Komputika: Jurnal Sistem Komputer
Volume 9, Nomor 1, April 2020, hlm. 75 - 83
Terakreditasi Peringkat 3, SK No. 28/E/KPT/2019
DOI: 10.34010/komputika.v9i1.2733

ISSN: 2252-9039 (print)
ISSN: 2655-3198 (online)

Identifikasi File JPEG dengan Metode Signature-Based Carving dalam Model Automata

Ardiansyah[1*], Nila Hardi[2], Windu Gata[3]

[1,2,3]Magister Ilmu Komputer, STMIK Nusa Mandiri
Jl. Kramat Raya No.18, Kota Jakarta Pusat, DKI Jakarta 10420
**email: 14002404@nusamandiri.ac.id*

(Naskah masuk: *21 Januari 2020*; diterima untuk diterbitkan: *19 Februari 2020*)

ABSTRAK *– Penggunaan format gambar JPEG semakin meningkat seiring dengan meningkatnya produksi foto digital yang dipicu oleh kemunculan smartphone dan perkembangan sosial media. Bahkan pada tahun 2017 diprediksi bahwa setiap orang akan memproduksi 1.600 foto digital dalam 1 tahun. Hal tersebut membuat file JPEG berperan penting dalam proses digital forensik, sehingga banyak metode identifikasi dan recovery file JPEG yang dikembangkan. Tulisan ini berusaha melihat proses identifikasi dan recovery file JPEG dengan metode Signature-Based Carving, sebuah metode carving yang paling sederhana, dengan sebuah model diagram Finite State Automata (FSA), sebuah model algoritma dasar dalam teori komputasi. Model FSA yang dibuat kemudian diuji menggunakan data berupa disk image yang disediakan untuk publik dari DigitalCorpora.org. Hasilnya model FSA tersebut dapat mengidentifikasi dan me-recovery file JPEG dengan metode Signature-Based Carving dengan syarat dan kondisi tertentu, diantaranya tidak terfragmentasi, memiliki header dan footer yang utuh, serta tidak ada thumbnail atau embedded JPEG files di dalam file JPEG tersebut.*

Kata Kunci *– digital forensic, file recovery, jpeg, file signature, finite state automata.*

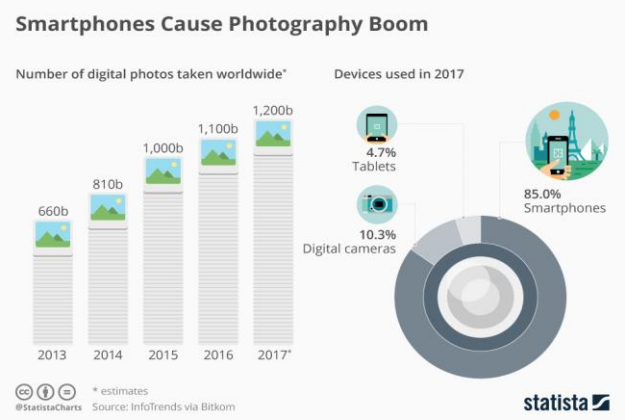
JPEG File Identification using Signature-Based Carving Method in the Automata Model

ABSTRACT *– The use of JPEG image formats is increasing along with the increase in digital photo production triggered by the emergence of smartphones and the development of social media. Even in 2017 it is predicted that each person will produce 1,600 digital photos in one year. This makes JPEG files play an important role in digital forensic processes so that many JPEG file identification and recovery methods are developed. This paper attempts to look at the process of identifying and recovering JPEG files with the Signature-Based Carving method, the simplest carving method, with a Finite State Automata (FSA) diagram model, a basic algorithm model in computational theory. The FSA model that was created was then tested using data in the form of a disk image made publicly available from DigitalCorpora.org. The result is that the FSA model can identify and recover JPEG files with the Signature-Based Carving method with certain terms and conditions, including unfragmented, full headers and footers, and no thumbnail or embedded JPEG files in the JPEG file.*

Keywords - digital forensic, file recovery, jpeg, file signature, finite state automata.

1. PENDAHULUAN

Perkembangan teknologi yang sangat pesat, khususnya dengan kemunculan *smartphone* dan perkembangan media sosial, membuat proses produksi dan distribusi file gambar sangat mudah dan sangat cepat. Dikutip dari artikel yang berjudul "Smartphones Cause Photography Boom", berdasarkan analisis dari InfoTrends yang dilakukan pada 2016, bahwa pada tahun 2017 akan ada 1,2 trilliun foto digital dibuat dalam 1 tahun. Dengan jumlah penduduk bumi sebanyak 7,5 miliar orang, maka rata-rata setiap orang di seluruh dunia akan memproduksi 1.600 buah foto digital dalam 1 tahun. Data pertumbuhan produksi foto digital diperlihatkan pada Gambar 1.

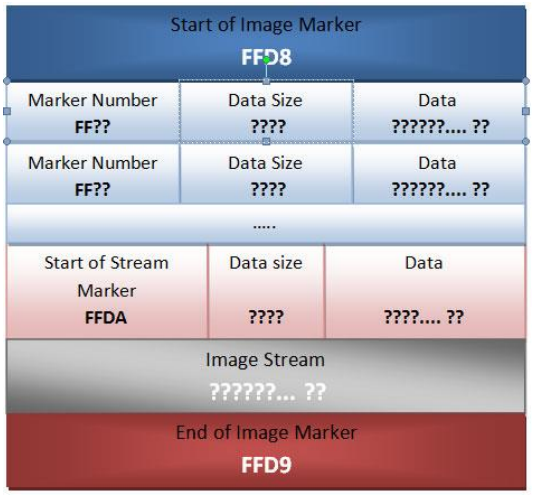


Gambar 1. Pertumbuhan produksi foto digital di dunia.

