# THE CREATION OF HICHAT: SECURE MOBILE CHAT USING HYBRID CRYPTOSYSTEM

**Linda Handayani**
*Informatics, Industrial Technology, Gunadarma University, Indonesia*
*Jl. Margonda Raya No. 100, Depok 16424, Jawa Barat*
linda.handa20@gmail.com

## Abstrak

*Keamanan transimi atau pertukaran chat sangat penting dalam penerapan komunikasi. Pesan rahasia dikirim atau ditukar melalui layanan komunikasi di dunia maya  di mana tidak boleh ada pihak lain yang memiliki otoritas selain pengirim dan penerima untuk melindungi dari bahaya. Hal ini menyebabkan perlu adanya sistem kriptografi yang digunakan untuk menjaga keamanan pada saat transmisi atau pertukaran chat berlangsung. Hybrid cryptosystem adalah teknik menggabungkan enkripsi simetris dan asimetris sehingga menciptakan sistem yang lebih aman dan efisien. Pada penelitian ini telah dibuat aplikasi chat berbasis mobile yang aman yaitu Hichat dengan menggabungkan algoritma AES (Advanced Encryption Standard) untuk mengenkripsi chat dan algoritma RSA (Rivest Shamir Adleman) untuk membuat kunci. Fungsi hash juga digunakan untuk otentikasi kunci API (Application Programming Interface) ke layanan server. Tujuan dari penelitian ini adalah membuat aplikasi chat berbasis mobile yang aman dan mengedepankan kerahasiaan, integritas, otentikasi dan non-repudiation. Pengujian fungsional yang dilakukan mendapatkan hasil bahwa semua fitur berjalan dengan baik. Pengujian keamanan dilakukan menggunakan serangan teknik sniffing yang menunjukkan bahwa informasi pesan dan kunci API terenkripsi dan hasil skenario menunjukkan setiap aspek keamanan berhasil. Kemudian tampilan aplikasi responsif di beberapa perangkat.*

***Kata Kunci**: AES, chat, cryptosystem, hybrid, RSA.*

## Abstract

*Security of chat transmission or exchange is very important in the application of communication. Secret messages are sent or exchanged through communication services in cyberspace where no other party should have authority other than the sender and recipient to protect against harm. This causes the need for a cryptographic system that is used to maintain security during the transmission or exchange of chats. Hybrid cryptosystem is a technique that combines symmetric and asymmetric encryption to create a safer and more efficient system. This study has been created of a secure mobile-based chat application, namely Hichat, by combining the AES (Advanced Encryption Standard) algorithm to encrypt chat and the RSA RSA (Rivest Shamir Adleman) algorithm to create keys. The hash function is also used to authenticate API (Application Programming Interface) keys to server services. The purpose of this study is to create a secure mobile-based chat application that prioritizes confidentiality, integrity, authentication and non-repudiation. The functional testing carried out obtained results that all features were running well. Security testing was carried out using a sniffing technique attack which showed that the message information and API keys were encrypted, and the scenario results showed that every aspect of security was successful. Then the application display is responsive on several devices.*

***Keywords**: AES, chat, cryptosystem, hybrid, RSA.*

## INTRODUCTION

Security of chat transmission or exchange is very important in the application of communication. In cyberspace, secret messages are sent or exchanged through communication services, ensuring that only the sender and recipient have authority over the communication to protect against potential threats. According to Ariyus (2008), some information security threats such as interruption, interception, modification and forgery [1]. In terms of security, cryptographic systems have several objectives such as confidentiality, integrity, authentication and non-repudiation, namely preventing a particular party from rejecting an action.

Database security from unauthorized access should not be ignored. Chat applications run on the Android operating system. API (Application Programming Interface) is used to connect Android applications to the database on the Server. Data storage like this has many advantages, especially in terms of data backup, but storing data on the internet will allow data to be hacked and read by other parties. By using two or more different cryptography, hybrid cryptosystem can easily encrypt and decrypt very long messages. A hybrid cryptosystem uses multiple ciphers from various algorithms used to generate symmetric keys and encrypt with asymmetric keys from public keys.

AES (Advanced Encryption Standard) is a symmetric key algorithm published by The National Institute of Standards and Technology (NIST) as a replacement for DES (Data Encryption Standard) [2,3]. The widely used symmetric key algorithm, AES, encrypts and decrypts sensitive data using the same key [4,5]. In the last two decades, researchers have developed many cryptography algorithms. Despite this, AES remains the most secure cryptographic algorithm due to its cost effectiveness and easier implementation in both hardware and software. The AES is categorized as AES-128, AES-192 and AES-256 based on the length of the key. Here, the 128, 192, and 256 indicates the length of the key used during the cryptographic process [6].

RSA (Rivest Shamir Adleman) is an asymmetric key algorithm that uses pair of public key and private key for data encryption and decryption, which enables secure communication over insecure channels [7]. RSA security is based on the complexity of factorizing a large integer that is the product of two large primes, making it difficult to crack without knowing the private key. RSA remains widely used due to its reliability and wide acceptance in various security protocols, such as SSL/TLS for secure internet communications. In addition, RSA is used in digital signatures to ensure message integrity and authentication [8].

In the research of Ashari and Ragin (2016), the RSA-CRT public key cryptography algorithm was implemented in an instant messaging application. The algorithm used is RSA which is included in the asymmetric algorithm. The obstacle that often occurs in the RSA decryption process is the relatively large size of the description key which can slow down the process, so the algorithm can be modified with CRT (Chinese Reminder Theorem) to speed up the decryption process. Implementation of the RSA-CRT cryptographic algorithm in instant messaging applications at n bit lengths from 56 bits to 88 bits, the RSA-CRT decryption process is twice as fast as RSA decryption [9].

Research by Saputra et al. in 2023 discussed the application of hybrid cryptosystem with a combination of RSA cipher algorithms and hash functions to maintain the security of messages sent. The results showed that messages sent could not be hacked by irresponsible people [10]. Research by Siburan et al. in 2023 stated that the use of hybrid cryptosystem can combine AES and RSA for chat encryption. This study uses a literature study method on the use of hybrid cryptosystem using AES and RSA. The integration of AES and RSA algorithms results in a secure hybrid cryptosystem capable of encrypting message texts, enhancing privacy and data confidentiality during transmission [11].

In the research of Akter, et al. (2023) discussed the use of RSA and AES-128 in cloud computing data. AES-128 refers to the key’s length of 128 bits, which offers a balance between security and performances. AES-128 is faster in encryption and decryption processes while maintaining robust security for general use cases in cloud environments. The data to be stored in the cloud is encrypted using AES-128, then the symmetric key is encrypted using the RSA algorithm, this makes the encryption process more secure. In the encryption time test, an average of 295.8774 bytes/ms was produced. The decryption time was not shown in the study and there was no testing of the threat of attacks [12].

