Journal of Computing and Informatics Research
ISSN 2808-375X (Media Online)
Vol 1, No 3, Juli 2022
Hal 88−94

Perancangan Aplikasi Duplicate Image Scanner Menerapkan Algoritma MD5

Gaberia Sanchia Melati BR Lumbantoruan

Fakultas Ilmu Komputer dan Teknologi Informasi, Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: gaberiasanchia06@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 MD5 ataupun *Message Digest 5* yang merupakan salah satu algoritma hashing yang sering digunakan untuk mengenkripsi data dengan panjang 128 bit. Dengan menerapkan algoritma ini 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 dan melihat isi file tersebut.

Kata Kunci: Kriptografi; Hash; Duplicat; Scanner; Algoritma MD5

Abstract−Currently we are in the era of the digital world, there are lots of equipment that have tools to record photos in the form of cameras found on smartphones. 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 MD5 method or Message Digest 5 which is one of the hashing algorithms that is often used to encrypt data with a length of 128 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; Hash; Duplicate; Scanner; MD5 Algorithm

1. PENDAHULUAN

Perkembangan teknologi yang semakin pesat membawa dampak terhadap kebiasaan hidup manusia. Saat ini data sudah tersedia dalam bentuk digital, banyak sekali file multimedia yang tersedia dalam bentuk digital. Hal ini dikarenakan file dalam bentuk digital sangat mudah untuk di olah dan di distribusikan melalui jaringan internet. Dengan kemudahan yang di miliki dalam pengolahan file berbentuk digital maka sering terjadi duplikasi atau penggandaan file yang sama dan disimpan dalam sebuah media penyimpanan salah satunya adalah citra digital. Hal ini tentunya sangat merugikan ruang penyimpanan karena menyimpan lebih dari satu citra digital dengan konten yang sama.

Untuk dapat menghemat ruang penyimpanan maka langkah yang harus di ambil adalah memanajemen file yang ada dalam ruang penyimpanan tersebut, seperti mengelompokkan file berdasarkan ekstensi atau jenisnya, menghapus file yang ganda atau duplikat. Untuk menghapus citra digital yang ganda atau duplikat akan sangat sulit dilakukan karena pengguna harus melihat isi dari citra digital tersebut kemudian membandingkan isinya dengan isi citra digital yang lainnya. Hal ini hampir tidak memungkinkan untuk dilakukan apalagi jika jumlah citra digital yang ada dalam media penyimpanan tersebut sangat banyak.

Hal yang dapat dilakukan untuk menghapus citra digital yang ganda atau duplikat adalah dengan cara memberikan identitas yang unik untuk setiap citra digital, di mana identitas itu merepresentasikan isi dari citra digital tersebut. Sehingga jika terdapat 2 atau lebih citra digital yang memiliki identitas yang sama bisa di pastikan citra digital tersebut ganda atau duplikat. Setelah di ketahui identitas citra digital yang sama maka pengguna dapat menghapus citra digital tersebut dan hanya menyisakan satu saja.

MD5 atau *Message-Digest* algoritma MD5 adalah fungsi *hash* kriptografi. Algoritma ini terutama digunakan untuk melakukan pemeriksaan integritas *file* dalam berbagai situasi. Dalam ilmu kriptografi, MD5 adalah salah satu algoritma *hash* yang paling populer. *Hash* atau *hashing* sendiri adalah proses perubahan suatu data lain dengan panjang tertentu. Sedemikian sehingga data itu tidak dapat dipulihkan kembali. Teknik ini biasa digunakan dalam *enkripsi* data, misalnya untuk menyimpan *password* agar tidak ada dapat mengetahuinya meskipun dia dapat melihat *hash* dari *pasword*.

