Improving Firebase BaaS Service Security in Counseling Chat Applications: AES-256 and CBC Approach for End-to-End Encryption

Mogar Nurhandhi[1*], Agus Suhendar[2]
[1]Program Studi Informatika, Fakultas Sains dan Teknologi, Universitas Teknologi Yogyakarta
[2] Program Studi Informatika, Fakultas Sains dan Teknologi, Universitas Teknologi Yogyakarta
Email: [1]mogarnurhandhi@gmail.com, [2]agus.suhendar@staff.uty.ac.id

Abstract – The activities of using the internet and exchanging information and sending messages have used a lot of internet media, one of which is chat message media, but over time data and information security problems in chat messages that are often encountered are active and passive wiretapping. In this research, the author wants to create a counseling chat message security application using the AES (Advanced Encryption Standard) algorithm cryptographic method combined with the CBC (Cipher Block Chaining) technique which is an advanced development of the ECB (Electronic Code Book) technique. AES basically uses a block cipher with a length of 128 bits as the default operation, and the key length size varies from 128, 192 and 256 bits. so AES uses a 4x4 matrix equation with each section having a size of 1 byte. From these problems, research will be conducted to develop an application to accommodate the counseling process using a chat application that has the main focus of securing messages with image types and stored in the Firebase database service (Backend as a Service). As well as using the End-to-End service principle so that users do not need to do the encryption or decryption process directly because the process has been carried out by the system, this will also provide more security aspects in terms of confidentiality of key data and initialization vectors. So that the process of exchanging information using the media chat counseling application is secured and avoids tapping by irresponsible parties.

Keywords – Counseling, Advanced Encryption Standard, Cipher Block Chaining, Cryptography, Chatting

I. INTRODUCTION

The need for counselling for school students is one of the means that must be available in every school, this is not just because of the formality in the subjects and curriculum accompanied by the school. But counselling has a position in building the personality and mentality of student participants in experiencing various problems such as career paths or further learning [1]. The counselling process can be tried by teachers with various procedures that have existed and tried so far, but with the rapid growth of technology and data, the teaching procedures related to counselling must be able to explore this growth and one of them is by using internet-based communication technology. Based on the explanation of the case, the author wants to create a practical message application or chat by implementing the AES algorithm, which is not only able to accommodate the counselling process but also takes into account the security aspects and confidentiality of student information as a form of privacy that must be protected. The procedure that can be used in securing chat messages is cryptography, which is a scientific discipline that focuses on methods of hiding messages so that they are not known to unauthorised parties. But in the modern definition of cryptography, it is closer to the basic science in the field of informatics with the aim of securing data, protecting the integrity of information or authenticating an entity so that modern cryptography has a more complex purpose [2].

In its application, cryptographic procedures can use methods of securing messages by changing messages and hiding the meaning of messages by encryption and decryption processes. In this procedure, something is known as encryption which can be interpreted as a code or chipper. Cryptographic procedures have many encoding methods and types that can be applied in various aspects of pc, data, or network security, not only that this procedure also allows the combination of some algorithms in it so that the system built by this procedure has a strong level of security [3]. Basically, cryptography can be categorized into two parts, symmetric cryptography and asymmetric cryptography, both of which have their own algorithmic derivatives [4]. In this research, the symmetric key cryptography method will be applied using the AES encryption algorithm. AES (Advanced Encryption Standard) is a continuation of the development of the DES (Data Encryption System) algorithm then the AES algorithm slowly acts as a replacement for DES because the DES algorithm has weaknesses in the hardware sector, namely the problem of using keys that are too long. The AES algorithm has advantages over its predecessor, namely a high security system and has several types of levels with different key lengths to support security. The AES algorithm has another advantage, namely relatively fast and efficient computing speed with 128-bit keys but also provides 192 and 256 keys for complex computations [5].

In the AES algorithm based on the shortest key length of 128-bit, the AES algorithm is resistant to exhaustive key search attacks and the AES algorithm is also proven to be able to withstand square attacks where this attack utilises the byte approach structure. If the length of AES is 128-bit then it allows this attack to accelerate iterations 6 times faster than AES, but this algorithm is able to survive with more cycles so that the resulting resilience by this type of attack [6]. Then the source of the literature review used by the author based on previous research with the same research family, among others, is the AES algorithm as a block algorithm based on the classic logic operation (XOR) which again operates on both binary states 0 and 1. This research proposes a modification by replacing the XOR operation between binary (0,1) with (#) on a 256 state table

e-ISSN: 2614-8404
p-ISSN:2776-3234

that works with a specified 8-bit. Although it provides more complexity, this modification succeeds in providing efficient encryption and decryption processing time and provides more security against differential attacks [7]. The process of securing data in email messages with the AES algorithm method is possible using the java programming language and the netbeans platform. The encryption process mechanism is based on the transformation of SubByte, ShiftRows, MixColumns and AddAroundKey while the decryption will produce an inverse cipher of the encryption process transformation [8]. Data security in cloud computing is also an important issue with the encryption data authentication method through the conversion of cipher blocks into cipher streams by separating text into several AES-256 encryption blocks [9]. In the process of sharing and transferring data on cloud computing technology, there is a risk of data theft, therefore the application of encryption using AES in the data transfer process is a worthy preventive measure as a security system [10]. Data in the form of audio can be encrypted using the AES-128 method and allows using analogue signals in audio communication. With an FPGA (Field Programmable Gate Array) device, namely Artix-7 (xc7a200tlffg1156-2L) and Kintex-7 (xc7k160tffg676-2L) as well as the VIVADO tool application. further simulation and delay configuration by reducing the column matrix operation in the AES algorithm so as to provide real-time peer to peer communication time [11]. The combination of the AES algorithm with the PRNG (Pseudorandom Number Generator) chaos system provides advantages in terms of the size of data that can be processed and the best key acquisition in securing medical image data. FPGA (Field Programmable Gate Array) devices are an option because circuitry is possible for specific algorithms as well as being dynamically programmable [12].

