Boyer-Moore Algorithm in retrieving deleted Short Message Service in Android Platform

R F Rahmat, D F Prayoga, D Gunawan and O S Sitompul

Department of Information Technology, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan, Indonesia

Email: romi.fadillah@usu.ac.id, dian.fajar.prayoga@students.usu.ac.id, danigunawan@usu.ac.id, opim@usu.ac.id.

Abstract. Short message service (SMS) can be used as digital evidence of disclosure of crime because it can strengthen the charges against the offenders. Criminals use various ways to destroy the evidence, including by deleting SMS. On the Android OS, SMS is stored in a SQLite database file. Deletion of SMS data is not followed by bit deletion in memory so that it is possible to rediscover the deleted SMS. Based on this case, the mobile forensic needs to be done to rediscover the short message service. The proposed method in this study is Boyer-Moore algorithm for searching string matching. An auto finds feature is designed to rediscover the short message service by searching using a particular pattern to rematch a text with the result of the hex value conversion in the database file. The system will redisplay the message for each of a match. From all the testing results, the proposed method has quite a high accuracy in rediscovering the short message service using the used dataset. The search results to rediscover the deleted SMS depend on the possibility of overwriting process and the vacuum procedure on the database file.

1. Introduction

From the beginning of digital cellular telephony era, SMS has become one of the services to do communication. Since then, it is also likely to facilitate the communication of criminal activity. Therefore, SMS has become a significant object in the criminal investigation until now. In Indonesia, the importance of digital evidence has been stated explicitly in Law No. 11 In 2000 Article 5 on Information and Electronic Transaction. It shows that digital evidence such as SMS is crucial for a criminal investigation [1]. There are four phases of digital evidence investigation, namely digital evidence identification, digital evidence storage, digital evidence analysis, and digital evidence presentation. Each of them has their function to reveal the data from digital evidence [2]. Several examples of digital evidence feature and services in the mobile phone include Subscriber Identity Module (SIM), internal memory, memory card and network service providers [3].

In the Android operating system, SMS will be stored in SQLite Database file, which consists of two parts, namely the table storage and its record. It means one single SMS will be stored in a record in the table storage. In the case of SMS deletion and removal, the termination of the bit in memory does not occur, which means deleted SMS will be removed to empty space in the memory. This process is usually called as vacuum procedure [4]. From this procedure, we can assume that SMS retrieval becomes possible if no overwriting process occurred in memory. If so, then SMS object will be permanently deleted from memory. Thus, we can define that overwriting process is a process of adding new SMS in

the memory space of preceding SMS [5]. The other aspect that can determine the existence of deleted SMS is how often the database file executes the vacuum procedure.

To find the deleted SMS, we need to build a system that can recover deleted SMS with string matching in its hexadecimal bytes. The algorithm for string matching requirement is Boyer-Moore Algorithm proven as a good and fast string-matching algorithm [6]. Previous work has been conducted in term of device forensic such as file-based content type identification [7], file undelete [8] and file type identification [9]

2. Research methodology

In figure 1, we describe the general architecture of our proposed method. It starts from connecting Android smartphone with our system. Then several steps of pre-processing are conducted, namely doing image making to get another copy of data from the smartphone, extracting the image file to get database file, which contains SMS file from the Android root directory and generating hex values from database file to get hexadecimal bytes pattern. Next, input generalization is used as hex pattern in the searching process. This specific hex pattern will be used in Boyer-Moore Pre-processing phase to count the shifting values along with generated hex string. The last processes are the searching process using Boyer-Moore Algorithm and filtering process.

