JUSSI: Jurnal Sains Dan Teknologi Informasi
Vol 1, No 4, September 2022, Hal 112-120
ISSN 2809-610X (Media Online)
Website https://ejurnal.seminar-id.com/index.php/jussi

Implementasi Fungsi Hash Dengan Algoritma Sha-256 Pada Aplikasi Duplicate Image Scanner

Nova Ariska

Program Studi Tenik Informatika, Fakultas Ilmu Komputer Dan Teknologi Informasi, Universitas Budi Darma,
Jalan Sisingamangaraja No.338, Medan, Sumatera Utara, Indonesia
Email: ariskanova8@gmail.com

Abstrak−Saat ini kita sudah berada dalam era dunia digital, banyak sekali peralatan yang memiliki alat untuk merekam foto berupa kamera yang terdapat pada smartphone. Kelemahan dari smartphone adalah ruang penyimpanan yang kecil, sehingga pengguna harus pandai memanajemen file yang tersimpan di dalam smartphone tersebut. Banyak sekali aplikasi dalam smartphone yang menyajikan fasilitas untuk berbagi gambar. Hal ini tentunya akan sangat membebani ruang penyimpanan dari smartphone tersebut karena menyimpan file gambar yang umumnya berukuran besar, terlebih jika file gambar tersebut sama atau duplikat, tentunya hal itu sangat membebani ruang penyimpanan. Akan sangat sulit untuk membedakan file gambar tanpa melihat isi dari file gambar tersebut. Sehingga jika ingin membedakan satu file gambar dengan yang lainnya maka pengguna harus melihat isi file gambar tersebut dan ini tentunya akan sangat menyulitkan. Untuk itu diperlukan suatu cara yang dapat digunakan untuk mengidentifikasi file gambar yang sama ataupun file gambar yang duplikat sehingga dapat diambil keputusan untuk menghapus file dokumen yang duplikat tersebut sehingga dapat menghemat ruang penyimpanan. Dalam kriptografi terdapat fungsi hash yang merupakan fungsi satu arah yang dapat membangkitkan identitas sebuah file, dimana jika file gambar itu isinya sama maka akan menghasilkan nilai hash yang sama pula. Hal ini dapat digunakan untuk mengidentifikasi file gambar yang sama ataupun duplikat. Dalam fungsi hash terdapat metode SHA-256 ataupun Secure Hash Algorithm 256 yang merupakan salah satu algoritma hashing yang sering digunakan untuk mengenkripsi data dengan panjang 256 bit. Dengan menerapkan algoritma ini pada aplikasi file scanner maka hasil dari pencarian file gambar tersebut di kelompokkan berdasarkan nilai hash yang sama dan pengguna dapat memudahkan dan mempercepat pengguna untuk menghapus file gambar yang duplikat tanpa harus membuka dan melihat isi file tersebut.

Kata Kunci: Kriptografi; Algoritma SHA-256; Fungsi Hash; File Gambar; Scanner

Abstract−Nowadays we are in the era of the digital world, there are lots of equipment that has a tool for recording photos in the form of a camera on a smartphone. The weakness of smartphones is the small storage space, so users must be good at managing files stored on the smartphone. There are so many applications in smartphones that provide facilities for sharing images. This of course will greatly burden the storage space of the smartphone because it stores image files which are generally large in size, especially if the image files are the same or duplicate, of course it is very burdensome for storage space. It will be very difficult to distinguish an image file without looking at the contents of the image file. So if you want to distinguish one image file from another, the user must see the contents of the image file and this will certainly be very difficult. For that we need a method that can be used to identify the same image file or duplicate image files so that a decision can be made to delete the duplicate document files so as to save storage space. In cryptography there is a hash function which is a one-way function that can generate the identity of a file, where if the image file contains the same content, it will produce the same hash value. It can be used to identify the same or duplicate image files. In the hash function there is the SHA-256 method or the Secure Hash Algorithm 256 which is one of the hashing algorithms that is often used to encrypt data with a length of 256 bits. By applying this algorithm to the file scanner application, the results of the image file search are grouped based on the same hash value and users can make it easier and faster for users to delete duplicate image files without having to open and view the contents of the file.