Format foto digital yang paling populer hingga saat ini adalah format JPEG yang dikembangkan oleh Joint Photographic Experts Group sejak tahun 1982 hingga disetujui sebagai ISO/IEC 10918-1 Standard pada tahun 1993. JPEG adalah sebuah standar kompresi internasional untuk *continuous-still image*, baik untuk gambar jenis *grayscale* maupun berwarna. Standar tersebut dirancang untuk dapat mendukung penggunaan gambar dan grafis dengan kualitas tinggi pada perangkat digital[1]–[3].

Setiap file JPEG memiliki penanda awal atau header yang disebut "*Start of Image*" (SOI) dan sebuah penanda akhir atau *trailer* yang disebut "*End of Image*" (EOI). Setiap file JPEG dimulai dengan nilai *binary* '0xFFD8' dan diakhiri dengan nilai *binary* '0xFFD9' [1], [4]–[8].

Sebuah file JPEG dimulai dengan data *binary* yang disebut *Markers* dengan format umum sebagai berikut: 0xFF + *Marker Number* (1 byte) + *Data Size* (2 bytes) + Data (n bytes). Beberapa *Markers* digunakan untuk mendeskripsikan data, sebelum SOI *Stream Marker* yang merupakan penanda awal dari JPEG *image stream* yang diakhiri dengan penanda EOI *Marker* [4], [9]. Struktur file JPEG secara umum ditunjukkan pada Gambar 2.



Gambar 2. Struktur file JPEG secara umum.

Beberapa *markers* yang penting untuk diketahui di dalam segmen *header* sebuah file JPEG ditunjukkan pada Tabel 1 [4].

Tabel 1. Daftar Markers Segmen Header File JPEG

Name	Bytes Value	Marker Use
SOI	0xFF 0xD8	Start of Image
APP0	0xFF 0xE0	JFIF Metadata
APP1	0xFF 0xE1	ExIF Metadata
DHT	0xFF 0xC4	Define Huffman Table
COM	0xFF 0xFD	Comment
EOI	0xFF 0xD9	End of Image

Sebuah file JPEG dapat memuat file JPEG lain yang lengkap dengan SOI/EOI nya. File JPEG di yang berada di dalam file JPEG tersebut disebut *thumbnail* atau *embedded image*. *Thumbnail* adalah versi kecil dari file gambar aslinya yang telah direduksi. Sebuah file JPEG dapat tidak memiliki *thumbnail* atau dapat memiliki hingga 2 buah *thumbnails* [1], [4], [5], [9].

Dengan pesatnya pertumbuhan produksi foto digital, maka kebutuhan akan kebutuhan akan *photo recovery* juga semakin meningkat mengingat karakter data digital yang sangat rentan mengalami kerusakan (*fragile*). Data digital, termasuk file foto, sangat mudah dihapus atau mengalami kerusakan baik disengaja maupun tidak. Selain itu, file-file gambar/foto juga seringkali menjadi bukti/petunjuk penting proses analisis forensik digital. Para pelaku kejahatan seringkali menyembunyikan file-file gambar/foto yang dapat menjadi bukti kejahatannya, sehingga dalam proses analisis forensik digital juga memerlukan proses identifikasi jenis file disamping proses *file recovery* [7], [8], [10]–[12].

File recovery adalah sebuah proses memunculkan kembali file yang sudah terhapus atau rusak, dari sebuah perangkat penyimpanan elektronik dengan metode tertentu. Secara garis besar, terdapat 2 metode yang digunakan untuk melakukan *file recovery*, yaitu dengan melakukan analisis terhadap *file system* dan analisis terhadap *raw data*. Salah satu metode *file recovery* dengan metode analisis terhadap *raw data* adalah metode *carving* [7], [10], [13], [14].

File carving adalah proses untuk mengembalikan file yang terhapus atau rusak, dari sebuah perangkat penyimpanan digital jika metadata *file system*-nya sudah tidak tersedia. Metode *carving* tidak memerlukan adanya metadata *file system*, karena akan melakukan analisis terhadap media penyimpanan secara *byte per byte*. Keuntungannya, file yang dapat di-*recovery* lebih banyak karena area pencariannya meliputi seluruh area media penyimpanan. Namun, karena tidak melakukan analisis terhadap metada *file system*, maka metode

carving tidak akan menemukan informasi seperti *filename*, *timestamp*, *exist/deleted*, fragmentasi, dsb [4], [5], [7], [15].

File carving dapat diklasifikasikan menjadi dua tingkatan, yaitu tingkat *basic* dan tingkat *advanced*. *Basic data carving* memiliki asumsi bahwa bagian awal dari sebuah file tidak rusak, file tersebut tidak terfragmentasi, dan file tersebut tidak terkompresi. Sedangkan pada tingkat *advanced* metode *carving* dapat digunakan meskipun terhadap file yang terfragmentasi karena metode *carving* yang digunakan melakukan analisis terhadap struktur file tersebut. Metode file *carving* dapat dibagi menjadi 3 (tiga) kategori yang diurutkan berdasarkan tingkat kerumitannya, yaitu *Signature-Based*, *Structure-Based*, dan *Content-Based* [3], [7], [13].

Untuk dapat menyelesaikan masalah dalam bidang komputasi, model algoritma dasar yang digunakan adalah model teori automata.

Automata adalah model abstrak dari mesin yang melakukan komputasi terhadap suatu input dengan bergerak melalui deretan states atau fungsi transisi untuk menuju konfigurasi berikutnya berdasarkan satu bagian dari konfigurasi saat ini [16]–[18].