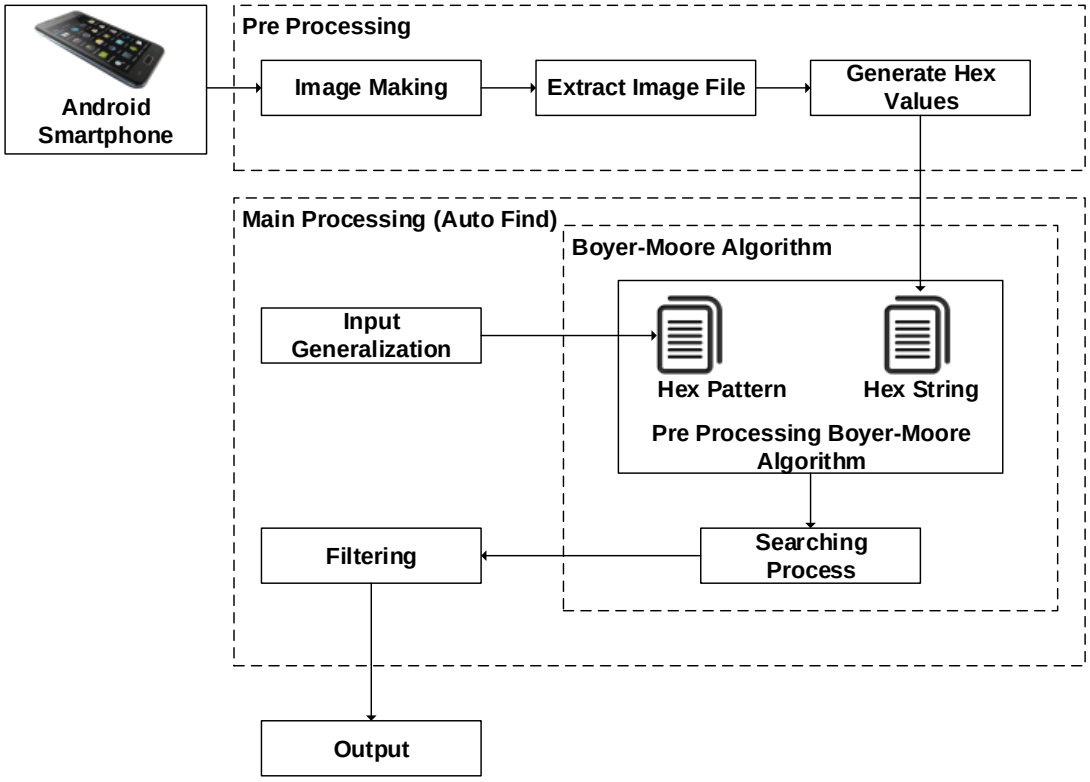

Figure 1. General Architecture

2.1. Dataset

The used dataset in this research is divided into two sets. Both of them are mmssms.db SQLite database file. These files are taken from /data/data/com.android.providers.telephony/databases/ Android directory files. The first dataset is mmssms1.db, mmssms2.db, mmssms3.db that consist of 10-30 SMS in one database, while mmssms4.db, mmssms5.db, and mmssms6.db have 0 SMS because it has been

deleted before. The last set is mmssms7.db, mmssms8.db, mmssms9.db, and mmssms10.db which has unknown SMS existed before and unknown deleted SMS. The dataset properties can be seen in table 1.

Table 1. Dataset properties.

File	SMS	Deleted SMS
mmssms1.db	10	0
mmssms2.db	20	0
mmssms3.db	30	0
mmssms4.db	0	*Unknown*
mmssms5.db	0	*Unknown*
mmssms6.db	0	*Unknown*
mmssms7.db	*Unknown*	*Unknown*
mmssms8.db	*Unknown*	*Unknown*
mmssms9.db	*Unknown*	*Unknown*
mmssms10.db	*Unknown*	*Unknown*

2.2. Image Making and Extracting Image File

In this step, we will copy the data in the Android mobile phone root directory where the SMS database files contained. The purpose of image making is to copy entire SMS file to a safe container and to prevent any damage, modification, or loss of the file.

Next step is to extract image file to get file folder from image making process. In this folder, we can get the system or user's applications data including SMS messages. The output of this process is SMS database.

2.3. Generating Hex Values

After SMS database file is found, the next step in this proposed method is generating hex values to get hexadecimal bytes value from the database file. In this hexadecimal bytes values, we can get deleted SMS. The output of this process is hex string which is saved inside a .txt file. In this hexadecimal byte representation, SMS will have SMS file pattern, containing a header, sms_number, sms_date, and sms_body. The structure of SMS file pattern can be shown in figure 2.