Keywords: Cryptography; SHA-256 algorithm; Hash Functions; Image Files; Scanner

1. PENDAHULUAN

Kriptografi berperan penting dalam layanan keamanan seperti kerahasiaan, integritas data, otentikasi dan pencegahan penyangkalan. Kriptografi modern menggunakan kunci yang harus dirahasiakan untuk mengatasi masalah keamanan kriptografi. Permasalahan dalam penggunaan satu kunci yang sama oleh dua entitas yang saling berkomunikasi untuk bertukar pesan adalah cara mendistribusikan kunci. Permasalahan ini dapat diatasi dengan penggunaan kriptografi kunci-public, yang memungkinkan pengguna berkomunikasi secara aman tanpa perlu berbagi kunci rahasia [1].

Saat ini kita sudah berada dalam era dunia digital, banyak sekali peralatan yang memiliki alat untuk merekam foto berupa kamera yang terdapat pada smartphone. Kelemahan dari smartphone adalah ruang penyimpanan yang kecil, sehingga pengguna harus pandai memanajemen file yang tersimpan di dalam smartphone tersebut. Banyak sekali aplikasi dalam smartphone yang menyajikan fasilitas untuk berbagi gambar. Hal ini tentunya akan sangat membebani ruang penyimpanan dari smartphone tersebut karena menyimpan file gambar yang umumnya berukuran besar, terlebih jika file gambar tersebut sama atau duplikat, tentunya hal itu sangat membebani ruang penyimpanan. Akan sangat sulit untuk membedakan file gambar tanpa melihat isi dari file gambar tersebut. Sehingga jika ingin membedakan satu file gambar dengan yang lainnya maka pengguna harus melihat isi file gambar tersebut dan ini tentunya akan sangat menyulitkan. Untuk itu diperlukan suatu cara yang dapat digunakan untuk mengidentifikasi file gambar yang sama ataupun file gambar yang duplikat sehingga dapat diambil keputusan untuk menghapus file gambar yang duplikat tersebut sehingga dapat menghemat ruang penyimpanan[2].

Dalam kriptografi terdapat fungsi hash yang merupakan fungsi satu arah yang dapat membangkitkan identitas sebuah file, dimana jika file gambar itu isinya sama maka akan menghasilkan nilai hash yang sama pula. Hal ini dapat digunakan untuk mengidentifikasi file gambar yang sama ataupun duplikat. Dalam fungsi hash terdapat metode SHA-256

JUSSI: Jurnal Sains Dan Teknologi Informasi
Vol 1, No 4, September 2022, Hal 112-120
ISSN 2809-610X (Media Online)
Website https://ejurnal.seminar-id.com/index.php/jussi

ataupun Secure Hash Algorithm 256 yang merupakan salah satu algoritma hashing yang sering digunakan untuk mengenkripsi data dengan panjang 256 bit. Dengan menerapkan algoritma ini pada aplikasi file scanner maka hasil dari pencarian file gambar tersebut di kelompokkan berdasarkan nilai hash yang sama dan pengguna dapat memudahkan dan mempercepat pengguna untuk menghapus file gambar yang duplikat tanpa harus membuka dan melihat isi file tersebut [3].

Berdasarkan permasalahan diatas maka penulis tertarik untuk mengangkat judul "Implementasi Fungsi Hash Dengan Algoritma SHA-256 Pada Aplikasi Duplicate Image Scanner". Dengan menerapkan algoritma SHA-256 pada aplikasi file scanner maka hasil dari pencarian file gambar tersebut dikelompokkan berdasarkan nilai hash yang sama dan pengguna dapat memudahkan dan mempercepat pengguna untuk menghapus file gambar yang duplikat tanpa harus membuka file tersebut.