Dengan istilah "*Enkripsi*' tidaklah tepat karena jika data itu di *enkripsi*, Artinya data yang sudah di *hash*- tidak dapat dipulihkan kembali menjadi seperti data awal. Algoritma hash MD5 sendiri menerima input berupa data dengan panjang bebas, dan menghasilkan *output heksadesimal* sepanjang 32 karakter. Jadi, seberapapun panjang data *input*, *output* yang dihasilkan akan selalu sepanjang 32 karakter[1].

2. METODOLOGI PENELITIAN

2.1 Kriptografi

Kriptografi adalah ilmu yang mempelajari teknik – teknik matematika yang behubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data. Dengan memanfaatkan ilmu kriptografi maka kerahasiaan data dapat di jamin pada saat dilakukan pengiriman data ataupun pada saat dilakukan penyimpanan data sehingga data hanya dapat di akses oleh pihak yang memiliki wewenang saja. Di era digital seperti sekarang ini aspek keamanan data dan informasi merupakan suatu hal yang bersifat vital, sehingga ilmu kriptografi sangat di butuhkan. Namun tidak semua aspek keamanan informasi ditangani oleh kriptografi. Enskripsi erat kaitannya dengan deskripsi, untuk itulah muncul istilah kriptanalisis. Kriptanalisis adalah ilmu dan seni untuk memecahkan informasi yang telah dienkripsi tanpa mengetahui kunci yang digunakan. Pelaku kriptanalisis disebut dengan kriptanalis.

2.2 Algoritma MD5

Algoritma MD5 menggunakan input pesan dengan panjang sembarang dan menghasilkan *message digest* dengan panjang 128 bit. Secara teori tidak memungkinkan dua pesan yang berbeda menghasilkan *message digest* yang sama (Rivest, 1992). Algoritma MD5 dirancang untuk dapat bekerja dengan cepat pada arsitektur 32 bit. Algoritma MD5 tidak memerlukan tabel subsitusi yang berukuran besar sehingga dapat bekerja dengan cepat. Algoritma MD5 lebih lambat dibandingkan dengan algoritma MD4, namun algoritma MD5 dirancang karena algoritma MD4 sudah kurang aman lagi. Langkah-langkah dalam algoritma MD5 adalah sebagai berikut:

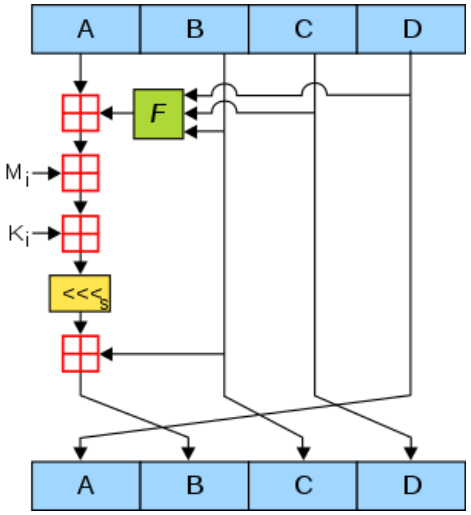
1. Penambahan bit pengganjal
   Pesan diisi sehingga panjangnya kongruen dengan 448, modulus 512. Bit pengganjal 1bit ditambahkan di akhir pesan, diikuti oleh banyaknya nol yang diperlukan sehingga panjang bit sama dengan 448 modulus 512
2. Penambahan panjang pesan
   Representasi panjang pesan 64bit ditambahkan pada hasil akhirnya, langkah ini untuk membuat panjang pesan kelipatan 512 bit.
3. Inisialisasi *MD Buffer*
   Empat *word buffer* (A,B,C,D) digunakan untuk menghitung *message digest.* Masing-masing A,B,C,D merupakan register berukuran 32 bit dan diinisialisasikan dengan bilangan heksadesimal sebagai berikut:
   A = 01234567
   B = 89ABCDEF
   C = FEDCBA98
   D = 7654321
4. Proses Pesan dalam 16 blok *word*
   Mendefinisikan fungsi tambahan yang masing-masing menggunakan masukkan berukuran 32 bit dan menghasilkan keluaran 32 bit.
   F(X,Y,Z) = X and Y or not (X) and Z
   G(X,Y,Z) = X and Z or Y not (Z)
   H(X,Y,Z) = X xor Y xor Z
   I(X,Y,Z) = Y xor (X or not (Z))
   Langkah ini menggunakan tabel 64 elemen yang di bangun dari fungsi sinus. Misalkan T[i] menunjukkan elemen ke i dari tabel tersebut yang sama dengan bagian bilangan bulat dari 4294967296 di kali abs(sin(i)), di mana i berada dalam radian. Elemen tabel T dapat dilihat pada tabel 1. berikut ini:

Tabel 1. Elemen Tabel T dan nilai rotasi kiri S

i	1	2	3	4	5	6	7	8
elemens	D76AA478	E8C7B756	242070DB	C1BDCEEE	F57C0FAF	4787C62A	A8304613	FD469501
	7	12	17	22	7	12	17	22
i	9	10	11	12	13	14	15	16
elemens	698098D8	8B44F7AF	FFFF5BB1	895CD7BE	6B901122	FD987193	A679438E	49B40821
	7	12	17	22	7	12	17	22
i	17	18	19	20	21	22	23	24
elemens	F61E2562	C040B340	265E5A51	E9B6C7AA	D62F105D	2441453	D8A1E681	E7D3FBC8
	5	9	14	20	5	9	14	20
i	25	26	27	28	29	30	31	32

i	1	2	3	4	5	6	7	8
elemen s	21E1CDE6 5	C33707D6 9	F4D50D87 14	455A14ED 20	A9E3E905 5	FCEFA3F8 9	676F02D9 14	8D2A4C8A 20
i	33	34	35	36	37	38	39	40
elemen s	FFFA3942 4	8771F681 11	6D9D6122 16	FDE5380C 23	A4BEEA44 4	4BDECFA9 11	F6BB4B60 16	BEBFBC70 23
i	41	42	45	44	45	46	47	48
elemen s	289B7EC6 4	EAA127FA 11	D4EF3085 16	4881D05 23	D9D4D039 4	E6DB99E5 11	1FA27CF8 16	C4AC5665 23
i	49	50	51	52	53	54	55	56
elemen s	F4292244 6	432AFF97 10	AB9423A7 15	FC93A039 21	655B59C3 6	8F0CCC92 10	FFEFF47D 15	85845DD1 21
i	57	58	59	60	61	62	63	64
elemen s	6FA87E4F 6	FE2CE6E0 10	A3014314 15	4E0811A1 21	F7537E82 6	BD3AF235 10	2AD7D2BB 15	EB86D391 21

Ilustrasi dari proses pesan dalam 16 blok word dapat dilihat pada gambar di bawah ini:



Gambar 1. Proses Pesan dalam 16 blok word

Operasi pada ronde 1
a = b + ((a + F (b,c,d) + X [k] + T[i] <<< s
Operasi pada ronde 2
a = b + ((a + G (b,c,d) + X [k] + T[i] <<< s
Operasi pada ronde 3
a = b + ((a + H (b,c,d) + X [k] + T[i] <<< s
Operasi pada ronde 4
a = b + ((a + I (b,c,d) + X [k] + T[i] <<< s
kemudian lakukan penambahan sebagai berikut yang merupakan perubahan dari masing-masing nilai register sebelum operasi blok selanjutnya dimulai
A = A + AA
B = B + BB
C = C + CC
D = D + DD

5. Keluaran
   Message digest yang dihasilkan sebagai keluaran adalah nilai A,B,C,D dimulai dari orde yang terendah yaitu A dan di akhiri dengan nilai D.

