JOIN (Jurnal Online Informatika)
Volume 4 No. 1 | Juni 2019 : 16-23
DOI: 10.15575/join.v4i1.304

Implementation of SHA512 Hash Function and Boyer-Moore String Matching Algorithm for JPEG/exif Message Digest Compilation

Rachmad Fitriyanto[1], Anton Yudhana[2], Sunardi[3]
[1]Magister of Informatics Engineering, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
[2,3]Electrical Engineering Department, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
[1]1fitriyanto7477@gmail.com, [2]yudhana@ee.uad.ac.id, [3]sunardi@mti.uad.ac.id

Abstract- Security information method for JPEG/exif documents generally aims to prevent security attacks by protecting documents with password and watermark. Both methods cannot be used to determine the condition of data integrity at the detection stage of the information security cycle. Message Digest is the essence of a file that has a function as a digital fingerprint to represent data integrity. This study aims to compile digital fingerprints to detect changes that occurred in JPEG / exif documents in information security. The research phase consists of five stages. The first stage, identification of the JPEG / exif document structure conducted using the Boyer-Moore string matching algorithm to find JPEG/exif segments location. The Second stage is segment content acquisition, conducted based on segment location and length obtained. The Third step, computing message digest for each segment using SHA512 hash function. Fourth stage, JPEG / exif document modification experiments to identified affected segments. The fifth stage is selecting and combining the hash value of the segment into the message digest. The obtained result shows the message digest for JPEG/exif documents composed of three hash values. The SOI segment hash value used to detect modifications for JPEG to png conversion and image editing. The APP1 hash value used to detect metadata editing. The SOF0 hash values use to detect modification for image recoloring, cropping and resizing — the combination from three hash values as JPEG/exif's message digest.

***Keywords*- Message digest, JPEG/exif, SHA512, Hash value, Boyer-Moore**

I. INTRODUCTION

The JPEG/exif document is a document format that results from the use of digital cameras such as a smartphone camera. The JPEG/exif documents as image files are widely used in digital communications such as on social media. The Exchange of information requires security to ensure the information received is the same as the information sent. Information security for JPEG/exif documents generally designed to prevent document modifications. The password usages in the JPEG/exif document has long been used but can still be overcome with a variety of password remover tools that are widely available. The other technique for securing JPEG/exif is shown in the study by Wijayanto [1]. This study shows exif metadata data from JPEG/exif documents can be used to prevent copyright theft. The exif metadata provided information for digital analysis to detect altered JPEG/exif. Gangwar in [2] presented the methodology by comparing two exif metadata to analysis the data integrity of JPEG/exif's file. Another image file securing technique found in [3]. Int this research, the JPEG file secured by replacing selected parts of an image with another object. The exif metadata has secured before replacing the process beginning. The other technique beside the usages of exif metadata is the usage of watermark as an information security method in a study by Sukarno [4] to protect preventing document for modification. The purpose of watermark usage for JPEG/exif file security is to inform the attacker that the files protected by the physical presence of owner copyright [5]. The watermarks as a security method given robust protection for image geometry attack found in [6]. The research shows hidden and extracted watermark make the image files have more security the visible watermarks. The use of passwords, exif metadata, and watermark given great protection to prevent an attack on image files, but cannot be used in the detection stage of information security cycle to detect changes that occur in received JPEG/exif documents.

Message Digest is the essence of a file that can use to represent data integrity. The Message Digest is widely used to detect the integrity of data in installers provided by open-source application developers. The Message digest compiled using the hash function. The Hash function is a cryptographic method for the one-way encryption process. The output hash function is a hash value that has characteristics that cannot translate or decrypt into the original form [7]. The Hash value is very sensitive to changes in the input of any size. Secure Hash Algorithm (SHA) is a hash function developed by the National Institute of Science and Technology (NIST).

The characteristics from SHA that used for information security are one-way hash functions. This means that SHA generates a hash value that cannot decrypt [8]. Another SHA characteristic is this hash algorithm has high sensitivity for every input even a small change. Any modification or change in the input message will give different results [9].

SHA implementation for information security found in research by [8]. The SHA used to secure wireless communication by creating a message digest from the transmitted messages. The hash function in this research use the second generation of SHA. SHA consists of several categories that distinguished based on the size of the output hash value [10]. SHA variant from [8] use SHA-1 which has a 160-bit size of the hash value. The weakness from SHA-1 found in 2005 by Rijmen and Oswald [11]. The attack on SHA-1 made NIST developed the next generation of SHA. The variants of SHA produced SHA-2 with SHA256 and SHA512. Both variants in SHA-2 have longer hash value. Between two variants from SHA-2, SHA512 have better performance from SHA256. SHA512 has an output size of 512 bits that make this variant better than the previous hash function [11].

The use of SHA512 for information security was found in a study by Refialy [12]. This study uses SHA512 to compile the hash value of a pdf document. The obtained results indicate the hash value can detect small changes in the modification of pdf documents. The JPEG/exif file has a larger size than a pdf document. The size of the JPEG/exif document is the result of the development of optical technology in digital cameras. The larger the size of the document, the process of composing the message digest requires more time. This study aims to compile a concise message digest to detect changes in JPEG /exif documents in information security.