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Untuk mendukung kelancaran penelitian ini, maka dilakukan tahapan penelitian sebagai berikut:

a. Studi pustaka
   Merupakan bentuk penelitian yang dilakukan berdasarkan kepustakaan atau buku-buku, jurnal, dan sumber-sumber sejenis (tertulis maupun dokumen) lain yang mempunyai hubungan dengan masalah yang sedang dibahas
b. Analisa dan Perancangan
   Analisa digunakan setelah data-data yang diperoleh dari studi pustaka kemudian dilakukan analisa sesuai dengan permasalahan penulis dalam penelitian ini
c. Perancangan dan Implementasi
   Tahap ini merupakan perancangan serta membangun aplikasi dan mengimplementasikan algoritma Algoritma SHA-256 yang digunakan dan aplikasi Duplicate Image Scanner yang dibangun pada penelitian ini..
d. Pengujian
   Pada tahap ini dilakukan Pengujian dari teori atau data yang telah dirancang dan diimplementasikan guna menguji kebenarannya.
e. Dokumentasi
   Pada tahap ini seluruh kegiatan dalam pembuatan sistem didokumentasikan kedalam bentuk tulisan berupa laporan penelitian.

2.2 *Secure Hash Algorithm* 256 (SHA-256)

Secure Hash Algorithm(SHA) 256 merupakan fungsi *hash* yang umum digunakan, sampai saat ini belum ada yang dapat memecahkan algoritma fungsi *hash* SHA-256. Algoritma SHA-256 memiliki 8 langkah pengerjaan yaitu sebagai berikut [5]:

a. Tambahkan bit *Padding*
   Pesan diisi sehingga panjangnya kongruen dengan 448, modulus 512. *Padding* 1bit ditambahkan di akhir pesan, diikuti oleh banyaknya nol yang diperlukan sehingga panjang bit sama dengan 448 modulus 512.
b. Panjang *Append*
   Representasi panjang pesan 64bit ditambahkan pada hasil akhirnya, langkah ini untuk membuat panjang pesan kelipatan 512 bit.
c. *Parsing* Pesan
   Pesan *padding* diuraikan menjadi N blok pesan 512 bit, M(1), M(2),...M(N), dengan menambahkan blok 64 bit.
d. Inisialisasi Nilai *Hash*
   Nilai *hash* awal, H(0) diatur, terdiri dari delapan kata 32 bit, dalam bentuk heksadesimal.

Tabel 1. *Initial Hash Value*

Variabel	*Initial 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

Sumber: Federal Information Processing Standards Publications, 2002 [6]

e. Mempersiapkan jadwal pesan

JUSSI: Jurnal Sains Dan Teknologi Informasi
Vol 1, No 4, September 2022, Hal 112-120
ISSN 2809-610X (Media Online)
Website https://ejurnal.seminar-id.com/index.php/jussi

SHA-256 menggunakan jadwal pesan enam puluh empat kata 32bit, kata-kata dari jadwal pesan diberi label W0, W1, ..., W63[6].

$$Wt = \begin{cases} M_t^{(t)} & 0 \le t \le 15 \\ \sigma_1^{(256)}(W_{i-2}) + W_{i-7} + \sigma_0^{(256)}(W_{i-15}) + W_{i-16}, & 16 \le t \le 63 \end{cases} \quad (1)$$

Di mana:

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

$$\sigma_0^{(256)}(W_{i-15}) = \big((W_{i-15})ROTR\ 7\big) \oplus \big((W_{i-15})ROTR\ 18\big) \oplus ((W_{i-15})SHR3) \quad (3)$$

