HASIL CEK_60010313_(15)(1)

by Sunardi 60010313

Submission date: 11-Dec-2020 11:20AM (UTC+0700)
Submission ID: 1471733178
File name: CEK15_60010313.pdf (779.57K)
Word count: 3604
Character count: 21639

JEPIN
(Jurnal Edukasi dan Penelitian Informatika)
ISSN(e): 2548-9364 / ISSN(p) : 2460-0741

Vol. 6
No. 1
April 2020

Penyusunan File Fingerprint untuk Berkas Jpeg/exif dengan Hash Function SHA512 dan Algoritma Boyer-Moore String Matching

Rachmad Fitriyanto[#1], Anton Yudhana[*2], Sunardi[*3]

[#]*Magister Teknik Informatika, Universitas Ahmad Dahlan, Yogyakarta*
Jl. Kapas No.9, Semaki, Kec. Umbulharjo, Kota Yogyakarta, Daerah Istimewa Yogyakarta
[1]fitriyanto7477@gmail.com

[*]*Program Studi Teknik Elektro, Universitas Ahmad Dahlan Yogyakarta*
Jl. Kapas No.9, Semaki, Kec. Umbulharjo, Kota Yogyakarta, Daerah Istimewa Yogyakarta
[2]yudhana@ee.uad.ac.id
[3]sunardi@mti.uad.ac.id

***Abstrak*— Pengamanan berkas jpeg/exif dalam komunikasi digital umumnya bersifat untuk pencegahan, belum dapat digunakan untuk pendeteksian keutuhan data. *File Fingerprint* merupakan konsep sidik jari yang disusun berdasarkan konten dari dokumen digital. Penelitian ini bertujuan untuk menyusun *file fingerprint* dari berkas jpeg/exif yang dapat digunakan untuk mendeteksi terjadinya modifikasi pada berkas. Penyusunan *file fingerprint* dilakukan dalam lima tahap. Tahap pertama adalah identifikasi struktur berkas jpeg/exif menggunakan algoritma Boyer-Moore *string matching* untuk menentukan indeks lokasi segmen jpeg/exif. Tahap kedua adalah akuisisi konten segmen. Tahap ketiga adalah penghitungan *hash value* menggunakan *hash function* SHA512. Tahap keempat adalah pengujian modifikasi berkas jpeg/exif. Tahap kelima adalah penyusunan *file fingerprint*. Hasil yang diperoleh menunjukkan *file fingerprint* dari berkas jpeg/exif berasal dari tiga segmen, SOI, APP1 dan SOF0. *Hash value* dari segmen SOI digunakan untuk mendeteksi modifikasi dalam bentuk konversi tipe berkas dan penambahan objek pada gambar. *Hash value* dari segmen APP1 untuk mendeteksi modifikasi pada metadata. *Hash value* dari segmen SOF0 untuk mendeteksi modifikasi dalam bentuk *resizing*, *recoloring* dan *cropping*. Berkas *file fingerprint* yang dihasilkan memiliki ukuran rata-rata 0,017% dari ukuran berkas gambar dari *smartphone* Asus Z00UD dan 0,015% dari ukuran berkas gambar dari *smartphone* Samsung Galaxy A5.**

***Kata kunci*— *File fingerprint*, *Hash value*, Jpeg/exif, SHA512, Boyer-Moore**

I. PENDAHULUAN

Berkas jpeg/exif merupakan tipe berkas yang banyak digunakan dalam komunikasi digital saat ini. Tipe berkas ini merupakan berkas gambar yang dihasilkan dari pemakaian kamera digital seperti pada *smartphone*. Metode pengamanan komunikasi untuk berkas jpeg/exif umumnya hanya bersifat pencegahan. Teknik umum pengamanan dokumen jpeg/exif dilakukan menggunaan password dan enkripsi. Kedua teknik tersebut bertujuan untuk mencegah pihak lain untuk memodifikasi berkas. Teknik lain seperti *watermark* dalam penelitian [1] bertujuan untuk mencegah tindakan pembajakan hak cipta. Pemakaian metadata dalam pengamanan dokumen ditemui dalam penelitian oleh Wijayanto [2]. Metadata dari berkas jpeg/exif dienkripsi sebagai acuan verifikasi keaslian berkas. Penelitian tentang keamanan informasi yang bersifat deteksi ditemui dalam penelitian oleh Refialy [3]. Penelitian ini menunjukkan *hash value* dari dokumen pdf dapat digunakan sebagai identitas unik untuk mendeteksi integritas data pada dokumen yang diterima. *Watermark* dan metada dalam dua penelitian terdahulu hanya dapat digunakan untuk mencegah modifikasi illegal pada dokumen jpeg/exif. Kedua metode tersebut belum dapat digunakan untuk mendeteksi keutuhan data jika dokumen jpeg/exif berhasil dimodifikasi.