Secara umum finite automata dapat dibagi menjadi dua, yaitu *Deterministic Finite-state Automata* (DFA) dan *Non-Deterministic Finite-state Automata* (NDFA/NFA). Pada DFA setiap input akan menuju ke satu *state* tertentu, sedangkan pada NFA setiap input dapat menuju ke lebih dari satu *state*.

Sebuah mesin automata dimana set Q hanya memiliki jumlah elemen berhingga disebut sebagai *finite-states machine* (FSM). FSM adalah sebuah mesin abstrak, yang terdiri dari suatu *set of states* (set Q), *set of input events* (set I), sebuah *set of output events* (set Z) dan sebuah *state* fungsi transisi. *State* fungsi transisi tersebut mengambil *state* terkini sebagai input dan menghasilkan suatu set baru dari *output events* dan *state* berikutnya. Sebuah *finite-state machine* secara formal didefinisikan sebagai 5 *tuple* (Q, I, Z, ∂, W) dimana[16] :

Q : *finite state of states*
I : *finite set of input symbols*
Z : *finite set of output symbols*
∂ : *mapping of* I x Q *into* Q *called the state transition function*, i.e. I x Q □ Q
W : *set of accept states where* F *is a subset of* Q.

Penelitian terhadap proses *recovery file* JPEG dengan metode *carving* sudah banyak dilakukan, diantaranya Rabei Raad Ali et. al. dalam papernya yang berjudul "*A Review Of Digital Forensics Methods for JPEG File Carving*" yang melakukan review terhadap beberapa metode *carving* dalam melakukan *file recovery* terhadap JPEG [7], Nurul Azma Abdullah et. al. dalam papernya yang berjudul "*Carving Thumbnail/s and Embedded JPEG Files Using Image Pattern Matching*" yang meneliti mengenai teknik *recovery thumbnail* dari file JPEG menggunakan algoritma PattrecCarv [1], Esra'a Alshammary dan Ali Hadi dalam papernya yang berjudul "*Reviewing and Evaluating Existing File Carving Techniques for JPEG Files*" yang melakukan penelaahan berbagai metode JPEG *file carving* dalam digital forensik [4], Xinyan Zha and Sartaj Sahni dalam paper yang berjudul "*Fast In-Place File Carving For Digital Forensics*" meneliti mengenai metode Scalpel dengan *multi-algorithm* untuk mempercepat proses *file carving* [19], serta Nadeem Alherbawi et. al. dalam "*Systematic Literature Review on Data Carving in Digital Forensic*" yang melakukan review terhadap berbagai literatur mengenai metode ekstraksi data dalam digital forensik [13].

Sebagian besar penelitian yang sudah ada lebih banyak membahas mengenai metode tingkat lanjut (*advanced*) untuk *file recovery* dalam digital forensik. Berbeda dengan penelitian-penelitian sebelumnya, paper ini akan membahas mengenai *data carving* file JPEG dengan metode *signature-based carving* dalam model *finite state automat*

2. METODE PENELITIAN

Di dalam penelitian ini menggunakan metode *Signature-Based Carving* untuk identifikasi dan *recovery* data file JPEG, sedangkan untuk model algoritma proses *carving* tersebut menggunakan model FSA sebagai sebuah model dasar dari teori komputasi. Untuk pengujian, model tersebut akan diterapkan untuk melakukan identifikasi dan *recovery* data file JPEG dari file *disk image* dari yang diunduh dari DigitalCorpora.org kemudian hasilnya akan dibandingkan dengan referensi dari DigitalCorpora.org mengenai daftar file JPEG yang terdapat di dalam *disk image* tersebut.

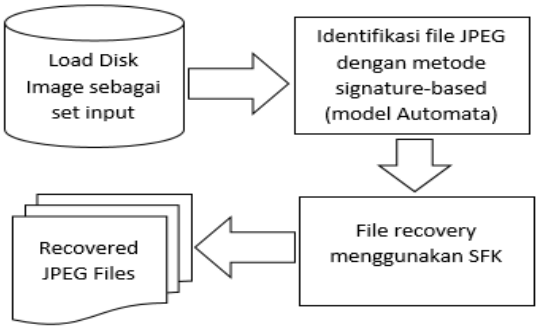
Metode yang digunakan di dalam FSA tersebut dengan mencari SOI dan EOI yang sesuai dengan karakteristik file JPEG berdasarkan nilai *binary* (dalam bentuk *hexadecimal*) yang dibaca *byte per byte* dari sebuah file. Dalam penelitian ini sebagai penanda awal file JPEG digunakan kombinasi dari nilai SOI, yaitu '0xFFD8', dan byte pertama dari *Marker Number*, yaitu '0xFF', sehingga untuk penanda awal file JPEG ditetapkan '0xFFD8FF'. Sedangkan sebagai penanda akhir sebuah file JPEG digunakan nilai EOI, yaitu '0xFFD9'.

Jika mesin automata sudah berhasil menemukan EOI tetapi masih terdapat set input yang tersisa, maka mesin akan kembali melakukan pencarian SOI, sehingga dimungkinkan mesin akan mengidentifikasi lebih dari 1 file JPEG di dalam satu set *input/file*.

Selanjutnya, dilakukan recovery data berdasarkan hasil identifikasi automata tersebut menggunakan tools Swiss File Knife (SFK), sebuah

aplikasi *command line* yang gratis dan *open source*, yang memiliki berbagai fungsi.

Proses identifikasi dan recovery file JPEG tersebut diperlihatkan pada Gambar 3.



Gambar 3. Diagram Proses Identifikasi dan Recovery File JPEG

Selanjutnya, berdasarkan model automata tersebut dibuat sebuah aplikasi sederhana dan dilakukan pengujian dengan menggunakan *disk image* yang diunduh dari DigitalCorpora.org (https://digitalcorpora.org/corpora/disk-images).