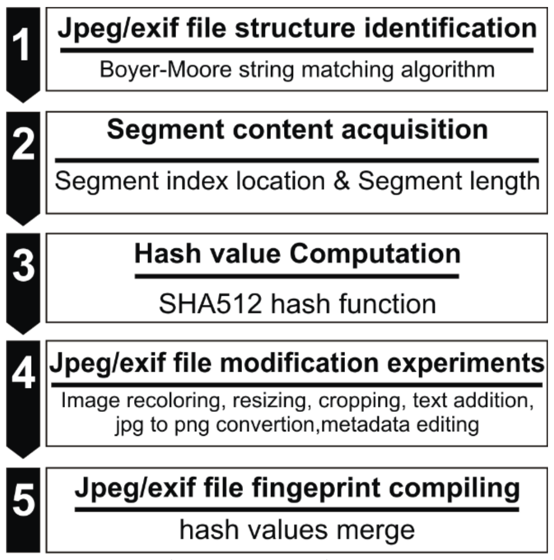
String matching algorithms are used to match or compare one or several characters and strings [13]. The Boyer-Moore string matching algorithm is included in the exact string matching algorithm that performs searches by comparing the characters of the strings tested with the pattern sought [14]. The implementation of the Boyer-Moore string matching algorithm found in research by Ramos-Frías [15]. This research was presenting the usage of the Boyer-Moore string matching algorithm for enhanced data processing in the graphics processing unit (GPU). Another implementation is in medical science for human DNA searching [16][17].

The Boyer-Moore has two rules for the searching process, Good-Character rule, and Bad-Character Rule. These two rules determined the direction of the searching process. Bad-Character rule occurred if a character from the pattern is not

the same as in string. This condition will make a comparison shift to the left to next character of pattern. Good-Character rule occurred if a character from the pattern is the same as in string after Bad-Character rule occurred. This condition will make the next comparison shift right aligning two-match characters. Those two rules searching make the searching process faster than other exact string matching algorithms by avoiding not-necessary character comparison [18].

II. METHOD

A research study conducted in five stages that shown as in Figure 1. The first stage is the identification of the image file segment. This stage has the purpose to identify the JPEG/exif file structure. Image files as research object acquired from two smartphone types, Asus Z00UD, and Samsung Galaxy A5. Each smartphone takes 10 images. Images file taken with embedded camera apps from each smartphone with mode auto from indoor and outdoor sites.


1
Jpeg/exif file structure identification
Boyer-Moore string matching algorithm
2
Segment content acquisition
Segment index location & Segment length
3
Hash value Computation
SHA512 hash function
4
Jpeg/exif file modification experiments
Image recoloring, resizing, cropping, text addition, jpg to png convertion,metadata editing
5
Jpeg/exif file fingeprint compiling
hash values merge
Figure 1. Research stages

A result of the first stage is the location index of each file part. This location index will use on the second stage as a parameter to identify the beginning and end of file parts. Identification process conducted by use the Boyer-Moore string matching algorithm for segment marker searching. Segment markers are data bit located at the beginning of each JPEG/exif segments [19]. JPEG/exif segment marker consists of four-bit hexadecimal. These bits have functioned as file structure signatures and contain specific information for computer and image viewer application to determine the suitable method to display image. Table 1 shows a segment marker value for each segment.

Table 1. JPEG/exif segment marker

Segment	Segment Marker
SOI (Start Of Image)	ffd8
APP1 (Application-1)	ffe1
DQT (Define Quantization Table)	ffdb
SOF0 (Start Of Frame-0)	ffc0
DHT (Define Huffman Table)	ffc4
SOS (Start Of Scan)	ffda

Segmen structured as sequences as shown in Table 1 for the image file that has not altered. Each segment marker used as a pattern that sought in string matching algorithm. Boyer-Moore string matching algorithm start pattern searching from most-right pattern character and shift to left until reaching the leftmost character from pattern [20]. Figure 2 shown the Boyer-Moore flowchart.

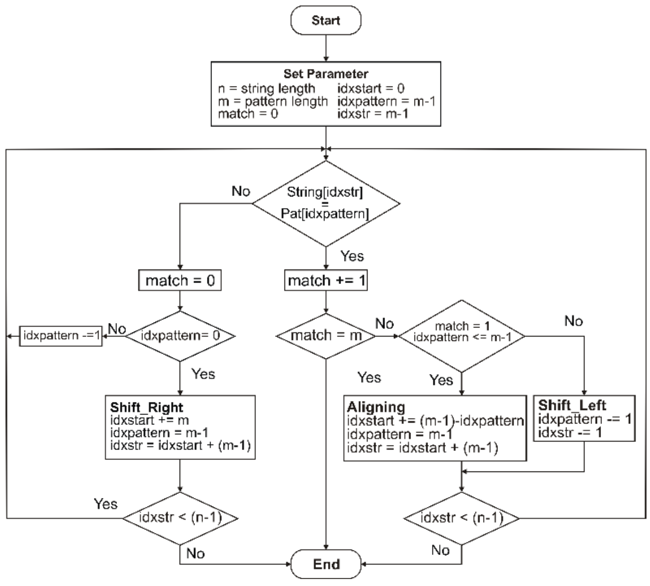


Figure 2. Boyer-Moore string matching flowchart

The Boyer-Moore string matching algorithm has two searching stages. The first stage is preprocessing, comprises of variable assignments such as m for pattern length, n for string length idxstr for string character index, idxpatter for pattern character index and match for character match comparison that occurred. The second stage is the character comparison. The second stage executed as an iteration loop that boundary by two conditions. Looping will stop if the match variable has valued the same as m variable or comparison has reached the end of the string. The segment location index identified by the last value of idxstr variable.

The second stage is segment content acquisition. To acquire segment content, we need two parameters, starting index and length of content. The starting index provided by the segment index location. Segment length computes from subtraction between index location values from two adjacent segments. The third stage is the hash value computation. The computation conducted for every segment. Hash value computation with SHA512 hash function consists of three stages, preprocessing, hash computation and hash value compilation. Figure 3 show the sequences of the stages.