File fingerprint merupakan konsep identitas unik yang digunakan untuk identifikasi keutuhan data [4]. *File fingerprint* umumnya digunakan untuk kepentingan forensik digital dengan cara membandingkan dua *hash value* [5]. Penyusunan *file fingerprint* dilakukan menggunakan *hash function* yang menghasilkan *hash value*. Berdasarkan konsep *file fingerprint* dan tiga penelitian terdahulu, penelitian ini bertujuan untuk menyusun *file fingerprint* dari berkas jpeg/exif yang dapat digunakan untuk mendeteksi keutuhan data apabila terjadi modifikasi. Karakteristik *file fingerprint tersebut* menjadi tujuan penelitian ini untuk mengimplementasikan penggunaan *file fingerprint* dalam mengidentifikasi keutuhan data pada dokumen jpeg/exif.

File fingerprint disusun dari *hash value* menggunakan *hash function* atau fungsi hash. *Hash function* merupakan teknik dalam kriptografi yang berfungsi mengenkripsi

Submitted 29-05-2019; Revised 03-12-2019; Accepted 10-12-2019

input menjadi *cipher text* [6]. *Secure Hash Algorithm* (SHA) merupakan *hash function* yang dikembangkan oleh *National Institute of Standard and Technology* [7]. SHA512 merupakan varian dari SHA2 yang memiliki panjang hash value sebesar 512-bit [8]. Pemilihan SHA untuk keamanan informasi disebabkan dua hal, sifat dan ukuran *hash value* yang dihasilkan. *Hash value* dari *hash function* bersifat tidak dapat diterjemahkan (*decrypt*) kembali seperti bentuk input [9]. Ukuran *hash value* yang dihasilkan selalu tetap sesuai varian *hash function*-nya untuk berbagai ukuran input [10].

Algoritma Boyer-Moore *string matching* merupakan pengembangan dari algoritma Brute Force *string matching* [11]. Algoritma Boyer-Moore memulai pencarian pola dengan membandingkan elemen paling kanan dari pola dengan elemen dari string yang diuji. Hasil perbandingan elemen menentukan proses yang akan dilakukan mengikuti dua aturan yaitu *Good-Character ruke* dan *Bad-Character rule* [12]. *Good-Character rule* digunakan apabila ditemukan elemen yang sama setelah perbandingan sebelumnya tidak ditemukan. Langkah yang diambil adalah mensejajarkan elemen pada pola dengan elemen pada string. *Bad-Character rule* digunakan apabila tidak ditemukan elemen yang sama. Kondisi ini dilanjutkan dengan menggeser elemen perbandingan pada pola ke sebelah kiri sementara elemen pada string tetap. Apabila elemen pola telah sampai pada elemen paling kiri dan tidak sama dengan elemen string, maka perbandingan dimulai dari awal dengan menggeser elemen string ke kanan sebanyak jumlah elemen pola. Kedua aturan ini membuat algoritma Boyer-Moore *string matching* memiliki kecepatan pencarian yang lebih baik dari algoritma Brute Force *string matching* [13].

II. METODOLOGI

Penyusunan *file fingeprint* dilakukan dalam lima tahapan seperti yang diilustrasikan pada Gambar 1. Tahap pertama adalah identifikasi struktur berkas jpeg/exif. Tahap ini bertujuan untuk mengetahui indeks lokasi dari segmen-segmen penyusun berkas jpeg/exif.

Berkas jpeg/exif tersusun dari enam segmen, SOI, APP1, DQT, DHT, SOF0, DHT dan SOS. Setiap segmen menyimpan data yang digunakan oleh komputer dan aplikasi *image viewer* untuk menampilkan gambar. Segmen-segmen dalam berkas jpeg/exif memiliki data unik yang digunakan untuk mengidentifikasi lokasinya yang disebut *segment marker* [14].

Tabel 1 menunjukkan nama segmen dan *segment marker* di dalam berkas jpeg/exif.