Dalam penelitian ini file *disk image* yang digunakan adalah file *disk image* dengan kategori "nps-2009-canon2" yang berisi *disk image* dari *memory card* 32 MB dari kamera "Canon PowerShot SD800IS". Di dalam kategori tersebut terdapat 6 versi *disk image* seperti ditunjukkan pada Tabel 2.

Tabel 2. Daftar File Disk Images "nps-2009-canon2"

No	File Name
1	nps-2009-canon2-gen1.raw
2	nps-2009-canon2-gen2.raw
3	nps-2009-canon2-gen3.raw
4	nps-2009-canon2-gen4.raw
5	nps-2009-canon2-gen5.raw
6	nps-2009-canon2-gen6.raw

Berdasarkan informasi yang disediakan oleh DigitalCorpora.org, bahwa file-file *disk image* tersebut dibuat dengan memasukkan sebuah SD Card ke dalam kamera, mengambil foto dengan kamera tersebut, melepaskan SD Card, menghapus beberapa foto, kemudian melakukan *disk imaging* terhadap SD Card tersebut. Proses tersebut diulang sebanyak 6 kali sehingga menghasilkan 6 versi file *disk image* dengan kondisi yang berbeda-beda. Pada kondisi terakhir (versi 6) beberapa file menjadi terfragmentasi, dan beberapa file sudah terhapus dari *file system* namun *raw data*-nya masih ada.

Setiap file foto tersebut berisi foto sebuah tulisan angka yang menunjukkan identitas file tersebut. Secara keseluruhan file foto yang diproduksi dalam proses pembuatan *disk image* tersebut ditunjukkan oleh Tabel 3.

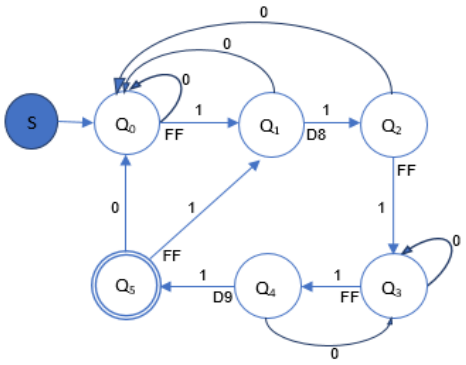
Tabel 3. Daftar file JPEG di dalam "nps-2009-canon2"

Filename	Text	Frag-ments	Filename	Text	Frag-ments
IMG_0001.JPG	1	-	IMG_0027.JPG	27	-
IMG_0002.JPG	2	-	IMG_0028.JPG	28	-
IMG_0003.JPG	3	-	IMG_0029.JPG	29	-
IMG_0004.JPG	4	-	IMG_0030.JPG	30	-
IMG_0005.JPG	5	-	IMG_0031.JPG	31	-
IMG_0006.JPG	6	-	IMG_0032.JPG	32	-
IMG_0007.JPG	7	-	IMG_0033.JPG	33	-
IMG_0008.JPG	8	-	IMG_0034.JPG	34	-
IMG_0009.JPG	9	-	IMG_0035.JPG	35	-
IMG_0010.JPG	10	-	IMG_0036.JPG	36	-
IMG_0011.JPG	11	-	IMG_0037.JPG	G2-1	-
IMG_0012.JPG	12	-	IMG_0038.JPG	G2-2	3
IMG_0013.JPG	13	-	IMG_0039.JPG	G2-3	-
IMG_0014.JPG	14	-	IMG_0040.JPG	G2-4	-
IMG_0015.JPG	15	-	IMG_0041.JPG	G3-1	-
IMG_0016.JPG	16	-	IMG_0042.JPG	G4-1	-
IMG_0017.JPG	17	-	IMG_0043.JPG	G5-1	4
IMG_0018.JPG	18	-	IMG_0044.JPG	G6-1	3
IMG_0019.JPG	19	-	IMG_0045.JPG	G6-2	-
IMG_0020.JPG	20	-	IMG_0046.JPG	G6-3	-
IMG_0021.JPG	21	-	IMG_0047.JPG	G6-4	-
IMG_0022.JPG	22	-	IMG_0048.JPG	G6-5	2
IMG_0023.JPG	23	-	IMG_0049.JPG	G6-6	-
IMG_0024.JPG	24	-	IMG_0050.JPG	G6-7	-
IMG_0025.JPG	25	-	IMG_0051.JPG	G6-8	-
IMG_0026.JPG	26	-			

Dari tabel tersebut diketahui bahwa file JPEG yang pernah disimpan di dalam *disk image* tersebut ada 51 file yang berisi tulisan seperti pada kolom "Text". Terdapat 4 file yang terfragmentasi yaitu file IMG_0038.JPG (3 fragments), IMG_0043.JPG (4 fragments), IMG_0044.JPG (3 fragments), dan IMG_0048 (2 fragments).

3. HASIL DAN PEMBAHASAN

Berdasarkan metode DFA yang telah dijelaskan, hasilnya adalah sebuah State diagram mesin automata untuk identifikasi file JPEG ditunjukkan pada Gambar 4.



Gambar 4. State diagram identifikasi file JPEG.