3. HASIL DAN PEMBAHASAN

Masalah yang di hadapi oleh pengguna atau *user* ketika menyimpan *file* citra di dalam sebuah media penyimpanan adalah melakukan penyimpanan lebih dari satu *file* citra yang berbeda, baik itu di letakkan di dalam satu folder atau di dalam folder lainnya. Hal ini tentunya akan memboroskan ruang penyimpanan karena ruang penyimpanan tersebut digunakan

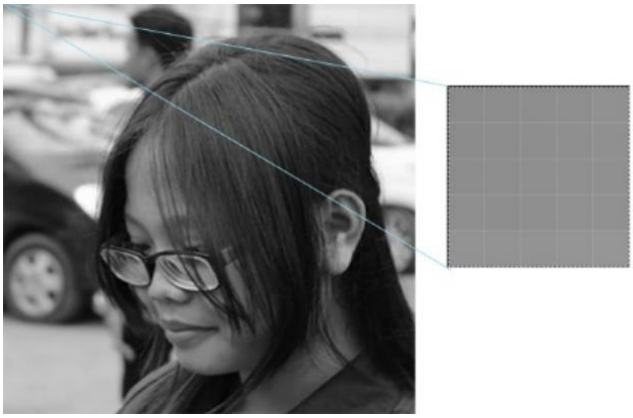
untuk menyimpan citra yang sama. Untuk menghemat ruang penyimpanan tersebut maka pengguna harus menghapus *file* citra yang sama atau duplikat. Untuk dapat membedakan *file* citra yang satu dengan yang lainnya maka pengguna harus melihat isi dari *file* citra tersebut satu persatu. Hal ini tentunya akan sangat merepotkan dan bahkan menjadi tidak mungkin untuk dilakukan jika *file* citra yang di simpan dalam jumlah yang banyak.

Solusi yang dapat digunakan untuk mengatasi permasalahan di atas adalah dengan memberikan identitas dari setiap *file* citra tersebut. Identitas *file* citra tersebut merupakan representasi dari isi citra yang disajikan dalam bentuk sebuah nilai tertentu. Sehingga jika terdapat lebih dari satu *file* citra yang memiliki identitas yang sama maka dapat di pastikan bahwa *file* citra tersebut ganda atau duplikat.

Untuk membangkitkan identitas dari sebuah *file* citra yang merupakan representasi dari isi citra tersebut dapat menggunakan algoritma MD5. Di mana algoritma MD5 ini mempunyai panjang nilai *hash* 128 bit. Dengan memanfaatkan algoritma MD5 ini maka dapat mempermudah identifikasi *file* citra yang ganda atau duplikat. Setiap *file* citra yang ada dalam sebuah media penyimpanan akan di beri identitas berupa nilai *hash* kemudian diurutkan berdasarkan nilai *hash*nya. Sehingga akan memudahkan pengguna untuk menghapus *file* citra yang ganda atau duplikat. Dengan menggunakan cara seperti ini sehingga sangat membantu para pengguna dalam memanajemen *file* citra yang ada di dalam media penyimpanan yang dimilikinya.

3.1 Penerapan Algoritma MD5

Pada penelitian ini citra yang digunakan sebagai contoh kasus berukuran 640 x 640 piksel dengan ekstensi penyimpanan .jpg, namun untuk menyederhanakan proses perhitungan maka di ambil sampel berukuran 5 x 5 piksel saja. Untuk lebih jelasnya dapat di lihat pada gambar 2. berikut ini:



Gambar 2. Citra Sampel

Untuk mendapatkan nilai dari setiap piksel maka digunakan aplikasi matlab untuk melihat nilai dari citra tersebut.

