Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 135 (2018) 283–291

Procedia Computer Science

www.elsevier.com/locate/procedia

3rd International Conference on Computer Science and Computational Intelligence 2018

Text Encryption in Android Chat Applications using Elliptical Curve Cryptography (ECC)

Dimas Natanael[a], Faisal[a], Dewi Suryani[b,*]

[a]Mathematics Department, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480
[b]Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480

Abstract

Information technology is an important aspect of human life that has provided comfort and ease in communicating. The most used communication technologies today is the smartphone, which is a mobile phone with the ability of a portable computer. Currently, to communicate each other, people feel more convenience using chat application in a smartphone than calling or short message services (SMS) features. Some of the advantages of chat application are there is no message size limitations, highest number of consumers use it; especially younger demographics,and it also works on mobile web without application download. Due to the convenience of communication using chat apps increases, the security demands are also higher, i.e., a proper cryptography scheme is needed to protect the messages. In this paper, we implement ECC algorithm to secure text message in messaging application of a smartphone. We will adopt the method proposed by Singh [1] to create a chat application in Android program that equipped end-to-end encryption. We also give the experimental result of our chat apps performance such as the accuracy of the received text message, average encryption and decryption time.

Keywords: Elliptical Curve Cryptography, Text Encryption, Chat Application, Cryptography

1. Introduction

In this era, technology has become an important aspect of human needs. Almost all their activities have been facilitated by technology, especially communication technology. Communication technology has always evolved over time and the most commonly used communication technology today is smartphones. The smartphone is a telecommunication device with a touch screen function that offers more capabilities than simply calling and sending messages, which also works as a portable computer that can be carried anywhere. Due to the capabilities of the smartphone, people do not have to rely on call and short message services (SMS) features anymore. They can use chat applications to communicate with each other. As it is easier and more convenient to communicate with smartphones, there are also

* Corresponding author. Tel.: +62-812-9337-3595.
E-mail address: dsuryani@binus.edu

1877-0509
10.1016/j.procs.2018.08.176

new threats to digital data that raise issues related to security in online data exchange. This security effort is carried out with the aim of preserving the integrity and confidentiality of any information that stored or transmitted by one party to another.

To maintain the security of the current data transmission in the communication process, there is an algorithm that usually handles the situation, i.e., cryptography. Cryptography is a field of study that studies the schemes of data encryptions and decryptions. These schemes are called cryptographic systems or ciphers. Encryption or also known as enciphering is a process of transforming a raw data or the original message, i.e. plaintext, into a coded message called ciphertext. Moreover, the decryption (deciphering) is the inverse process of encryption that converts the ciphertext back to the plaintext. Cryptography itself consists of several algorithms that developed by researchers, such as Caesar cipher, RivestShamirAdleman (RSA), data encryption standard (DES), triple data encryption standard (3DES), elliptic curve cryptography (ECC), and advanced encryption standard (AES). These algorithms can improve the security of a system with the objective of providing a level of complexity that is required by a specific program to safeguard the integrity and confidentiality of the data.

Nowadays, among all the cryptography algorithms, only ECC and RSA are the most common algorithms for information security. The RSA algorithm was first introduced by Ron **Rivest**, Adi **Shamir**, and Leonard **Adleman** in 1978[2]. RSA is motivated by the published works of Diffie and Hellman in 1976[3]. However, to obtain the best algorithm for information security, Chandraseka et al.[4] have compared the ECC and RSA algorithms based on the size of encryption results at the same level of security. The comparison outcomes show the ECC has smaller sizes than RSA algorithm. In Lenstra and Verheul[5] it also state that ECC can offer equivalent security with substantially smaller key sizes. Therefore, many researchers focus on studying the behaviors of the ECC algorithm. Elliptic curve cryptography (ECC) is a public key cryptographic algorithm that uses elliptic curves and finite fields. Elliptic Curve Cryptography (ECC) was first proposed by victor Miller[6] in 1986 and independently by Neal Koblitz[7] in 1987. With its advantage in the key sizes, the ECC algorithm can be implemented for devices with limited resources, such as smartphones.