Q = {Q_0, Q_1, Q_2, Q_3, Q_4, Q_5}
∑ = {0, 1}
S = {I_0, I_1, I_2, … , I_n}
F = {Q_5}

Fungsi transisi dalam state diagram pada Gambar 4 dapat dijelaskan sebagai berikut:

S = himpunan input berupa himpunan bytes dari sebuah file/data yang akan dianalisis.
Q_0 = bernilai True (1) jika input I_n = 0xFF. Jika tidak, maka bernilai False (0).
Jika $Q_0 = 1 \rightarrow Q_1$, Jika $Q_0 = 0 \rightarrow Q_0$.
Q_1 = bernilai True (1) jika input I_n = 0xD8. Jika tidak, maka bernilai False (0).
Jika $Q_1 = 1 \rightarrow Q_2$, Jika $Q_1 = 0 \rightarrow Q_0$.
Q_2 = bernilai True (1) jika input I_n = 0xFF. Jika tidak, maka bernilai False (0).
Jika $Q_2 = 1 \rightarrow Q_3$, Jika $Q_2 = 0 \rightarrow Q_0$.
Q_3 = bernilai True (1) jika input I_n = 0xFF. Jika tidak, maka bernilai False (0).
Jika $Q_3 = 1 \rightarrow Q_4$, Jika $Q_3 = 0 \rightarrow Q_1$.
Q_4 = bernilai True (1) jika input I_n = 0xD9. Jika tidak, maka bernilai False (0).
Jika $Q_4 = 1 \rightarrow Q_5$, Jika $Q_4 = 0 \rightarrow Q_3$.
Q_5 = Final state.
Jika S = {} → Finish, Jika S = {I_n} → Q_0.

Fungsi-fungsi transisi pada diagram tersebut dapat dituliskan sebagai berikut:

$\delta(Q_0 \cdot 0) = Q_0$
$\delta(Q_0 \cdot 1) = Q_1$
$\delta(Q_1 \cdot 0) = Q_0$
$\delta(Q_1 \cdot 1) = Q_2$
$\delta(Q_2 \cdot 0) = Q_0$
$\delta(Q_2 \cdot 1) = Q_3$
$\delta(Q_3 \cdot 0) = Q_3$
$\delta(Q_3 \cdot 1) = Q_4$
$\delta(Q_4 \cdot 0) = Q_3$
$\delta(Q_4 \cdot 1) = Q_5$
$\delta(Q_5 \cdot 0) = Q_0$
$\delta(Q_5 \cdot 1) = Q_1$

Fungsi transisi tersebut dapat ditampilkan dalam sebuah tabel transisi pada Tabel 4.

Tabel 4. Daftar Markers Segmen Header JPEG

Δ	0	1
Q_0	Q_0	Q_1
Q_1	Q_0	Q_2
Q_2	Q_0	Q_3
Q_3	Q_3	Q_4
Q_4	Q_3	Q_5
Q_5	Q_0	Q_1

Untuk menguji model automata tersebut, dibuat sebuah aplikasi sederhana berdasarkan model automata pada Gambar 4 dengan pseudocode sebagai berikut:

```
Input lokasi file (source file) yang akan diidentifikasi
Load source file
def main():
    buat file "result.txt" untuk menampung hasil identifikasi
    baca source file per 1 byte
    tetapkan variabel state = "start"
    tetapkan variabel no_urut = 0
    tetapkan variabel cursor_pos = 0
    while byte != "":
      cursor_pos = cursor_pos + 1
      jika(state=="start") maka state = "q0"
      jika (state=="q0") maka:
        jika Q0(byte) == True maka state = "q1"
        jika Q0(byte) == False maka state = "q0"
      jika (state=="q1") maka:
        jika Q1(byte) == True maka state = "q2"
        jika Q1(byte) == False maka state = "q0"
      jika (state=="q2") maka:
        jika Q2(byte) == True maka:
          state = "q3"
          SOI_offset = cursor_pos - 1
          SOI_chr = source_file_content[SOI_pos:2]
          tuliskan no_urut  + "," + SOI_offset pada result file
          no_urut = no_urut + 1
        jika Q2(byte) == False maka state = "q0"
      jika (state=="q3") maka:
        jika Q3(byte) == True maka state = "q4"
        jika Q3(byte) == False maka state = "q3"
      jika (state=="q4") maka:
        jika Q4(byte) == True maka:
          state = "q5"
          EOI_offset = cursor_pos - 1
          EOI_chr = source_file_content[EOI_pos:2]
          tuliskan EOI_offset + new line pada result file
        jika Q4(byte) == False maka state = "q3"

# Definisi fungsi-fungsi transisi

def Q0(byte):
  Jika (byte=="FF") maka True
  Jika (byte!="FF") maka False

def Q1(byte):
  Jika (byte=="D8") maka True
Jika (byte!="D8") maka False
```

```
def Q2(byte):
    Jika (byte=="FF") maka True
    Jika (byte!="FF") maka False

def Q3(byte):
    Jika (byte=="FF") maka True
    Jika (byte!="FF") maka False

def Q4(byte):
    Jika (byte=="D9") maka True
    Jika (byte!="D9") maka False
```

Hasil *output* dari aplikasi tersebut adalah sebuah daftar lokasi (*offset*) dari SOI dan EOI *marker* yang ditemukan di dalam *source file*, dengan format nomor urut, SOI *offset*, EOI *offset*.

Dari daftar tersebut, dilakukan *file recovery* dengan cara menyalin (*copy*) *source file* secara parsial dari lokasi SOI *offset* sampai dengan lokasi EOI *offset* menggunakan *tools* SFK dengan format perintah sebagai berikut:

```
sfk partcopy source_file -fromto SOI_offset
EOI_offset no_urut.jpg -yes
```

Perintah tersebut dijalankan untuk setiap baris pasangan SOI dan EOI yang tercatat di dalam file result.txt.