Based on the background above, this study uses a hybrid cryptosystem that combines the RSA algorithm to create keys and the AES algorithm to keep messages secret. The Hash function is also used for API key authentication as an access service to the server database. The Hash function is used to ensure that messages remain original and unaltered during transmission. The hash function will generate a unique authentication code by combining the message and the secret key [13]. The FCM (Firebase Cloud Messaging) protocol is used in the system for real-time message exchange and for push notifications. The purpose of this study is to create a secure mobile-based chat application that prioritizes confidentiality, integrity, authentication and non-repudiation.

## METHODS

The prototyping method is used in this study to match user needs with the system to be created [14]. The stages carried out in the prototyping method are identification of needs, analysis and design, creation and testing. These stages are described in the research framework in Figure 1. At the requirement identification, a literature study related to chat application security is carried out. At the functional requirement analysis consisting of facilities needed by users such as features for registering user accounts, logging in, a main page containing conversations and friends and viewing profiles of other users. Non-functional requirement analysis consisting of hardware specification requirements, software used by developers to create mobile-based chat applications and testing. is a hybrid cryptosystem design created based on the results of the functional outcome analysis performed. The design of the hybrid cryptosystem is done. This stage will integrate AES and RSA algorithms to ensure the security of message data, then add a hash function for authentication on the server. Then database design is created to support the management of information that will be stored by the application. This includes table structure, relationships between entities, and efficient data storage mechanisms. Application design contains navigation structure, use case and activity diagram. In the app's creation stage, the developer uses non-functional requirements to support the creation of apps in accordance with the functional requirements analysis and design that has been made. At the testing stage, testing is done to ensure that the application works according to specifications and is free of bugs or errors. This testing also involves evaluating data security, performance, and ease of use.

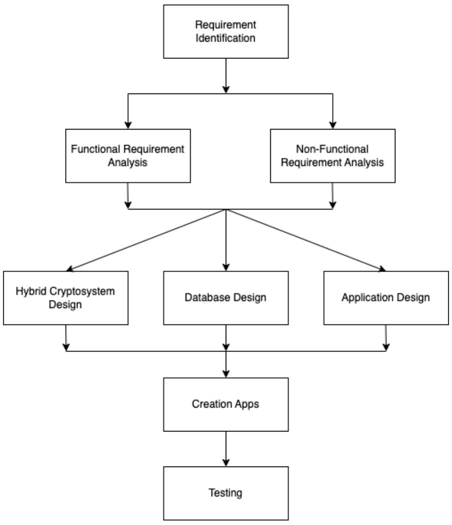

Requirement Identification
Functional Requirement Analysis
Non-Functional Requirement Analysis
Hybrid Cryptosystem Design
Database Design
Application Design
Creation Apps
Testing

Figure 1. Research Framework

**Hybrid Cryptosystem Application Design**

The message transmission process is carried out using the user's application account as authentication. User authentication is encrypted using a hash function. This method is used to secure user data as well as for message transmission and exchange. In the encryption and decryption process using a combination of AES and RSA algorithms. The received plaintext process will be encrypted using the AES algorithm to produce the ciphertext and its key. Then the key in the session will be reprocessed using the RSA algorithm to obtain the key creation in the form of ciphertext. Then the ciphertext results are resampled to obtain the ciphertext which is a combination of the previous ones.

Figure 2. shows the encryption and decryption scheme of the hybrid cryptosystem. The encryption process is done by the sender sending a message in the form of plaintext which will be encrypted using the AES algorithm and will generate the same key for encryption and decryption. The result of encryption is ciphertext. The session key generated from the random generate key in the AES process is used during one communication session and plays an important role in maintaining communication security. The generated session key is then encrypted using the RSA algorithm. RSA is an asymmetric cryptography algorithm, which uses a public key for encryption and a private key for decryption. RSA ensures that only recipients who have the corresponding private key can decrypt the session key. The result of this process is the ciphertext for the session key. The final ciphertext is a combination of the AES ciphertext (encrypted data) and the RSA ciphertext (encrypted session key).

The decryption process starts from the data received in the form of ciphertext from data encryption using the AES algorithm and ciphertext from session key encryption using RSA (with the recipient's public key). The session key used is randomly generated during the previous encryption process. This process ensures that each communication session has a unique level of security. The session key can only be accessed by the recipient through decryption using the RSA private key. Once the session key is obtained, the main ciphertext containing the encrypted data is decrypted using the AES algorithm. AES utilizes the session key to process the decryption so as to produce plaintext in the form of the original data before encryption.

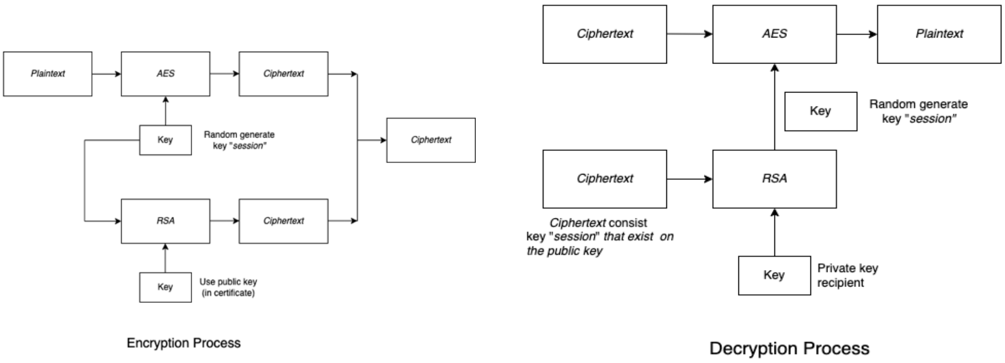

Plaintext
AES
Ciphertext
Key
Random generate key "session"
Ciphertext
RSA
Ciphertext
Key
Use public key (in certificate)
Encryption Process
Ciphertext
AES
Plaintext
Key
Random generate key "session"
Ciphertext
RSA
Ciphertext consist key "session" that exist on the public key
Key
Private key recipient
Decryption Process

Figure 2. Encryption and Decryption Scheme

**Hybrid Cryptosystem Service Protocol Design**

The sender sends a message to the recipient must send a request to be able to communicate with the recipient using an authenticated key generation protocol. The key generation process uses the RSA algorithm to generate and distribute the "session" key. The distribution of the "session" key will be forwarded by the FCM (Firebase Cloud Messaging) protocol using HTTP access and received by the message recipient. When the recipient has a private key and has the same "session" key as the sender, the sender will encrypt the text message using the AES algorithm and send the encryption results to the server and forward the message to be decrypted by the recipient.