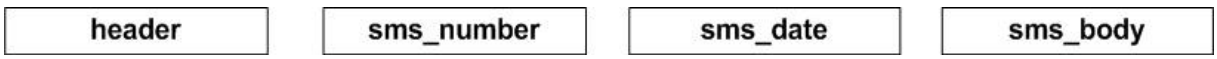

Figure 2. SMS File Pattern

2.4. Input Generalization

In this step, we generalize input string that will be searched in hex string. This input string will be used as a hex pattern in the searching process. Generalization is aimed to get the desired output. In the implementation, our proposed system uses hexadecimal the SMS header pattern (see Figure 2). 08 08 08 bytes from this header will be used as hex pattern for the next process.

2.5. Auto Find

Auto finds process is designed to retrieve deleted SMS file automatically. In the implementation, Boyer-Moore algorithm is used to support searching process. This feature assigned to search hex pattern 08 08 08. The result from this feature will give a whole content of SMS that has been found including its supporting attributes inside hexadecimal bytes.

2.5.1. Boyer-Moore Pre-processing Algorithm. Generated hex values create hex string, while input generalization creates hex pattern, these two values are used to calculate next shifting value based on bad character shift and good suffix shift functions. Thus, those two-shifting value is stored in two different table BmBc and BmGs. Shifting value of BmBc and BmGs can be described in Table 2 and Table 3 respectively.

Table 2. Shifting value BmBc in auto find feature

char	BmBc[char]
0	1
8	2
0	1
8	2
0	1
8	2

Table 3. Shifting value BmGs in auto find feature

i	Char	BmGs[char]
0	0	2
1	8	2
2	0	4
3	8	4
4	0	6
5	8	1

2.5.2. Searching Process. Our proposed system will do pattern searching using two shifting tables bad-character shift and good-suffix shift, wherein every unmatched character, the system will choose the biggest summation of shifting value for the next shifting. The pseudo code of Boyer-Moore Algorithm in the searching process can be shown below:

```
procedure BM(input pattern, text: array of char)
Variable Declaration
       i, j: integer
       bmGs : array [0..XSIZE] of integer
       bmBc : array [0..ASIZE] of integer
Algorithms
       /* Preprocessing */
       preBmGs(pattern, BmGs)
       preBmBc(pattern, BmBc)

       /* Searching */
       i = 0
       while (i <= text.Length - pattern.Length);
       for (j = pattern.Length - 1; j >= 0 and pattern[j]=text[i+j]; --i)
       if (j < 0)
              arr.Add(i);
              i ← i + BmGs[0];
       else
              i ← i + MAX(BmGs[j],BmBc[text[i+j]]- pattern.Length+1+j)
```

As has been described before, hexadecimal bytes from generated hex values is transformed into hex string, then the searching of hex pattern from input generalization will become a string matching process. When the criteria matches, the system will take some bytes from the file structure that contains SMS number pattern, SMS date, and SMS body. This method refers to the fact that undamaged SMS has similar pattern in their representation of hexadecimal bytes. The pattern for this auto finds method is shown in figure 3.

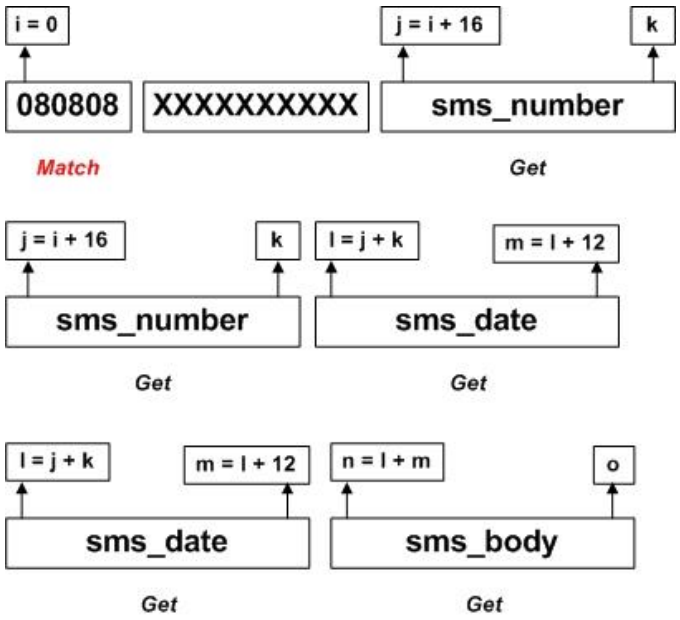

Figure 3. Auto Find Pattern Features

2.6. Filtering Process

In auto find step, there is possibility that system will display the message repeatedly. Therefore, an additional process is required to overcome it. Filtering is aimed to reduce the duplication of the data that frequently occurs in auto find. We use date of message as a comparator variable for implementing the filtering process. When the date of message is compatible, SMS will be stored in the system then it will be matched again. If the date of message cannot be found later, SMS will be displayed. Otherwise, it will not be displayed if the date of message is found. Figure 4 shows the filtering process.