In this paper, we implement the ECC algorithm to encrypt and decrypt text messages in chat applications of a smartphone. Currently, there are two kinds of platforms that usually installed and popularly used in smartphones, i.e., Android and iOS. Based on the Statista survey[8] that conducted in December 2017, Android has dominated the smartphone market which its percentage had reached 88.37% compared to the other platforms. Therefore, our chat application is built on an Android smartphone with the help of Android Studio, the real-time database firebase, and the local storage room persistence library. Two novelties given in this article are first we implement the algorithm that Singh[1] uses in Smartphones. Secondly we validate the encryption and decryption algorithms that are not given in Singh[1]. So this article is a support for the algorithm created in Singh[1].

The rest of this paper is organized as follows. Section 2 describes any prior works that related to ours. Next, our proposed system and the ECC algorithm will be explained in Section 3. Afterward, the experimental and result will be discussed further in Section 4. Finally, Section 5 concludes this paper and describes future improvements.

2. Related Works

Elliptical curve cryptography (ECC) is a trending cryptography scheme that currently many researchers[1,9,10] experiment about. The key size and security level of ECC are the factor which motivates many researchers to learn and explore it to know its strength and limit. The ECC is suitable for mobile devices since it is relatively small key size compared to other cryptography schemes such as RSA and Diffie Hellman. Below is a table from the NSA(Nasional Security Agency)s Case for Elliptic Cryptography showing the disparity

In 2015, L.D. Singh and K. M. Singh[1] offered the faster method to encrypt and decrypt data using ECC rather than common lookup table in their research titled ”Implementation of Text Encryption using Elliptic Curve Cryptography”. The ECC was used in the process of transforming plaintext into coordinates to be encrypted with the help of ASCII code. This research showed that their method is effective to encrypt and decrypt text data.

Moreover, in their research titled ”Design of an Android Application for Secure Chatting”, Ammar H. Ali and Ali M. Sagheer[11] implemented many algorithms such as Elliptic Curve Diffie Hellman (ECDH), AES, and RC4 in their chatting application with Android platform. Elliptic Curve was used as the key exchange protocol which will be mixed into other symmetric algorithms, AES and RC4. This research proved that ECDH, AES, and RC4 can be hybrid

Table 1: Comparison table between ECC and other asymmetric cryptography schemes

Simmetric Key Size(bits)	RSA and DH Key Size(bits)	ECC Key Size(bits)	Ratio of DH Cost : EC Cost
80	1024	160	3:1
112	2048	224	6:1
128	3072	256	10:1
192	7680	384	32:1
256	15360	521	64:1

implemented in chatting application using the Android platform. In contrast to Vigila[9], the plaintext is transformed as a ASCII numbers. Then they used this number in doubling operation of a point of an elliptic curve.

In this paper, we create a chat application that applies cryptography system using ECC algorithm proposed by Singh et al.[1] in Android chat application. Unlike Ali et al.[11], we do not mix other algorithms besides ECC algorithm only.

3. Proposed System

Our proposed system contains two major processes, i.e., encryption and decryption process by Singh[1]. Each process implements the ECC algorithm, which will be described in this section. Figure 1 illustrates the full processing pipeline of our proposed system.

3.1. Elliptical Curve Cryptography (ECC)

In 1985, Neal Koblitz and Victor Miller realized that elliptic curve (EC) can be adapted and applied for public key systems. Note that elliptic curve is not the same as an ellipse. Ellipse has two lines of symmetry, i.e., vertical and horizontal. In the other hand, the elliptical curve has only the horizontal line. According to the proposed method of Singh et al.[1], ECC can be represented by the following EC equation:

$$y^2 = x^3 + ax + b \tag{1}$$

The EC equation (Equation 1) is also known as the Weierstrass equation, where a and b are constants that determine the shape of the curve, as well as meet this condition:

$$4a^3 + 27b^2 \neq 0 \tag{2}$$

The ECC equation for a finite field Z_p with p primes can be written as follows:

$$y^2 = (x^3 + ax + b) \; mod \; p \tag{3}$$

which has 2 main operations, i.e., point addition and point doubling. Point addition is performed when we want to add two different points. The coordinate result is the third point through by the line which originates from the first and second coordinates. To perform this operation, firstly looking for the gradient (λ) by this way:

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \; mod \; p \tag{4}$$

and then, followed by a third search point

$$\begin{aligned} x_3 &= \{\lambda^2 - x_1 - x_2\} \; mod \; p \\ y_3 &= \{\lambda(x_1 - x_3) - y_1\} \; mod \; p \end{aligned} \tag{5}$$

where (x_1, y_1) is the first point and (x_2, y_2) is the second point.