Figure 3. show the key exchange protocol, Sender (Q) and Recipient (R) use a trusted Server (S) to distribute asymmetric (public key and private key) on request. These steps of cryptography protocol as follows:

1. The sender (Q) create message $M$ that will be sent to the recipient (R). Message $M$ is encrypted using the AES algorithm with a symmetric key $KQ$ while the encryption result is $C_M$. AES key $KQ$ then encrypted using RSA with the server public key, resulting in $C_{KQ}$. The sender sends a packet $\{C_{M,}C_{KQ}\}$ contain encrypted nessage and key to server S.
2. Server S receive $\{C_{M,}C_{KQ}\}$ from sender and forward to recipient (R).
3. The recipient (R) receives the packet decrypt $C_{KQ}$ and decrypt it using the private key RSA $K_A$ to obtain the *KQ*. With *KQ*, recipient decrypt $C_M$ using AES and obtain the origin message.
4. The recipient makes a reply to *M'* which is encrypted with the symmetric key *KR* with AES algorithm, resulting in $C_{M'}$. *KR* is also encrypted using RSA with the server public key, resulting in $C_{KR}$. The recipient sends $\{C_{M',}C_{KR}\}$ to server and forward to sender.
5. The sender decrypt $C_{KR}$ using private key RSA $K_B$ to get $K_R$. With $K_R$ sender decrypt $C_{M'}$ using AES to get the original reply to *M'*.

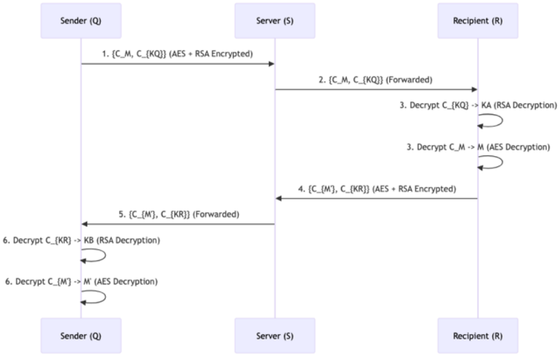

Sender (Q)
Server (S)
Recipient (R)
1. {C_M, C_{KQ}} (AES + RSA Encrypted)
2. {C_M, C_{KQ}} (Forwarded)
3. Decrypt C_{KQ} -> KA (RSA Decryption)
3. Decrypt C_M -> M (AES Decryption)
4. {C_{M'}, C_{KR}} (AES + RSA Encrypted)
5. {C_{M'}, C_{KR}} (Forwarded)
6. Decrypt C_{KR} -> KB (RSA Decryption)
6. Decrypt C_{M'} -> M' (AES Decryption)
Sender (Q)
Server (S)
Recipient (R)

Figure 3. Sequence Diagram Related to the Key Exchange Protocol

**Database Design**

The database design is shown in Figure 4. Class diagram describes the relationship between classes in the application, which contains the class name, attributes, and operations performed. There are seven interconnected classes, including the Account class, which will be used for class A, namely the ongoing transaction log, then manage the Friend class, which contains the AddFriend and InvitationAdapter activities and View ProfileActivity. The Friend class will use class A for transaction logs and manage the Conversation class, which is used for message exchange. The Conversation class will connect to a service that will later perform the data security process from the Cryptography class. Every transaction made in class A, which contains a transaction log, also contains access information from the API class.

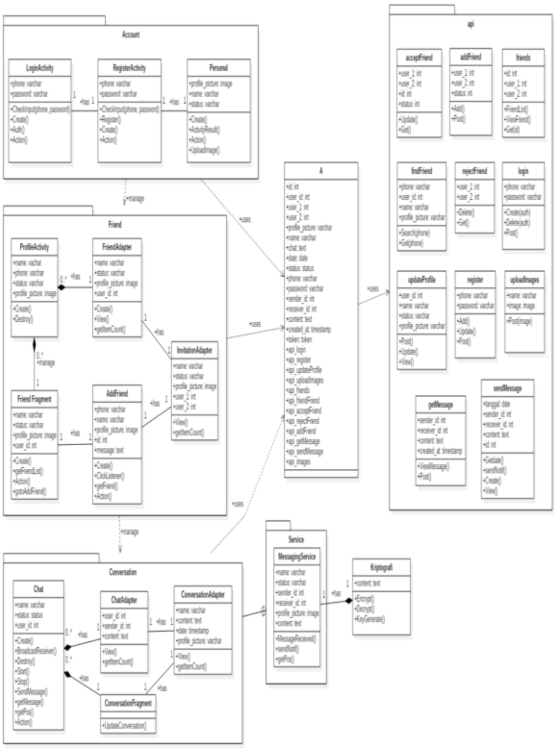


Figure 4. Class Diagram

## Application Design

The design of the interface display consists of the application navigation structure, use case diagram, activity diagram and class diagram. The application navigation structure describes the flow in building the application. The type of navigation structure used is a mixed navigation structure (linear and hierarchical) shown in Figure 5.

Use case diagram describes the user interaction in the application shown in Figure 6. Activity diagram describes the flow of activity control in the application. The user enters the mobile phone number and password, then the system will check the validity of the user account. If the account is verified, the user can select the information that needs to be managed. If the user account has not been registered, the user must register an account first.

Figure 7 shows an activity diagram which is used to model the workflow or activities in a system. The features in the application are adding, updating, searching and deleting messages and friends' contacts.