Untuk pengujian di dalam penelitian ini hanya akan menggunakan 2 versi *disk image* dari 6 *disk image* yang tersedia, yaitu versi pertama (nps-2009-canon2-gen1.raw) dan versi terakhir (nps-2009-canon2-gen6.raw). Meskipun dalam keterangannya DigitalCorpora.org menyebutkan disk image tersebut memiliki format "raw dd", namun pada laman download hanya tersedia file dengan format "E01". Sehingga terhadap *disk images* yang telah diunduh tersebut dilakukan reakuisisi ke dalam format "raw dd" menggunakan aplikasi FTK Imager. Laman download folder "nps-2009-canon2" diperlihatkan pada Gambar 5.

Index of /corpora/drives/nps-2009-canon2

Name	Last modified	Size	Description
Parent Directory		-	
files-gen6.zip	29-Aug-2012 08:42	28M	
narrative.txt	30-Jan-2010 22:25	2.6K	
nps-2009-canon2-gen1.E01	21-Feb-2013 17:11	28M	
nps-2009-canon2-gen2.E01	12-May-2015 12:06	29M	
nps-2009-canon2-gen3.E01	21-Feb-2013 17:11	29M	
nps-2009-canon2-gen4.E01	21-Feb-2013 17:11	29M	
nps-2009-canon2-gen5.E01	21-Feb-2013 17:11	29M	
nps-2009-canon2-gen6.E01	05-Aug-2010 20:06	30M	
report.txt	29-Jan-2009 15:32	9.2K	
report.xml	29-Jan-2009 15:32	24K	

Apache/2.2.15 (CentOS) Server at downloads.digitalcorpora.org Port 80

Gambar 5. Laman download folder "nps-2009-canon2"

File *disk images* yang diunduh dan akan digunakan sebagai bahan pengujian ditunjukkan pada Tabel 5.

Tabel 5. Daftar File Uji yang Diunduh dari DigitalCorpora.org

Filename	MD5 Checksum
nps-2009-canon2-gen1.E01	8aa75ca97507f20c-0887bb2b826cbc48
nps-2009-canon2-gen6.E01	750b509d8fbed37-a5213480aaccfdc61

Setelah diunduh, dilakukan reakuisisi (*reimaging*) terhadap file-file tersebut menggunakan FTK Imager dengan format *output* raw/dd, sehingga menghasilkan file seperti pada Tabel 6.

Tabel 6. Daftar File Uji Hasil Reakuisisi

Filename	MD5 Checksum
nps-2009-canon2-gen1.001	8aa75ca97507f20c-0887bb2b826cbc48 [verified]
nps-2009-canon2-gen6.001	750b509d8fbed37-a5213480aaccfdc61 [verified]

Selanjutnya kedua file tersebut dijadikan sebagai bahan uji secara berurutan dimulai dari file "nps-2009-canon2-gen1.001". Pengujian dilakukan dengan melakukan identifikasi file JPEG menggunakan script Python kemudian melakukan file recovery menggunakan SFK berdasarkan output hasil identifikasi tersebut.

Hasil pengujian terhadap file "nps-2009-canon2-gen1.001" ditemukan 72 files JPEG berdasarkan identifikasi file signature-nya seperti pada Tabel 7.

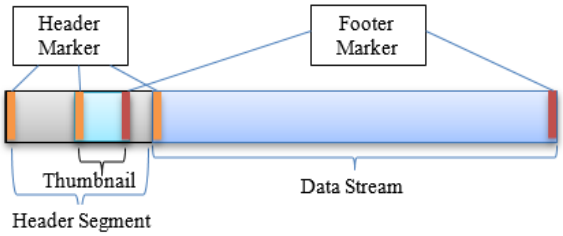
Tabel 7. Resume Hasil Identifikasi dan Recovery Disk Image "nps-2009-canon2-gen1.001"

Hasil	Jumlah
Gambar dapat ditampilkan secara utuh	36
Gambar dapat ditampilkan sebagian	0
Gambar tidak dapat ditampilkan	36
Jumlah file yang direcovery	72

Berdasarkan informasi pada Tabel 3, seharusnya terdapat 36 file JPEG di dalam file "nps-2009-canon2-gen1.001", namun dari hasil pengujian ditemukan 72 buah SOI dan EOI atau ditemukan 72 file JPEG. Jumlah tersebut berarti 2 kali lipat dari jumlah file

yang seharusnya. Namun, hanya 36 file yang menampilkan gambar dengan sempurna, sedangkan 36 file lainnya tidak dapat menampilkan gambar (#N/A).

Setelah dilakukan analisis terhadap file-file tersebut, bahwa file yang tidak dapat menampilkan gambar seharusnya merupakan bagian dari segmen header sedangkan file yang dapat menampilkan gambar merupakan bagian dari segmen data stream. Di dalam segmen header tersebut terdapat thumbnail image dalam format JPEG, sedangkan gambar utamanya terdapat di dalam segmen data stream. Sehingga di dalam setiap file JPEG tersebut terdapat 3 buah SOI dan 2 buah EOI. File thumbnail tersebut tidak dapat ditampilkan karena SOI yang diperoleh adalah SOI dari awal file JPEG, bukan SOI dari awal thumbnail image tersebut. Sedangkan file gambar yang diperoleh dari segmen data stream dapat tampil sempurna karena SOI yang diperoleh adalah SOI dari awal data stream. Struktur File JPEG dengan Embedded Thumbnail ditampilkan pada Gambar 6.



Gambar 6. Struktur File JPEG dengan Embedded Thumbnail

Secara umum bahwa hasil pengujian terhadap "nps-2009-canon2-gen1.001" dapat mengidentifikasi dan me-recovery seluruh file JPEG yang ada di dalam disk image tersebut.

Selanjutnya dilakukan pengujian terhadap disk image "nps-2009-canon2-gen6.001" yang merupakan versi terakhir dari "nps-2009-canon2". Karena sudah ada file yang dihapus (delete) dan ada file baru yang dibuat (create), maka di dalam disk image ini terdapat beberapa file JPEG yang terfragmentasi.