Tabel 2. Nilai Piksel Citra Sampel

118	119	120	121	123
118	119	120	121	122
119	120	121	122	122
121	120	121	123	123
125	125	126	128	129

Tabel 3. Nilai Citra Sampel Dalam Biner

01110110	01110111	01111000	01111001	01111011
01110110	01110111	01111000	01111001	01111010
01110111	01111000	01111001	01111010	01111010
01111001	01111000	01111001	01111011	01111011
01111101	01111101	01111110	10000000	10000001

1. Penambahan *Padding Bit*
   Algoritma MD5 membutuhkan *input* 512 bit, sehingga jika *input* kurang dari 512 bit maka di tambahkan bit pengganjal yang di mulai dari 1 dan diikuti dengan 0 sampai panjangnya kongruen dengan 448 modulus 512.
   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 3.3. di bawah ini:

Tabel 4. Penambahan *Padding Bit*

01110110	01110111	01111000	01111001	01111011	01110110	01110111	01111000
01111001	01111010	01110111	01111000	01111001	01111010	01111010	01111001
01111000	01111001	01111011	01111011	01111101	01111101	01111110	10000000
10000001	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

2. Penambahan Panjang Pesan
   Penambahan panjang pesan dilakukan dengan penambahan panjang data sebanyak 64 bit di akhir. Panjang data adalah 200 bit sehingga ditambahkan panjang pesan 11001000 di akhir sebanyak 64bit sebagai berikut:

Tabel 5. Penambahan Panjang Pesan

01110110	01110111	01111000	01111001	01111011	01110110	01110111	01111000
01111001	01111010	01110111	01111000	01111001	01111010	01111010	01111001
01111000	01111001	01111011	01111011	01111101	01111101	01111110	10000000
10000001	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

3. Inisialisasi MD *Buffer*
   Melakukan proses inisialisasi penyangga *(buffer) message digest*. Empat kata *buffer* yang didefinisikan dengan A, B, C, D digunakan dalam melakukan komputasi *message digest*. Setiap dari A, B, C, D merupakan sebuah data register yang terdiri dari 32 bit. Total panjang penyangga adalah 128 bit. Register- register ini diinisialisasikan sebagai berikut:

Tabel 6. Inisialisasi MD *Buffer*

A	B	C	D
01234567	89ABCDEF	FEDCBA98	76543210

4. Proses Pesan Dalam 16 Blok *Word*
   Langkah ke 4 adalah melakukan proses pesan menggunakan Fungsi F, G, H dan I untuk setiap 16 blok *word* yaitu pesan 512 bit yang di bagi menjadi 16 bagian dan setiap bagian terdiri dari 32 bit. Proses kompresi terdiri dari 4 ronde untuk setiap 16 blok *word* pesan. Untuk ronde pertama memproses pesan menggunakan fungsi F, ronde kedua memproses pesan dengan menggunakan fungsi G, dan pada ronde ketiga memproses pesan menggunakan fungsi H dan ronde keempat menggunakan fungsi I. Sebelum masuk ronde pertama maka pesan di bagi menjadi 16 blok *word* yaitu X0-X15 yang dapat di lihat pada tabel 6.. berikut ini:

Tabel 7. 16 Blok *Word* Pesan

X0	=	01110110	01110111	01111000	01111001	=	76777879
X1	=	01111011	01110110	01110111	01111000	=	7B767778
X2	=	01111001	01111010	01110111	01111000	=	797A7778
X3	=	01111001	01111010	01111010	01111001	=	797A7A79
X4	=	01111000	01111001	01111011	01111011	=	78797B7B
X5	=	01111101	01111101	01111110	10000000	=	7D7D7E80
X6	=	10000001	00000000	00000000	00000000	=	00000000
X7	=	00000000	00000000	00000000	00000000	=	00000000
X8	=	00000000	00000000	00000000	00000000	=	00000000
X9	=	00000000	00000000	00000000	00000000	=	00000000
X10	=	00000000	00000000	00000000	00000000	=	00000000
X11	=	00000000	00000000	00000000	00000000	=	00000000
X12	=	00000000	00000000	00000000	00000000	=	00000000
X13	=	00000000	00000000	00000000	00000000	=	00000000
X14	=	00000000	00000000	00000000	00000000	=	00000000
X15	=	00000000	00000000	00000000	11001000	=	000000C8

