Ju Ti : Indonesian Engineering Journal
ISSN : 2964-0490 (online)
Vol. 2, no. 01, 2023, pp. 1-7
https://jurnal.seaninstitute.or.id/index.php/juti

Security Analysis of Diffie-Hellman Algorithm in Cryptographic Key Exchange

Jeni Nova Situmoran[1], Eunike Nainggolan[2], Adelwin Saputra Loi[3], Andreas Wendi Hutablian[4], Anju Franjein Hutasoit[5]
[1]Faculty of Computer Science, Catholic University of Saint Thomas, Jalan Setia Budi No.479, Tanjung Sari Medan, Indonesia

Article Info

Keywords:
Cryptography,
Diffie-Helman,
Key Exchange,
Encryption,
Description.

ABSTRACT

The Diffie-Hellman algorithm is a cryptographic method used for secure key exchange between two parties over an insecure channel. This study aims to analyze the implementation of the Diffie-Hellman algorithm, focusing on its security and efficiency aspects. The research process begins with manual step-by-step calculations of this algorithm, including prime number selection, modulus calculation, and key exchange. Furthermore, key encryption and decryption tests are carried out to assess the algorithm's resilience to threats. Although this algorithm has proven effective in protecting data, the efficiency of the encryption and decryption processes can be affected by the size of the key used. The test results show that Diffie-Hellman can maintain the confidentiality of communications well, although its processing speed may be affected in systems with large-scale requirements. This study concludes that the Diffie-Hellman algorithm is suitable for use in communications that require a high level of security, but its efficiency needs to be optimized by considering factors such as key size and algorithm optimization. These findings provide important insights for the development of cryptography-based security systems for modern communication applications.

Corresponding Author:
Jeni Nova Situmorang,
Faculty of Computer Science,
Santo Thomas Catholic University Medan,
Jl. Medan-Binjai km 13.5.
Jenisitumorang6@gmail.com

1. INTRODUCTION

The need for data security is becoming very crucial in the increasingly developing digital era. One method used to maintain the confidentiality of information is by using cryptography. The Diffie-Hellman algorithm is a cryptographic key exchange method that allows two parties to create a shared secret key over a public channel without having to exchange any prior secret information.(Saepulrohman & Negara, 2021).

The Diffie-Hellman algorithm works by exploiting the mathematical properties of the discrete logarithm. The process involves choosing a large prime number pp and a base gg, which is then used by both parties to generate a shared secret key through a modulo exponentiation operation. While this process is theoretically secure, its practical implementation can be vulnerable to attack if parameters such as pp and gg are not chosen carefully1. For example, research has shown that many internet applications use weak parameters that are vulnerable to brute force or man-in-the-middle attacks.(Gunardi, nd).

The most important aspect is data security. Therefore, it is important to protect communication and information exchange through vulnerable channels when there is a possibility of attack by third parties.(Purwanto & Informatics, nd). The technique for protecting data sent over a network is cryptography, which is currently very developed, becoming one of the most widely implemented

solutions.(Thahara & Siregar, 2021). There are many different cryptographic algorithms in the world, from simple to complex and very reliable, developed to ensure that no one can decrypt the data except the authorized sender and recipient.(Pratiwi et al., 2022).

Caesar Cipher, one of the most popular classical cryptography methods, changes each letter in a message to another letter located at a specific position in the alphabet based on an encryption key.(Febrianingsih et al., 2019). Although easy to use, the Caesar Cipher has a significant weakness: the encryption key remains static and is easy to guess when an attacker tries all possible key combinations to obtain the hidden message.(Limbong et al., 2017). With the increasing threats and technological advances, this method is no longer suitable for communication needs that require a higher level of security. One way to use the Diffie-Hellman Key Exchange algorithm as a key exchange mechanism can be used to overcome the weaknesses of Caesar Cipher. This algorithm allows two people to generate a shared secret key without having to send the key directly over a less secure communication medium, and can increase communication security.(Wahyuni, 2011). By combining the Diffie-Hellman algorithm with the Caesar Cipher, it is expected that there will be an increase in communication security.

Various studies have been conducted to analyze the security of the Diffie-Hellman algorithm. One study found that the combination of the Diffie-Hellman algorithm with other cryptographic methods, such as RSA, can increase the level of security in the key exchange process. In addition, other studies have shown that the use of larger parameters and additional randomization techniques can strengthen this algorithm against statistical and computational attacks.(7.Ahmad_, nd). However, despite various efforts to improve the security of the Diffie-Hellman algorithm, challenges remain in overcoming computational limitations and threats from malicious actors with large resources.

In this study, we use the Python programming language to create a program that implements the Diffie-Hellman algorithm in the key exchange process, which is then used for encryption and decryption using the Caesar Cipher method. The purpose of this program is to test how well the combination of the two algorithms improves communication security. Python allows for automation of the key exchange calculation process and then encryption and decryption which allows for simulation in security. With this implementation, further analysis of the strength of the encryption and decryption created and its possible vulnerability to third-party attacks can be carried out.

