence
Volume 1, No 1, April 2022 Page: 25−32
ISSN 2962-3944 (media online)
https://journal.grahamitra.id/index.php/buai

4. KESIMPULAN

Kesimpulan yang dapat diambil dari hasil perancanagan aplikasi duplicate video Scanner menerapkan algoritma sha-256 Dalam mencari file duplicate video scenner dapat memberikan identitas dari setiap file video dan melakukan penghapusan yang di dalamnya terdapat banyak file video. Dengan menarapkan algoritma SHA-256 dapat membantu dalam setiap file video yang duplicate. Untuk menambah wawasan dan pengetahuan, bisa mempelajari kelemahan aplikasi duplicate video scanner dan menjadi bahan referensi bagi penelitian.

REFERENCES

[1] Imam Saputra, "Analisa Algoritma SHA-256 Untuk Mendekteksi Orisinalitas Citra Digital," In Seminar Nasional Riset Information Science (SENARIS), 2019.
[2] S.J Patil, N.P Jagtap, S.H Rajput, and R.B Sangore,, "A Duplicate File Finder System," International Journal of Scienci Spirituality Business and Tecnology, pp. 10-14, 2017.
[3] Hellman, Whitfield Diffie and Martin E, "New Direction in Cryptography," Vol.dari IT 22,no 6, November 1976.
[4] D. Ariyus, PengantarIlmu Kriptografi, 2008.
[5] Prayudi, Yudi, and Idham Halik, "Studi Analisis Algoritma Rivest Code 6 (RC6) Dalam Enkripsi/Dekripsi Data," Seminar Nasional Aplikasi Teknologi Informasi 2005(SNATI 2005, 2005.
[6] M Syafriadi, "Analisa Kecepatan dan Keamanan Algoritma Secure Hash Algoritma 256 (SHA-256) Untuk Otentikasi pesan Teks," 2006.
[7] Booch, G James, and R Ivar, "The Unified Modeling Language User Guide Second Edition," addison Wesley Profesional, 2005.
[8] Adi Nugroho,, "Rekayasa Perangkat Lunak Menggunakan UML dan Java," Andi Offset, 2010.
[9] A. R. a. M. Shalahuddin, "Rekayasa Perangkat Lunak Struktur dan berorientasi Objek," Informatika, 2014.