N-TEA (New-Text Encryption Algorithm) For Secure Chat In Android Based *Application*

Sugiyanto
Informatics Department
Adhi Tama Institute of Technology Surabaya
Arief Rachman Hakim 100, Surabaya, Indonesia
sugianto@itats.ac.id

***Abstract*—Smartphones have become an essential communication device where people share much information through this communication channel. The Android-based chat app allows users to create groups and interacts with other users by exchanging messages over a local network or Internet network. In the Android-based chat application, there are problems with data security issues. Messages sent from chat apps become insecure because the messages are on an accessible network allowing a hacker to infiltrate and steal words in data packets. Encryption becomes a good solution with the key exchange to prevent intruders or hackers trying to take data. This research proposes N-TEA algorithm as a solution in securing chat messages in Android applications. N-TEA is an encryption mechanism that works in real-time and does not depend on what data to send. This algorithm serves to support the ability of a real-time and robust encryption mechanism. The results of this study show that the change of plaintext character results in an avalanche effect of 36.72%. In another part, the change of encryption key character results in an avalanche effect of 42.9%. This research proves that the N-TEA algorithm can secure to send messages well.**

Keywords—android, chatting, communication, encryption, security

I. INTRODUCTION

Smartphones have become an essential communication device where users share much information through this communication channel [1]. The Android operating system has become popular in the Smartphone market. One of the Android apps that users often use is chatting apps. Some types of chat apps allow users to create groups or groups for family and friends to share messages [2]. The chat app lets users interact with other users by exchanging messages over a local network or Internet network [3]. In the chat app, there are problems with data security [4]. Messages sent by users become insecure because through a public access network that allows a hacker to infiltrate and steal messages in data packets [5].

Security of public communications during data exchange applications required to make sure security, and privacy [6]. Encryption becomes a good solution with the key exchange to prevent intruders or hackers trying to steal data [7]. Various ways have been done to discuss the problem of data theft. Some previous algorithms for text encryption include Elliptic Curve Cryptography [8], Extended Euclidean Algorithm [9], and Advanced Encryption Standard [10]. Dependency on the transmitted data is a source of system weakness which will increase the chances of violating the encryption algorithm [11]. This research proposes N-TEA as a solution in securing chat messages to make it more difficult for hackers to solve. N-TEA is an encryption mechanism that works in real-time, so it does not depend on what data to send.

The N-TEA algorithm supports to maintain real-time capabilities and powerful encryption mechanisms. The basic idea of N-TEA encryption mechanism depends on the dictionary or dictionary 'AM' which maintains data reduction and security. Both versions of the published 'AM' code were developed as a text compression framework [12]. The next code is used for bandwidth optimization in both text exchanges and mobile applications such as e-mail and chat apps. In this study, the N-TEA algorithm is based on the 'AM' framework used to encrypt digital data.

II. METHODOLOGY

A. Requirements Analysis

Analysis of this system needs includes software and hardware requirements for the design of making Android-based chat encryption application with N-TEA algorithm. Software requirements used for the manufacture of this Android-based chat encryption application are as follows:

a. Microsoft Windows 7 Operating System,
b. Android SDK Operating System,
c. Android Studio,
d. Java (Programming Language),
e. Firebase.

Hardware that becomes the need for the manufacture of Android-based chat encryption application with this cryptography is as follows:

a. Notebook
   Processor : Intel(R) Core(TM) i3 CPU T6500 @ 2.10 GHz (2 CPUs),
   Memory : 4 GB, HDD 500 GB,
b. Smartphone has an Android operating system.

The many exchanges of information that occur in this system, so many people want the information for personal interests and group interests. This condition may occur in the absence of high security from the valuable information exchange. Based on the problem, the system to be built is a data security system that serves to secure chat content from the tapers are not responsible. Some features requirement in the system as follows :

a. Display incoming messages from the sender,
b. Send encrypted messages to recipients,
c. Describe ciphertext to plaintext.

B. System Design

At this stage, the data design and application interface design and followed by application development. In the design of this Android-based chat application, this research using network topology as shown in Fig. 1.