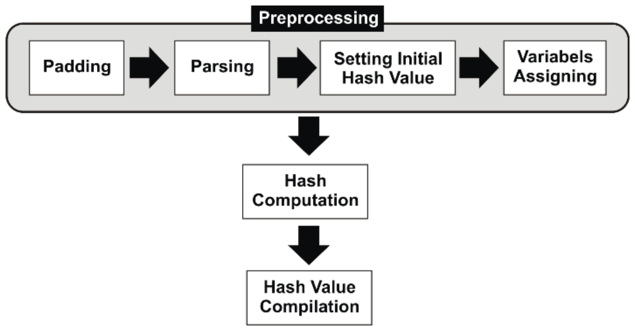


Figure 3. SHA512 computation stages

The preprocessing stages consist of four processes, padding, parsing, setting initial hash value and variable assigning [3]. The padding process is the adjustment of the size of input data so that the processed data has a size of multiples of 512 bits. The parsing process is to divide the data bits into groups of data with 64-bit size. The process of determining initial hash values and register assignments arranged as shown in Table 2.

Table 2. JPEG/exif segment marker

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

The eight registers (a, b, c, d, e, f, g, h) and blocks of input data are used in computing the hash value according to the message expanding block sequence shown in Figure 4.

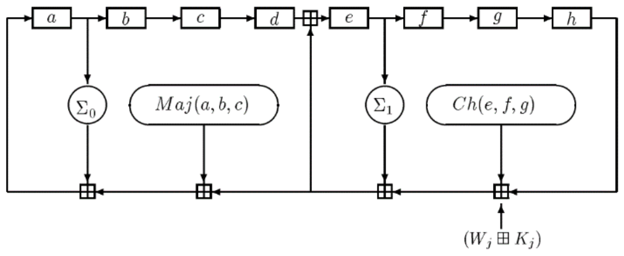


Figure 4. Message expanding block sequence diagram

The hash value computing executed iteratively as much as the block of data bits generated from the parsing stage. The mathematical equation in Figure 4 used according to what is shown in equations 1 to 6.

$$Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z) \quad (1)$$
$$Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \quad (2)$$
$$\sum 0(x) = S^{28}(x) \oplus S^{34}(x) \oplus S^{39}(x) \quad (3)$$
$$\sum 1(x) = S^{14}(x) \oplus S^{18}(x) \oplus S^{41}(x) \quad (4)$$
$$\sigma_0(x) = S^{1}(x) \oplus S^{8}(x) \oplus R^{7}(x) \quad (5)$$
$$\sigma_1(x) = S^{19}(x) \oplus S^{61}(x) \oplus R^{6}(x) \quad (6)$$

The results of the computing process are eight hash values stored in eight registers. Figure 5 shows the results of calculating the hash value for the input "abc" string.

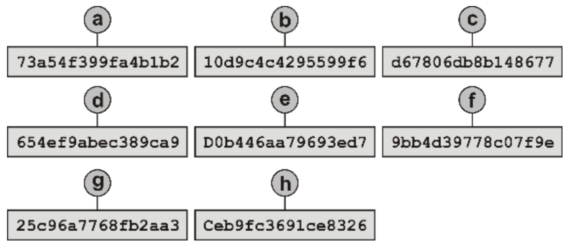
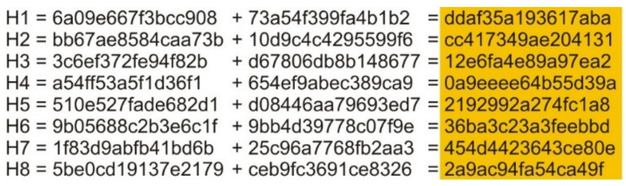


Figure 5. Hash value stored in eight registers

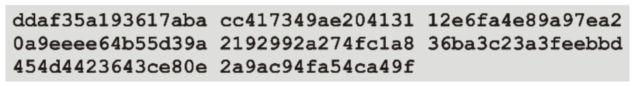
Each hash value in eight registers (a, b, c, d, e, f, g, h) then add with the initial hash value as shown in Figure 6.



```
H1 = 6a09e667f3bcc908  + 73a54f399fa4b1b2  = ddaf35a193617aba
H2 = bb67ae8584caa73b + 10d9c4c4295599f6  = cc417349ae204131
H3 = 3c6ef372fe94f82b  + d67806db8b148677 = 12e6fa4e89a97ea2
H4 = a54ff53a5f1d36f1   + 654ef9abec389ca9  = 0a9eeee64b55d39a
H5 = 510e527fade682d1  + d08446aa79693ed7 = 2192992a274fc1a8
H6 = 9b05688c2b3e6c1f  + 9bb4d39778c07f9e  = 36ba3c23a3feebbd
H7 = 1f83d9abfb41bd6b  + 25c96a7768fb2aa3  = 454d4423643ce80e
H8 = 5be0cd19137e2179 + ceb9fc3691ce8326  = 2a9ac94fa54ca49f
```

Figure 6. registers hash values and initial hash values

The result of the hash value of SHA512 is a combination of eight hash values from Figure 6 composed sequentially as shown in Figure 7.



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

Figure 7. SHA512 result

Research's third stage is modification experiments that consist of six types of file modification. This stage consists of three main steps as shown in Figure 8.

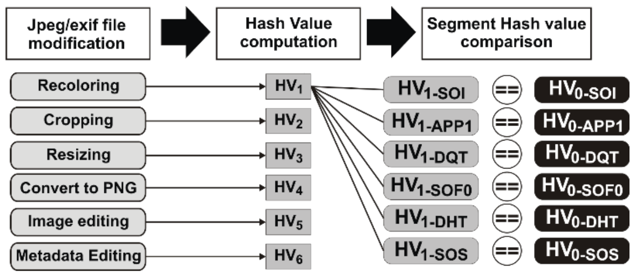


Figure 8. Image file modification experiments

Image recoloring, resizing, cropping conducted with ACDSee Pro.8 application. Metadata manipulation conducted with Hex Editor Neo. Image file format conversion from JPEG to png conducted by using FormatFactory application and image text addition conducted by using Paint application. This third stage has the purpose of identified altered segments caused by image modification. Identification conducted by comparing each segment's hash value from the original image with hash values from a modified image as shown in Figure 8. Before the image file modified, the hash value calculated from the original file (HV_0). The hash value of the original file used in comparison to the hash value of the modified file (HV_1, HV_2, HV_3, HV_4, HV_5, HV_5, and HV6). Each modified form will have the value hash of each segment compiled.