Wt = Blok pesan yang baru

Mt = Blok pesan yang lama
W_{i-2} = Blok pesan dari *W* ke i-2
W_{i-15} = Blok pesan dari *W* ke i-15
ROTR = *Rotate Right*
SHR = *Shift Right*
⊕ = Operator *XOR*

f. Inisialisasi delapan variabel kerja *a*, *b*, *c*, *d*, *e*, *f*, *g*, dan *h* dengan nilai *hash* (i-1)[6]
For t= 0 *to* 63
{

$$T_1 = h + \sum_1^{(256)}(e) + Ch(e,f,g) + K_1^{(256)} + W_t \quad (4)$$

$$T_2 = \sum_0^{(256)}(a) + Maj(a,b,c) \quad (5)$$

$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

}
Di mana:

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

$$\sum_0^{(256)}(a) = (a\ ROTR\ 2) \oplus (a\ ROTR\ 13) \oplus (a\ ROTR\ 22) \quad (7)$$

$$Ch(e,f,g) = (e\ \wedge\ f) \oplus (\sim e\ \wedge\ g) \quad (8)$$

$$Maj(a,b,c) = (a\ \wedge\ b) \oplus (a\ \wedge\ c) \oplus (b\ \wedge\ c) \quad (9)$$

a, *b*, *c*, *d*, *e*, *f*, *g*, *h* = Variabel yang berisi pesan heksadesimal
$K_1^{(256)}$ = Konstansta SHA-256
ROTR = *Rotate Right*
⊕ = Operator *XOR*
∧ = Operator *AND*

Tabel 2. Konstanta SHA-256

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

Sumber : Federal Information Processing Standards Publications, 2002 [6]

g. Menjumlahkan hasil akhir *a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*dengan inisial *hash value* $H^{(i)}$
$\boldsymbol{H_0^{(i)}}$ = $\boldsymbol{a + H_0^{(i)}}$

JUSSI: Jurnal Sains Dan Teknologi Informasi
Vol 1, No 4, September 2022, Hal 112-120
ISSN 2809-610X (Media Online)
Website https://ejurnal.seminar-id.com/index.php/jussi

$$H_1^{(i)} = b + H_1^{(i)}$$
$$H_2^{(i)} = c + H_2^{(i)}$$
$$H_3^{(i)} = d + H_3^{(i)}$$
$$H_4^{(i)} = e + H_4^{(i)}$$
$$H_5^{(i)} = f + H_5^{(i)}$$
$$H_6^{(i)} = g + H_6^{(i)}$$
$$H_7^{(i)} = h + H_7^{(i)}$$

h. *Output*
Setelah mengulangi langkah 1 hingga 4 sebanyak N kali, fungsi *hash* yang dihasilkan adalah sebagai berikut:
$H_0^{(N)} \,||\, H_1^{(N)} \,||\, H_2^{(N)} \,||\, H_3^{(N)} \,||\, H_4^{(N)} \,||\, H_5^{(N)} \,||\, H_6^{(N)} \,||\, H_7^{(N)}$

2.3 *Duplicate Image Scanner*

Dalam proses pencarian *file* yang duplikat atau ganda sebagian besar menggunakan algoritma fungsi *hash* standar[7]. Algoritma fungsi *hash* bertindak sebagai enkripsi satu arah dan menghasilkan identitas dari sebuah citra dan setiap citra menghasilkan nilai *hash* yang unik. Untuk mendeteksi citra digital yang duplikat atau ganda maka di lakukan perbandingan antara nilai *hash* dari masing-masing citra digital. Jika terdapat citra digital yang memiliki nilai *hash* yang sama berarti citra digital tersebut ganda atau duplikat. Ada beberapa teknik yang digunakan untuk mencari citra digital yang ganda atau duplikat yaitu [7]:

a. Pencarian *checksum*
Pencarian jenis ini akan mencari *file* dengan nama, ekstensi dan ukuran dengan panjang *checksum* 128bit. Cara ini sangat cepat dan memiliki tingkat akurasi yang tinggi.
b. Pencocokan *file* dengan ekstensi berbeda
Pencarian jenis ini akan mencocokkan *byte* demi *byte* dan akan mencari *file* dengan ekstensi yang berbeda.
c. Pencarian berdasarkan konten
Pencarian ini dilakukan dengan mencocokkan *byte* demi *byte* dan dengan mengambil sampel yang berbeda dalam sebuah *file* dan membandingkannya dengan sampel *file* yang lain.