TABEL I
SEGMEN DAN SEGMENT MARKER PADA BERKAS JPEG/EXIF

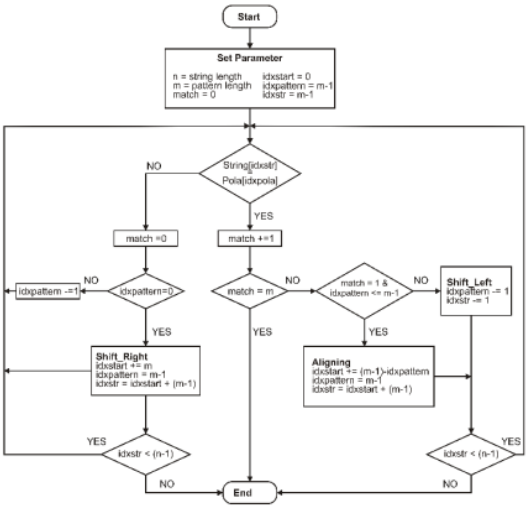
Segmen	Segment marker
SOI (*Start of Image*)	ffd8
APP1 *(Application-1)*	ffe1
DQT (*Define Quantization Table*)	ffdb
SOF0 (*Start of Frame-0*)	ffc0
DHT (*Define Huffmans Table*)	ffc4
SOS (*Start of Scan*)	ffda

Gambar. 1 Tahap penyusunan fle fingerprint

Nilai *segment marker* pada Tabel 1 disusun dalam format *hexadecimal*. Berkas jpeg/exif yang diuji adalah berkas gambar yang dihasilkan dari pemakaian kamera pada *smartphone*. Tipe *smartphone* yang digunakan adalah Asus Z00UD dan Samsung Galaxy A5. Jumlah berkas gambar dari setiap *smartphone* sebanyak 10 berkas. Metode pengambilan gambar menggunakan aplikasi *default* (*embedded*) dari setiap *smartphone* dengan mode *auto*. Lokasi pengambilan terdiri dari *indoor* dan *outdoor*. Tahap identifikasi struktur berkas jpeg/exif diawali dengan proses konversi bit-bit data berkas gambar ke dalam format string. Hasil konversi ini digunakan dalam proses pencarian lokasi segmen berkas gambar pada tahap pertama.

Setiap *Segment marker* seperti yang ditunjukkan di Tabel 1, dicari indeks lokasinya. Lokasi setiap *segment marker* yang dicari adalah titik awal setiap segmen dimulai. Pencarian titik awal segmen menggunakan *segment marker* sebagai pola (*pattern*) yang dicari pada algoritma Boyer-Moore *string matching*. Gambar 2 menunjukkan *flowchart* dari algoritma Boyer-Moore *string matching*.

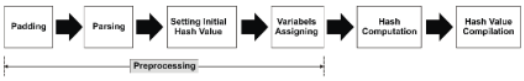
Proses pencarian pada algoritma Boyer-Moore dilakukan dengan perbandingan karakter pada pola dengan string. Perbandingan dilakukan secara iteratif yang dibatasi dengan dua kondisi. Perbandingan berhenti atau selesai apabila variabel match memiliki nilai yang sama dengan panjang pola atau variabel m. perbandingan juga berhenti apabila karakter string yang dibandingkan adalah karakter string terakhir. Kondisi kedua ditunjukkan dengan variabel idxstr yang memiliki nilai sama dengan panjang string (variabel n). Indeks lokasi segmen ditunjukkan dengan nilai idxstr terakhir saat proses perbandingan berhenti.



Gambar. 2 *Flowchart* algoritma Boyer-Moore *string matching*

Tahap kedua penyusunan *file fingerprint* adalah akuisisi konten yang terdapat di setiap segmen. Tahap kedua bertujuan untuk menyalin isi atau konten yang tersimpan di setiap segmen. Konten setiap segmen akan diolah untuk menghitung nilai hash pada tahap selanjutnya. Nilai indeks lokasi yang diperoleh dari proses identifikasi lokasi segmen pada tahap pertama digunakan sebagai acuan untuk mengakuisisi konten segmen. Akuisisi konten membutuhkan parameter panjang segmen yang diperoleh dengan mencari selisih dari indeks lokasi dua segmen yang berdekatan. Panjang segmen SOI diperoleh dari selisih indeks lokasi APP1 dikurangi indeks lokasi SOI. Panjang segmen APP1 diperoleh dari selisih antara indeks lokasi DQT dengan APP1. Panjang segmen SOS diperoleh dari selisih variabel n dengan indeks lokasi SOS. Indeks lokasi segmen menunjukkan titik awal akuisisi. Panjang segmen menunjukkan titik akhir akuisisi.

Tahap ketiga penyusunan *file fingerprint* adalah penghitungan *hash value* yang dilakukan untuk setiap segmen. Tahap ketiga bertujuan untuk menghasilkan enam buah *hash value* yang merepresentasikan konten dari enam segmen berkas jpeg/exif. Proses penghitungan *hash value* dengan SHA512 terdiri dari enam langkah seperti yang diilustrasikan pada Gambar 3.