The AES algorithm is available in several programming language libraries and one of them is java which will be implemented in this research. In addition, the development does not only use the pure AES cryptography method, but also combined using the CBC (Cipher Block Chaining) block cipher technique, this will strengthen the system with the use of two security key parameters and initialization vector. So in this research cryptography will be applied using the java programming language with the output of android mobile applications with the AES-256 encryption algorithm.

II. RESEARCH METHODOLOGY

In completing this research, the author uses a method to design systematic system development, while the method used is the waterfall model. The waterfall model emphasises a system development scheme in an orderly and sequential manner at each stage, starting from initial identification to the testing or maintenance stage [13]. The waterfall model used in the system development process in this study can be seen in Figure 1 below.

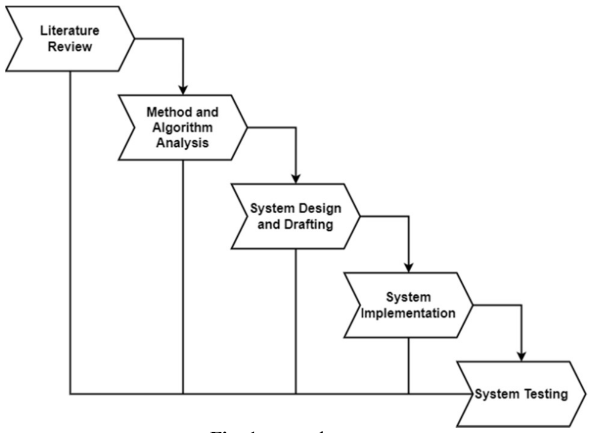


Fig. 1 research stages

A. Literature Review

The first stage in this research and its relationship to system development is a literature study, this stage is carried out by analysing and studying previously existing related research such as in journal media, proceedings, books, and other publication media. The results at this stage can be seen in the literature review in the previous chapter which explains the process and results achieved by previous researchers.

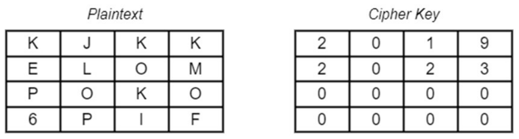
B. Method and Algorithm Analysis

In the previous chapter it was explained that cryptography methods can be broadly divided into two, namely based on symmetric and asymmetric keys. In this research, AES is included in symmetric cryptography so that both the same key will be used in the encryption process and the decryption process of a data. The AES algorithm is a type of symmetric block ciphertext algorithm in which it has a key length variation which is divided into 3 namely 128-bit, 192-bit, and 256-bit. The key length will determine the number of rounds that will be carried out in the encryption and decryption process, as for the key length and other information can be seen in table 1 below.

Table 1. AES Key Variations

AES Key	Parameter AES		
	Key Length	Block Size	Turnaround
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Before entering the main stage in the basic AES algorithm encryption process, there are several preparations that must be made, including the conversion of plaintext and cipher keys in the form of a 4x4 matrix, then both are converted in hexadecimal form and then converted again into binnary. The results of the conversion mentioned earlier can be seen in some of the images below.



Plaintext

K	J	K	K
E	L	O	M
P	O	K	O
6	P	I	F

Cipher Key

2	0	1	9
2	0	2	3
0	0	0	0
0	0	0	0

Fig 2. 4x4 Matrix Conversion

Plaintext

4B	4A	4B	4B
45	4C	4F	4D
50	4F	4B	30
36	50	49	46

Cipher Key

32	30	31	39
32	30	32	33
30	30	30	30
30	30	30	30

Fig 3. Hexadecimal Conversion

Plaintext

01001011	01001010	01001011	01001011
01000101	01001100	01001111	01001101
01010000	01001111	01001011	00110000
00110110	01010000	01001001	01000110

Cipher Key

00110010	00110000	00110001	00111001
00110010	00110000	00110010	00110011
00110000	00110000	00110000	00110000
00110000	00110000	00110000	00110000

Fig 4. Binary Conversion

Next is the initial round where this stage is actually part of the add roun key. XOR operation is performed between plaintext and cipher key, starting from 4B XOR 32, 45 XOR 32, 30 XOR 30, 36 XOR 30 so that the complete equation is as follows.

$$01001011 \text{ XOR } 00110010 = 01111001 \tag{1}$$

$$01000101 \text{ XOR } 00110010 = 01110111 \tag{2}$$

$$01010000 \text{ XOR } 00110000 = 01100000 \tag{3}$$

$$00110110 \text{ XOR } 00110000 = 00000110 \tag{4}$$

With the same calculation method, all binnary columns of plaintext and cipher key will produce a new matrix and must be converted again into hexadecimal, as for the results can be seen in the following figure.

01111001	01111010	01111010	01110010
01110111	01111100	01111101	01111110
01100000	01111111	01111011	00000000
00000110	01100000	01111001	01110110

Fig 5. XOR Calculation Result

79	7A	7A	72
77	7C	7D	7E
60	7F	7B	00
06	60	79	76

Fig 6. Hexadecimal Conversion

After several processes have been carried out, the next stage is the main stage in the encryption process using the AES Algorithm, this stage is divided into four parts, namely sub bytes, shift rows, mix columns, and add arround key.

a. Sub Bytes

Is a transformation process by utilising the S-Box (Substitution Box) table, this table is an indexing-based tool for the efficiency of the block cipher algorithm substitution process [14], as for the transformation results as follows.

Hexadecimal

79	7A	7A	72
77	7C	7D	7E
60	7F	7B	00
06	60	79	76

Sub-bytes

B6	DA	DA	40
F5	10	FF	F3
D0	D2	21	63
6F	3C	B6	38

Fig 7. Sub-bytes Transformation

b. Shift Rows

Is a row move stage where the first row is not moved, but the second to fourth row is moved consecutively between 1 to 3 bytes.. The shift result is as shown in the following image.