Setelah membagi pesan menjadi 16 bagian maka selanjutnya adalah masuk ke dalam ronde pertama dengan menggunakan fungsi F, ronde kedua dengan menggunakan fungsi G, dan ronde ketiga menggunakan fungsi H serta ronde keempat dengan fungsi I.
Ronde 1

Putaran 0
K = 0, S = 7, I = 1

i	A	B	C	D
initial	**01234567**	**89ABCDEF**	**FEDCBA98**	**76543210**

A = B + (A + F (B,C,D) + X [K] + T[I] <<< S)
A = B + (A + F (B,C,D) + X [0] + T[1] <<< 7)
A = 89ABCDEF + (01234567 + F (B,C,D) + 76777879 + D76AA478 <<< 7)
F (B,C,D) = B and C or not (B) and D
F (B,C,D) = 89ABCDEF and FEDCBA98 or not (89ABCDEF) and 76543210
F (B,C,D) = 88888888 or 76543210
F (B,C,D) = FEDCBA98
A = 89ABCDEF + (01234567 + FEDCBA98 + 76777879 + D76AA478 <<< 7)
A = 89ABCDEF + (4DE21CF0 <<< 7)
A = 89ABCDEF + F10E7826
A = 7ABA4615

i	A	B	C	D
0	**7ABA4615**	**89ABCDEF**	**FEDCBA98**	**76543210**

Putaran 1
K = 1, S = 12, I = 2

i	A	B	C	D
0	**76543210**	**7ABA4615**	**89ABCDEF**	**FEDCBA98**

A = B + (A + F (B,C,D) + X [K] + T[I] <<< S)
A = 7ABA4615 + (76543210 + F (B,C,D) + X [1] + T[2] <<< 12)
A = 7ABA4615 + (76543210 + F (B,C,D) + 7B767778 + E8C7B756<<< 12)
F (B,C,D) = B and C or not (B) and D
F (B,C,D) = 7ABA4615 and 89ABCDEF or not (7ABA4615) and FEDCBA98
F (B,C,D) = 8AA4405 or 8444B888
F (B,C,D) = 8444B888
A = 7ABA4615 + (76543210 + 8444B888 + 7B767778 + E8C7B756 <<< 12)
A = 7ABA4615 + (5ED71966 <<< 12)
A = 7ABA4615 + 719665ED
A = EC50AC02

i	A	B	C	D
1	**EC50AC02**	**7ABA4615**	**89ABCDEF**	**FEDCBA98**

Putaran 2
K = 2, S = 17, I = 3

i	A	B	C	D
1	**FEDCBA98**	**EC50AC02**	**7ABA4615**	**89ABCDEF**

A = B + (A + F (B,C,D) + X [K] + T[I] <<< S)
A = EC50AC02 + (FEDCBA98 + F (B,C,D) + X [2] + T[3] <<< 17)
A = EC50AC02 + (FEDCBA98 + F (B,C,D) + 797A7778 + 242070DB <<< 17)
F (B,C,D) = B and C or not (B) and D
F (B,C,D) = EC50AC02 and 7ABA4615 or not (EC50AC02) and 89ABCDEF
F (B,C,D) = 68100400 or 01AB41ED
F (B,C,D) = 69BB45ED
A = EC50AC02 + (FEDCBA98 + 69BB45ED + 797A7778 + 242070DB <<< 17)
A = EC50AC02 + (0632E8D8 <<< 17)
A = 7ABA4615 + D1B00C65
A = 4C6A527A