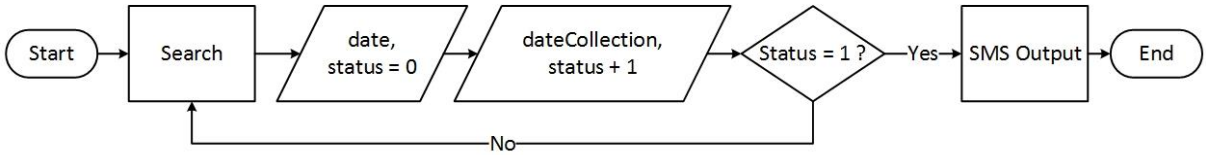

Figure 4. Filtering Process

2.7. Target Output

Target output in our proposed method is that system can summarize and retrieve the deleted SMS and identify if it is still intact or removed. The system will display the number, date, content and the status of SMS whether it is still integrated or deleted. For the message that can be found in the system and is still stored in the database file, then it can be identified that it is still integrated, otherwise, if it cannot be found in the database file, it has been deleted. The date of message will be used as epoch time and will be matched again with the database file for receiving the status of SMS. Table 4 below shows the proposed target output.

Table 4. Target output

Position	Number	Date	Body	Status
111222	+6281311112222	1 May 2015, 17:32:15	reference item (partial)	Exist
222333	+6282300005555	abstract body	abstract heading (also in Bold)	Delete

333444	+628112012012		level-2 heading, level-3 heading, author affiliation	Exist
…	…	…	…	…

3. Result and discussion

3.1. Generating hex values

The generated hex values will be stored in .txt file before being used in auto find feature. Figure 5 shows the generated hex values, and table 5 describes Byte Frequency Distribution (BFD) from mmssms7.db.

Table 5. BFD Table mmssms7.db dataset.

Byte (hexadecimal)	Byte (decimal)	Frequency
00	0	24980
01	1	1795
02	2	723
03	3	482
04	4	237
...	...	...
FD	253	0
FE	254	1
FF	255	31

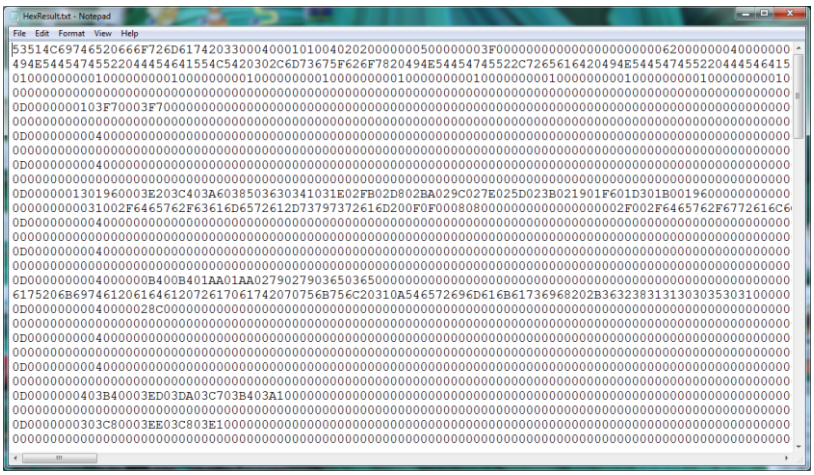

Figure 5. Result of Generating Hex Values

3.2. Auto Find

In this feature, the system will do searching by using hex pattern and hex string in each dataset that contains the generated hex values. The system will display the message when hex pattern and hex string matches. Several parameters that affect the searching process, include:

- Length of pattern used by default in auto finds feature is header byte of SMS data that consists of 6 bytes of hexadecimal, 080808.
- Length of character counted based on the result of generated hex values in each file of SMS database.
- The number of matching pattern and text in generated hex values counted for each file of SMS database.