The comparison of hash values conducted for each of the same segments. The hash value that has changed in each form of modification will be used as the compiler of the fingerprint file at a later stage. The last stage is a message digest compiling. Each hash value from the third stage compiles into one string file to form one message digest.

III. RESULTS AND DISCUSSION

Table 3 shows the image file size and picture mode, taken from smartphone Asus Z00UD.

Table 3. Image file from smartphone Asus Z00UD

JPEG/Exif file	Size (KB)	Picture Mode
P_20180723_ 41211	2.117	Auto/Outdoor
P_20180731_ 34049	1.915	Auto/Outdoor
P_20180823_ 24724	2.488	Auto/Indoor
P_20180905_ 85850	1.977	Auto/Outdoor
P_20190110_ 00735	1.482	Auto/Indoor
P_20190324_10003	1.601	Auto/Indoor
P_20190324_10004	1.564	Auto/Indoor
P_20190324_11402	2.315	Auto/Indoor
P_20190324_11530	1.538	Auto/Outdoor
P_20190324_12100	1.703	Auto/Outdoor

The images data shown in Table 3, taken from two categories place, indoor and outdoor. These two locations did not affect the file size. The biggest file from Table 3 shows that images with multiple colors will make a file size bigger. Table 4 shows image files from the smartphone Samsung Galaxy A5.

Table 4. Image file from smartphone Samsung Galaxy A5

JPEG/Exif file	Size (KB)	Picture Mode
01_20180825_131753	4.968	Auto/Outdoor
02_20171225_111236	3.346	Auto/Outdoor
03_20171201_124906	3.075	Auto/Outdoor
04_20171201_130420	3.304	Auto/Outdoor
05_20180825_134942	2.203	AutoOutdoor
06_20181213_172235	1.208	Auto/Indoor
07_20190114_154205	2.422	Auto/Indoor
08_20190114_154209	2.101	Auto/Indoor
09_20190114_154215	2.065	Auto/indoor
10_20190114_154220	2.151	Auto/indoor

The image files from the smartphone Samsung Galaxy A5 have different characteristics than image files from smartphone Asus Z00UD. The file size from Table 4 influenced by the amount of light and color captured in the image. The result of JPEG / exif file segment identification shown in Tables 3 and 4. Table 3 shown the segment location index for an image file from the Asus Z00UD smartphone.

Table 3. Segment index location for an image file from smartphone Asus Z00UD

JPEG/Exif file	Segment Index					
	SOI	APP1	DQT	SOF0	DHT	SOS
P_20180723_ 41211	0	4	26400	26844	26726	27630
P_20180731_ 34049	0	4	26368	61656	26694	27598
P_20180823_ 24724	0	4	26378	26664	26704	27610
P_20180905_ 85850	0	4	25350	25636	25676	26580
P_20190110_ 00735	0	4	26318	26602	26642	27548

JPEG/Exif file	Segment Index					
	SOI	APP1	DQT	SOF0	DHT	SOS
P_20190324_10003	0	4	34212	25544	25584	26490
P_20190324_10004	0	4	25378	25662	25704	26608
P_20190324_11402	0	4	25278	25769	25809	26713
P_20190324_11530	0	4	25348	25789	25792	26713
P_20190324_12100	0	4	25405	25663	25564	26580

Segments index value obtained from variable idxstr that shown in Figure 2. Variable idxstr that uses is the last value before the looping process stopped. The location index of the SOI and APP1 segment in all JPEG / exif files in Table 3 have the same values 0 and 4. This is because the SOI segment is located in the initial bit of the image file and consists of only four bits containing the segment marker segment, ffd8. The location index and length of the segments DQT, SOF0, DHT and SOS in ten JPEG / exif files have different values. This makes the index location of a segment of a JPEG / exif file not be used to identify the location of the segment in another file. Therefore, the Boyer-Moore algorithm matching string is always used to identify the location of the segment for each time the message digest compiled. The identification of the location of the JPEG / exif file segment from the Samsung Galaxy A5 smartphone shown in Table 4.

Table 4. Segment index location for an image file from smartphone Samsung Galaxy A5

JPEG/Exif file	Segment Index					
	SOI	APP1	DQT	SOF0	DHT	SOS
01_20180825_131753	0	4	2000	2286	2326	3218
02_20171225_111236	0	4	2000	2286	2326	3218
03_20171201_124906	0	4	2000	2286	2326	3218
04_20171201_130420	0	4	2000	2286	2326	3218
05_20180825_134942	0	4	2000	2286	2326	3218
06_20181213_172235	0	4	2000	2286	2326	3218
07_20190114_154205	0	4	2000	2286	2326	3218
08_20190114_154209	0	4	2000	2286	2326	3218
09_20190114_154215	0	4	2000	2286	2326	3218
10_20190114_154220	0	4	2000	2286	2326	3218

The location index of the SOI and APP1 segment in all JPEG / exif files in Table 4 has the same values 0 and 4. This is because the SOI segment is located in the initial bit of the image file and the segment location index on the JPEG/exif file from the Samsung Galaxy A5 smartphone shows that each file has the same segment location. Therefore, identification of the location of the JPEG/exif segment that will be conducted in the future does not require a search from the start.