Hasil pengujian terhadap disk image "nps-2009-canon2-gen6.001" ditemukan 68 files JPEG berdasarkan identifikasi file signature-nya seperti ditunjukkan pada Tabel 8.

Tabel 8. Resume Hasil Identifikasi dan Recovery Disk Image "nps-2009-canon2-gen6.001"

Hasil	Jumlah
Gambar dapat ditampilkan secara utuh	31
Gambar dapat ditampilkan sebagian	8
Gambar tidak dapat ditampilkan	29
Jumlah file yang direcovery	68

Jumlah file yang berhasil di-*recovery* lebih sedikit karena setelah melalui proses beberapa kali penghapusan file dan pembuatan file baru, maka terdapat beberapa file yang terfragmentasi dan residu file yang tidak lengkap (tidak memiliki header/footer). Hal tersebut menyebabkan metode *Signature-Based Carving* tidak dapat melakukan *file recovery* dengan sempurna terhadap file-file tersebut.

Berdasarkan informasi yang disertakan oleh DigitalCorpora.org bahwa file JPEG yang dapat di-*recovery* dari *disk image* "nps-2009-canon2-gen6.001" sebanyak 38 file JPEG. Jika dilihat berdasarkan status keberadaan file tersebut, terdapat 5 buah file yang sudah dihapus (*residual file*). Sedangkan jika ditiinjau dari fragmentasinya, terdapat 4 buah file yang terfragmentasi (*fragmented file*). Hasil selengkapnya dapat dilihat pada tabel 9, Tabel 10, Tabel 11 dan Tabel 12.

Tabel 9. Resume Hasil Recovery Disk Image "nps-2009-canon2-gen1.001" Berdasarkan Status Keberadaanya

Hasil	Jumlah
Resident file	33
Residual File	5

Tabel 10. Resume Hasil Recovery Disk Image "nps-2009-canon2-gen1.001" Berdasarkan Fragmentasinya

Hasil	Jumlah
Non-fragmented	34
Fragmented	4

Tabel 11. Daftar Residual Files pada Disk Image "nps-2009-canon2-gen1.001"

No	Nama File
1	IMG_0014.JPG
2	IMG_0020.JPG
3	IMG_0025.JPG
4	IMG_0030.JPG
5	IMG_0035.JPG

Tabel 12. Daftar Fragmented Files pada Disk Image "nps-2009-canon2-gen1.001"

No	Nama File	Jumlah Fragmentasi
1	IMG_0038.JPG	3
2	IMG_0043.JPG	4
3	IMG_0044.JPG	3
4	IMG_0048.JPG	2

Merujuk pada daftar file JPEG yang pernah disimpan di dalam *disk image* tersebut (Tabel 3), file JPEG yang berhasil diidentifikasi dan di-*recovery* ditunjukkan pada Tabel 13.

Tabel 13. Daftar Hasil Identifikasi dan *Recovery Disk Image*

Filename	G1	G6	Filename	G1	G6
IMG_0001.JPG	✓	✓	IMG_0027.JPG	✓	✓
IMG_0002.JPG	✓	×	IMG_0028.JPG	✓	✓
IMG_0003.JPG	✓	✓	IMG_0029.JPG	✓	✓
IMG_0004.JPG	✓	×	IMG_0030.JPG	✓	×
IMG_0005.JPG	✓	×	IMG_0031.JPG	✓	✓
IMG_0006.JPG	✓	×	IMG_0032.JPG	✓	✓
IMG_0007.JPG	✓	✓	IMG_0033.JPG	✓	✓
IMG_0008.JPG	✓	×	IMG_0034.JPG	✓	✓
IMG_0009.JPG	✓	✓	IMG_0035.JPG	✓	×
IMG_0010.JPG	✓	×	IMG_0036.JPG	✓	✓
IMG_0011.JPG	✓	✓	IMG_0037.JPG	×	×
IMG_0012.JPG	✓	×	IMG_0038.JPG	×	×
IMG_0013.JPG	✓	✓	IMG_0039.JPG	×	×
IMG_0014.JPG	✓	✓	IMG_0040.JPG	×	×
IMG_0015.JPG	✓	×	IMG_0041.JPG	×	×
IMG_0016.JPG	✓	✓	IMG_0042.JPG	×	✓
IMG_0017.JPG	✓	✓	IMG_0043.JPG	×	×
IMG_0018.JPG	✓	✓	IMG_0044.JPG	×	×
IMG_0019.JPG	✓	✓	IMG_0045.JPG	×	✓
IMG_0020.JPG	✓	×	IMG_0046.JPG	×	✓
IMG_0021.JPG	✓	✓	IMG_0047.JPG	×	✓
IMG_0022.JPG	✓	✓	IMG_0048.JPG	×	×
IMG_0023.JPG	✓	✓	IMG_0049.JPG	×	✓
IMG_0024.JPG	✓	✓	IMG_0050.JPG	×	✓
IMG_0025.JPG	✓	×	IMG_0051.JPG	×	✓
IMG_0026.JPG	✓	✓			

Keterangan:
G1 : nps-2009-canon2-gen1 ✓ : Berhasil
G6 : nps-2009-canon2-gen6 × : Tidak berhasil

Hasil proses terhadap *disk image* G1 dapat mengidentifikasi dan me-*recovery* 36 file JPEG (IMG_0001.JPG sampai dengan IMG_0036). File IMG_0037.JPG sampai dengan IMG_0051.JPG tidak dapat teridentifikasi karena sesuai informasi yang disertakan oleh DigitalCorpora.org bahwa *disk image* nps-2009-canon2-gen1 menyimpan 36 file JPEG. Artinya bahwa model FSA yang digunakan memiliki tingkat keberhasilan 100% pada *disk image* nps-2009-canon2-gen1.