Sub-bytes

B6	DA	DA	40
F5	10	FF	F3
D0	D2	21	63
6F	3C	B6	38

Shift-rows

B6	DA	DA	40
10	FF	F3	F5
21	63	D0	D2
38	6F	3C	B6

Fig 8. Shift-rows Transformation

c. Mix Columns

It is the process of multiplying the results obtained previously with a fixed polynominal matrix, as for example in row 1 column 1 in equation 5 and the calculation is in equation 6 below.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} x \begin{bmatrix} B6 \\ 10 \\ 21 \\ 38 \end{bmatrix} \tag{5}$$

$$\begin{aligned}
&02 \times B6 = 00000010\ XOR\ 10110110 \\
&= x \bullet x7 + x5 + x4 + x2 + x \\
&= x8 + x6 + x5 + x3 + x2\ mod\ x8 + x4 + x3 + x + 1 \\
&= x6 + x5 + x4 + x2 + x + 1 \\
&= 01110111 \\
&03 \times 10 = 00000011\ XOR\ 00010000 \\
&= (x + 1) \bullet (x4) \\
&= x5 + x4 = 00110000 \\
&01 \times 21 = 00000001\ XOR\ 00100001 \\
&= 1 \bullet x5 + 1 \\
&= 00100001 \\
&01 \times 38 = 00000001\ XOR\ 00111000 \\
&= 1 \bullet x5 + x4 + x3 \\
&= 00111000
\end{aligned} \tag{6}$$

From the calculation process, the next step is to add up the results as in equation 7 below.

$$01110111 + 00110000 + 00100001 + 00111000 = 01011110\ (5F) \tag{7}$$

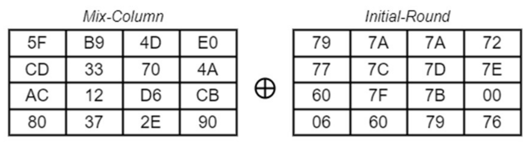
So with the same calculation method, all columns in the matrix will produce mix columns as follows.

5F	B9	4D	E0
CD	33	70	4A
AC	12	D6	CB
80	37	2E	90

Fig 9. Result Mix Column

d. Add Round Key

It is the final process in the AES encryption system where this process uses the XOR operation of the mix-columns result with the initial round as shown below.



Mix-Column

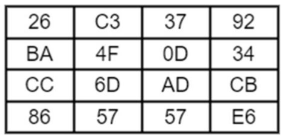
5F	B9	4D	E0
CD	33	70	4A
AC	12	D6	CB
80	37	2E	90

⊕

Initial-Round

79	7A	7A	72
77	7C	7D	7E
60	7F	7B	00
06	60	79	76

Fig 10. XOR Operation



26	C3	37	92
BA	4F	0D	34
CC	6D	AD	CB
86	57	57	E6

Fig 11. XOR Result

The completion is adjusted to the round used based on the length of the AES key, because only as an example, the calculation only reaches the round 0 stage. To do the decryption process, it is done with the inverse transformation of all previous transformations.

e. CBC Method

The AES algorithm is one type of block cipher cryptography algorithm where in this type of cryptography there are several advanced methods or modes for security and one of them is CBC (Cipher Block Chaining). In its flow the CBC method performs the encryption process with the initialization vector parameter in its first block. Parameter IV operates the OR logic gate which will perform the XOR advanced logic operation on the next plaintext, the encryption data is generated using the process to produce the first ciphertext block. so that the new plaintext block only performs the XOR operation with the previous ciphertext output, the inverse function will be used in the next decryption stage, and the plaintext is generated. So that it can be said that this encryption mechanism is like a chain that is related to each other in each block. [15]. The use of the CBC method can provide advantages from the security aspect, where each IV will be different in each encryption session with the same key that will produce different ciphertexts., as for how the CBC method works as in the following figure 12.

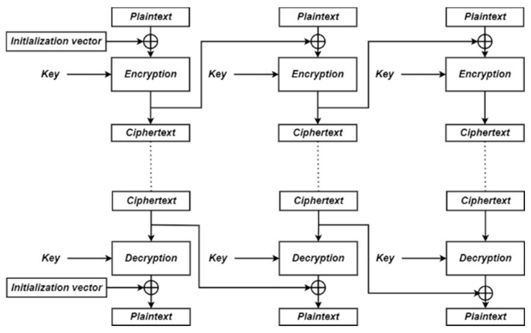


Fig 12. How the CBC Method Works

In this method originally developed in 1976 [16] known operations using XOR bolean logic as described earlier so that the basic mathematical calculation of the CBC method is as in equation 8 below.

$$C_1 = E_k(P_1 \oplus IV)\ ,\ C_i = E_k(P_i \oplus C_{i-1})\ ,\ i = \overline{2,n} \qquad (8)$$

C. System Design and Drafting

In the research conducted by the author in relation to the development of the application system, the author uses DFD (Data Flow Diagram) which in its implementation is displayed in two parts, namely the message encryption and message decryption processes.

a. DFD of Encryption Process

The sender enters the original image message or plaintext and goes through the encryption process into ciphertext which then continues the contact authentication process to find out the purpose of sending the chat message. Then for the process of sending messages and receiving them as ciphertext, decryption is carried out first before displaying, and more details can be seen in Figure 13 below.

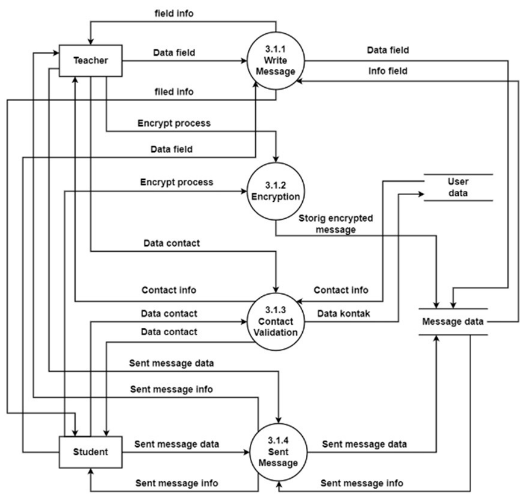


Fig 13. Encryption Process

b. DFD of Decryption Process

The following is a process that describes the reception of messages between teachers and students and vice versa up to the stage of message reading. It contains three processes that can be explained in more detail. Message reception begins. The message data sent as an encrypted message is decrypted by the system and displayed to the message recipient as a normal message, and you can see more details in Figure 14 below.