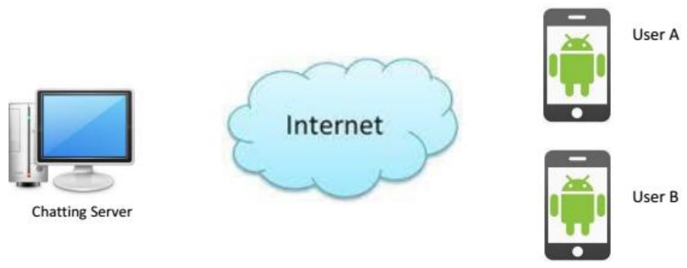

Fig. 1. Network Topology

The flow of this Android-based chat application starts from connecting the Android smartphone to the server via the internet. Once connected to the server, then this chat application using TCP protocol. In this chat application, there are features register, login, contact, and chat. After the user register, the user gets the ID as the identity that will communicate with other users. Users one with other users can communicate when it is a friend by way of exchanging ID. The message sent will be given a choice, using encryption or not.

The message is data or information that can be read and understood the meaning. Another name for the message is plaintext. For the message to be unreadable by other unauthorized parties, the message needs to be encoded into another unintelligible form. The encoded form of the message is called the ciphertext. Ciphertext must be able to be transformed back into original plaintext to be accepted and readable by the recipient.

Data communication involves exchanging messages between two entities. The sender is the entity that sends messages to other entities. The recipient is the entity receiving the message. The sender would want the message to be sent securely, i.e., the sender is sure that the other party cannot read the contents of the sent message. The solution is to encode the message into ciphertext. Encryption is a process of making an explicit message (plaintext) into a random message that cannot be read (ciphertext). While the decryption process is a reverse process of encryption, this process will convert the ciphertext into plaintext by using the same critical algorithms and key.

Although both options are equally encrypted, there is the only difference in receiving the message. If the sender uses encryption, all received messages will be the ciphertext, requiring a key to encrypt the message. Conversely, if the sender does not use encryption, then the client automatic receives a plaintext message and can be directly read. Some ways to distribute the key to receiving the message in the form of plaintext can be through chat, SMS, phone, email, and so forth.

In this system design is done parable there are two users, that is A and B. User A as the message sender, while user B as the message recipient. Before user A sends a message, user A is given a choice of how to send a message, i.e., whether user A wants to use encryption or not. When the message is encrypted, the message will enter the encryption function. For more details can be seen in Fig. 2.