Gambar. 3 Langkah penghitungan *hash value* dengan SHA512

Langkah *padding* adalah penyesuaian panjang input string sehingga memiliki panjang dengan kelipatan 512-bit. Berkas jpeg/exif yang telah dikonversi ke dalam bit data biner akan disesuaikan panjang bit-nya menjadi 512-bit. Bit data yang ditambanhkan adalah bit-0 sebanyak kebutuhan sehingga panjang bit berkas jpeg/exif menjadi 512-bit. Langkah kedua *parsing* yang bertujuan untuk membagi bit data 512-bit hasil langkah pertama menjadi 80 blok data (M^0, M^1, …M^{79}) yang memiliki ukuran 64-bit untuk setiap bloknya [7]. Langkah ketiga adalah penentuan nilai *initial hash value* ($H_{(n)}{}^0$) sebanyak delapan nilai seperti ditunjukkan pada Tabel 2.

TABEL II
INITIAL HASH VALUE UNTUK SETIAP VARIABEL

Hash Value	Variabel
$H_{(0)}{}^0$ = 6a09e667f3bcc908	a
$H_{(1)}{}^0$ = bb67ae8584caa73b	b
$H_{(2)}{}^0$ = 3c6ef372fe94f82b	c
$H_{(3)}{}^0$ = a54ff53a5f1d36f1	d
$H_{(4)}{}^0$ = 510e527fade682d1	e
$H_{(5)}{}^0$ = 9b05688c2b3e6c1f	f
$H_{(6)}{}^0$ = 1f83d9abfb41bd6b	g
$H_{(7)}{}^0$ = 5be0cd19137e2179	h

Kedelapan *initial hash value* pada Tabel 2 merupakan konstanta yang mengacu pada ketentuan yang disusun oleh NIST sebagai pengembang algoritma *Hash function* SHA512. Kedelapan *initial hash value* disimpan di dalam delapan variabel (a, b, c, d, e, f, g, h).

Langkah kelima adalah penghitungan *hash value* SHA512 menggunakan enam persamaan yang ditunjukkan pada persamaan 1 sampai 6.

$$Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z) \quad (1)$$

$$Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \quad (2)$$

$$\sum_0^{512}(x) = ROTR^{28}(x) \oplus ROTR^{34}(x) \oplus ROTR^{39}(x) \quad (3)$$

$$\sum_0^{512}(x) = ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{21}(x) \quad (4)$$

$$\sigma_0^{512}(x) = ROTR^{1}(x) \oplus ROTR^{8}(x) \oplus SHR^{7}(x) \quad (5)$$

$$\sigma_1^{512}(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^{6}(x) \quad (6)$$

Input yang telah dikembangkan menjadi beberapa blok diproses dengan dua langkah penghitungan yang dilakukan sesuai urutan blok input menggunakan enam persamaan. Penghitungan pertama untuk blok ke-0 sampai 79 mengikuti persamaan 7.

For t=0 to 79:
{
$T_1 = h + \sum_1^{\{512\}}(e) + Ch(e,f,g) + K_t^{\{512\}} + W_t$
$T_2 = \sum_0^{\{512\}}(a) + Maj(a,b,c)$
$h = g$
$g = f$
$f = e$
$e = d + T_1$
$d = c$
$c = b$
$b = a$
$a = T_1 + T_2$
} (7)

Penghitungan kedua untuk menentukan nilai *intermediate hash value* (H^i)menggunakan persamaan 8.

$$\begin{aligned} H_0^{(i)} &= a + H_0^{(i-1)} \\ H_1^{(i)} &= b + H_1^{(i-1)} \\ H_2^{(i)} &= c + H_2^{(i-1)} \\ H_3^{(i)} &= d + H_3^{(i-1)} \\ H_4^{(i)} &= e + H_4^{(i-1)} \\ H_5^{(i)} &= f + H_5^{(i-1)} \\ H_6^{(i)} &= g + H_6^{(i-1)} \\ H_7^{(i)} &= h + H_7^{(i-1)} \end{aligned} \quad (8)$$

Hasil akhir SHA512 adalah *hash value* yang tersimpan di delapan register seperti yang ditunjukkan pada Gambar 4.

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$$

Gambar. 4 Delapan *hash value*

Kedelapan *hash value* dari delapan register disusun sesuai urutan mulai dari register a sampai h sehingga membentuk satu rangkaian string. Gambar 5 menunjukkan contoh *hash value* dari input "abc".