The location index of Tables 3 and 4 used to calculate the length of each segment. This calculation accomplished by finding the difference in location index values from two adjacent segments as formulated in equation 7.

$$P_{segmen(n)} \quad = Index(n) - Index(m) \qquad (7)$$

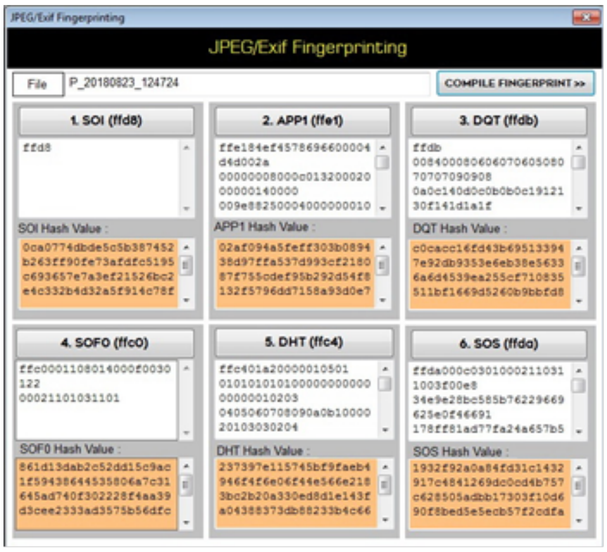
An example of the calculation of segment length for JPEG / exif documents from the Asus Z00UD smartphone shown in Table 5.

Table 5. Image file Segment length

File name	Segment Length (bit)					
	SOI	APP1	DQT	SOF0	DHT	SOS
P_20180723_141211	4	26396	444	118	904	9181254
P_20180823_124724	4	26374	286	40	906	5386020
P_20180905_085850	4	25346	286	40	904	5319634
P_20190324_100013	4	34208	8668	40	906	3197654
P_20190324_100202	4	63584	38180	40	906	5009834
01_20180825_131753	4	1996	286	40	892	10805918
02_20171225_111236	4	1996	286	40	892	7275684
03_20171201_124906	4	1996	286	40	892	6686880
04_20171201_130420	4	1996	286	40	892	6598378
05_20180825_134942	4	1996	286	40	892	4788372

The APP1 segment length in the first file in Table 3 has a value of 26396 obtained using equation 7, which is the result of a reduction between the DQT segment location index and the APP segment1. The results of the long calculation of all segments show that the SOS segment has the largest segment length. This is because the SOS segment contains image data that is the main data from the Image file. The bigger the size of the JPEG / exif document, the longer the SOS segment will be. The calculation of SOS segment length accomplished by operating a subtraction between the overall image bit length and the SOS segment location index. The bit length of image data, obtained from the length calculation of the converted file at the beginning of the identification phase of the segment location that is also stored in the n variable in the searching process with the Boyer-Moore string matching algorithm.

Figure 9 shows the application interface for content acquisition and calculation of hash values for each segment. Segment content acquired use location index value and segment length as boundary parameters. Location index to identify an acquisition starting point and segment length for numbers of characters that should acquire

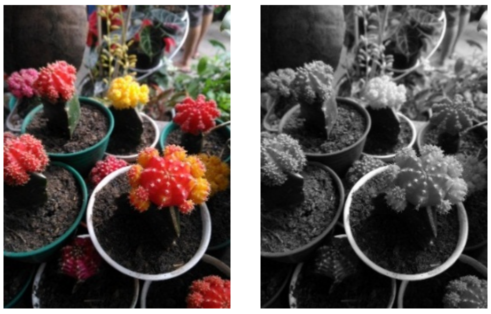


```
P_20180823_124724

1. SOI (ffd8)
ffd8
SOI Hash Value :
0ca0774dbde5c5b387452
b263ff90fe73afdfc5195
c693657e7a3ef21526bc2
e4c332b4d32a5f914c78f

2. APP1 (ffe1)
ffe184ef4578696600004
d4d002a
00000008000c013200020
00000140000
009e88250004000000010
APP1 Hash Value :
02af094a5feff303b0894
38d97ffa537d993cf2180
87f755cdef95b292d54f8
132f5796dd7158a93d0e7

3. DQT (ffdb)
ffdb
00840008060607060508O
70707090908
0a0c140d0c0b0b0c19121
30f141d1a1f
DQT Hash Value :
c0cacc16fd43b69513394
7e92db9353e6eb38e5633
6a6d4539ea255cf710835
511bf1669d5260b9bbfd8

4. SOFO (ffc0)
ffc0001108014000f0030
122
00021101031101
SOF0 Hash Value :
861d13dab2c52dd15c9ac
1f59438644535806a7c31
645ad740f302228f4aa39
d3cee2333ad3575b56dfc

5. DHT (ffc4)
ffc401a20000010501
0101010101000000000000
00000010203
0405060708090a0b10000
20103030204
DHT Hash Value :
237397e115745bf9faeb4
946f4f6e06f44e566e218
3bc2b20a330ed8d1e143f
a04388373db88233b4c66

6. SOS (ffda)
ffda000c0301000211031
1003f00e8
34e9e28bc585b76229669
625e0f46691
178ff81ad77fa24a657b5
SOS Hash Value :
1932f92a0a84fd31c1432
917c4841269dc0cd4b757
c628505adbb17303f10d6
90f8bed5e5ecb57f2cdfa
```

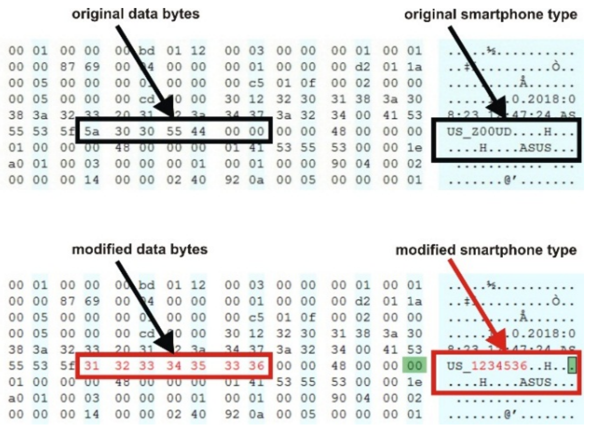
Figure 9. segment content acquisition and hashing

Six hash values from six JPEG/segments form figure 9 can use as message digest elements that will conduct on research fourth stage.

