Combination of advanced encryption standard 256 bits with md5 to secure documents on android smartphone

Hendra Pasaribu, Delima Sitanggang, Rudolfo Rizki Damanik, Alex Chandra Rudianto Sitompul
Faculty of Technology and Computer Science, Universitas Prima Indonesia, Indonesia

E-mail: *kevinbadboyz@gmail.com

Abstract. File transfer by using a smartphone has some security issues like data theft by irresponsible parties. To improve the quality of data security systems on smartphones, in this research the integration of AES 256 bit algorithm by using MD5 hashing is proposed. The use of MD5 aims to increase the key strength of the encryption and decryption process of document files. The test results show that the proposed method can increase the key strength of the encryption and decryption process in the document file. Encryption and decryption time by using AES and MD5 combination is faster than using AES only on *.txt file type and reverse results for *.docx, *.xlsx, *.pptx and *.pdf file files.

1. Introduction
Smartphone is a communication tool that used by human every day. As the increasing use of smartphones, tablets, and computers, information security becomes a very important issue. Information theft by strangers aims to read and change information when the transmission of information occurs. Cryptography is a technique for protecting information from unauthorized persons. Cryptography has always been an important task, where the main purpose of cryptography is data security [1]. Cryptography uses fairly complex mathematical calculations in the process of encryption and decryption. Cryptography can be used to secure document files. Cryptography is widely used on applications such as chatting, communication system, mobile devices, and various other embedded applications. To secure communication in chatting, it can be overcome by using cryptography technique end to end encryption [2]. For embedded applications, it can be applied on employee monitoring system. This monitoring system can track information from an employee. This application can provide data protection from the tappers who can steal employee information without their knowledge by using AES cryptography techniques [3]. Cryptographic usage is also available on mobile devices such as encryption and decryption applications using AES algorithms on Android phones [2-4].

There are several cryptographic techniques such as DES, 3DES, Blowfish, RSA and others. DES has a short key length. 3DES uses three keys and key sizes longer than DES, but the process is slower than other cryptographic algorithms. AES is not only suitable for securing data but also faster in terms of encryption and decryption processes. AES can be implemented on various platforms from smartphones [4]. A message digest is a cryptographic hash function that contains a series of digits created with a one-way hashing formula. Message digest is designed to protect the integrity of a piece of data or media to detect changes in the message section [5]. Based on the above description, AES
cryptography techniques can be used to secure document files from attacks or bugs. Applications designed by researchers also use a combination of AES cryptography techniques and hashing MD5. Addition of MD5 hashing is used when the key is inputted by the user then the key will go through MD5 hashing process. This applies during encryption and decryption processes. The purpose of adding MD5 hashing is to strengthen cryptographic techniques of an attack and the application is implemented on Android-based smartphones.

2. Method
Cryptography is used on digital communications system. It is used to secure data. Based on the keys used in encryption and decryption, cryptographic keys use two key types: symmetric keys and asymmetric keys. A symmetric key is a key used for both encryption and decryption. While asymmetric keys are keys used for encryption and decryption are not the same, where the sender uses public key and receiver uses private key [6]. AES uses symmetric keys in encryption and decryption [7]. AES is a symmetrical block cipher with 128 bit bit size [8]. The 128 bit AES key length is 128 bits and uses 10 rounds. The 192 bit AES key length is 192 bits and uses 12 rounds. The key length of AES 256 bits is 256 bits and uses 14 rounds [9,10,11,12]. Each round on AES 128, 192 and 256 bits in each encryption and decryption process has identical rotation, except the last round. The AES encryption and decryption process consists of four types of byte transformations:

- Sub Byte Transformation: using an S-box table to substitute every byte of a message block. This non linear layer is useful for resistance to differential and linear cryptanalysis attacks.
- Shift Row Transform: shifts every byte of the message block. The byte shift is done from the leftmost byte to the rightmost byte. This transformation is applied on the second, third and fourth lines, while the first line does not shift. The second row will experience a one-time shift, the third row undergoes two shifts and the fourth row undergoes three shifts.
- Mix Column Transform: substitution that utilizes GF arithmetic (28).
- Add Round Key (ARK): a round key added to a state with bitwise XOR operations.