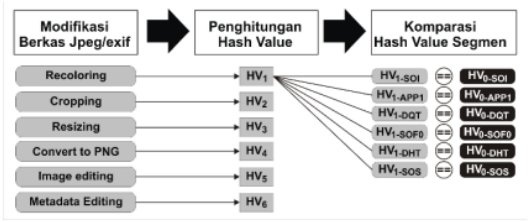
```
ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a
2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f
```

Gambar. 5 Contoh *hash value* SHA512 dari input "abc"

Panjang *hash value* pada Gambar 5 adalah sepanjang 128 digit yang tersusun dalam format *hexadecimal*. Besaran output dalam format biner adalah 128 x 4 = 512-bit.

Tahap keempat dalam penyusunan *file fingerprint* jpeg/exif adalah modifikasi berkas gambar. Tahap keempat bertujuan untuk mengetahui segmen-segmen yang mengalami perubahan ketika mengalami modifikasi. Percobaan modifikasi dilakukan untuk mengetahui segmen yang mengalami perubahan dari setiap bentuk modifikasi. Modifikasi konversi tipe berkas dari jpeg ke png menggunakan aplikasi FormatFactory. Modifikasi metadata berkas dengan aplikasi Neo Hex Editor. Modifikasi penambahan teks atau objek pada gambar dengan aplikasi Corel Photo-Paint X3. Modifikasi *recoloring*, *resizing* dan *cropping* menggunakan aplikasi ACDSee Pro.8. Langkah-langkah percobaan modifikasi gambar ditunjukkan pada Gambar 6.

Sebelum berkas gambar dimodifikasi, dilakukan penghitungan *hash value* dari berkas asli (HV_0). *Hash value* dari berkas asli digunakan dalam komparasi *hash value* dari berkas yang telah dimodifikasi (HV_1, HV_2, HV_3, HV_4, HV_5, HV_6). Setiap bentuk modifikasi akan disusun *hash value* setiap segmennya. Komparasi *hash value* dilakukan untuk setiap segmen yang sama. *Hash value* yang mengalami perubahan disetiap bentuk modifikasi akan digunakan sebagai penyusun *file fingerprint* pada tahap selanjutnya.



Gambar. 6 Langkah modifikasi berkas jpeg/exif

Tahap kelima, penyusunan *file fingerprint* adalah proses penggabungan beberapa *hash value* yang telah ditentukan dari tahap keempat. Tahap kelima bertujuan untuk menghasilkan *file fingerprint* dari berkas jpeg/exif.

III. Hasil dan Pembahasan

Hasil identifikasi lokasi segmen berkas jpeg/exif ditunjukkan pada Tabel 3 untuk berkas dari *smartphone* Asus Z00UD dan Tabel 4 dari *smartphone* Samsung Galaxy A5.

TABEL III
INDEKS LOKASI SEGMEN JPEG/EXIF SMARTPHONE ASUS Z00UD

Nama berkas	Indeks lokasi segmen					
	SOI	APP1	DQT	SOF0	DHT	SOS
P_20180723_141211	0	4	26400	26844	26726	27630
P_20180731_134049	0	4	26368	61656	26694	27598
P_20180823_124724	0	4	26378	26664	26704	27610
P_20180905_085850	0	4	25350	25636	25676	26580
P_20190110_100735	0	4	26318	26602	26642	27548
P_20190324_100013	0	4	34212	25544	25584	26490
P_20190324_100040	0	4	25378	25662	25704	26608
P_20190324_114023	0	4	25278	25769	25809	26713
P_20190324_115302	0	4	25348	25789	25792	26713
P_20190324_121005	0	4	25405	25663	25564	26580

Lokasi segmen SOI dan APP1 dari sepuluh berkas jpeg/exif pada Tabel 3 memiliki nilai yang sama. Hal ini disebabkan segmen SOI berada pada bagian awal berkas dan memiliki panjang yang sama untuk semua berkas gambar. Indeks lokasi segmen DQT, SOF0, DHT dan SOS memiliki indeks lokasi dan panjang yang berbeda. Kondisi ini membuat pencarian lokasi setiap segmen harus dilakukan untuk setiap berkas gambar.

TABEL IV
INDEKS LOKASI SEGMEN JPEG/EXIF SMARTPHONE ASUS Z00UD

Nama berkas	Indeks lokasi segmen					
	SOI	APP1	DQT	SOF0	DHT	SOS
20180825_131753	0	4	2000	2286	2326	3218
20171225_111236	0	4	2000	2286	2326	3218
20171201_124906	0	4	2000	2286	2326	3218
20171201_130420	0	4	2000	2286	2326	3218
20180825_134942	0	4	2000	2286	2326	3218
20181213_172235	0	4	2000	2286	2326	3218
20190114_154205	0	4	2000	2286	2326	3218
20190114_154209	0	4	2000	2286	2326	3218
20190114_154215	0	4	2000	2286	2326	3218
20190114_154220	0	4	2000	2286	2326	3218