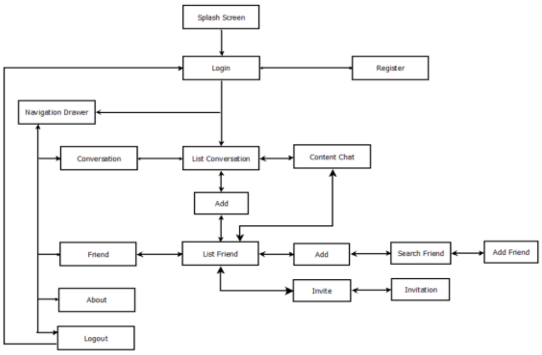


Figure 5. Application Navigation Structure

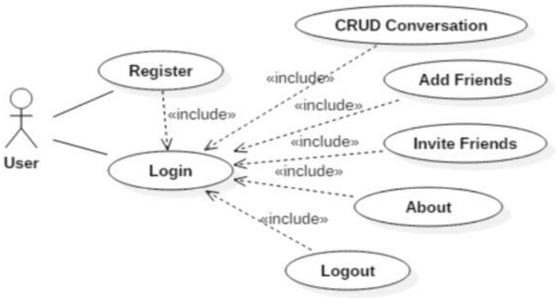


Figure 6. Use Case Diagram

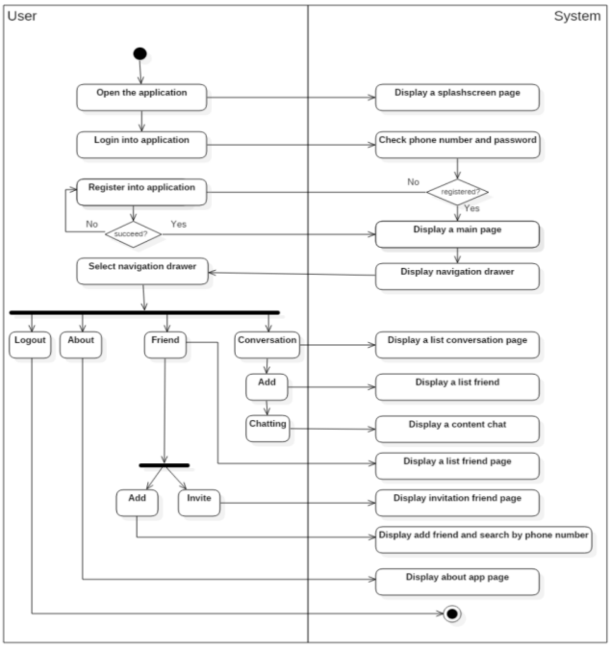

User
System
Open the application
Display a splashscreen page
Login into application
Check phone number and password
No
registered?
Yes
Register into application
No
succeed?
Yes
Display a main page
Select navigation drawer
Display navigation drawer
Logout
About
Friend
Conversation
Display a list conversation page
Add
Display a list friend
Chatting
Display a content chat
Display a list friend page
Add
Invite
Display invitation friend page
Display add friend and search by phone number
Display about app page

Figure 7. Activity Diagram

## RESULTS AND DISCUSSION

The results of the Hichat application creation display nine pages features including splash screen, login, register, navigation drawer, friend, add friend, profile page, conversation and content chat. When the user runs Hichat application, the first page that will be displayed is the splash screen page (Figure 8). After the splash screen, the user will be directed to the login page (Figure 9). The user needs to type in the cellphone number and password then click the submit button. Then the system checks and confirms the user account. If the login is successful, the user can access and manage services and features on the main page of the apps.

On the Friend page (Figure 10), users must press the add button to find friends, so they can have a conversation. Other users (friendship invitation recipients) will then take actions such as Accept, Delete or Cancel on the Friend page. After the user accepts the friendship, the user can immediately chat in the conversation feature. The conversation page (Figure 11) displays a list of users who are chatting. If not chatting, then the user can click the link to start a chat "Click to start a chat". Then the user will be directed to the chat content page. The chat content page displays the contents of the chat conducted by the sender and user. The chat conducted can only be seen by the sender and recipient because the chat will be encrypted and decrypted so that it cannot be read by third parties. The application will provide a notification if there is a chat entering the recipient's keypad. The chatting process occurs in real time and there is information about the time the message was sent and received.

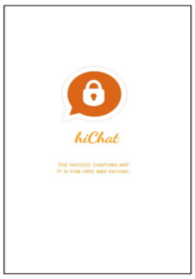


Figure 8. Splash Screen

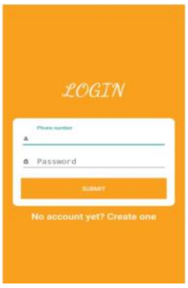
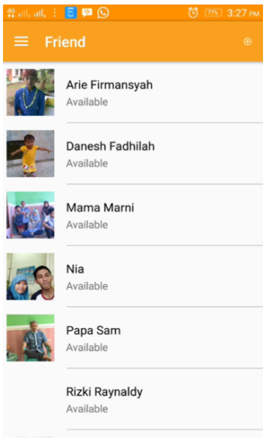


Figure 9. Login



Arie Firmansyah
Available

Danesh Fadhilah
Available

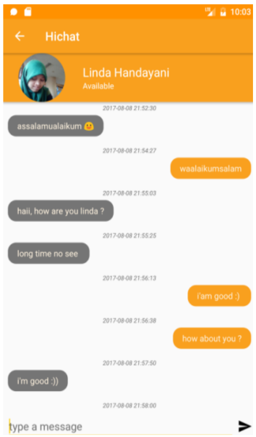
Mama Marni
Available

Nia
Available

Papa Sam
Available

Rizki Raynaldy
Available

Figure 10. Friend Page



Linda Handayani
Available

2017-08-08 21:52:30
assalamualaikum 😃

2017-08-08 21:54:27
waalaikumsalam

2017-08-08 21:55:03
haii, how are you linda ?

2017-08-08 21:55:25
long time no see

2017-08-08 21:56:13
i'am good :)

2017-08-08 21:56:38
how about you ?

2017-08-08 21:57:50
i'm good :))

2017-08-08 21:58:00

type a message

Figure 11. Conversation

Security testing using sniffing is a technique in network security used to monitor and analyse data traffic passing through a computer network [15] with wireshark tools. The process of this technique is to capture network traffic to see data or chats sent by users. Previously, researchers designed the topology used to capture the data shown in Figure 12. The sniffing technique is carried out by capturing the data sending process carried out by the cellphone via the wifi interface in the wireshark tool. The protocol to be taken is the HTTP protocol. The results of the data sending process in the form of a packet listing window from sending messages and receiving messages are shown in Figures 13, 14, 15, 16. The test scenario is used to prove the security aspects of the system shown in Figure 17 and Table 1.

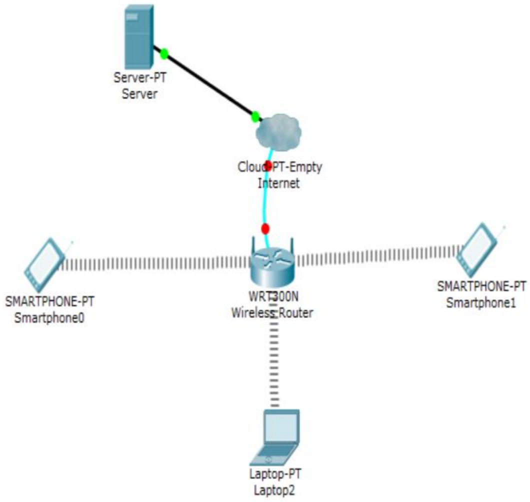
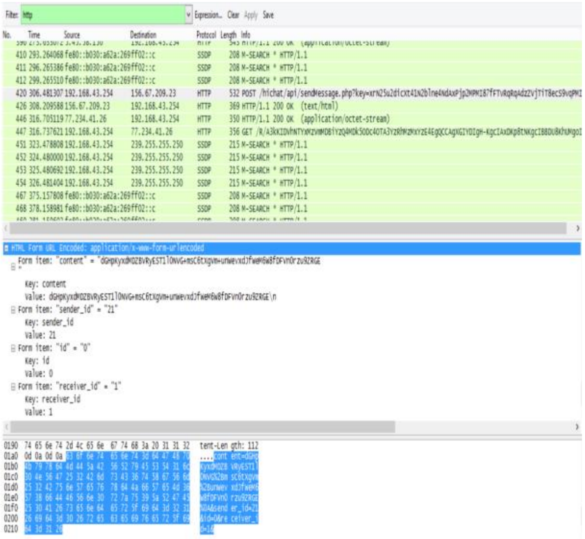