2.4 Citra Digital

Citra digital merupakan citra yang menunjuk pada pemrosesan gambar 2 dimensi menggunakan komputer. Dalam konteks yang lebih luas, pengolahan citra digital mengacu pada mengacu pada pemrosesan setiap data 2 dimensi. Citra digital merupakan sebuah larik (*array*) yang berisi nilai-nilai *real* maupun kompleks yang direpresentasikan dengan deretan bit tertentu [8].

3. HASIL DAN PEMBAHASAN

3.1 Analisa Masalah *Duplicate Image Scanner*

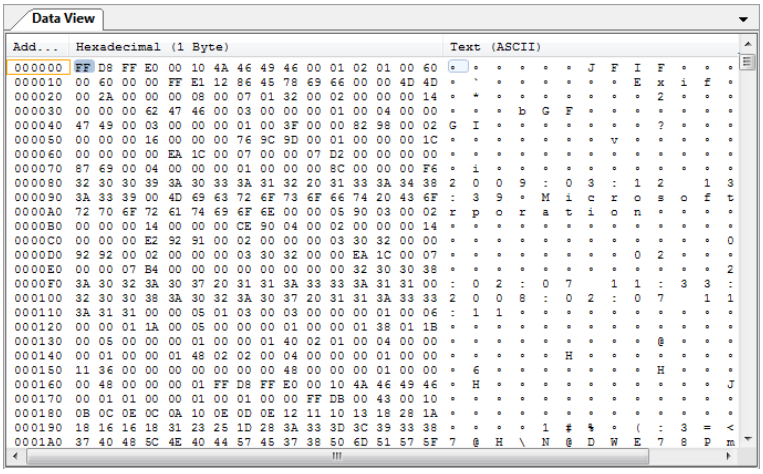
Masalah yang dihadapi ketika akan melakukan pencarian *file* gambar yang duplikat atau ganda pada sebuah media penyimpanan membutuhkan ketelitian dan ingatan yang kuat dari penggunanya karena harus melihat isi *file* satu persatu. Masalah ini dapat diatasi dengan cara memberikan identitas dari setiap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

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

Gambar 1. Bagan Alir Proses Pencarian File Gambar Duplikat

Objek pada penelitian ini adalah *file* gambar dengan ekstensi jpg berukuran 606 KB. Untuk memudahkan proses analisa maka diambil sampel dari *file*gambar menggunakan aplikasi Binary Viewerdan diambil sampel sebanyak 25 byte.

JUSSI: Jurnal Sains Dan Teknologi Informasi
Vol 1, No 4, September 2022, Hal 112-120
ISSN 2809-610X (Media Online)
Website https://ejurnal.seminar-id.com/index.php/jussi



Gambar 2. Nilai sampel *file*gambar

Nilai-nilai dalam bentuk heksadesimal tersebut yang akan diproses dengan menggunakan Algoritma SHA-256 untuk mengetahui identitas dari sebuah *file*gambar itu sendiri.