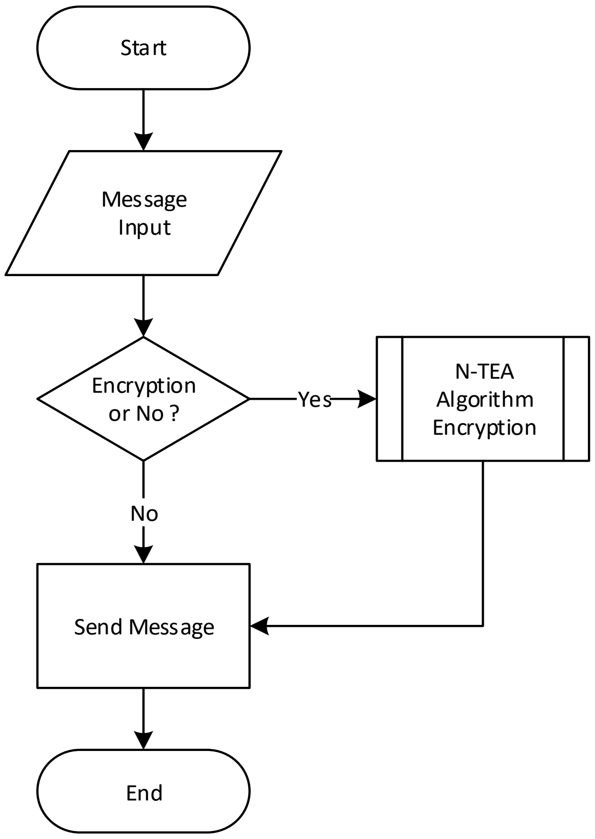

Fig. 2. Flowchart Encryption of chat messages

On the chat chats encryption flowchart, there is an N-TEA encryption algorithm that has several processes flows that can be seen in Fig. 3. The plaintext-shaped message is split into 64-bit blocks. Then divided into two namely L0 and R0, each of 32 bits. After that, the system created the key block 128 bits and divided into four parts. Each part is 32 bits, so it becomes k [0], k [1], k [2] and k [3]. The next step is the process of shifting the bits to the left and right, then the Y and Z that have been shifted will be added with the key k [0] -k [3]. Then, the initial Y and Z will be added to the sum (delta). Results from XOR cannot be over 32 bits. When it is fulfilled, the last step is to combine Y and Z, so it becomes a ciphertext.

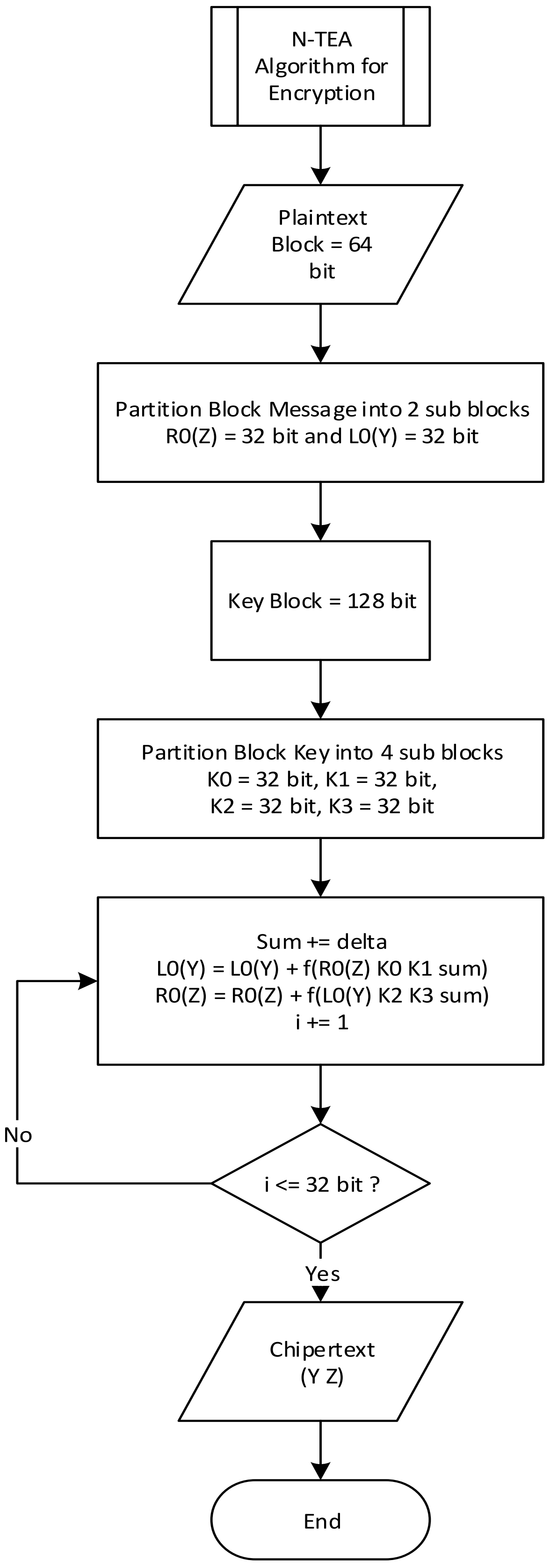
N-TEA Algorithm for Encryption
Plaintext Block = 64 bit
Partition Block Message into 2 sub blocks
R0(Z) = 32 bit and L0(Y) = 32 bit
Key Block = 128 bit
Partition Block Key into 4 sub blocks
K0 = 32 bit, K1 = 32 bit,
K2 = 32 bit, K3 = 32 bit
Sum += delta
L0(Y) = L0(Y) + f(R0(Z) K0 K1 sum)
R0(Z) = R0(Z) + f(L0(Y) K2 K3 sum)
i += 1
No
i <= 32 bit ?
Yes
Chipertext (Y Z)
End

Fig. 3. N-TEA Encryption Flowchart

It is in this decryption that distinguishes the display of messages that have been sent by user A. If user A sends using encryption, then the message received by user B in the form of a ciphertext and a key to change to plaintext again. Whereas if user A sends without using encryption, then directly user B can read the message without decryption first. The decryption process in the chat application is shown in Fig. 4.