Figure 12. Application Testing Topology

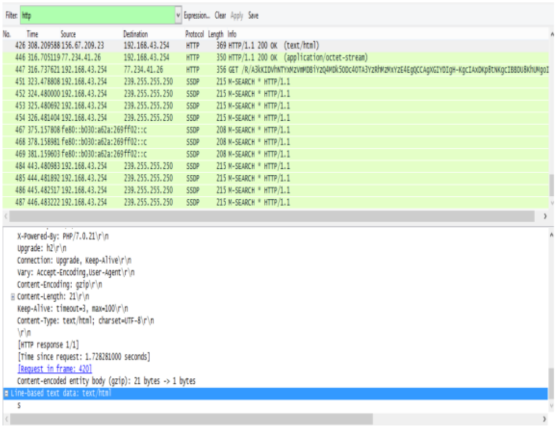


```
No.  Time       Source                       Destination      Protocol Length Info
410 293.264068 fe80::b030:a62a:269ff02::c                     SSDP     208 M-SEARCH * HTTP/1.1
411 296.265386 fe80::b030:a62a:269ff02::c                     SSDP     208 M-SEARCH * HTTP/1.1
412 299.265510 fe80::b030:a62a:269ff02::c                     SSDP     208 M-SEARCH * HTTP/1.1
420 306.481307 192.168.43.254  156.67.209.23                  HTTP     532 POST /hichat/api/sendMessage.php?key=xrN25u2d1cXt41N2bIne4NdAxPjp2MPM187fFTVRqRqAdzZVjTiT8ecS9vqPMI
426 308.209588 156.67.209.23   192.168.43.254                 HTTP     369 HTTP/1.1 200 OK  (text/html)
446 316.705119 77.234.41.26    192.168.43.254                 HTTP     350 HTTP/1.1 200 OK  (application/octet-stream)
447 316.737621 192.168.43.254  77.234.41.26                   HTTP     356 GET /R/A3kKIDVhNTYxMzVmMDBiYzQ4MDk5ODc4OTA3YzRhM2MxYzE4EgQCCAgXGIYDIgH-KgcIAxDKp8tNKgcIB8DU8KhUMgoI
451 323.478808 192.168.43.254  239.255.255.250                SSDP     215 M-SEARCH * HTTP/1.1
452 324.480000 192.168.43.254  239.255.255.250                SSDP     215 M-SEARCH * HTTP/1.1
453 325.480692 192.168.43.254  239.255.255.250                SSDP     215 M-SEARCH * HTTP/1.1
454 326.481404 192.168.43.254  239.255.255.250                SSDP     215 M-SEARCH * HTTP/1.1
467 375.157808 fe80::b030:a62a:269ff02::c                     SSDP     208 M-SEARCH * HTTP/1.1
468 378.158981 fe80::b030:a62a:269ff02::c                     SSDP     208 M-SEARCH * HTTP/1.1

⊟ HTML Form URL Encoded: application/x-www-form-urlencoded
 ⊟ Form item: "content" = "dGHpKyxdMDZBVRyEST1lONVG+msC6tXgVm+unWevxdJfweM6W8fDFVn0rzu9ZRGE
   "
    Key: content
    Value: dGHpKyxdMDZBVRyEST1lONVG+msC6tXgVm+unWevxdJfweM6W8fDFVn0rzu9ZRGE\n
 ⊟ Form item: "sender_id" = "21"
    Key: sender_id
    Value: 21
 ⊟ Form item: "id" = "0"
    Key: id
    Value: 0
 ⊟ Form item: "receiver_id" = "1"
    Key: receiver_id
    Value: 1

0190  74 65 6e 74 2d 4c 65 6e  67 74 68 3a 20 31 31 32   tent-Len gth: 112
01a0  0d 0a 0d 0a 63 6f 6e 74  65 6e 74 3d 64 47 48 70   ....cont ent=dGHp
01b0  4b 79 78 64 4d 44 5a 42  56 52 79 45 53 54 31 6c   KyxdMDZB VRyEST1l
01c0  30 4e 56 47 25 32 42 6d  73 43 36 74 58 67 56 6d   0NVG%2Bm sC6tXgVm
01d0  25 32 42 75 6e 57 65 76  78 64 4a 66 57 65 4d 36   %2Bunwev xdJfweM6
01e0  57 38 66 44 46 56 6e 30  72 7a 75 39 5a 52 47 45   W8fDFVn0 rzu9ZRGE
01f0  25 30 41 26 73 65 6e 64  65 72 5f 69 64 3d 32 31   %0A&send er_id=21
0200  26 69 64 3d 30 26 72 65  63 65 69 76 65 72 5f 69   &id=0&re ceiver_i
0210  64 3d 31 26                                        d=1&
```