Point doubling is calculated when both points are in the same coordinates. The result coordinates are the points that the line passes with the *tan* position against the initial coordinates. The operation of this point doubling is not so different from the point addition, which is starting with the gradient (λ) search as follows:

$$\lambda = \frac{3{x_1}^2 + a}{2y_1} \tag{6}$$

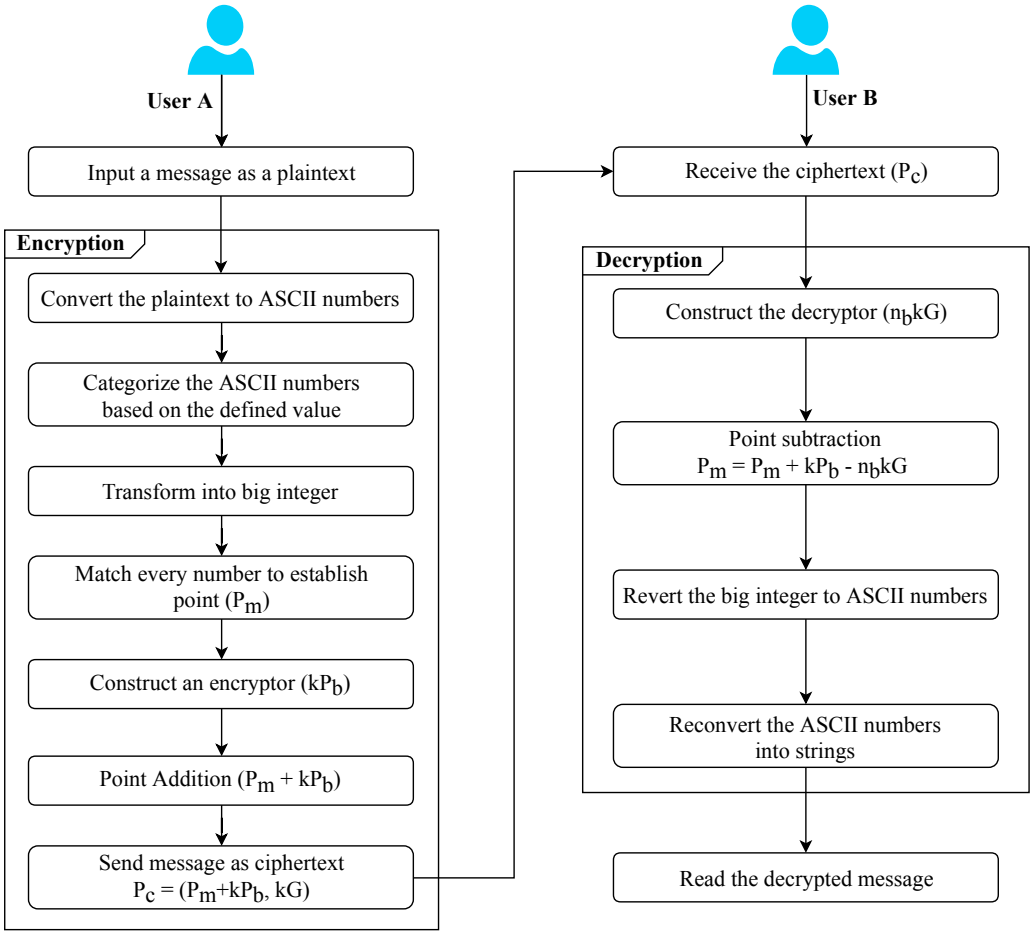


Fig. 1: Full overview of our proposed system. Two users are connected by a message that will be sent and received. Encryption is processed when the user A wants to send a message to the user B whereas the user B needs to perform the decryption in order to read the message from the user A.