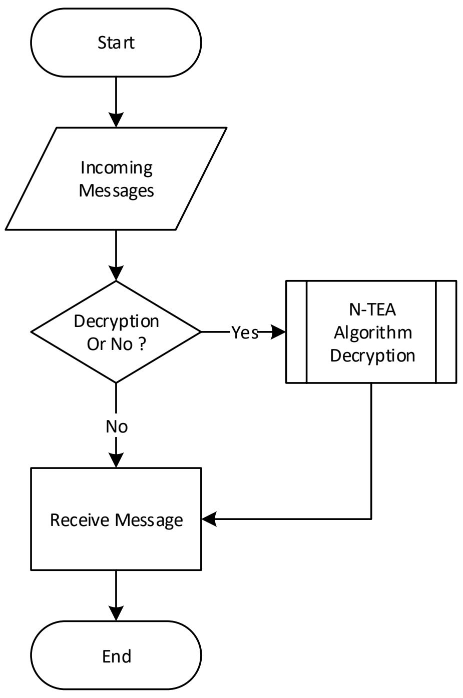

Fig. 4. Flowchart Decryption chat messages

In the decryption flowchart chat message there is an N-TEA decryption algorithm. N-TEA decryption description path can be seen in Fig. 5. The message is already shaped ciphertext divided into per-block 64 bit. Then divided into two, i.e., L0 and R0, every 32 bits. After that, the ciphertext is converted into the 128-bit key block, and divided into four parts. Each part is 32 bits, so it becomes k[0], k[1], k[2] and k[3]. The next step is to shift one ciphertext bits to left or right. Y and Z which have been shifted will be added to the key k[0]-k[3], while the initial Y and Z will be added to the sum (delta). Results from XOR cannot be over 32 bits. When it is fulfilled, the final step is to combine Y and Z, so it becomes plaintext.

C. Implementation

After doing the data design and application interface, the application is developed by adjusting based on the design that has been made. This application is developed with Java programming language.