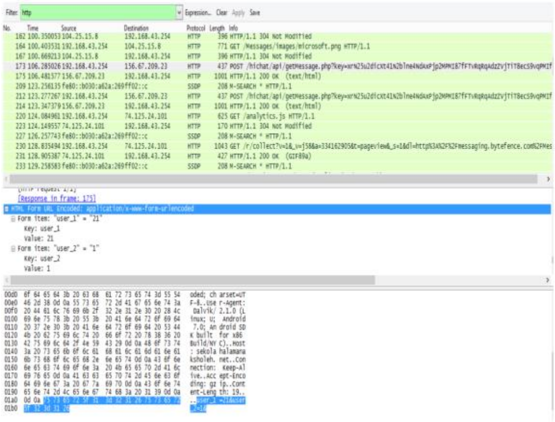
Figure 13. Send Message (Apps to Server)



```
No.  Time       Source                       Destination      Protocol Length Info
426 308.209588 156.67.209.23   192.168.43.254                 HTTP     369 HTTP/1.1 200 OK  (text/html)
446 316.705119 77.234.41.26    192.168.43.254                 HTTP     350 HTTP/1.1 200 OK  (application/octet-stream)
447 316.737621 192.168.43.254  77.234.41.26                   HTTP     356 GET /R/A3kKIDVhNTYxMzVmMDBiYzQ4MDk5ODc4OTA3YzRhM2MxYzE4EgQCCAgXGIYDIgH-KgcIAxDKp8tNKgcIB8DU8KhUMgoI
451 323.478808 192.168.43.254  239.255.255.250                SSDP     215 M-SEARCH * HTTP/1.1
452 324.480000 192.168.43.254  239.255.255.250                SSDP     215 M-SEARCH * HTTP/1.1
453 325.480692 192.168.43.254  239.255.255.250                SSDP     215 M-SEARCH * HTTP/1.1
454 326.481404 192.168.43.254  239.255.255.250                SSDP     215 M-SEARCH * HTTP/1.1
467 375.157808 fe80::b030:a62a:269ff02::c                     SSDP     208 M-SEARCH * HTTP/1.1
468 378.158981 fe80::b030:a62a:269ff02::c                     SSDP     208 M-SEARCH * HTTP/1.1
469 381.159603 fe80::b030:a62a:269ff02::c                     SSDP     208 M-SEARCH * HTTP/1.1
484 443.480983 192.168.43.254  239.255.255.250                SSDP     215 M-SEARCH * HTTP/1.1
485 444.481892 192.168.43.254  239.255.255.250                SSDP     215 M-SEARCH * HTTP/1.1
486 445.482517 192.168.43.254  239.255.255.250                SSDP     215 M-SEARCH * HTTP/1.1
487 446.483222 192.168.43.254  239.255.255.250                SSDP     215 M-SEARCH * HTTP/1.1

    X-Powered-By: PHP/7.0.21\r\n
    Upgrade: h2\r\n
    Connection: Upgrade, Keep-Alive\r\n
    Vary: Accept-Encoding,User-Agent\r\n
    Content-Encoding: gzip\r\n
  ⊞ Content-Length: 21\r\n
    Keep-Alive: timeout=3, max=100\r\n
    Content-Type: text/html; charset=UTF-8\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 1.728281000 seconds]
    [Request in frame: 420]
    Content-encoded entity body (gzip): 21 bytes -> 1 bytes
⊟ Line-based text data: text/html
    5
```

Figure 14. Send Message (Server to Apps)

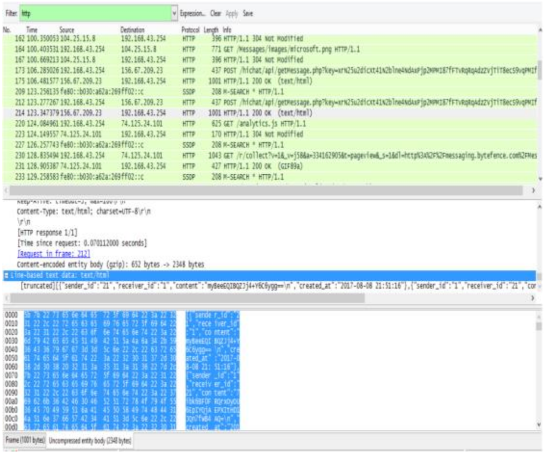


```
No.  Time        Source                      Destination     Protocol Length Info
162 100.350053 104.25.15.8                  192.168.43.254  HTTP   396 HTTP/1.1 304 Not Modified
164 100.403531 192.168.43.254               104.25.15.8     HTTP   771 GET /Messages/images/microsoft.png HTTP/1.1
167 100.669213 104.25.15.8                  192.168.43.254  HTTP   396 HTTP/1.1 304 Not Modified
173 106.285026 192.168.43.254               156.67.209.23   HTTP   437 POST /hichat/api/getMessage.php?key=xrN25u2dicXt41N2blne4NdAxPjp2MPMI87fFTvRqRqAdzZVjTiT8ecS9vqPMIf
175 106.481577 156.67.209.23                192.168.43.254  HTTP  1001 HTTP/1.1 200 OK  (text/html)
209 123.256135 fe80::b030:a62a:269ff02::c                   SSDP   208 M-SEARCH * HTTP/1.1
212 123.277267 192.168.43.254               156.67.209.23   HTTP   437 POST /hichat/api/getMessage.php?key=xrN25u2dicXt41N2blne4NdAxPjp2MPMI87fFTvRqRqAdzZVjTiT8ecS9vqPMIf
214 123.347379 156.67.209.23                192.168.43.254  HTTP  1001 HTTP/1.1 200 OK  (text/html)
220 124.084961 192.168.43.254               74.125.24.101   HTTP   625 GET /analytics.js HTTP/1.1
223 124.149557 74.125.24.101                192.168.43.254  HTTP   170 HTTP/1.1 304 Not Modified
227 126.257743 fe80::b030:a62a:269ff02::c                   SSDP   208 M-SEARCH * HTTP/1.1
230 128.835494 192.168.43.254               74.125.24.101   HTTP  1043 GET /r/collect?v=1&_v=j58&a=334162905&t=pageview&_s=1&dl=http%3A%2F%2Fmessaging.bytefence.com%2FMes
231 128.905387 74.125.24.101                192.168.43.254  HTTP   427 HTTP/1.1 200 OK  (GIF89a)
233 129.258583 fe80::b030:a62a:269ff02::c                   SSDP   208 M-SEARCH * HTTP/1.1

    [Response in frame: 175]
⊟ HTML Form URL Encoded: application/x-www-form-urlencoded
  ⊟ Form item: "user_1" = "21"
      Key: user_1
      Value: 21
  ⊟ Form item: "user_2" = "1"
      Key: user_2
      Value: 1

00d0  6f 64 65 64 3b 20 63 68  61 72 73 65 74 3d 55 54   oded; ch arset=UT
00e0  46 2d 38 0d 0a 55 73 65  72 2d 41 67 65 6e 74 3a   F-8..Use r-Agent:
00f0  20 44 61 6c 76 69 6b 2f  32 2e 31 2e 30 20 28 4c    Dalvik/ 2.1.0 (L
0100  69 6e 75 78 3b 20 55 3b  20 41 6e 64 72 6f 69 64   inux; U;  Android
0110  20 37 2e 30 3b 20 41 6e  64 72 6f 69 64 20 53 44    7.0; An droid SD
0120  4b 20 62 75 69 6c 74 20  66 6f 72 20 78 38 36 20   K built  for x86 
0130  42 75 69 6c 64 2f 4e 59  43 29 0d 0a 48 6f 73 74   Build/NY C)..Host
0140  3a 20 73 65 6b 6f 6c 61  68 61 6c 61 6d 61 6e 61   : sekola halamana
0150  6b 73 68 6f 6c 65 68 2e  6e 65 74 0d 0a 43 6f 6e   ksholeh. net..Con
0160  6e 65 63 74 69 6f 6e 3a  20 4b 65 65 70 2d 41 6c   nection:  Keep-Al
0170  69 76 65 0d 0a 41 63 63  65 70 74 2d 45 6e 63 6f   ive..Acc ept-Enco
0180  64 69 6e 67 3a 20 67 7a  69 70 0d 0a 43 6f 6e 74   ding: gz ip..Cont
0190  65 6e 74 2d 4c 65 6e 67  74 68 3a 20 31 39 0d 0a   ent-Leng th: 19..
01a0  0d 0a 75 73 65 72 5f 31  3d 32 31 26 75 73 65 72   ..user_1 =21&user
01b0  5f 32 3d 31 26                                     _2=1&
```