which is derived from the implicit decrease of the EC equations in Equation 1, followed by the search of x_3 and y_3.

$$\begin{aligned} x_3 &= \{\lambda^2 - 2x_1\} \; mod \; p \\ y_3 &= \{\lambda(x_1 - x_3) - y_1\} \; mod \; p \end{aligned} \tag{7}$$

where (x_1, y_1) is the origin coordinate. If P is a point of an elliptic curve, then the multiplication of a positive integer k to P is defined as a point addition that is repeated k times. The multiplication can be written as $kP = P+P+P+P+...+P$. If $x_1 = x_2$ and $y_1 = y_2 = 0$ or $x_1 = x_2$ and $y_1 = -y_2$, then these points are intersect at infinity, denoted by 0.

Before starting the ECC algorithm, the two parties who wish to exchange information must first approve the use of a curve and its parameters, such as the coefficients of a and b, the upper bound of p in EC equation, and a base point G to be used. In order to create a public key, let's say that there are two sides A and B that select a private key n_A and n_B, respectively. The private keys n_A and n_B are an integer in the set with p upper bound. It is usually used for decryption and confidential. The public key of each part is obtained in a way:

$$P_A = n_A G \quad \text{and} \quad P_B = n_B G \tag{8}$$

By using the public key, each party can encrypt a message to a ciphertext in the following form:

$$P_c = \{P_m + kP_B, kG\} \tag{9}$$

where P_m is the message that has been changed and encrypted by the public key of user B (P_B), then k is a random integer. The use of k is to obtain different results of ciphertext from the same plaintext without formatting. Data decryption can be performed by determining a point addition operation against $P_m + kP_B$ with mirror point from $n_B kG$.

3.2. Encryption

Encryption is the process of encoding a message so that messages can only be accessed by the authorized one. All procedures for encrypting a message are depicted in Figure 1. The details of the encryption procedures are described as follows:

1. Convert the plaintext to ASCII numbers. A message (plaintext) that inputted by the user will be converted into ASCII numbers for each character.
2. Categorize the ASCII numbers based on the defined value. After the plaintext changed into ASCII numbers, the numbers are categorized into some groups that the total members of the group depend on the defined value. The grouping is done by calculating the 65536 based number (b) that generates primes (p).
3. Transform into big integers. Every number of the groups is transformed into a big integer by multiplying the ASCII with the b numbers included its power.
4. Match every number to establish point (P_m). Each big integer number are matched with the next big integer to be a pair which represents a point to be the encryption key. If there is a big integer without a partner, then a padding of 32's number which stands for an empty symbol of ASCII will be added.
5. Construct an encryptor (kP_B). In this phase, the system will generate an encryptor (kP_B) for encrypting the message, which is obtained by multiplying the k stands for the random number and the receiver's public key (P_B).
6. Point addition ($P_m + kP_B$). Here, the plaintext that has been altered (P_m) will be encrypted with the encryptor by applying point addition method in elliptic curve operation.
7. Send message as ciphertext $P_c = (P_m + kP_B, kG)$. The encrypted message ($P_m + kP_B$) will be sent including the encryption key (kG) as the ciphertext (P_c).

3.3. Decryption

In cryptography, decryption is the reverse process of encryption where the ciphertext will be converted back into plaintext using any kind of selected algorithms. The steps to perform the decryption are described in Figure 1. The full descriptions of the decryption stages are discussed as follows:

1. Construct the decryptor ($n_B kG$). This phase is conducted after the receiver's system gets the ciphertext (P_c) from the sender. Here, the point multiplication will be performed between the encryption key (kG) and the receiver's private key (n_B) in order to generate the decryptor.
2. Point subtraction ($P_m = P_m + kP_B - n_B kG$). After the decryptor is generated, the ciphertext will be decrypted by applying the point subtraction with the decryptor. Basically, the point subtraction cannot be done in elliptic curve operation. Therefore, we reverse the decryptor and employ the point addition operation ($P_m = (P_m + kP_B) + mirror(n_B kG)$) instead.
3. Revert the big integer to ASCII numbers. In this phase, the decrypted data is still in big integer form. Therefore every big integer numbers will be converted back into ASCII with 65536 based numbers in order to perform the next step.
4. Reconvert the ASCII numbers into strings. Finally, the decrypted message will be transformed back into strings from ASCII number so that the receiver can read the message.