The research's fourth stage begins with modifying the image file. The modification process consists of six type experiments. The first modification is recoloring image, change the image file into a grayscale image as shown in Figure 10.


(a) (b)
Figure 10. Example of recoloring image
(a).Original image (b).Modified image

Recoloring image conducted shown from Figure 10, uses ACDSee Pro.8 application. The Second modification is metadata editing. This experiment conducted using the Neo Hex editor application to change smartphone type inside the APP1 segment. Figure 11 shows a metadata comparison between the original and modified image.



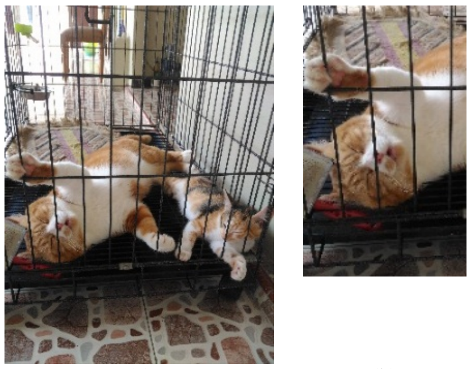
```
00 01 00 00  00 bd 01 12  00 03 00 00  00 01 00 01   .....½..........
00 00 87 69  00 04 00 00  00 01 00 00  00 d2 01 1a   ..‡i........Ò..
00 05 00 00  00 01 00 00  00 c5 01 0f  00 02 00 00   .........Å......
00 05 00 00  00 cd 00 00  30 12 32 30  31 38 3a 30   .......0.2018:0
38 3a 32 33  20 31 32 3a  34 37 3a 32  34 00 41 53   8:23 12:47:24 AS
55 53 5f 5a  30 30 55 44  00 00 00 00  48 00 00 00   US_Z00UD....H...
01 00 00 00  48 00 00 00  01 41 53 55  53 00 00 1e   ....H....ASUS...
a0 01 00 03  00 00 00 01  00 01 00 00  90 04 00 02   ................
00 00 00 14  00 00 02 40  92 0a 00 05  00 00 00 01   .......@'.......
```

```
00 01 00 00  00 bd 01 12  00 03 00 00  00 01 00 01   .....½..........
00 00 87 69  00 04 00 00  00 01 00 00  00 d2 01 1a   ..‡i........Ò..
00 05 00 00  00 01 00 00  00 c5 01 0f  00 02 00 00   .........Å......
00 05 00 00  00 cd 00 00  30 12 32 30  31 38 3a 30   .......0.2018:0
38 3a 32 33  20 31 32 3a  34 37 3a 32  34 00 41 53   8:23 12:47:24 AS
55 53 5f 31  32 33 34 35  33 36 00 00  48 00 00 00   US_1234536..H...
01 00 00 00  48 00 00 00  01 41 53 55  53 00 00 1e   ....H....ASUS...
a0 01 00 03  00 00 00 01  00 01 00 00  90 04 00 02   ................
00 00 00 14  00 00 02 40  92 0a 00 05  00 00 00 01   .......@'.......
```

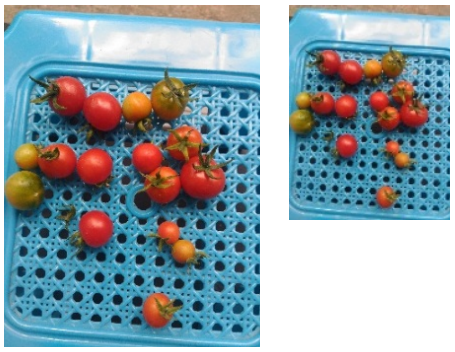
Figure 11. Example of metadata editing

Metadata editing from Figure 11, modified smartphone type from "Z00UD" into "123456". The modification conducted by change the original data bytes 5a 30 30 55 44 00 into 31 32 33 34 35 36 in hexadecimal number format. The result from this modification is the metadata section has altered while the image still can be displayed.

The third experiment selects a part of an image using the ACDSee Pro.8 application. Figure 12 shows a comparison between the original image and the cropped image.


(a) (b)
Figure 12. Example of cropping image
(a).Original image (b).Cropped image

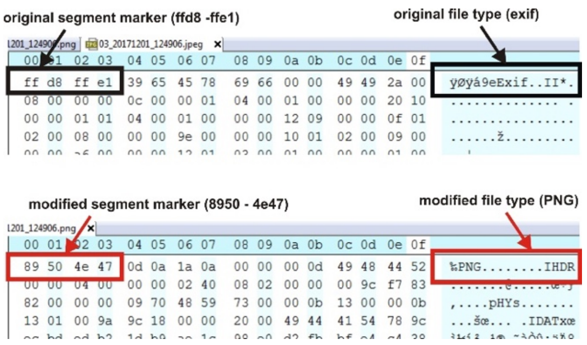
The fourth modification experiment is resizing. Figure 13 shows a comparison between the original and modified image.


(a) (b)
Figure 13. Example of resizing image

(a).Original image (b).Modified image

Image dimension (width-height) on the fourth modification experiment, reduced into 50% smaller from the original image. This modification conducted uses ACDSee Pro.8 application.