Figure 15. Message Information (Apps to Server)



```
No.  Time        Source                      Destination     Protocol Length Info
162 100.350053 104.25.15.8                  192.168.43.254  HTTP   396 HTTP/1.1 304 Not Modified
164 100.403531 192.168.43.254               104.25.15.8     HTTP   771 GET /Messages/images/microsoft.png HTTP/1.1
167 100.669213 104.25.15.8                  192.168.43.254  HTTP   396 HTTP/1.1 304 Not Modified
173 106.285026 192.168.43.254               156.67.209.23   HTTP   437 POST /hichat/api/getMessage.php?key=xrN25u2dicXt41N2blne4NdAxPjp2MPMI87fFTvRqRqAdzZVjTiT8ecS9vqPMIf
175 106.481577 156.67.209.23                192.168.43.254  HTTP  1001 HTTP/1.1 200 OK  (text/html)
209 123.256135 fe80::b030:a62a:269ff02::c                   SSDP   208 M-SEARCH * HTTP/1.1
212 123.277267 192.168.43.254               156.67.209.23   HTTP   437 POST /hichat/api/getMessage.php?key=xrN25u2dicXt41N2blne4NdAxPjp2MPMI87fFTvRqRqAdzZVjTiT8ecS9vqPMIf
214 123.347379 156.67.209.23                192.168.43.254  HTTP  1001 HTTP/1.1 200 OK  (text/html)
220 124.084961 192.168.43.254               74.125.24.101   HTTP   625 GET /analytics.js HTTP/1.1
223 124.149557 74.125.24.101                192.168.43.254  HTTP   170 HTTP/1.1 304 Not Modified
227 126.257743 fe80::b030:a62a:269ff02::c                   SSDP   208 M-SEARCH * HTTP/1.1
230 128.835494 192.168.43.254               74.125.24.101   HTTP  1043 GET /r/collect?v=1&_v=j58&a=334162905&t=pageview&_s=1&dl=http%3A%2F%2Fmessaging.bytefence.com%2FMes
231 128.905387 74.125.24.101                192.168.43.254  HTTP   427 HTTP/1.1 200 OK  (GIF89a)
233 129.258583 fe80::b030:a62a:269ff02::c                   SSDP   208 M-SEARCH * HTTP/1.1

    Content-Type: text/html; charset=UTF-8\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.070112000 seconds]
    [Request in frame: 212]
    Content-encoded entity body (gzip): 652 bytes -> 2348 bytes
⊟ Line-based text data: text/html
    [truncated][{"sender_id":"21","receiver_id":"1","content":"myBeeEQIBQZJj4+Y6C6ygg==\n","created_at":"2017-08-08 21:51:16"},{"sender_id":"1","receiver_id":"21","cor

0000  5b 7b 22 73 65 6e 64 65  72 5f 69 64 22 3a 22 32   [{"sende r_id":"2
0010  31 22 2c 22 72 65 63 65  69 76 65 72 5f 69 64 22   1","rece iver_id"
0020  3a 22 31 22 2c 22 63 6f  6e 74 65 6e 74 22 3a 22   :"1","co ntent":"
0030  6d 79 42 65 65 45 51 49  42 51 5a 4a 6a 34 2b 59   myBeeEQI BQZJj4+Y
0040  36 43 36 79 67 67 3d 3d  5c 6e 22 2c 22 63 72 65   6C6ygg== \n","cre
0050  61 74 65 64 5f 61 74 22  3a 22 32 30 31 37 2d 30   ated_at" :"2017-0
0060  38 2d 30 38 20 32 31 3a  35 31 3a 31 36 22 7d 2c   8-08 21: 51:16"},
0070  7b 22 73 65 6e 64 65 72  5f 69 64 22 3a 22 31 22   {"sender _id":"1"
0080  2c 22 72 65 63 65 69 76  65 72 5f 69 64 22 3a 22   ,"receiv er_id":"
0090  32 31 22 2c 22 63 6f 6e  74 65 6e 74 22 3a 22 37   21","con tent":"7
00a0  69 62 6b 36 42 46 30 46  52 51 72 78 4f 79 4f 55   ibk6BF0F RQrxOyOU
00b0  36 45 70 49 59 51 6a 41  45 50 58 49 74 48 44 31   6EpIYQjA EPXItHD1
00c0  4a 51 6e 37 66 57 42 34  41 51 3d 5c 6e 22 2c 22   JQn7fWB4 AQ=\n","
00d0  63 72 65 61 74 65 64 5f  61 74 22 3a 22 32 30 31   created_ at":"201
```

Figure 16. Message Information (Server to Apps)

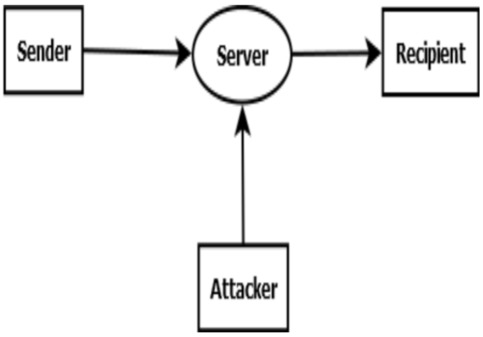


Figure 17. Security Testing Scenario Concept

Table 1. Security Testing Scenarios

| Security Aspect | Scenario Testing | Result |
|---|---|---|
| Confidentiality | The sender sends a message to the recipient via server. The attacker tries to see the contents of the message sent by the sender to the recipient via tools. | Messages stored on the server contain ciphertext and cannot be read by the attacker. |
| Integrity | The attacker tries to intercept the traffic of the message sent via tools, so that it can change the message sent by the sender to the recipient. | Traffic through the attacker via tools in the system has used an encrypted key id before entering the server. So, the attacker can't change the message that was received by the recipient. |
| Authentication | The attacker tries to steal user data in the form of a username and password, in order to authenticate the login with the user data. | User data when register will be protected with a system key id and the password entered will be encrypted. So that the data stored on the server is a chipper text. |
| Nonrepudiation | Every user activity carried out in the application system will be recorded in service. | The system can do user logging that the activity being carried out can't be manipulated by the attacker. |

The third test was conducted to see the responsiveness of the application display on several devices, there are five cell phones with different specifications that will display the results of the interface, resolution, performance time and chat content. The test results are shown in Table 2.