Tabel 2. Nilai File Gambar 25 Byte

Address (heksadesimal)	Value Heksadesimal	Address (heksadesimal)	Value Heksadesimal
00 00 00	**FF**	**00 00 0D**	**01**
00 00 01	**D8**	**00 00 0E**	**00**
00 00 02	**FF**	**00 00 0F**	**60**
00 00 03	**E0**	**00 00 10**	**00**
00 00 04	**00**	**00 00 11**	**60**
00 00 05	**10**	**00 00 12**	**00**
00 00 06	**4A**	**00 00 13**	**00**
00 00 07	**46**	**00 00 14**	**FF**
00 00 08	**49**	**00 00 15**	**E1**
00 00 09	**46**	**00 00 16**	**12**
00 00 0A	**00**	**00 00 17**	**86**
00 00 0B	**01**	**00 00 18**	**45**
00 00 0C	**02**	**00 0 19**	**78**

3.2 Penerapan Algoritma SHA-256

Berikut ini langkah-langkah penerapan metode Algoritma SHA-256 untuk mengetahui duplikat *file*gambar, dimana sebelumnya terlebih dahulu dilakukan pengubahan nilai heksadesimal dari *file* gambar menjadi bilangan biner.

Taebel 3. Input Nilai Biner

11111111	11011000	11111111	11100000	00000000
00010000	01001010	01000110	01001001	01000110
00000000	00000001	00000010	00000001	00000000
01100000	00000000	01100000	00000000	00000000
11111111	11100001	00010010	10000110	01000101

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 mod 512
k = 200 + 1 = 448 mod 512
k = 201= 448 mod 512
k = 448 – 201
k = 247
Maka banyaknya *padding bit* 1 dan selanjutnya nilai 0 sebanyak 247 dapat di lihat pada tabel 4 di bawah ini:

Tabel 4. Penambahan *Padding*Bit

11111111	11011000	11111111	11100000	00000000	00010000	01001010	01000110
01001001	01000110	00000000	00000001	00000010	00000001	00000000	01100000
00000000	01100000	00000000	00000000	11111111	11100001	00010010	10000110

JUSSI: Jurnal Sains Dan Teknologi Informasi
Vol 1, No 4, September 2022, Hal 112-120
ISSN 2809-610X (Media Online)
Website https://ejurnal.seminar-id.com/index.php/jussi

01000101	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

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

Tabel 5. Penambahan Panjang *Append*

11111111	11011000	11111111	11100000	00000000	00010000	01001010	01000110
01001001	01000110	00000000	00000001	00000010	00000001	00000000	01100000
00000000	01100000	00000000	00000000	11111111	11100001	00010010	10000110
01000101	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*Pesan
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* pesan dengan membagi setiap blok 512bit menjadi 16 blok berukuran 32 bit.

Tabel 6. .*Parsing* Pesan

Data	Biner	Hexadesimal
$M_0^{(0)}$	11111111 11011000 11111111 11100000	FFD8FFE0
$M_1^{(0)}$	00000000 00010000 01001010 01000110	00104A46
$M_2^{(0)}$	01001001 01000110 00000000 00000001	49460001
$M_3^{(0)}$	00000010 00000001 00000000 01100000	02010060
$M_4^{(0)}$	00000000 01100000 00000000 00000000	00600000
$M_5^{(0)}$	11111111 11100001 00010010 10000110	FFE11286
$M_6^{(0)}$	01000101 10000000 00000000 00000000	45800000
$M_7^{(0)}$	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 7. 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:

JUSSI: Jurnal Sains Dan Teknologi Informasi
Vol 1, No 4, September 2022, Hal 112-120
ISSN 2809-610X (Media Online)
Website https://ejurnal.seminar-id.com/index.php/jussi

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

Untuk menjadwalkan data ke 16 sampai 63 dilakukan perhitungan sebagaiberikut:

Data ke 16

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

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

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

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

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

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

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

$((W_{16-15})ROTR\ 7) = ((W_1)ROTR\ 7)$
$= ((00104A46)ROTR\ 7)$
$= (8C002094)$

$((W_{16-15})ROTR\ 18) = ((W_1)ROTR\ 18)$
$= ((00104A46)ROTR\ 18)$
$= (12918004)$

$((W_{16-15})SHR\ 3) = ((W_1)SHR\ 3)$
$= ((00104A46)SHR\ 3)$
$= (00020948)$

$\sigma_0^{(256)}(W_{i-15}) = (8C002094) \oplus (12918004) \oplus (00020948)$
$= (\mathbf{12918004})$

$W_{16-16} = W_0$
$= \mathbf{FFD8FFE0}$

$Wt = \sigma_1^{(256)}(W_{i-2}) + W_{i-7} + \sigma_0^{(256)}(W_{i-15}) + W_{i-16}$
$Wt = 00000000 + 00000000 + 12918004 + FFD8FFE0$
$Wt = \mathbf{126A7FE4}$

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.