In cryptography there is a function used to authenticate the security and integrity of a message called a hash function. The hash function is a function that receives a string input of any length and converts it into a fixed length length string. There are two hash functions: Message Digest (MD5) and Secure Hash Algorithm 1 (SHA1). MD5 generates message digest 128 bit and SHA1 produces 160 bit message digest [5,9]. The MD5 algorithm uses a mathematical function to generate a message digest with different input strings. MD5 algorithm has the advantage that MD5 will generate a different message digest if one of the bits has changed the input message. Another advantage is that the message digest size is smaller than the document size generated by MD5 [13]. The following are the steps of making MD5 hashing [14-15]: i) The addition of the wedge bits. The purpose of adding bit bits is for a congruent message length of 448 modulo 512. Messages converted to binary form are added by bit "0" followed by bit "1" to length equal to 448 modulo 512; ii) Increase the value of the original message length. The message that has been given the bits of the wedge then added again with 64 bits that state the length of the original message. Once added with 64 bits, the message length now becomes 512 bits; iii) Initialize the MD buffer. MD requires four buffers each of which is 32 bits in length. The buffer is named A, B, C, D. The total length of the buffer is 128 bits. Each buffer is initialized with values in hexadecimal notation in the form of A: 01234567, B: 89ABCDEF, C: FEDCBA98, D: 76543210; iv) The functions of fF, fG, fH, fI each contain 16 basic operations of the input. The function fF, fG, fH, fI is a function to manipulate inputs A, B, C, D with 32 bit size. Each function is:

\[
\begin{align*}
    f_F &= (B \land C) \lor (\neg B \land D) \\
    f_G &= (B \land C) \lor (B \land \neg D) \\
    f_H &= B \oplus C \oplus D \\
    f_I &= C \oplus (B \land \neg D) 
\end{align*}
\]

v) The output of the message digest is the result of connecting bits in A, B, C, D.

Procedures of encryption and decryption of files on AES 256 bit and MD5 are:
File encryption algorithm with AES 256 bit and MD5: i) The user selects the plaintext file to be encrypted; ii) User entering key; iii) The application will hashing the key with MD5; iv) AES 256 bit encryption process; v) The application will generate a ciphertext file.

File decryption algorithm with AES 256 bit and MD5: i) User selects the ciphertext file to be decrypted; ii) User entering key; iii) The application will hashing the key with MD5; iv) AES 256 bit decryption process; v) The application will generate a plaintext file.

3. Results and Discussions

AES 256 bit application that allows users to encrypt and decrypt documents on Android smartphones like txt files, doc, docx, xls, xlsx, ppt, pptx, pdf. This application uses the bottom navigation menu, so users simply click on the menu located at the bottom of the application. This app is built using the Android Studio editor. For AES encryption and decryption process using AES class developed by Popa Tiberiu 2011 using Java programming language. The following packages and classes are used in this program:

```java
import com.example.user.enkripsideskripsifile.aes.AES; // the class that used for encryption and decryption
```

The implementation of AES encryption and decryption class show in Figure 1.

![Figure 1. Implementation of encryption and decryption AES 256 class.](image1)

Here are the packages and classes used on MD5:

```java
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class MD5 {
    public static String getMD5(String input) {
        try {
            MessageDigest md = MessageDigest.getInstance("MD5");
            byte[] messageDigest = md.digest(input.getBytes());
            BigInteger number = new BigInteger(1, messageDigest);
            String hashtext = number.toString(16);
            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }
            return hashtext;
        }
        catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }
}
```

![Figure 2. The MD5 class.](image2)

Implementation of MD5 class shown in Figure 3 and Figure 4
Figure 3. MD5 implementation in AES encryption.

Figure 4. MD5 implementation in AES decryption.

Encryption Menu: As it is shown in the screen shot in Figure 5, the user interface is created and designed by researchers using the .xml layout in Android Studio and name it layout_enkripsi.xml. This menu is used for the encryption process, the user simply select the plaintext file and enter a key or password to encrypt the document. The time required for the decryption process is also displayed at this stage.

Decryption Menu: As it is shown in the screen shot in Figure 6, the user interface is created and designed by researchers using the .xml layout in Android Studio and name it layout_dekripsi.xml. This menu is used for the decryption process, the user simply select the ciphertext file and enter a key or password to decrypt the document. The time required for the decryption process is also displayed at this stage.