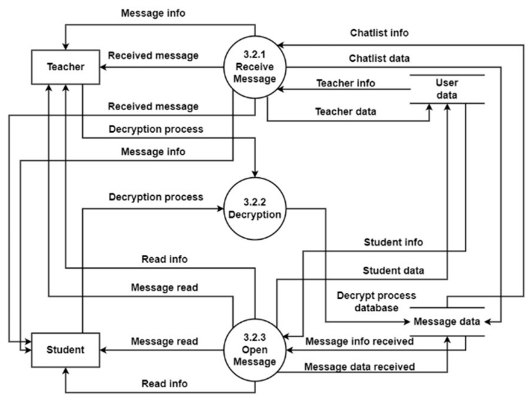


Fig 14. Decryption Process

D. Implementation System

In the implementation stage, a system will be developed to secure chat messages used in the counseling procession. The type or type of message that is the main focus in this research is the type of image or image, besides that the cryptographic method using the AES-256 algorithm is not only used independently but will also be combined using the CBC method. The developed application system produces an android mobile application output by utilizing the Backend As a Service service in the form of a Firebase database with three main types of storage, namely realtime and storage, with firebase authentication as an authentication API for every user who registers or enters the system. Thus the system will use the End-to-End User principle so that both sending or receiving parties only see plaintext data during counseling sessions. The flow of system implementation can be seen in Figure 15 below.

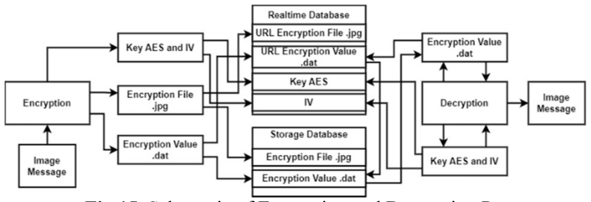


Fig 15. Schematic of Encryption and Decryption Process

E. *System Testing*

At this stage, testing of the application system that has been developed will be carried out, while the testing method that will be used is the blackbox testing method to test the functional structure of the encryption and decryption system. In addition to the blackbox approach, other tests will be carried out using brute force attacks to determine the resilience of the application system in the security domain. And especially to test the ability to hack the keys used in the encryption process.

III. Results And Discussion

At this stage the process of securing counseling messages using the AES-256 algorithm and the CBC method will be explained in two stages, namely the encryption and decryption processes. Then proceed with the results of the implementation of the application that has been developed for the final stage tested with two test parameters.

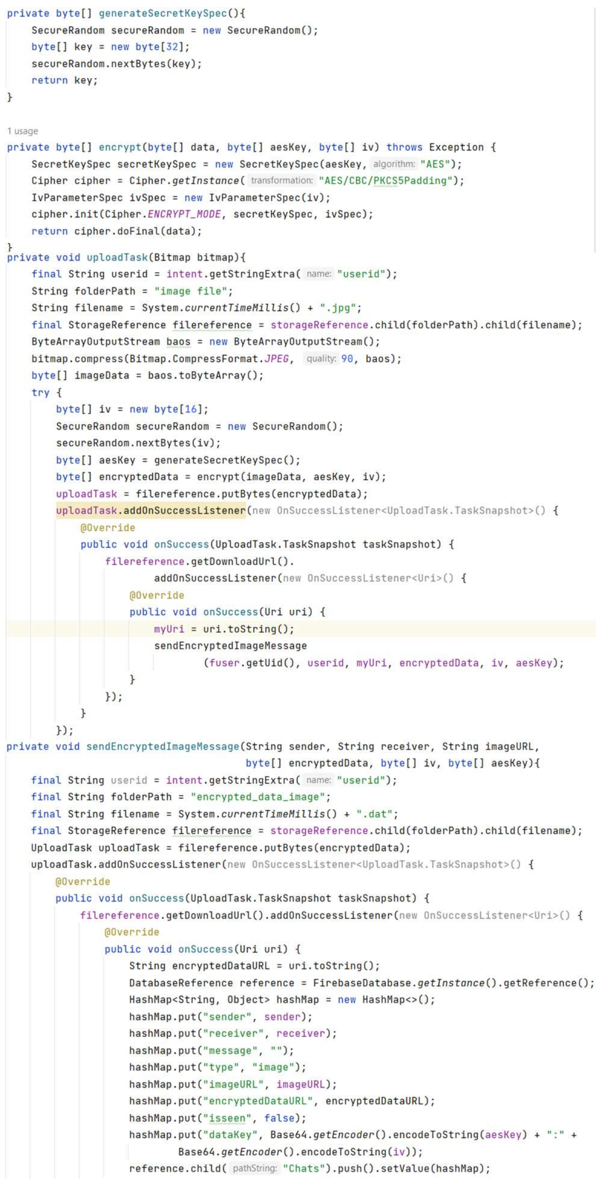
A. *Flow of Encryption Code*

In the encryption process, the author uses three main functions in the script code logic, namely *generateSecretKeySpec()* to generate AES keys with a size of 32-byte randomly. Next is *encrypt()* whose role is to retrieve data, keys, and iv as a condition for performing AES encryption with a combination of CBC. The last function is *uploadTask(Bitmap bitmap)* whose role is to upload the encrypted image message data to the firebase BaaS service.

B. *Flow of Decryption Code*

In the decryption process, the author uses three main functions in the script code logic, namely *generateSecretKeySpec(byte[] aesKey)* to generate objects and hold the AES key. Next *decrypt(byte[] encryptedData, byte[] aesKey, byte[] iv)* whose role is to decrypt using the AES key and iv used in the previous encryption process. And the last function *decryptAndDisplayImage(String encryptedDataURL, final ImageView imageView, final byte[] aesKey, final byte[] iv)* which has the role of downloading the encrypted image and decrypting it before displaying it again.