Untuk t=0 lakukan perhitungan sebagai berikut :

a	***b***	***c***	***D***	***E***	***F***	***g***	***h***
6A09E667	BB67AE85	3C6EF372	A54FF53A	510E527F	9B05688C	1F83D9AB	5BE0CD19

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

$h = 5BE0CD19$

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

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

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

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

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

$Ch(e,f,g) = (510E527F\ ^\wedge\ 9B05688C) \oplus (\sim 510E527F\ ^\wedge\ 1F83D9AB)$

JUSSI: Jurnal Sains Dan Teknologi Informasi
Vol 1, No 4, September 2022, Hal 112-120
ISSN 2809-610X (Media Online)
Website https://ejurnal.seminar-id.com/index.php/jussi

$Ch(e,f,g)$ = (1104400C)⊕(0E818980)
$Ch(e,f,g)$ = 1F85C98C
$K_0^{(256)}$ = 428A2F98
W_t = FFD8FFE0
T_1 = 5BE0CD19 + 3587272B + 1F85C98C + 428A2F98 + FFD8FFE0
T_1 =**F350ED48**

$T_2 = \sum_0^{(256)}(a) + Maj(a,b,c)$
$\sum_0^{(256)}(a) = ((6A09E667)ROTR\ 2) \oplus ((6A09E667)ROTR\ 13) \oplus ((6A09E667)ROTR\ 22)$
$\sum_0^{(256)}(a) = (DA827999) \oplus (333B504F) \oplus (27999DA8)$
$\sum_0^{(256)}(a) = CE20B47E$
$Maj(a,b,c)$ = $(a \wedge b) \oplus (a \wedge c) \oplus (b \wedge c)$
$Maj(a,b,c)$ =(6A09E667^BB67AE85)⊕(6A09E667^3C6EF372)⊕(BB67AE85^3C6EF372)
$Maj(a,b,c)$ = (2A01A605)⊕(2808E262)⊕(3866A200)
$Maj(a,b,c)$ = 3A6FE667
$T_2 = CE20B47E$ + 3A6FE667
$T_2 =$ **08909AE5**
h =g
=1F83D9AB
g = f
= 9B05688C
f = e
= 510E527F
e = d + T_1
=A54FF53A + F350ED48
= 98A0E282
d = c
=3C6EF372
c = b
=BB67AE85
b = a
=6A09E667
a = T_1 + T_2
=F350ED48 + 8909AE5
= FBE1882D

g. Menghitung *Intermediate Hash Value*
$H_0^{(1)} = a + H_0^{(1-1)}$
$H_0^{(1)}$ = EAA2123A + 6A09E667
$H_0^{(1)} =$ **54ABF8A1**
$H_1^{(1)} = b + H_0^{(1-1)}$
$H_1^{(1)}$ = A7DAE359 + BB67AE85
$H_1^{(1)} =$ **634291DE**
$H_2^{(1)} = c + H_0^{(1-1)}$
$H_2^{(1)}$ = DCDF2B26 + 3$C6EF$372
$H_2^{(1)} =$ **194E1E98**
$H_3^{(1)} = d + H_0^{(1-1)}$
$H_3^{(1)}$ = 1C96A388 + A54FF53A
$H_3^{(1)} =$ **C1E698C2**
$H_4^{(1)} = e + H_0^{(1-1)}$
$H_4^{(1)}$ = F1D1A869 + 510E527F
$H_4^{(1)} =$ **42DFFAE8**