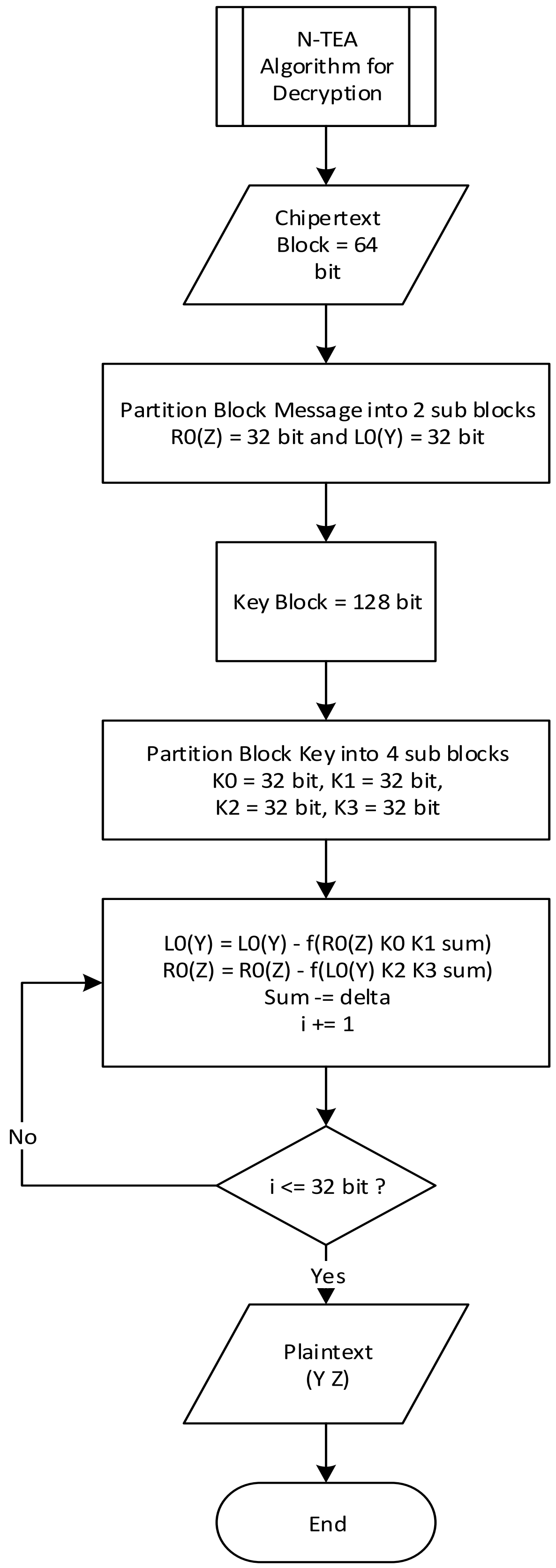
N-TEA Algorithm for Decryption
Chipertext Block = 64 bit
Partition Block Message into 2 sub blocks R0(Z) = 32 bit and L0(Y) = 32 bit
Key Block = 128 bit
Partition Block Key into 4 sub blocks K0 = 32 bit, K1 = 32 bit, K2 = 32 bit, K3 = 32 bit
L0(Y) = L0(Y) - f(R0(Z) K0 K1 sum)
R0(Z) = R0(Z) - f(L0(Y) K2 K3 sum)
Sum -= delta
i += 1
i <= 32 bit ?
No
Yes
Plaintext (Y Z)
End

Fig. 5. Flowchart decryption of N-TEA

D. Evaluation

At this stage, the system evaluated to determine N-TEA algorithm perform. In the process of testing the system, researchers simulate to test the application that has been made by doing running on the emulator and handset Android. The parameter to test the effect of bit change is avalanche effect [10].

$$\text{Avalanche Effect} = \frac{\text{Number of flipped bits in ciphered text}}{\text{Number of bits in ciphered text}} \text{ x } 100\ \% \quad (1)$$

III. RESULT AND DISCUSSION

The first plaintext test is to change one character from plaintext. Then we get the Indonesian word “jayalahITATS” (change the character “j” (01101010) to “k” (01101011)). The detail evaluation can be seen in table 1.

TABLE I. CHANGE OF PLAINTEXT CHARACTER

Plaintext	jayalahITATS
Key	1234
Plaintext Encryption	28E8F80A04C8446862028D57E4119E8C
Plaintext Modification	kayalahITATS
Encryption Modification	FABA8EB7DB068F79655F795A34D71166
Bit Exchange	47
Avalanche Effect	36.72%

With the change of plaintext of one character, then there is bit exchange as much as 47 bits. With a value of 36.72%, this proves that the N-TEA algorithm has good encryption. The next test of the key is to change a last character of the key. Then we get the key “1234” (change the character “4” (00110100) to “5” (00110101)). The detail evaluation can be seen in table 2.

TABLE II. CHANGE OF LAST KEY CHARACTER

Plaintext	jayalahITATS
Key	1234
Plaintext Encryption	28E8F80A04C8446862028D57E4119E8C
Key Modification	1235
Encryption Modification	05BC28B3540FEF4A5514983DFC8F4EF
Bit Exchange	55
Avalanche Effect	42.9%

Besides changing plaintext, evaluation is done by another mechanism such as modification the key. There is a significant change of the results got before and after the key is changed the avalanche effect value of 42.9%. This result is better than earlier research that has been done by researchers using Advanced Encryption Standard algorithm [10]. From calculate avalanche effect, the average value of AES algorithm is only 37.24%. The next test is a computational trial. This computation test aims to figure the speed of time of encryption and decryption on N-TEA algorithm. The following test results are shown in table 3.