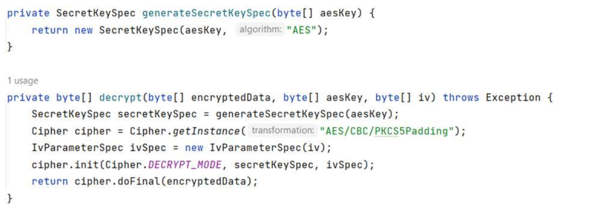
then for each encryption and decryption process code can be seen in the following figures 16 and 17.



```java
private byte[] generateSecretKeySpec(){
    SecureRandom secureRandom = new SecureRandom();
    byte[] key = new byte[32];
    secureRandom.nextBytes(key);
    return key;
}

1 usage
private byte[] encrypt(byte[] data, byte[] aesKey, byte[] iv) throws Exception {
    SecretKeySpec secretKeySpec = new SecretKeySpec(aesKey, algorithm: "AES");
    Cipher cipher = Cipher.getInstance( transformation: "AES/CBC/PKCS5Padding");
    IvParameterSpec ivSpec = new IvParameterSpec(iv);
    cipher.init(Cipher.ENCRYPT_MODE, secretKeySpec, ivSpec);
    return cipher.doFinal(data);
}
private void uploadTask(Bitmap bitmap){
    final String userid = intent.getStringExtra( name: "userid");
    String folderPath = "image file";
    String filename = System.currentTimeMillis() + ".jpg";
    final StorageReference filereference = storageReference.child(folderPath).child(filename);
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    bitmap.compress(Bitmap.CompressFormat.JPEG,  quality: 90, baos);
    byte[] imageData = baos.toByteArray();
    try {
        byte[] iv = new byte[16];
        SecureRandom secureRandom = new SecureRandom();
        secureRandom.nextBytes(iv);
        byte[] aesKey = generateSecretKeySpec();
        byte[] encryptedData = encrypt(imageData, aesKey, iv);
        uploadTask = filereference.putBytes(encryptedData);
        uploadTask.addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                filereference.getDownloadUrl().
                        addOnSuccessListener(new OnSuccessListener<Uri>() {
                    @Override
                    public void onSuccess(Uri uri) {
                        myUri = uri.toString();
                        sendEncryptedImageMessage
                                (fuser.getUid(), userid, myUri, encryptedData, iv, aesKey);
                    }
                });
            }
        });
private void sendEncryptedImageMessage(String sender, String receiver, String imageURL,
                                       byte[] encryptedData, byte[] iv, byte[] aesKey){
    final String userid = intent.getStringExtra( name: "userid");
    final String folderPath = "encrypted_data_image";
    final String filename = System.currentTimeMillis() + ".dat";
    final StorageReference filereference = storageReference.child(folderPath).child(filename);
    UploadTask uploadTask = filereference.putBytes(encryptedData);
    uploadTask.addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
            filereference.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                @Override
                public void onSuccess(Uri uri) {
                    String encryptedDataURL = uri.toString();
                    DatabaseReference reference = FirebaseDatabase.getInstance().getReference();
                    HashMap<String, Object> hashMap = new HashMap<>();
                    hashMap.put("sender", sender);
                    hashMap.put("receiver", receiver);
                    hashMap.put("message", "");
                    hashMap.put("type", "image");
                    hashMap.put("imageURL", imageURL);
                    hashMap.put("encryptedDataURL", encryptedDataURL);
                    hashMap.put("isseen", false);
                    hashMap.put("dataKey", Base64.getEncoder().encodeToString(aesKey) + ":" +
                            Base64.getEncoder().encodeToString(iv));
                    reference.child( pathString: "Chats").push().setValue(hashMap);
```

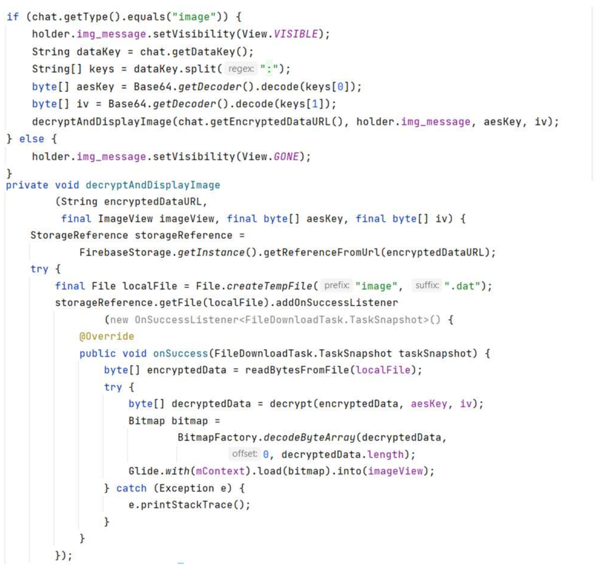
Fig 16. Encryption Code Process



```java
private SecretKeySpec generateSecretKeySpec(byte[] aesKey) {
    return new SecretKeySpec(aesKey,  algorithm: "AES");
}

1 usage
private byte[] decrypt(byte[] encryptedData, byte[] aesKey, byte[] iv) throws Exception {
    SecretKeySpec secretKeySpec = generateSecretKeySpec(aesKey);
    Cipher cipher = Cipher.getInstance( transformation: "AES/CBC/PKCS5Padding");
    IvParameterSpec ivSpec = new IvParameterSpec(iv);
    cipher.init(Cipher.DECRYPT_MODE, secretKeySpec, ivSpec);
    return cipher.doFinal(encryptedData);
}
```

e-ISSN: 2614-8404
p-ISSN:2776-3234



```
if (chat.getType().equals("image")) {
    holder.img_message.setVisibility(View.VISIBLE);
    String dataKey = chat.getDataKey();
    String[] keys = dataKey.split( regex: ":");
    byte[] aesKey = Base64.getDecoder().decode(keys[0]);
    byte[] iv = Base64.getDecoder().decode(keys[1]);
    decryptAndDisplayImage(chat.getEncryptedDataURL(), holder.img_message, aesKey, iv);
} else {
    holder.img_message.setVisibility(View.GONE);
}
private void decryptAndDisplayImage
        (String encryptedDataURL,
         final ImageView imageView, final byte[] aesKey, final byte[] iv) {
    StorageReference storageReference =
            FirebaseStorage.getInstance().getReferenceFromUrl(encryptedDataURL);
    try {
        final File localFile = File.createTempFile( prefix: "image",  suffix: ".dat");
        storageReference.getFile(localFile).addOnSuccessListener
                (new OnSuccessListener<FileDownloadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {
                byte[] encryptedData = readBytesFromFile(localFile);
                try {
                    byte[] decryptedData = decrypt(encryptedData, aesKey, iv);
                    Bitmap bitmap =
                            BitmapFactory.decodeByteArray(decryptedData,
                                    offset: 0, decryptedData.length);
                    Glide.with(mContext).load(bitmap).into(imageView);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
```