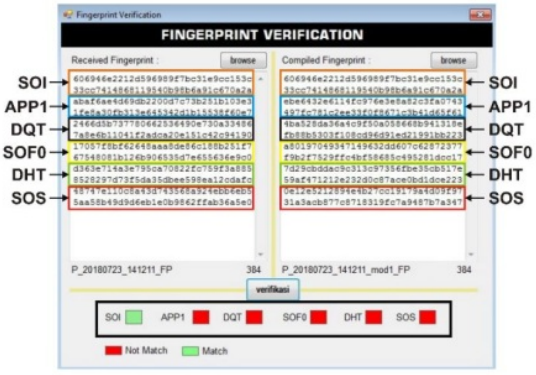
Lokasi segmen berkas jpeg/exif dari *smartphone* Samsung Galaxy A5 memiliki nilai yang sama dari sepuluh berkas yang diuji. Kondisi ini dapat membuat proses pencarian lokasi segmen hanya dilakukan sekali.

Nilai indeks lokasi yang diperoleh digunakan untuk menentukan panjang segmen. Penghitungan panjang segmen dilakukan dengan mencari selisih dari dua nilai indeks lokasi dari dua segmen yang berdekatan. Panjang segmen DQT pada berkas pertama pada Tabel 3 adalah selisih dari indeks lokasi SOF0 dan DQT, yaitu 26844 – 26400 = 444.

Indeks lokasi segmen digunakan sebagai titik awal proses akuisisi konten segmen. Nilai panjang segmen digunakan sebagai panjang segmen yang diakuisisi. Gambar 7 menunjukkan tampilan antarmuka dari aplikasi yang berisikan konten setiap segmen dan *hash value* masing-masing segmen.

Gambar. 7 Antarmuka aplikasi akuisisi konten segmen

Bagian berwarna putih berisikan bit data pada setiap segmen. Bagian yang berwarna coklat menunjukkan *hash value* hasil perhitungan untuk setiap konten segmen. Hasil penghitungan *hash value* yang diperoleh kemudian disatukan menjadi *file fingerprint* yang dikomparasikan dengan *hash value* dari berkas jpeg/exif hasil modifikasi. Gambar 8 menunjukkan tampilan aplikasi untuk membandingkan dua *file fingerprint*.



Gambar. 8 Antarmuka aplikasi komparasi dua *file fingerprint*

Bagian kiri aplikasi menunjukkan *file fingerprint* dari berkas jpeg/exif asli, sedangkan bagian kanan dari berkas jpeg/exif hasil modifikasi *recoloring*. Komparasi antar segmen menunjukkan segmen APP1, DQT, SOF0, DHT dan SOS mengalami perubahan. Hasil komparasi dari enam bentuk modifikasi terhadap perubahan konten segmen jpeg/exif ditunjukkan pada Tabel 5.

TABELV
PERUBAHAN KONTEN SEGMEN

No.	Bentuk modifikasi	Perubahan konten segmen					
		SOI	APP1	DQT	SOF0	DHT	SOS
1	*Recoloring*	-	✓	✓	✓	✓	✓
2	Modifikasi metadata	-	✓	-	-	-	-
3	*Resizing*	-	✓	✓	✓	✓	✓
4	Konversi jpg-png	✓	✓	✓	✓	✓	✓
5	Penambahan teks/gambar	✓	✓	✓	✓	✓	✓
6	*Cropping*	-	✓	✓	✓	✓	✓

Segmen yang bertanda centang pada Tabel 5 menunjukkan terjadinya perubahan pada kontennya. Hasil percobaan enam modifikasi dikelompokkan menjadi tiga. Kelompok pertama adalah perubahan pada segmen APP1. Modifikasi metadata hanya mengakibatkan perubahan pada satu segmen tersebut. Kelompok kedua adalah perubahan pada seluruh segmen untuk modifikasi konversi jpg-png dan penambahan teks/gambar. Modifikasi ke-4 mengubah struktur berkas gambar secara keseluruhan sehingga segmen-segmen jpeg/exif tidak lagi terdapat pada berkas gambar. Modifikasi ke-5 menyebabkan bertambahnya satu

segmen APP0 yang terletak diantara segmen SOI dan APP1 seperti yang ditunjukkan pada Gambar 9.

Gambar. 9 Perbandingan struktur segmen dari dua berkas jpeg/exif (a). Gambar asli (b). Gambar hasil modifikasi

Segmen tambahan APP0 yang berada setelah segmen SOI membuat aplikasi yang dikembangkan membaca segmen SOI memiliki panjang yang berbeda. Hal ini disebabkan aplikasi mengidentifikasi panjang segmen SOI dimulai dari awal data sampai dengan ditemukannya segmen APP1.