Hasil proses terhadap *disk image* G6 dapat mengidentifikasi dan me-recovery 31 file JPEG dari 51 file yang pernah disimpan di dalam *disk image* tersebut. Hal tersebut terjadi karena *disk image* G6 telah mengalami beberapa kali penambahan dan penghapusan file sehingga terdapat beberapa bagian file yang sudah terhapus dan beberapa file JPEG yang terfragmentasi.

4. KESIMPULAN

Berdasarkan hasil penelitian dan pengujian yang telah dilakukan, dapat disimpulkan bahwa proses identifikasi dan *recovery* file JPEG dengan metode *Signature-Based Carving* dapat dibuat dalam sebuah model FSA meskipun terdapat beberapa kondisi yang harus dipenuhi agar mesin automata tersebut dapat memberikan hasil yang lebih baik. Kondisi yang harus dipenuhi tersebut diantaranya : file tidak terfragmentasi, file memiliki *header* dan *footer* yang utuh, serta tidak ada *thumbnail* atau *embedded JPEG files* di dalam file JPEG tersebut.

Selanjutnya hasil penelitian ini masih dapat dikembangkan lagi untuk membuat model automata proses file *recovery* menggunakan teknik yang lebih *advanced* sehingga dapat menghasilkan *output* yang lebih.

UCAPAN TERIMA KASIH

Terima kasih kami ucapkan kepada STMIK Nusa Mandiri yang telah memberikan dukungan dalam penelitian ini.

DAFTAR PUSTAKA

[1] N. A. Abdullah, R. Ibrahim, and K. M. Mohamad, "Carving Thumbnail/s and Embedded JPEG Files Using Image Pattern Matching," *J. Softw. Eng. Appl.*, vol. 06, no. 03, pp. 62–66, 2013.

[2] M. I. Cohen, "Advanced carving techniques," *Digit. Investig.*, vol. 4, no. 3–4, pp. 119–128, 2007.

[3] P. Alvarez, "Using extended file information (EXIF) file headers in digital evidence analysis," *Int. J. Digit. Evid.*, vol. 2, no. 3, pp. 1–5, 2004.

[4] E. Alshammary and A. Hadi, "Reviewing and evaluating existing file carving techniques for JPEG files," in *Proceedings - 2016 Cybersecurity and Cyberforensics Conference, CCC 2016*, 2016, no. October 2017, pp. 55–59.

[5] B. Schildendorfer, "Carving fragmented JPEG images," University of Applied Science St. Poelten, 2012.

[6] S. Flowerday and R. Von Solms, "Security and Privacy in Dynamic Environments," *IFIP Int. Fed. Inf. Process.*, vol. 201, no. May 2006, pp. 340–350, 2006.

[7] R. R. Ali, K. M. Mohamad, S. Jamel, and S. K. A. Khalid, "A review of digital forensics methods for JPEG file carving," *J. Theor. Appl. Inf. Technol.*, vol. 96, no. 17, pp. 5841–5856, 2018.

[8] M. Karresand and N. Shahmehri, "Reassembly of fragmented JPEG images containing restart

markers," *Proc. - 4th Annu. Eur. Conf. Comput. Netw. Defense, EC2ND 2008*, pp. 25–32, 2008.

[9] P. Mullan, C. Riess, and F. Freiling, "Forensic source identification using JPEG image headers: The case of smartphones," *Digit. Investig.*, vol. 28, pp. S68–S76, 2019.

[10] S. L. Garfinkel, "Digital forensics research: The next 10 years," *Digit. Investig.*, vol. 7, no. SUPPL., pp. S64–S73, 2010.

[11] N. Memon and A. Pal, "Automated reassembly of file fragmented images using greedy algorithms," *IEEE Trans. Image Process.*, vol. 15, no. 2, pp. 385–393, 2006.

[12] R. F. Sri Supatmi, Taufiq Nuzwir Nizar, "Perangkat Pendukung Forensik Lalu Lintas Jaringan," *J. Tek. Komput. Unikom – Komputika*, vol. 3, no. 2, pp. 32–33, 2014.

[13] N. Alherbawi, Z. Shukur, and R. Sulaiman, "Systematic Literature Review on Data Carving in Digital Forensic," *Procedia Technol.*, vol. 11, no. March 2014, pp. 86–92, 2013.

[14] V. L. L. Thing, T. W. Chua, and M. L. Cheong, "Design of a digital forensics evidence reconstruction system for complex and obscure fragmented file carving," *Proc. - 2011 7th Int. Conf. Comput. Intell. Secur. CIS 2011*, pp. 793–797, 2011.

[15] O. S. Sitompul, A. Handoko, and R. F. Rahmat, "File reconstruction in digital forensic," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 16, no. 2, pp. 776–794, 2018.

[16] R. O. Akinyede and O. Fajuyigbe, "Theory of Computation, Automata and Languages," *Ife J. Sci.*, vol. 10, no. 1, pp. 199–206, 2008.

[17] S. Jalan, P. Kumar, and S. Das, "Formalization of digital forensic theory by using Buchi Automaton," *Proc. 2015 3rd Int. Conf. Image Inf. Process. ICIIP 2015*, pp. 102–108, 2016.

[18] J. L. Massey, "Applications of Automata Theory in Coding," *Applied Automata Theory*, 1968. [Online]. Available: https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/automata-theory/basics.html.

[19] X. Zha and S. Sahni, "Fast in-Place File Carving for Digital Forensics," *Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng.*, vol. 56, pp. 141–158, 2011.