The fifth modification is file type conversion. The original file type is JPEG/exif, converted into png uses Format Factory application. The conversion result did not show the significance result from the displayed image. The differences clearly are shown if the converted image opened in the hex editor application as shown in Figure 14.


original segment marker (ffd8 -ffe1)
original file type (exif)
modified segment marker (8950 - 4e47)
modified file type (PNG)

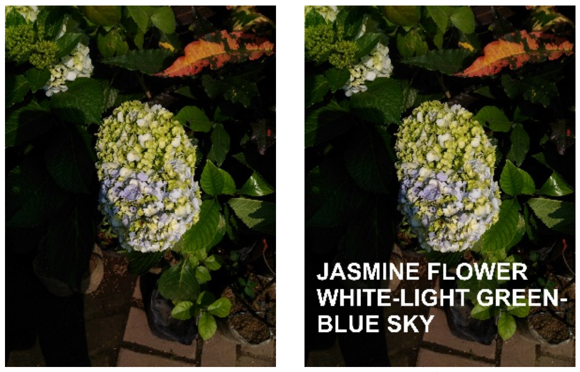
```
1201_124906.png | 03_20171201_124906.jpeg ×
00 01 02 03  04 05 06 07  08 09 0a 0b  0c 0d 0e 0f
ff d8 ff e1  39 65 45 78  69 66 00 00  49 49 2a 00   ÿØÿá9eExif..II*.
08 00 00 00  0c 00 00 01  04 00 01 00  00 00 20 10   ...............
00 00 01 01  04 00 01 00  00 00 12 09  00 00 0f 01   ................
02 00 08 00  00 00 9e 00  00 00 10 01  02 00 09 00   ......ž.........
```

```
1201_124906.png ×
00 01 02 03  04 05 06 07  08 09 0a 0b  0c 0d 0e 0f
89 50 4e 47  0d 0a 1a 0a  00 00 00 0d  49 48 44 52   ‰PNG........IHDR
00 00 04 00  00 00 02 40  08 02 00 00  00 9c f7 83   .......@.....œ÷ƒ
82 00 00 00  09 70 48 59  73 00 00 0b  13 00 00 0b   ‚....pHYs.......
13 01 00 9a  9c 18 00 00  20 00 49 44  41 54 78 9c   ...šœ... .IDATxœ
```

Figure 14. Example of a comparison image's data bytes from the converted file

The file type from the two files the different values. The original file has exif type while the modified file has png file type. The change of file type made identified by searching the file signature from both files. The file signature used by the computer to determine how to open a file. The file signature located at the beginning of the data file. The original segment marker on JPEG/exif file type, have file signatures that identified as ffd8. File type conversion on this modification experiments, not just affected the change of file signature only, but also altered the entire data bytes of the image file. This condition caused by the changed of the file structure.

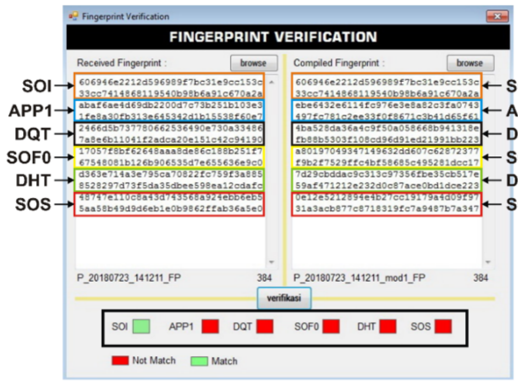
The sixth image modification is text addition. This experiment conducted by uses Core Photo-Paint X3. Figure 15 shows a comparison between the original image and the modified image. The experiment conducted by add text into the image.



(a) (b)

Figure 15. Example of text modification on the image file (a).Original image (b).modified image

Each original and modified image processed with SHA512 to produce hash values from each segment inside each image file. The obtained hash value then compared between the original and modified images. Figure 16 shows hash values comparison from recoloring image modification.



```
Received Fingerprint :                           Compiled Fingerprint :
SOI   606946e2212d596989f7bc31e9cc153c           606946e2212d596989f7bc31e9cc153c   SOI
      33cc7414868119540b98b6a91c670a2a           33cc7414868119540b98b6a91c670a2a
APP1  abaf6ae4d69db2200d7c73b251b103e3           ebe6432e6114fc976e3e8a82c3fa0743   APP1
      1fe8a30fb313e645342d1b15538f60e7           497fc781c2ee33f0f8671c3b41d65f61
DQT   2466d5b737780662536490e730a33486           4ba528da36a4c9f50a058668b941318e   DQT
      7a8e6b11041f2adca20e151c42c94190           fb88b5303f108cd96d91ed21991bb223
SOF0  17057f8bf62648aaa8de86c188b251f7           a8019704934714963 2dd607c62872377   SOF0
      67548081b126b906535d7e655636e9c0           f9b2f7529ffc4bf58685c495281dcc17
DHT   d363e714a3e795ca70822fc759f3a885           7d29cbddac9c313c97356fbe35cb517e   DHT
      8528297d73f5da35dbee598ea12cdafc           59af471212e232d0c87ace0bd1dce223
SOS   48747e110c8a43d743568a924ebb6eb5           0e12e5212894e4b27cc19179a4d09f97   SOS
      5aa58b49d9d6eb1e0b9862ffab36a5e0           31a3acb877c8718319fc7a9487b7a347
P_20180723_141211_FP                384          P_20180723_141211_mod1_FP          384
SOI 🟩  APP1 🟥  DQT 🟥  SOF0 🟥  DHT 🟥  SOS 🟥
🟥 Not Match   🟩 Match
```

Figure 16. Application interface for hash values comparison

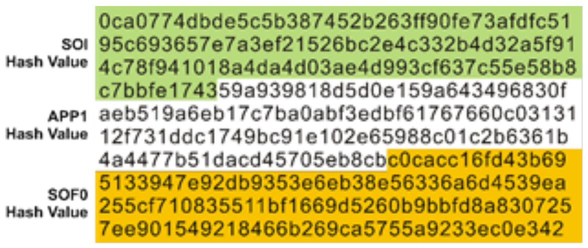
The left section of Figure 16 shows the hash value of six segments from the original file. The right section shows the hash value of five segments from the recoloring modification file. The comparison conducted by comparing two hash values from the same segment. The comparison results consist of two conditions, "Match" if the hash values of the two segments are equal and "Not match" if not the same. The comparison result from Figure 16 shows, only the value of the SOI segment hash value that altered. These results show that four other segments' content altered when recoloring modifications occurred. The affected segments from each modified experiments shown in Table 6. The result from Table 6 categorized into three groups. The first group has shown that the metadata modification experiment affected the APP1 segment only. The second group showed that the SOI segment altered for image file conversion and text/object addition experiments.