Fig 17. Decryption Code Process

C. *Results Of System Implementation*

The results of the implementation of the developed system can be seen in the following interfaces starting from the initial registration login page to the main focus of sending counseling messages with the type of image or image, until at the end it will be shown the acquisition of encryption results in the form of encrypted file data and the encryption value of the data.

a. Login Interface

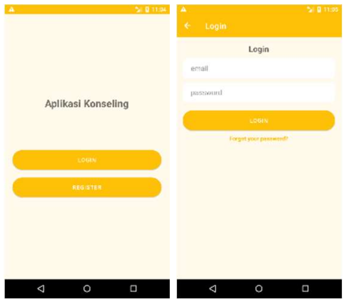


Fig 18. Interface Login Page

b. Registration Interface

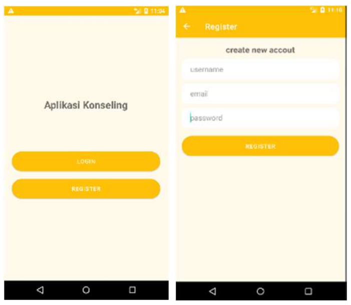


Fig 19. Interface Registration Page

c. Reset Password Interface

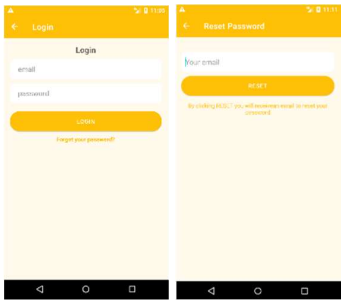


Fig 20. Interface Reset Password Page

d. Dashboard User Interface

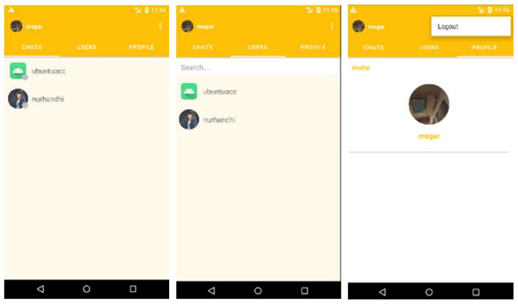


Fig 21. Interface Dashboard

e. Chat Room Interface

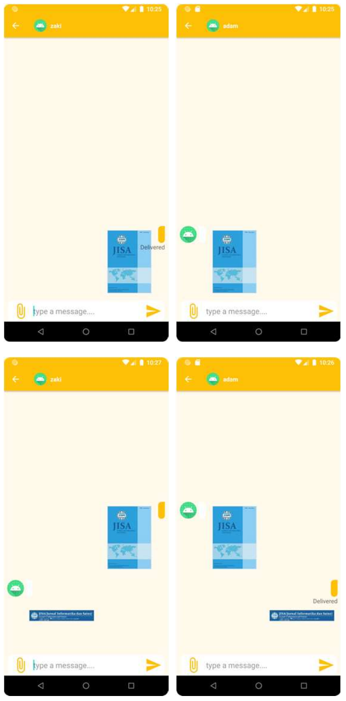
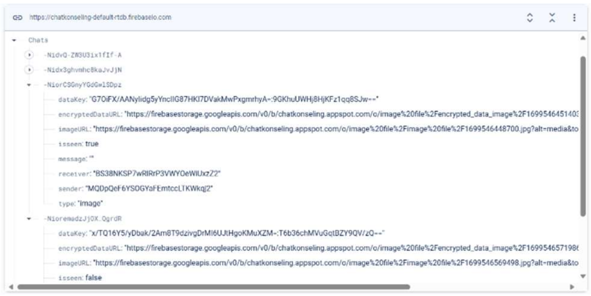