In this section the researcher conducted an encryption experiment on the type of *.txt file with the size of 659 bytes shown in Figure 7. The ciphertext file will generate a 672-byte *.ct file extension shown in Figure 8. In the experiment's experiment, the ciphertext file the decrypted looks the same as the original file, but the ciphertext file is 672 bytes. All the encryption and decryption process files are shown in Figures 7, 8, 9 which are opened by using the Notepad ++ application.
In this section the researcher also performed encryption experiments on the *.docx file type with the size of 864465 bytes shown in Figure 10. The ciphertext file will produce a file extension *.ct of size 864480 bytes shown in Figure 11. In the experiment, the decrypted ciphertext looks the same as the original file, but the ciphertext file is 864480 bytes. All the encryption and decryption process files are shown in Figure 10, 12 which are opened using the Office WPS application and Figure 11 opens with the Notepad ++ application.

Encryption and decryption applications have been installed and tested on smartphone devices based on the Android operating system. The smartphone specs used during the experiment were Asus Zenfone 6, Intel Atom 2 GHz CPU Model, 2 GB RAM and Android 5.0 Lollipop. The results of encryption and decryption of files using AES 256 bits are shown in Table 1.

| No | File Name       | The size of the original | File Encryption | Duration of Encryption | File Size Decryption | Duration of Decryption |
|----|-----------------|--------------------------|-----------------|------------------------|----------------------|------------------------|

Table 1. The results of encryption and decryption of files using AES 256 bit.
Table 2. The results of encryption and decryption of files using AES 256 bit combination with MD5

| No | File Name               | The size of the original file (byte) | File Encryption Size (byte) | Duration of Encryption (ms) | File Size Decryption (byte) | Duration of Decryption (ms) |
|----|-------------------------|-------------------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|
| 1  | data.txt                | 659                                 | 672                         | 107                         | 672                         | 43                          |
| 2  | Data Teknis.xlsx        | 12,877                              | 12,880                      | 533                         | 12,880                      | 969                         |
| 3  | readme.txt              | 15,637                              | 15,648                      | 789                         | 15,648                      | 1,329                       |
| 4  | Chapter I.docx          | 23,556                              | 23,568                      | 1039                        | 23,568                      | 2,040                       |
| 5  | Slide.pptx              | 71,847                              | 71,856                      | 2,960                       | 71,856                      | 7,876                       |
| 6  | Student data.xlsx       | 97,585                              | 97,600                      | 3,915                       | 97,600                      | 12,330                      |
| 7  | SQL Tutorial.pdf        | 506,113                             | 506,128                     | 21,529                      | 506,128                     | 152,161                     |
| 8  | Chapter III.docx        | 864,465                             | 864,480                     | 40,383                      | 864,480                     | 390,521                     |
| 9  | Writing guidelines.pdf  | 867,691                             | 867,696                     | 36,382                      | 867,696                     | 386,393                     |
| 10 | PPS TI.pptx             | 1,632,198                           | 1,632,208                   | 72,274                      | 1,632,208                   | 1,217,620                   |

Table 1. shows that the decryption process takes considerably longer than the encryption process. The larger the file size, the encryption and decryption process will take a long time and this applies to all document file types as well. The process of encrypting and decrypting a text file has a fairly short duration of time. Encryption and decryption of * .xlsx files has a fairly fast time duration. While encryption and decryption of document files such as * .docx, * .pptx, * .pdf where in the file there are text and images that have a long duration of time. Table 2 shows that the results are similar to Table 1. In some experiments conducted by researchers with the same document file, the results of encryption and decryption in document files such as data.txt and readme.txt files show faster results using a
combination of AES 256 bits with MD5 than just using AES only. While the document file containing graphic data such as tables, images, lines in the file *.docx, *.xlsx, *.pptx, *.pdf then encryption using only AES 256 bit method faster than using AES 256 combination bit and MD5.

4. Conclusion
In this paper, it shows that the successful implementation of file document encryption as well as decryption on Android smartphones. Experience from experimental results was conducted by researchers, the encryption process is faster than decryption. It shows that the combination of AES 256 bit encryption and decryption algorithms and MD5 runs faster on Android smartphones. This app provides better mobile security than unauthorized access. This application ensures the security of end to end data transfer without experiencing corrupt data, but for file size, this is larger than the original file. The time required for encryption and decryption for document files using a combination of AES and MD5 is faster than just using AES. While the files contain graphic data such as tables, lines, images of encryption time and decryption faster by using AES alone than using a combination of AES and MD5.