Table 2. Device Testing

| No | Mobile | Specification | Result |
|---|---|---|---|
| 1 | Lenovo A7000plus | Resolution: 1080 x 1920p (~401 ppi)<br>Screen: 5.5 inch<br>OS: Android 5.1<br>Processor: Octa-core 1.7 GHz Cortex-A53<br>RAM: 2 GB | Interface: according to the resolution and screen mobile (no stacked components)<br>Performance: real time<br>Message: corresponding |
| 2 | Oppo A39 | Resolution: 720 x 1280p (~267 ppi)<br>Screen: 5.2 inch<br>OS: Android 5.1<br>Processor: Octa-core 1.5 GHz Cortex-A53<br>RAM: 3 GB | Interface: according to the resolution and screen mobile (no stacked components)<br>Performance: real time<br>Message: corresponding |
| 3 | Himax Pure III | Resolution: 720 x 1280p<br>Screen: 4.7 inch<br>OS: Android 4.2.2<br>Processor: Octa Core MT6592 1,7 GHz<br>RAM: 1 GB | Interface: according to the resolution and screen mobile (no stacked components)<br>Performance: real time<br>Message: corresponding |
| 4. | IMO S79 | Resolution: 320 x 480p<br>Screen: 3.5 inch<br>OS: Android 2.3.6<br>Processor: MTK6575M, Cortex A9 1 GHz<br>RAM: 512 MB | Interface: according to the resolution and screen mobile (no stacked components)<br>Performance: real time<br>Message: corresponding |

|  |  |  |  |
|---|---|---|---|
| 5. | MITO T99 | Resolution: 1024 x 600p<br>Screen: 7 inch<br>OS: Android 5.1 Lollipop<br>Processor: Quad Core 1.2 GHz Cortex A-7<br>RAM: 512 MB | Interface: according to the resolution and screen mobile (no stacked components)<br>Performance: real time<br>Message: corresponding |

## CONCLUSION

Based on the results and discussions that have been carried out, the creation of the Hichat application, a secure mobile-based chat using a hybrid cryptosystem has been successful. Based on the results of functional testing, all features run and are in accordance with the design and functional requirements. In security testing using sniffing techniques in the data sending process, it shows that the security of the sender's data and recipient's data is indicated by the id value and content containing the ciphertext resulting from the encryption process. The results show that data with the same value (plaintext) has a different ciphertext value. This proves that the Hichat application has met the security criteria including Confidentiality, Integrity, Authentication and Non-repudiation. Then in the third test, it was found that testing with several different devices showed the display, resolution according to the response time, namely real time and the messages sent and received were appropriate. For further development, it is expected that brute force testing can be used to determine the reliability of the hybrid cryptosystem used and UI/UX development can be used in this application.

## REFERENCES

[1] D. Ariyus, *Pengantar Ilmu KRIPTOGRAFI Teori, Analisis, dan Implementasi*, Yogyakarta, Indonesia: Andi, 2008.

[2] A. M. Abdulazeez and A. S. Tahir, "Design and Implementation of Advanced Encryption Standard Security Algorithm using FPGA," *International Journal of Computer Technology*, vol. 4, pp. 1988–1993, 2013.

[3] H. Mestiri, F. Kahri, B. Bouallegue, and M. Machhout, "A high-speed AES design resistant to fault injection attacks," *Microprocessors and Microsystems*, vol. 41, pp. 47–55, 2016, doi: 10.1016/j.micpro.2015.12.002.

[4] K. Shahbazi and S. B. Ko, "Area-Efficient Nano-AES Implementation for Internet-of-Things Devices," *IEEE Transactions on Very Large Scale*

*Integration (VLSI) Systems*, vol. 29, pp. 136–148, 2021, doi: 10.1109/TVLSI.2020.3033928.

[5] D. S. Kundi, A. Aziz, and N. Ikram, "A high performance ST-Box based unified AES encryption/decryption architecture on FPGA," *Microprocessors and Microsystems*, vol. 41, pp. 37–46, 2016, doi: 10.1016/j.micpro.2015.11.015.

[6] National Institute of Standards and Technology (NIST), "Announcing the Advanced Encryption Standard (AES)," FIPS PUB 197, Nov. 26, 2001. Updated May 9, 2023. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf. [Accessed: Dec. 27, 2024].

[7] H. T. Sihotang, S. Efendi, E. M. Elvyawati, Z. Zamzami, and H. Mawengkang, "Design and implementation of Rivest Shamir Adleman’s (RSA) cryptography algorithm in text file data security," in *Journal of Physics: Conference Series*, vol. 1641, no. 1, p. 012042, 2020, doi: 10.1088/1742-6596/1641/1/012042.

[8] M. S. A. Mohammad, R. Din, and J. I. Ahmad, "Research trends review on RSA scheme of asymmetric cryptography techniques," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 1, pp. 487–492, Feb. 2021, doi: 10.11591/eei.v10i1.2493.

[9] A. Ashari and S. Ragil, "Implementasi Kriptografi Kunci Publik dengan Algoritma RSA-CRT pada Aplikasi Instant Messaging," *Scientific Journal of Informatics*, vol. 3, no. 1, 2016.

[10] M. W. Saputra, A. Sapitri, and M. A. Putri, "Penerapan Kriptosistem Hybrid untuk Mengenkripsi Pesan Menggunakan Algoritma RSA Cipher," *Jurnal JOCOTIS – Journal Science Informatica and Robotics*, vol. 1, no. 1, pp. 10–21, Sep. 2023.

[11] S. R. Siburian, P. Sultan, R. A. S. Sinaga, and F. Yudistira, "Kriptosistem Hybrid Menggunakan Kombinasi AES dan RSA Untuk Enkripsi Teks Pesan," *Jurnal JOCOTIS – Journal Science Informatica and Robotics*, vol. 1, no. 1, pp. 22–31, Sep. 2023.

[12] R. Akter, "RSA and AES Based Hybrid Encryption Technique for Enhancing Data Security in Cloud Computing," *International Journal of Computational and Applied Mathematics & Computer Science*, 2023, doi: 10.37394/232028.2023.3.8.

[13] L. Chi and X. Zhu, “Hashing Techniques: A Survey and Taxonomy,” *ACM Computing Surveys (CSUR)*, vol. 50, no. 11, pp. 1-36, Apr. 2017, doi: 10.1145/3047307.

[14] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 10th ed. New York, NY, USA: McGraw-Hill, 2020.

[15] S. Chengwai, W. Quanhong, W. Zhenjun, and Y. Xiaoyi, “Research and demonstration of measuring and evaluation system of electronic resources relying on sniffer,” in *Proc. 1st Int. Conf. E-commerce, E-Business and E-Governance (ICEEG ’17)*, 2017, doi: 10.1145/3108421.3108439.