i	A	B	C	D
2	**4C6A527A**	**EC50AC02**	**7ABA4615**	**89ABCDEF**

Putaran 3
K = 3, S = 22, I = 4

i	A	B	C	D
2	**89ABCDEF**	**4C6A527A**	**EC50AC02**	**7ABA4615**

A = B + (A + F (B,C,D) + X [K] + T[I] <<< S)
A = 4C6A527A + (89ABCDEF + F (B,C,D) + X [3] + T[4] <<< 22)
A = EC50AC02 + (FEDCBA98 + F (B,C,D) + 797A7A79 + C1BDCEEE <<< 22)
F (B,C,D) = B and C or not (B) and D
F (B,C,D) = 4C6A527A and EC50AC02 or not (4C6A527A) and 7ABA4615
F (B,C,D) = 4C400002 or 32900405
F (B,C,D) = 7ED00407
A = EC50AC02 + (FEDCBA98 + 7ED00407 + 797A7A79 + C1BDCEEE <<< 22)
A = EC50AC02 + (B8E50806 <<< 22)
A = 7ABA4615 + 01AE3942
A = 7C687F57
Proses terus dilanjutkan hingga putaran ke-64

5. *Output*
   Message digest yang dihasilkan sebagai *output* merupakan penjumlahan dari *intermediate hash value* dengan *output* dari proses 64 ronde.
   A = A + AA
   A = 01234567 + BB8E8374
   A = BCB1C8DB

   B = B + BB
   B = 89ABCDEF + 1051A8C5
   B = 99FD76B4

   C = C + CC
   C = FEDCBA98 + 731253A4
   C = 71EF0E3C

   D = D + DD
   D = 76543210 + 614DCB5B
   D = D7A1FD6B

i	A	B	C	D
output	**BCB1C8DB**	**99FD76B4**	**71EF0E3C**	**D7A1FD6B**

Sehingga nilai *hash*nya adalah BCB1C8DB99FD76B471EF0E3CD7A1FD6B, nilai inilah yang dijadikan patokan untuk mengetahui citra atau gambar yang ganda atau duplikat.

4. KESIMPULAN

Dari hasil penulisan dan analisa dari bab-bab sebelumnya, maka dapat diambil kesimpulan, dimana kesimpulan-kesimpulan tersebut kiranya dapat berguna bagi para pembaca, sehingga penulisan skripsi ini dapat lebih bermanfaat. Adapun kesimpulan-kesimpulan tersebut adalah sebagai berikut Pencarian *file* citra yang ganda atau duplikat di dalam sebuah media penyimpanan dapat di lakukan dengan memberikan identitas dari setiap *file* citra menggunakan fungsi *hash* sehingga tidak perlu membuka dan membaca isi *file* citra satu persatu Algoritma fungsi *hash* MD5 dapat digunakan untuk merepresentasikan isi dari *file* citra sehingga dapat digunakan untuk membandingkan *file* citra yang ganda atau duplikat.

REFERENCES

[1] Irwanto, D. (2007). Perancangan Object Oriented Software Dengan UML. Yogyakarta: Andi.
[2] Ketut Darmayuda. (2010). Pemograman Aplikasi Database dengan Microsoft Visual Basic. NET 2008. Bandung.
[3] Kurniawan, Y. (2017). Kriptografi Keamanan Internet dan Jaringan Komunikasi. Bandung: Informatika.
[4] Munir, R. (2006). Kriptografi. Bandung: R. Munir.
[5] Munir, R. (2006). Kriptografi. Bandung: Informatika Bandung.
[6] Munir, R. (2007). Pengolahan Citra Digital Dengan Pendekatan Algoritmik. Bandung: Informatika.
[7] Patil, S., Jagtap, N., Rajput, S., & Sangore, R. (2017). A Duplicate File Finder System. International Journal of Science Spirituality Business and Technology, 10-14.
[8] Putra, D. (2010). Pengolahan Citra Digital. Yogyakarta: Andi.
[9] Rivest, R. (1992). The MD5 Message-Digest Algorithm. Massachuset: MIT Laboratory for Computer Science.
[10] Sadikin, R. (2012). Kriptografi Untuk Kemanan Jaringan dan Implementasinya Dalam Bahasa Java. Yogyakarta: Andi.
[11] Solution, C., & Community, S. (2010). Membangun Aplikasi Database Dengan Visual Basic 2008 dan SQL Server 2008. Jakarta: PT. Elex Media Komputindo.