4. Experimental and Result

Our proposed system is realized into an application for an Android smartphone. The application is in the form of chat application which built using Android studio. This application has several features, such as add friends, end-to-end encrypted messages, etc. The looks of this chat app are illustrated in Figure 2. This paper focuses on two main things, i.e., proof of the decryption process using ECC algorithm through a code and performance comparisons of the ECC secured chat application.

Unlike in Singh [1], we give the proof of decryption using ECC algorithm is done by implementing it in Python with the help of Sympy library. Figure 3 proves that the decryption process using the ECC algorithm is possible either mathematically or program.

$$(P_m + kP_B) - n_B kG = P_m \tag{10}$$

where $P_m = (x_1, y_1)$, $kP_B = n_B kG = (x_2, y_2)$, and assume that $x_1 \neq x_2$.

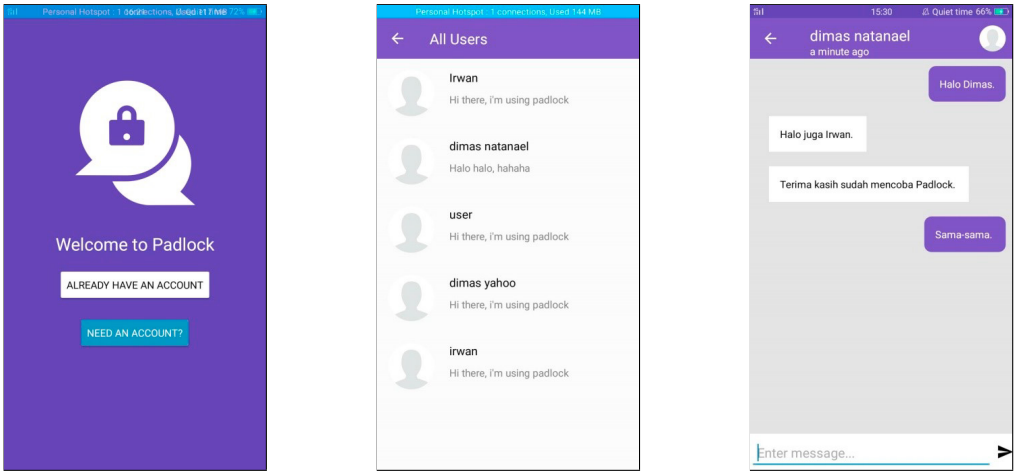
By using point addition method, we get $(P_m + kP_B) = (s, t)$, where

$$\begin{aligned} s &= \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - (x_1 + x_2) \\ t &= -\left(\frac{y_2 - y_1}{x_2 - x_1}\right)^3 + \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 + x_2) + \frac{x_1 y_2 - x_2 y_1}{x_2 - x_1} \end{aligned} \tag{11}$$

Let's assume $(P_m + kP_B) - n_B kG = (CC, DD)$, where

$$CC = \left(\frac{-y_2 - t}{x_2 - s}\right)^2 - (s + x_2) \quad \text{and} \quad DD = -\left(\frac{-y_2 - t}{x_2 - s}\right)^3 + \left(\frac{-y_2 - t}{x_2 - s}\right)(s + x2) + \left(\frac{s(-y_2)) - (tx_2)}{(x_2) - s}\right) \tag{12}$$

To demonstrate $(P_m + kP_B) - n_B kG = P_m$, the conditions that must be fulfilled are $CC = x_1$ and $DD = y_1$, which has been proved by the Figure 3. Moreover, the performance that is measured in here consists of time and accuracy which detailed in Table 2 and Table 4.