- The number of call number found in generated hex values, where the default number is 24 bytes of hexadecimal or 12 ASCII characters.
- The number of the integrated message found and still stored in database file.
- The number of deleted message found and not stored in database file.
- Unreadable message caused by vacuum procedure or not integrated.

The sample result of auto finds feature using mmssms7.db dataset is shown in figure 9 and table 6. It shows that there are 29 match messages (23 integrate messages and 6 deleted messages).

Table 6. Result of auto find

Parameter	Value
Length of pattern	6
Length of character	151552
Match message	29
Number of call number as SMS sender	3
Number of integrating message	23
Number of deleted message	6

3.3. Testing Result

Table 7, table 8 and table 9 show the testing result using testing dataset and training dataset. The obtained accuracy for finding integrate message is 100% as shown in Table 4. While for the deleted message in training dataset, the system only can find 26 deleted SMS (around 43.33%), and in testing dataset, the system can discover 87 integrate message and 20 deleted message. The finding of deleted message depends on several processes before, namely overwriting or vacuum procedure in the database file.

Table 7. Testing result using training dataset for integrate file.

File	Integrate Messages	Found Messages	Accuracy
mmssms1.db	10	10	100%
mmssms2.db	20	20	100%
mmssms3.db	30	30	100%
Total	60	60	100%

Table 8. Testing result using training dataset for deleted file.

File	Integrate Messages	Deleted Messages	Total
mmssms4.db	0	9	9
mmssms5.db	0	9	9
mmssms6.db	0	8	8
Total	0	26	26

Table 9. Testing result using testing dataset.

File	Integrate Messages	Deleted SMS	Total
mmssms7.db	22	6	28
mmssms8.db	20	4	24
mmssms9.db	31	5	36
mmssms10.db	14	5	19
Total	87	20	107

4. Conclusion

From all testing results, there are several obtained conclusion for retrieving the deleted SMS in Android mobile phone. First, our proposed method can be used for retrieving the deleted SMS in Android mobile phone with 100% integrate message and 43.33% deleted message in the training dataset. In testing dataset, system can find 87 integrate message and 20 deleted message. The result of finding depends on the entire contents of database file. If the database file has overwritten and vacuum procedure, then it will be difficult to find because the database file has the different pattern. However, our proposed method cannot be used to retrieve message permanently deleted by overwriting and limited to specific digital

device and platform. Another additional approach is required to identify the message in other device and platform.

For further research, we suggest considering record recover method to retrieve SMS data deleted permanently. To get higher accuracy, the data searching should use a different pattern, and the data should be more and more various. Another research possibly done is retrieving data in SQLite database using the same method.

5. References

[1] Carrier B 2003 Defining Digital Forensic Examination and Analysis Tools Using Abstraction Layers *Int. J. Digital Evidence* **1** 1 – 12

[2] Casey E 2011 *Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet* (Waltham: Elsevier)

[3] Zareen A and Baig S 2010 Mobile Phone Forensics: Challenges, Analysis and Tools Classification *Proc. of the 2010 Int. Workshop on Systematic Approaches to Digital Forensic Engineering* pp 47-55

[4] Stahlberg P, Miklau G and Levine B N 2007 Threats to Privacy in the Forensic Analysis of Database Systems *Proc. of the 2007 ACM SIGMOD Int. Conf. on Management of Data* pp 91 - 102

[5] Hoog A 2011 *Android Forensic: Investigation, Analysis, and Mobile Security for Google Android* (Waltham: Elsevier)

[6] Boyer R S and Moore J S 1977 A Fast String Searching Algorithm *Communications of the ACM* **20** 762 – 772

[7] Aaron, Sitompul O S and Rahmat R F 2014 Distributed autonomous Neuro-Gen Learning Engine for content-based document file type identification *International Conference on Cyber and IT Service Management* pp. 63–68.

[8] Sitompul O S, Handoko A, and Rahmat R F 2016 A file undelete with Aho-Corasick algorithm in file recovery *International Conference on Informatics and Computing (lCIC)* pp. 427–431.

[9] Rahmat R F, Nicholas F, Purnamawati S and Sitompul O S 2017 File type identification of file fragments using longest common subsequence (LCS) *Journal of Physics: Conference Series* **801** p. 12054.