Fig 22. Interface Chat Room

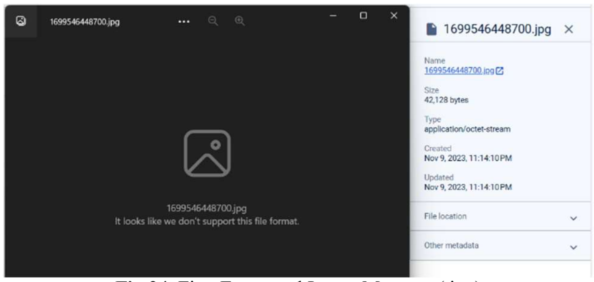
f. Encryption Result

From the results of sending messages with the type of image or image above, the system automatically carries out the encryption and decryption process. This process produces encryption data stored in the firebase database, as for the encryption data as follows.



```
https://chatkonseling-default-rtdb.firebaseio.com
Chats
  -NidvQ-ZW3U3ix1fIf-A
  -Nidx3ghvmhc8kaJvJjN
  -NiorCSGnyYGdGwlSDpz
    dataKey: "G7OiFX/AANylidg5yYncIIG87HKI7DVakMwPxgmrhyA=:9GKhuUWHj8HjKFz1qq8SJw=="
    encryptedDataURL: "https://firebasestorage.googleapis.com/v0/b/chatkonseling.appspot.com/o/image%20file%2Fencrypted_data_image%2F169954645140
    imageURL: "https://firebasestorage.googleapis.com/v0/b/chatkonseling.appspot.com/o/image%20file%2Fimage%20file%2F1699546448700.jpg?alt=media&to
    isseen: true
    message: ""
    receiver: "BS38NKSP7wRIRrP3VWYOeWIUxzZ2"
    sender: "MQDpQeF6YSOGYaFEmtccLTKWkqj2"
    type: "image"
  -NioremadzJjOX.QgrdR
    dataKey: "x/TQ16Y5/yDbak/2Am8T9dzivgDrMI6UJtHgoKMuXZM=:T6b36chMVuGqtBZY9QV/zQ=="
    encryptedDataURL: "https://firebasestorage.googleapis.com/v0/b/chatkonseling.appspot.com/o/image%20file%2Fencrypted_data_image%2F16995465719
    imageURL: "https://firebasestorage.googleapis.com/v0/b/chatkonseling.appspot.com/o/image%20file%2Fimage%20file%2F1699546569498.jpg?alt=media&to
    isseen: false
```

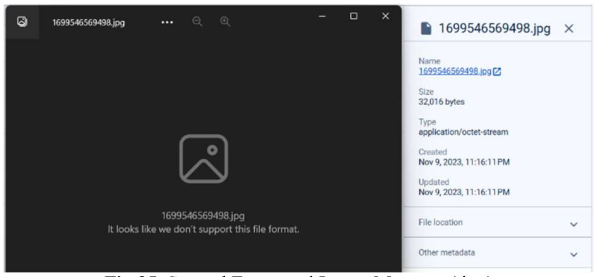
Fig 23. Realtime Database



1699546448700.jpg
It looks like we don't support this file format.

1699546448700.jpg
Name: 1699546448700.jpg
Size: 42,128 bytes
Type: application/octet-stream
Created: Nov 9, 2023, 11:14:10PM
Updated: Nov 9, 2023, 11:14:10PM
File location
Other metadata

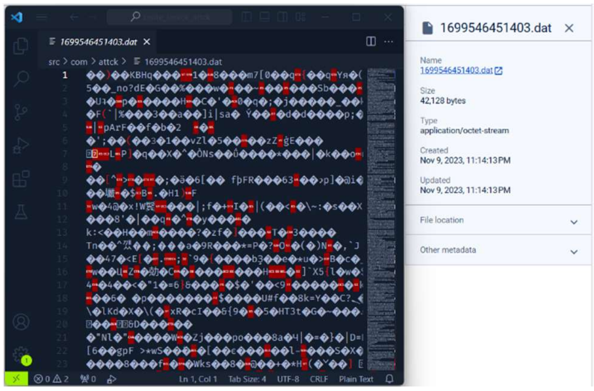
Fig 24. First Encrypted Image Message (.jpg)



1699546569498.jpg
It looks like we don't support this file format.

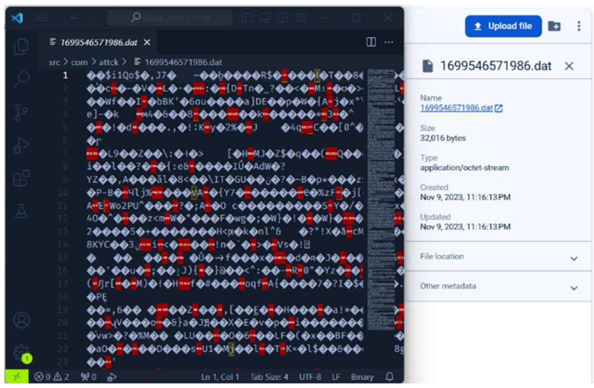
1699546569498.jpg
Name: 1699546569498.jpg
Size: 32,016 bytes
Type: application/octet-stream
Created: Nov 9, 2023, 11:16:11PM
Updated: Nov 9, 2023, 11:16:11PM
File location
Other metadata

Fig 25. Second Encrypted Image Message (.jpg)



1699546451403.dat
Name: 1699546451403.dat
Size: 42,128 bytes
Type: application/octet-stream
Created: Nov 9, 2023, 11:14:13PM
Updated: Nov 9, 2023, 11:14:13PM
File location
Other metadata

Fig 26. First Encryption Value File (.jpg)



1699546571986.dat
Name: 1699546571986.dat
Size: 32,016 bytes
Type: application/octet-stream
Created: Nov 9, 2023, 11:16:13PM
Updated: Nov 9, 2023, 11:16:13PM
File location
Other metadata

Fig 27. Second Encryption Value File (.jpg)

Apart from the encryption results in the form of encrypted image messages and encryption values above, the realtime database also stores the AES key and initialization vector used as parameters for the encryption and decryption process, as for the data can be seen in table 2 below.

Table 2. Key Acquisition and IV

No	AES Parameters	
	AES Key	Initialization Vector
1.	G7OiFX/AANyIidg5yYncIIG8 7HKI7DVakMwPxgmrhyA=	9GKhuUWHj8 HjKFz1qq8SJw==
2.	x/TQ16Y5/yDbak/2Am8T 9dzivgDrMI6UJtHgoKMuXZM=	T6b36chMVu GqtBZY9QV/zQ==

D. *Results Of System Testing*

In this testing stage, two methods will be carried out, namely blackbox and brute force attacks, the results of the testing method are as follows.

Table 3. Blackbox Testing

Testing Schema	Estimated Results	Results
Trying the registration process	The system displays the register form and registers the user	Correct
Trying the login process	The system displays the login form and authenticates the user.	Correct
Sorting users on the search tab	The system displays user search results based on keywords	Correct
Trying to personalize the profile	The system displays the profile tab and opens the file manager to update the profile.	Correct
Create a new message	The system opens the gallery before the image message is sent and encrypted	Correct
Attempting to encrypt a message	The system encrypts the image message using the key and IV	Correct
Storing message data in the database	The system stores the AES key and IV data in the realtime database, and stores the encrypted image and encryption result value in the storage database.	Correct
Send a message	The system sends a specific encrypted image message based on the registered user id.	Correct
Attempt the message decryption process	The system decrypts the encrypted image message using the key and IV used in the previous encryption process.	Correct
Display a message to the end-user	The system displays the decrypted image message again.	Correct
Send message notification	The system sends a notification of the encrypted image message according to the message destination.	Correct
Attempt message notification decryption	The system decrypts the image message notification using the same key and IV	Correct
Display a message notification to the end-user	The system displays a decrypted image message notification	Correct

Based on the tests in Table 3, the results obtained show that the tested system functionality can run well. Testing is done with several scenarios on the main functionality of the application system developed.