Table 6. Affected segment for image modification

Modification Experiments	Affected Segment					
	SOI	APP1	DQT	SOF0	DHT	SOS
Recoloring	-	✓	✓	✓	✓	✓
Metadata Modification	-	✓	-	-	-	-
Resizing	-	✓	✓	✓	✓	✓
Convert to PNG	✓	✓	✓	✓	✓	✓
Text addition	✓	✓	✓	✓	✓	✓
Cropping	-	✓	✓	✓	✓	✓

The third group shown all segments altered for image display modification (recoloring, resizing, cropping). Each result group used as a base to determine the file fingerprint components as JPEG/exif message digest. The first component is SOI hash value from the first group and the second component is APP1 hash value from the second group. Third component selected from four segments (DQT, SOF0, DHT, SOS) based on segment content and segment length.

SOS segment has the most length content from other segments because this segment stored the image data. This condition made the segment marker searching process need more time because the Boyer-Moore string matching algorithm has O(mn) time complexity for worst-case conditions. SOF0 has the smallest size than others except for the SOI segment. SOF0 store image information such as image dimension and the number of color components that always change if image altered as in 1st, 2nd, 5th and 6th experiments. Figure 17 shown JPEG/exif file message digest that arranged from three hash values, SOI, APP1, and SOF0.


SOI
Hash Value
APP1
Hash Value
SOF0
Hash Value

```
0ca0774dbde5c5b387452b263ff90fe73afdfc51
95c693657e7a3ef21526bc2e4c332b4d32a5f91
4c78f941018a4da4d03ae4d993cf637c55e58b8
c7bbfe17459a939818d5d0e159a643496830f
aeb519a6eb17c7ba0abf3edbf61767660c03131
12f731ddc1749bc91e102e65988c01c2b6361b
4a4477b51dacd45705eb8cbc0cacc16fd43b69
5133947e92db9353e6eb38e56336a6d4539ea
255cf710835511bf1669d5260b9bbfd8a830725
7ee901549218466b269ca5755a9233ec0e342
```

Figure 17 JPEG/exif message digest

Hash value from each segment in Figure 17, has 128-bit length. This caused by number format from an application is in hexadecimal which in binary can convert to 512-bit length by computing 128 x 4.

IV. CONCLUSION

JPEG/exif message-digest consist of three hash values that represent modified experiments. SOI hash values use for identifying file conversion and text addition modification. APP1 hash values use for identifying metadata editing modification and SOF0 hash value for identifying recoloring, resizing and cropping modification. Identifying that three-segment will make segments searching faster for the Boyer-Moore string matching algorithm. The compilation of Message digest from SHA512 hash value has an advantage because of its small size and cannot decrypt. Future research hope can develop message digest compilation for other file types that common use in digital communication such as video and audio.

V. REFERENCES

[1] 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.

[2] D. P. Gangwar and A. Pathania, "Authentication of Digital Image using Exif Metadata and Decoding Properties," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 3, no. January, pp. 335–341, 2019.

[3] L. Yuan and T. Ebrahimi, "Image Privacy Protection with Secure JPEG ransmorphing," *Inst. Eng. Technol. Journals*, vol. 11, no. 9, pp. 1–8, 2017.

[4] 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.

[5] A. Chauhan, "Digital Watermarking-Revisit," *Int. J. Comput. Sci. Inf. Technol.*, vol. 6, no. 1, pp. 833–838, 2015.

[6] D. Vaishnavi and T. S. Subashini, "A Secure and Robust Image Watermarking System Using Normalization and Arnold Scrambling," *Int. J. Netw. Secur.*, vol. 18, no. 5, pp. 832–841, 2017.

[7] C. B. Shah and D. R. Panchal, "Secured Hash Algorithm-1: Review Paper," *Int. J. Adv. Res. Eng. Technol.*, vol. 2, no. 10, pp. 26–30, 2014.

[8] N. Tiwari and A. Sinhal, "An Implementation on Secure Hash Algorithm in Wireless Algorithms to Ensure the Integrity," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. January, pp. 4779–4781, 2014.

[9] N. C. Iyer and S. Mandal, "Implementation of Secure Hash Algorithm-1 using FPGA," *Int. J. Inf. Comput. Technol.*, vol. 3, no. 8, pp. 757–764, 2013.

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

[11] 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.

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

[13] 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.

[14] 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.

[15] R. Ramos-frí and M. Vargas-lombardo, "A Review of String Matching Algorithms and Recent Implementations using GPU," *Int. J. Secur. Its Appl.*, vol. 11, no. 6, pp. 69–76, 2017.

[16] J. Bhandari and A. Kumar, "String Matching Rules Used By Variants of Boyer-Moore Algorithm," *J. Glob. Res. Comput. Sci.*, vol. 5, no. 1, pp. 8–11, 2014.

[17] Y. D. Prabowo, "Pencocokan DNA NR_108049 dan DNA DI203322 Menggunakan Algoritma Boyer Moore," *Pros. Semin. Nas. Teknol. Inf. dan Komun.*, no. 40, pp. 18–19, 2016.

[18] D. R. Candra and K. D. Tania, "Application of Knowledge Sharing Features Using the algorithm Boyer-moore On Knowledge Management System (KMS)," *J. Sist. Inf.*, vol. 9, no. 1, pp. 1216–1221, 2017.

[19] 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.

[20] N. Jiji and T. Mahalaksmi, "An Efficient String Matching Algorithm for Detecting Pattern Using Forward and Backward Searching Approach," *Int. J. Comput. Sci.*, vol. 6, no. 2, pp. 16–26, 2018.