(a) Home Screen (b) Friends List (c) Chat Room

Fig. 2: Screenshots of the chat application

```
from sympy import *

x1, y1, x2, y2 = symbols('x1 y1 x2 y2')
init_printing(use_unicode=True)

s=((y2-y1)/(x2-x1))**2 - (x1+x2)
t=-((y2-y1)/(x2-x1))**3 + (((y2-y1)/(x2-x1))*(x1+x2)) + \
    (((x1*y2)-(x2*y1))/(x2-x1))

CC = ((-y2-t)/(x2-s))**2 - (s+x2)
DD = -((-y2-t)/(x2-s))**3 + (((-y2-t)/(x2-s))*(s+x2)) + \
    ((s*(-y2))-(t*x2))/(x2-s)

resultCC = simplify(CC)
resultDD = simplify(DD)

print "RESULTS"
print "-------"
print "CC = ", resultCC
print "DD = ", resultDD
```

```
RESULTS
-------
CC =  x1
DD =  y1
>>>
```

Fig. 3: Proofing of decryption formula in ECC algorithm using python.

4.1. Time Performance

Our chat application is created in order to provide an end-to-end encrypted message that secured when sending a message between two parties. For experimental, this app is installed on two smartphones (Xiaomi Mi 5) with the specification of Android 7.0 Nougat, Qualcomm Snapdragon 820, and 3 GB RAM. We apply the elliptic curve with the secpl92r1 standard in F_p field by tuning the following parameters:

$$p = FFFFFFFF\ FFFFFFFF\ FFFFFFFF\ FFFFFFFE\ FFFFFFFF\ FFFFFFFF = 2^{192} - 2^{64} - 1$$
$$a = FFFFFFFF\ FFFFFFFF\ FFFFFFFF\ FFFFFFFE\ FFFFFFFF\ FFFFFFFC$$
$$b = 64210519\ E59C80E7\ 0FA7E9AB\ 72243049\ FEB8DEEC\ C146B9B1$$
$$G = (04\ 188DA80E\ B03090F6\ 7CBF20EB\ 43A18800\ F4FF0AFD\ 82FF1012,$$
$$07192B95\ FFC8DA78\ 631011ED\ 6B24CDD5\ 73F977A1\ 1E794811)$$

This experiment is conducted according to the total number of characters per message to measure the encryption, decryption, and sending time in millisecond (ms). The results of this experiment are detailed in Table 2.

Table 2: Time Performance Experimental Results

# chars	Encryption time (ms)	Decryption time (ms)	Sending time (ms)
500	53	100	4299
1000	96	130	4172
1500	139	121	4636
2000	212	135	4934
2500	289	145	4301
3000	333	163	4338

Based on the experimental result in Table 2, either encryption, decryption, or sending time are not stable although we expected the time consume should be incrementally increased over the total characters per message. That situation can be caused by the random number (k) when encryption and decryption processes. To verify our speculation, we perform another experiment using a static k value, i.e., $k = 28186466$. Table 3 shows the results of the second experiment, which describes the encryption and decryption time always increased depends on the total characters per message.

Table 3: Time Performance Experimental Result (with static k value)

# chars	Encryption time (ms)	Decryption time (ms)
500	101	100
1000	116	124
1500	184	142
2000	218	168
2500	286	183
3000	360	216

4.2. Accuracy

In this paper, we also measure the accuracy of the chat application from encrypting, transferring, and then decrypting the message. Every message contains approximately 6000 characters with the size of 5.85 KB. The experiments are executed for 25, 50, 75, and 100 messages that are sent per one measurement. The accuracy results are presented in Table 4

Table 4: Accuracy Experimental Results

# chats	Average encryption time (ms)	Average decryption time (ms)	Accuracy (%)
25	689.2	701.8	100
50	698.44	716.66	100
75	695.6133	725.7867	100
100	678.68	732.18	100

The experimental result shows all messages are sent between two sides, 100 % successfully received and correctly encrypted. The average time of encryption and decryption is 678.68 ms and 732.18 ms, respectively.