JUSSI: Jurnal Sains Dan Teknologi Informasi
Vol 1, No 4, September 2022, Hal 112-120
ISSN 2809-610X (Media Online)
Website https://ejurnal.seminar-id.com/index.php/jussi

$H_5^{(1)} = f + H_0^{(1-1)}$
$H_5^{(1)} = \text{A4623850} + \text{9B05688C}$
$H_5^{(1)} = \mathbf{3F67A0DC}$
$H_6^{(1)} = g + H_0^{(1-1)}$
$H_6^{(1)} = \text{994C1208} + \text{1F83D9AB}$
$H_6^{(1)} = \mathbf{B8CFEBB3}$
$H_7^{(1)} = h + H_0^{(1-1)}$
$H_7^{(1)} = \text{3541F2D7} + \text{5BE0CD19}$
$H_7^{(1)} = \mathbf{9122BFF0}$

h. *Output*

Berdasarkan perhitungan yang telah di lakukan pada langkah di atas maka di dapatkan nilai *hash* dengan panjang 256bit. Nilai inilah yang akan di jadikan sebagai identitas dari setiap citra digital dalam ruang penyimpanan tersebut. Sehingga jika terdapat citra digital yang memiliki nilai *hash* yang sama maka dapat di pastikan bahwa citra tersebut ganda atau duplikat. Berikut ini adalah nilai hash dari algoritma SHA-256 berdasarkan perhitungan pada langkah di atas.**54ABF8A1||634291DE||C1E698C2||C1E698C2||42DFFAE8||B8CFEBB3||9122BFF0**

4. KESIMPULAN

Kesimpulan yang dapat diambil dari hasil implementasi fungsi hash dengan algoritma sha-256 pada aplikasi duplicate image scanner, dengan menerapkan algoritma SHA-256 pada aplikasi duplicate image scanner maka hasil dari pencarian file gambar tersebut di kelompokkan berdasarkan nilai hash yang sama. Dengan menarapkan algoritma SHA-256 dapat mempermudah dalam pencarian setiap file gambar yang duplicat dengan cara memberikan identitas dari setiap *file* gambar, sehingga ketika didapatkan *file* gambar yang memiliki identitas yang sama maka *file* gambar tersebut merupakan *file* gambar yang duplikat atau ganda. Dalam penelitian ini cara yang digunakan untuk mendapatkan identitas *file* gambar tersebut menggunakan Algoritma SHA-256.

REFERENCES

[1] R. Sadikin, Kriptografi Untuk Kemanan Jaringan dan Implementasinya Dalam Bahasa Java, Yogyakarta: Andi, 2012.
[2] R. Munir, Kriptografi, Bandung: R. Munir, 2006.
[3] Y. Kurniawan, Kriptografi Keamanan Internet dan Jaringan Komunikasi, Bandung: Informatika, 2017.
[4] R. Munir, Kriptografi, Bandung: Informatika Bandung, 2006.
[5] D. Rachamawati, J. T. Tarigan and A. B. C. Ginting, "A Comparative Study of Message Digest 5 (MD5) and SHA-256 Algorithm," in 2nd International Conference on Computing and Applied Informatics 2017, Medan, 2018.
[6] F. I. P. Standards Publications, "Secure Hash Standard," National Institute of Standards and Technology, Amerika Serikat, 2002.
[7] S. Patil, N. Jagtap, S. Rajput and R. Sangore, "A Duplicate File Finder System," International Journal of Science Spirituality Business and Technology, pp. 10-14, 2017.
[8] D. Putra, Pengolahan Citra Digital, Yogyakarta: Andi, 2010.
[9] R. Munir, Pengolahan Citra Digital Dengan Pendekatan Algoritmik, Bandung: Informatika, 2007.