Hasil modifikasi kelompok ketiga adalah adanya perubahan konten pada konten-konten yang berhubungan langsung dengan gambar, yaitu DQT, SOF0, DHT dan SOS. Keempat segmen tersebut menyimpan data yang berkaitan langsung dengan tampilan gambar seperti dimensi gambar, jenis warna yang digunakan dan metode kompresi gambar. Perubahan yang terjadi pada tampilan gambar akan mengubah keempat segmen tersebut.

Tiga kelompok hasil pengujian modifikasi digunakan untuk memilih bagian-bagian penyusun *file fingerprint* pada tahap penelitian yang kelima. *Hash value* dari segmen SOI dan APP1 digunakan untuk mengidentifikasi bila terjadi perubahan tipe berkas, penambahan objek pada gambar dan modifikasi metadata. *Hash value* segmen SOF0 dipilih untuk mengidentifikasi tiga bentuk modifikasi, *recoloring*, *resizing* dan *cropping*. Pemilihan *hash value* segmen SOF0 dikarenakan segmen ini memiliki ukuran yang lebih kecil dibandingkan tiga segmen lainnya. Ukuran sebuah segmen akan mempengaruhi kecepatan akuisisi konten untuk proses penghitungan *hash value*. Hal ini dilakukan untuk mengantisipasi ukuran berkas gambar yang memiliki kualitas semakin baik seiring perkembangan teknologi perangkat kamera digital dan *smartphone*. Gambar 10 menunjukkan gabungan tiga *hash value* yang membentuk sebuah *file fingerprint*.

Total panjang *file fingerprint* adalah 3x128 = 384-bit *hexadecimal* atau 1536-bit dalam format biner. *File fingerprint* yang terbentuk memiliki ukuran berkas yang sangat kecil di bawah 1 KB yaitu 771 byte. Ukuran tersebut berlaku untuk setiap *file fingerprint* yang disusun dari berkas gambar dari kedua jenis *smartphone*.

SOI Hash Value
0ca0774dbde5c5b387452b263ff90fe73afdfc51
95c693657e7a3ef21526bc2e4c332b4d32a5f91
4c78f941018a4da4d03ae4d993cf637c55e58b8
c7bbfe174359a939818d5d0e159a643496830f

APP1 Hash Value
aeb519a6eb17c7ba0abf3edbf61767660c03131
12f731ddc1749bc91e102e65988c01c2b6361b
4a4477b51dacd45705eb8cbc0cacc16fd43b69

SOF0 Hash Value
5133947e92db9353e6eb38e56336a6d4539ea
255cf710835511bf1669d5260b9bbfd8a830725
7ee901549218466b269ca5755a9233ec0e342

Gambar. 10 *File fingerprint* sebuah berkas jpeg/exif

IV. KESIMPULAN

Tahapan penyusunan *file fingerprint* untuk berkas jpeg/exif dimulai dengan mencari lokasi setiap segmen berkas. Pencarian dilakukan berdasarkan *segment-marker* menggunakan algoritma Boyer-Moore *string matching*. Lokasi segmen digunakan sebagai batasan untuk menyalin konten segmen. Konten yang diperoleh kemudian diolah menggunakan algoritma SHA512 untuk menghasilkan nilai hash untuk setiap segmennya. Keenam nilai hash dari enam segmen diseleksi berdasarkan karakteristik perubahan strukturnya terhadap setiap jenis modifikasi berkas jpeg/exif. Nilai hash yang terpilih disusun secara sequensial untuk membentuk file fingerprint jpeg/exif.

Penyusunan *file fingerprint* untuk berkas jpeg/exif harus memperhatikan struktur berkas tersebut. Berkas gambar yang dihasilkan dari berbagai *smarthone* memiliki perbedaan dalam ukuran panjang segmennya. Berkas gambar yang memiliki panjang segmen yang tetap hanya memerlukan pencarian lokasi segmen sebanyak satu kali, sedangkan berkas gambar yang memiliki ukuran panjang segmen yang bervariasi membutuhkan pencarian lokasi segmen setiap kali dilakukan penyusunan *file fingerprint*. Pemilihan *hash value* segmen sebagai penyusun *file fingerprint* dilakukan berdasarkan perubahan yang terjadi pada segmen untuk setiap bentuk modifikasi.

Penyusunan *file fingerprint* pada penelitian ini hanya untuk berkas gambar tipe jpeg/exif. Penelitian selanjutnya diharapkan dapat menyusun *file fingerprint* untuk tipe berkas multimedia seperti audio dan video yang mulai banyak digunakan dalam komunikasi dengan merangkat *smartphone*.