Here, we compare our proposed approach with other methods performance. The comparison results are presented in Table 5. It turns out our method outperform compared to Vigila and Muneeswaran's work either encryption or decryption process. However, we still need to improve our performance by optimizing the encryption and decryption functions including the hyperparameter.

Table 5: Comparison Result of Several Approaches

Methods	# chars	Encryption time	Decryption time	Ciphertext size	Mapping	Common lookup table
Vigila and Munees [9]	409	1.95 s	0.83 s	459.118 KB	Yes	Yes
Singh and Singh [1]	409	0.093 s	0.14 s	21.017 KB	No	No
Our system	409	0.263 s	0.206 s	15.587 KB	No	No

Furthermore, we also compared our result with Megha Kolhekar and Anita Jadhav's [10] approach which also used maps and common lookup tables. The results obtained are superior with an encryption time of 0.012 seconds that produces an encrypted text of 1 KB and a deciphering of 0.061 seconds, while the methods of Megha Kolhekar and Anita Jadhav [10] require 0.20 seconds for the encryption that produces an encrypted text of 1,146 KB and 0.30 seconds for decryption.

5. Conclusion and Future Works

We present our proposed system to create an android based chat application that comes with end-to-end encryption. The form of chat application using the ECC algorithm for text encryption and decryption proposed by Singh[1]. In this research It proves that the ECC algorithm can be implemented and has competitive performance in both time and accuracy. However, the application of ECC in the Android app still needs to be optimized for better results. For the future, therefore, the optimization of ECC methods can be implemented and also performed in the encryption of images and videos.

References

1. Singh, L.D., Singh, K.M.. Implementation of text encryption using elliptic curve cryptography. *Procedia Computer Science* 2015;**54**:73–82.
2. Rivest, R.L., Shamir, A., Adleman, L.. A method for obtaining digital signatures and public-key cryptosystems. *Commun ACM* 1978;**21**(2):120–126. doi:\bibinfo{doi}{10.1145/359340.359342}. URL http://doi.acm.org/10.1145/359340.359342.
3. Diffie, W., Hellman, M.. New directions in cryptography. *IEEE Trans Inf Theor* 2006;**22**(6):644–654. doi:\bibinfo{doi}{10.1109/TIT.1976.1055638}. URL http://dx.doi.org/10.1109/TIT.1976.1055638.
4. Chandrasekar, A., Rajasekar, V.R., Vasudevan, V.. Improved authentication and key agreement protocol using elliptic curve cryptography. *International Journal of Computer Science and Security* 2009;**3**(4):325–333.
5. Lenstra, A.K., Verheul, E.R.. Selecting cryptographic key sizes. *J Cryptol* 2001;**14**(4):255–293. doi:\bibinfo{doi}{10.1007/s00145-001-0009-4}. URL http://dx.doi.org/10.1007/s00145-001-0009-4.
6. Miller, V.S.. Use of elliptic curves in cryptography. In: *Lecture Notes in Computer Sciences; 218 on Advances in cryptology—CRYPTO 85*. Berlin, Heidelberg: Springer-Verlag. ISBN 0-387-16463-4; 1986, p. 417–426. URL http://dl.acm.org/citation.cfm?id=18262.25413.
7. Koblitz, N.. Elliptic curve cryptosystems. *Mathematics of Computation* 1987;**48**(177):203–209.
8. Statista, . Market share of mobile operating systems in indonesia from january 2012 to december 2017. 2018. URL https://www.statista.com/statistics/262205/market-share-held-by-mobile-operating-systems-in-indonesia.
9. Vigila, S.M.C., Muneeswaran, K.. Implementation of text based cryptosystem using elliptic curve cryptography. In: *Advanced Computing, 2009. ICAC 2009. First International Conference on*. IEEE; 2009, p. 82–85.
10. Kolhekar, M., Jadhav, A.. Implementation of elliptic curve cryptography on text and image. *International Journal of Enterprise Computing and Business Systems* 2011;**1**(2).
11. Ali, A.H., Sagheer, A.M.. Design of an android application for secure chatting. *International Journal of Computer Network and Information Security* 2017;**9**(2):29.