Table 4. Brute Force Testing

AES Key	Key Size	Possible Number of Keys	Valid Key
128 bit	16 byte	3.402823669209385E38	0
192 bit	24 byte	6.277101735386681E57	0
256 bit	32 byte	1.157920892373162E77	0

Based on Table 4, the results of system robustness testing are obtained using the brute force attack test. The test case is done by trying all possible AES key lengths and counting the number of possible keys. because the number of possible keys is very large, it is written in scientific notation. Despite finding the number of possible keys, it failed to find a valid key for the encryption process.

IV. CONCLUSION

The conclusion obtained from this research is the application of a combination of the AES 256 algorithm and the CBC method on the BaaS service security system in the chat counseling application can run well. Combining these two methods produces stronger encrypted data because the use of initialization vectors and message authentication are additional security parameters. In addition, based on testing conducted with the blackbox method, it shows that the system functionality has worked properly and in accordance with their respective parameters. Testing the system against brute force attacks also shows strong resistance, so that valid keys cannot be guessed. Further research development can further include the use of double encryption on encrypted data to increase system data security and reduce the risk of data leakage.

REFERENCES

[1] Alimzhanova, Z., Skublewska-Paszkowska, M., & Nazarbayev, D. (2023). Periodicity Detection of the Substitution Box in the CBC Mode of Operation: Experiment and Study. *IEEE Access*, *11*, 75686–75695. https://doi.org/10.1109/ACCESS.2023.3295909

[2] Al-Mashhadani, M., & Shujaa, M. (2022). IoT security using AES encryption technology based ESP32 platform. *Int. Arab J. Inf. Technol.*, *19*(2), 214–223.

[3] Arianto, B., Kurniadi, H., & Kurniasari, I. (2023). IMPLEMENTASI PENGARSIPAN ELEKTRONIK MENGGUNAKAN ENKRIPSI DAN DEKRIPSI DENGAN METODE AES DI UNISKA. *JURNAL FASILKOM*, *13*(02), 259–268. https://doi.org/10.37859/jf.v13i02.5060

[4] Gupta, M., & Sinha, A. (2021). Enhanced-AES encryption mechanism with S-box splitting for wireless sensor networks. *International Journal of Information Technology*, *13*(3), 933–941. https://doi.org/10.1007/s41870-021-00626-w

[5] Hafsa, A., Gafsi, M., Malek, J., & Machhout, M. (2021). FPGA Implementation of Improved Security Approach for Medical Image Encryption and Decryption. *Scientific Programming*, *2021*, 6610655. https://doi.org/10.1155/2021/6610655

[6] Hidayat, A. (2022). Application of the AES Cryptographic Algorithm for E-mail Encryption and Description. *INFOKUM*, *10*(5), 494–500. http://infor.seaninstitute.org/index.php/infokum/article/view/1001

[7] Hidayat, T. (2019). ENCRYPTION SECURITY SHARING DATA CLOUD COMPUTING BY USING AES ALGORITHM: A SYSTEMATIC REVIEW. *TEKNOKOM*, *2*(2), 11–16. https://doi.org/10.31943/teknokom.v2i2.39

[8] K. R., R., Aithal, G., Shetty, S., & K., B. (2020). Image encryption scheme in public key cryptography based on cubic pells quadratic case. *Indonesian Journal of Electrical Engineering and Computer Science*, *20*(1), 385. https://doi.org/10.11591/ijeecs.v20.i1.pp385-394

[9] Kareem, S. M., & Rahma, A. M. S. (2021). New method for improving add round key in the advanced encryption standard algorithm. *Information Security Journal: A Global Perspective*, *30*(6), 371–383. https://doi.org/10.1080/19393555.2020.1859654

[10] Khaliq, K. F. (2021). Pengamanan Data Akta Dengan Metode Aes Berbasis Cloud Computing. *JURNAL TEKNOLOGI DAN ILMU KOMPUTER PRIMA (JUTIKOMP)*, *4*(1), 509–512.

[11] Kumar, K., Ramkumar, K. R., & Kaur, A. (2022). A lightweight AES algorithm implementation for encrypting voice messages using field programmable gate arrays. *Journal of King Saud University - Computer and Information Sciences*, *34*(6), 3878–3885. https://doi.org/10.1016/j.jksuci.2020.08.005

[12] Muttaqin, K., & Rahmadoni, J. (2020). Analysis And Design of File Security System AES (Advanced Encryption Standard) Cryptography Based. *Journal of Applied Engineering and Technological Science (JAETS)*, *1*(2), 113–123. https://doi.org/10.37385/jaets.v1i2.78

[13] Nida, K., & Usiono, U. (2023). Peranan Bimbingan dan Konseling Dalam Pembentukan Karakteristik Siswa. *Jurnal Pendidikan Dan Konseling (JPDK)*, *5*(3), 64–72.

[14] Panagiotou, P., Sklavos, N., Darra, E., & Zaharakis, I. D. (2020). Cryptographic system for data applications, in the context of internet of things. *Microprocessors and Microsystems*, *72*, 102921. https://doi.org/10.1016/j.micpro.2019.102921

[15] Rantelinggi, P. H., & Saputra, E. (2020). Algoritma Kriptografi Triple Des dan Steganografi LSB sebagai Metode Gabungan dalam Keamanan Data. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *7*(4), 661. https://doi.org/10.25126/jtiik.2020741838

[16] Rifki Sadikin. (2012). *Kriptografi untuk Keamanan Jaringan* (Th. Arie Prabawati, Ed.). Andi Publisher .