References
[1] Nishika and R. Kumar Yadav,”Cryptography on Android Message Applications – A Review”, International Journal on Computer Science and Engineering (IJCSE) ISSN : 0975-3397, vol. 5 no. 05, May 2013
[2] A.H Ali and A. Makki Sagheer, 2017, “Design and Implementation of Secure Chatting Application with End to End Encryption”, Journal of Engineering and Applied Sciences 12 (1) ISSN : 1816-949X, 156-160
[3] A. Pandey and U. Kumar Lilhore, 2017, “An Improved AES Cryptosystem Based Genetic Method on S-Box, With, 256 Key Sizes and 14-Rounds”, International Journal of Advanced Engineering Research and Science (IJAERS), ISSN: 2349-6495(P) | 2456-1908(O) vol.4 no. 3 March 2017, pp. 166-171.
[4] S. Tayde and S. Siledar, 2015, “File Encryption, Decryption Using AES Algorithm in Android Phone”, International Journal of Advanced Research in Computer Science and Software Engineering, vol. 5 no. 05 May 2015, pp. 550-554.
[5] S.Mishra, Shikha Mishra, N.Kumar, 2013, “Hashing Algorithm: MD5”, IJSRD - International Journal for Scientific Research & Development ISSN (online): 2321-0613 vol. 1, no.9, 2013, pp.1931-1933.
[6] R. Verma and Shikha Agrawal, 2013, "Data Security For Any Organization By Using Public Key Infrastructure Components And MD5, RSA Algorithms", IJRET: International Journal of Research in Engineering and Technology ISSN: 2319-1163 , vol.2 Issue: 05, May-2013
[7] A. Pandey and U. Kumar Lilhore, 2017, “An Improved AES Cryptosystem Based Genetic Method on S-Box, With, 256 Key Sizes and 14-Rounds”, International Journal of Advanced Engineering Research and Science (IJAERS), ISSN: 2349-6495(P) | 2456-1908(O) vol.4 no. 3 March 2017, pp. 166-171.
[8] Preeti Singh et al, 2014, "Symmetric Key Cryptography: Current Trends", International Journal of Computer Science and Mobile Computing ISSN 2320–088X, vol.3 Issue.12, December-2014, pp. 410-415.
[9] N.K. Jharbade† and R. Shrivastava, 2012, “Network based Security model using Symmetric Key Cryptography (AES 256– Rijndael Algorithm) with Public Key Exchange Protocol (Diffie-Hellman Key Exchange Protocol), IJCSNS International Journal of Computer Science and Network Security, vol.12 No.8, August 2012, pp. 69-73.
[10] M.Gavhane, S.More, G.Ghodke, Patil S.S., Nimbalkar P.P, 2015, "Spy Android App", International Journal of Emerging Technologies and Engineering (IJETE) ISSN 2348 – 8050 vol.2 no.2, February 2015, pp.33-37.
[11] S. El Adib and N. Raissouni, 2012, "AES Encryption Algorithm Hardware Implementation Architecture: Resource and Execution Time Optimization", International Journal of Information & Network Security (IJINS) ISSN: 2089-3299, vol.1, no.2, June 2012, pp. 110-118

[12] S.R.Paddhan, T.R. Wagh, S.N. Shelke, 2014, "Secured Wireless Communication Using AES-256", Global Journal of Advanced Engineering Technologies, Special Issue (CTCNSF-2014) ISSN (Online): 2277-6370 & ISSN (Print): 2394-092

[13] S. Ranjan Lenka and B. Nayak, 2014, “Enhancing Data Security in Cloud Computing Using RSA Encryption and MD5 Algorithm”, International Journal of Computer Science Trends and Technology (IJCST), ISSN : 2347-8578 vol. 2 no. 3 June 2014, pp.60-64.

[14] P. Walia and V. Thapar, 2014, "Implementation of New Modified MD5-512 bit Algorithm for Cryptography", International Journal of Innovative Research in Advanced Engineering (IJIRAE) ISSN: 2349-2163, vol.1 issue:6, July 2014

[15] A. Kumar Kasgar, M. Kumar Dhariwal, N. Tantubay, H. Malviya, 2013,"A Review Paper of Message Digest 5 (MD5)", International Journal of Modern Engineering & Management Research ISSN: 2320-9984, vol.1, Issue 4, December 2013

Acknowledgements
Researchers give many thanks to the Faculty of Technology and Computer Science, Universitas Prima Indonesia. Moreover, special thanks to the Abdi Dharma (Dean of Faculty of Technology and Computer Science) and Arjon Turnip for supporting us in this research.