REFERENSI

[1] A. S. Sukarno, "Pengembangan Aplikasi Pengamanan Dokumen Digital Memanfaatkan Algoritma Advance Encryption Standard, RSA Digital Signature dan Invisible Watermarking," *Pros. Semin. Nas. Apl. Teknol. Inf. 2013*, pp. 1–8, NaN-5022, 2013.

[2] H. Wijayanto, I. Riadi, and Y. Prayudi, "Encryption EXIF Metadata for Protection Photographic Image of Copyright Piracy," *Int. J. Res. Comput. Commun. Technol.*, vol. 5, no. 5, 2016.

[3] L. Refialy, E. Sediyono, and A. Setiawan, "Pengamanan Sertifikat Tanah Digital Menggunakan Digital Signature SHA-512 dan," *JUTISI*, vol. 1, pp. 229–234, 2015.

[4] V. Roussev, "An Evaluation of Forensic Similarity Hashes Vassil Roussev An evaluation of forensic similarity hashes," *Proc. Digit. Forencsic Res. Conf.*, pp. s34–s41, 2011.

[5] V. Roussev, "Hashing and Data Fingerprinting in Digital Forensics," E Secur. Priv., no. April, pp. 49–55, 2009.

[6] W. Stallings, *Cryptography and Network Security Principles and Practice*, 6th ed. New Jersey: Pearson Education Inc., 2014.

[7] NIST, *FIPS PUB 180-4 Secure Hash Standard (SHS)*, no. August. Gaithersburg: National Institute of Standards and Technology, 2015.

[8] A. Sahu and S. M. Ghosh, "Review Paper on Secure Hash Algorithm With Its Variants," *Int. J. Tech. Innov. Mod. Eng. Sci.*, vol. 3, no. 5, pp. 0–7, 2018.

[9] S. Hameed, "Digital Signature Based on Hash Functions," *Int. J. Adv. Eng. Technol. Manag. Appl. Sci.*, vol. 4, no. 1, pp. 88–89, 2017.

[10] I. Riadi and M. Sumagita, "Analysis of Secure Hash Algorithm (SHA) 512 for Encryption Process on Web Based Application," *Int. J. Cyber-Security Digit. Forensics*, vol. 7, no. 4, 2018.

[11] K. Al-Khamaiseh and S. Al-Shagarin, "A Survey of String Matching Algorithms," *Int. J. Eng. Res. Appl.*, vol. 4, no. June 2015, pp. 144–156, 2014.

[12] N. Jiji and T. Mahalakshmi, "Survey of Exact String Matching Algorithm for Detecting Patterns in Protein Sequence," *Adv. Comput. Sci. Technol.*, vol. 10, no. 8, pp. 2707–2720, 2017.

[13] M. Zubair, F. Wahab, I. Hussain, and M. Ikram, "Text Scanning approach for Exact String Matching," *Int. Conf. Netw. Inf. Technol.*, pp. 118–122, 2010.

[14] A. L. . Sandoval, D. M. . Gonzales, L. J. . Villaba, and J. Hernandez-Castro, "Analysis of errors in exif metadata on mobile devices," *Multimed Tools Appl*, no. 74, pp. 4735–4763, 2015.

HASIL CEK_60010313_(15)(1)

ORIGINALITY REPORT

7%	6%	6%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universitas Brawijaya Student Paper	1%
2	core.ac.uk Internet Source	1%
3	Eko Prianto. "ANALISIS EMPIRIS PERBANDINGAN KINERJA METODE HASHING PROGRESSIVE OVERFLOW DAN LINEAR QUOTIENT DALAM STUDI PEMBUATAN APLIKASI DEKSTOP ADMINISTRASI KEPEGAWAIAN", ILKOM Jurnal Ilmiah, 2016 Publication	1%
4	slidelegend.com Internet Source	1%
5	sinta3.ristekdikti.go.id Internet Source	1%
6	Submitted to School of Business and Management ITB Student Paper	1%

7	Imam Saputra, Surya Darma Nasution. "Analisa Algoritma SHA-256 Untuk Mendeteksi Orisinalitas Citra Digital", Prosiding Seminar Nasional Riset Information Science (SENARIS), 2019 Publication	1%
8	R. Abiraami, R. Anuradha, V. Johnpaul, M. Guruprasad, R. Gobinath, A. Sivakrishna, S. Shrihari. "Mechanical and flexural behaviour study on fibrillated concrete as partial replacement of M-Sand and metakaolin", Materials Today: Proceedings, 2020 Publication	1%

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		