TABLE III. COMPUTATION TEST OF N-TEA ALGORITHM

Data	Size	Encryption Time (sec)	Decryption Time (sec)
1	1 byte	0.34	0.82
2	2 byte	0.41	0.94
3	3 byte	0.47	1.23
4	1 byte	0.29	0.69
5	3 byte	0.48	0.98
6	1 byte	0.34	0.82

The larger the size of the encrypted text, the longer the encryption time. In 1-byte text data requires encryption time of 0.34 second, while 2-byte text data requires encryption time of 0.41 second. The speed of message encryption depends on the number of words entered. If the number of words sent sender more, then the process becomes longer. The decryption process takes longer than the encryption process. In the first text data, the encryption process takes 0.34 seconds. The decryption process time takes longer than encryption process is 0.82 second. About security, this algorithm is more guaranteed. Because all messages received by getting the message is not the original message, but rather the ciphertext. Recipient must use a key that has been entered from the message sender to open a message. If the key entered by the message recipient differs from the key that the sender entered, then the message will not be recipient read.

IV. CONCLUSION

The conclusion of this study indicates that N-TEA algorithm can make chatting communication more secure. The results of this study show that the change of plaintext character results in an avalanche effect of 36.72%. In another part, the change of encryption key character results in an avalanche effect of 42.9%. For further application development and adding encryption strength, this algorithm can be joint using the extended Euclidean algorithm or using linear congruence generator feature combination.

REFERENCES

[1] D.P. Baviskar, S.N. Patil, and O.K. Pawar, “Android-Based Message Encryption/Decryption Using Matrix,” International Journal of Research in Engineering and Technology, vol. 4, issue 1, 2015.

[2] Sugiyanto, and R.K. Hapsari, “Sistem Keamanan Short Message Service (SMS) Berbasis Android Menggunakan Algoritma Advanced Encryption Standard (AES),” Seminar Nasional Sains dan Teknologi Terapan (SNTEKPAN) IV 2016, 2016, pp. 55-60.

[3] A. Kadhim, and S. Khalaf, “New Approach for Security Chatting in Real Time,” International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), vol. 4, issue 3, 2015.

[4] K. Chouhan, and S. Ravi, “Public Key Encryption Techniques Provide Extreme Secure Chat Environment,” International Journal of Scientific & Engineering Research, vol. 4, issue 6, 2013.

[5] A.H. Ali, and A.M. Sagheer, “Design and Implementation of Secure Chatting Application with End to End Encryption,” Journal of Engineering and Applied Sciences, vol. 12(1), 2017, pp. 156-160.

[6] U. Singh, and U. Garg, “An ASCII value based text data encryption System,” International Journal of Scientific and Research Publications, vol. 3, issue 11, 2013.

[7] O.M.A. Al-Hazaimeh, “A New Approach For Complex Encrypting And Decrypting Data,” International Journal of Computer Networks & Communications (IJCNC), vol. 5, no. 2, 2013.

[8] L.D. Singh, and K.M. Singh, “Implementation of Text Encryption using Elliptic Curve Cryptography,” Procedia Computer Science, vol. 54, 2015, pp. 73–82.

[9] S. Pattanayak, and Dipankar Dey, “Text Encryption And Decryption With Extended Euclidean Algorithm And Combining The Features Of Linear Congruence Generator,” International Journal of Development Research, vol. 06, issue 07, 2016, pp. 8753-8756.

[10] Sugiyanto, and R.K. Hapsari, “Pengembangan Algoritma Advanced Encryption Standard pada Sistem Keamanan SMS Berbasis Android Menggunakan Algoritma Vigenere,” Jurnal ULTIMATICS Universitas Multimedia Nusantara, vol. 8, issue 2, 2016, pp. 131-138.

[11] R, Padate, and A. Patel, “Encryption and Decryption of Text using AES Algorithm,” International Journal of Emerging Technology and Advanced Engineering, vol. 4, issue 5, 2014.

[12] A.M.A. Mansour, and M.A.M. Fouad, “N-TEA: New Text Encryption Algorithm—Secured Data Exchange,” International Journal of Software Engineering and Its Applications, vol. 8, no. 9, 2014, pp.199-206.M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.