2. RESEARCH METHODS

This research was conducted through several systematic stages to ensure the accuracy and effectiveness of the method used. In the process, a literature study was conducted to understand the basic concepts of the Diffie-Hellman and Caesar Cipher algorithms. After that, manual calculations were carried out as an initial step to verify the key exchange process before being implemented inprogram. The results of the manual calculations were then tested using Python code to ensure that the algorithm functions correctly and can improve the security of encryption and decryption. Figure 1 shows a workflow diagram that provides an overview of the main stages of this research.

Figure 1. Research Stages

1. Literature Study
Studying the basic theory of cryptography, Caesar Cipher algorithm, and Diffie-Hellman algorithm. Researching the combination of key exchange techniques with encryption methods and to improve data security.

2. Manual calculation
Manual calculations are carried out by following the formula of the Diffe-Helman algorithm. The following is the formula and the stages of its implementation.

1. Select Public Number:
    a. Prime number (p)
    b. base number (g) and both are mutually agreed upon.
2. Select Private Key:
    a. Party A chooses a private key a (random number, a < p)

   b. Party B chooses a private key b (random number, b < p)
3. Calculate Public Key:
   a. Party A calculates:$A = g^a \ mod\ p$
   b. Party B calculates:$B = g^b \ mod\ p$
4. Exchange Public Key:
   a. Party A sends the results of A's calculations to Party B.
   b. Party B sends the results of B's calculations to Party A.
5. Calculate Shared Secret Key:
   a. Party A calculates:$K = B^a \ mod\ p$
   b. Party B calculates:$K = A^b \ mod\ p$

Next, after the calculation is done and K is found, the next stage is to encrypt and describe the message that you want to crack.(Ng, 2017). The formula used is Caesar cipher. Here is the Encryption Formula and description.

Encryption Formula: Encrypted ASCII= (ASCII Original+K) mod 256

Formula Description: Decrypted ASCII= (Encrypted ASCII−K) mod 256

3. Implementation and Testing with Python
Developing program code to automate key exchange using Diffie-Hellman. Integrating the key exchange results with encrypting and decrypting messages using Caesar Cipher. Testing various scenarios to ensure the security and effectiveness of the algorithm.

4. Evaluation Model
Verify whether the manual calculation results match the program results. Test the security of encryption and description using various methods, Evaluate the advantages and disadvantages of the system.

3. RESULTS AND DISCUSSION

The first stage of the Diffe-Helman algorithm process is to perform manual calculations to understand the key exchange process using the Diffie-Hellman algorithm up to the encryption and decryption process. Here are the calculation steps:

Initial Information:

Table 1 Plaintext to ASCII and Binary Conversion:

Character	ASCII	Binary
K	75	01001011
O	79	01001111
T	84	01010100
A	65	01000001
K	75	01001011
R	82	01010010
A	65	01000001
H	72	01001000
A	65	01000001
S	83	01010011
I	73	01001001
A	65	01000001
A	65	01000001
D	68	01000100
A	65	01000001
D	68	01000100
I	73	01001001
B	66	01000010
A	65	01000001
W	87	01010111
A	65	01000001

Character	ASCII	Binary
H	72	01001000
M	77	01001101
E	69	01000101
J	74	01001010
A	65	01000001

Public Number Selection

Two parties who want to share a secret key (for example Wendi and Jeni) must agree on two public numbers: Prime number p = 23; Base number g = 5.

1. Private Key Selection
   Each party chooses their own private key secretly:
   1. Wendi chooses the private key a = 6
   2. Jeni chooses the private key b = 15
2. Public Key Calculation
   Each party calculates their public key 5. using the formula:
   $A = g^a \bmod p$ And $B = g^b \bmod p$
   Calculation by Wendi:
   $A = 5^6 \bmod 23$
   A=156.25 mod 23 = 8
   Calculation by Jeni:
   $A = 5^{15} \bmod 23$
   A=30,517,578,125 mod 23 = 19
3. Public Key Exchange
   Wendi sends A = 8 to Jeni, and Jeni sends B = 19 to Wendi. Since A and B are not secret keys, they can be sent.
4. Shared Secret Key Calculation
   Each party uses the received public key to calculate a shared secret key using the formula:
   $$K = B^a \bmod p$$
   $$K = A^b \bmod p$$
   Calculation by Wendi:
   $$K = 19^6 \bmod 23$$
   A=47,045,82 mod 23 = 2
   Calculation by Jeni:
   $A = 8^{15} \bmod 23$
   A=35,184,372,088,832 mod 23 = 2
   Both parties get a shared secret key K = 2.

Encryption and Description Calculations

The following is the encryption result for the word "SECRET BOXISUNDERTHE DESK" using the secret key K = 2:

Table 2 encryption using secret key K = 2

Plaintext	ASCII	Binary	Encryption (M+K) mod 256	Ciphertext	Binary Results
K	75	01001011	(75+2) mod 256 = 77	M	01001101
O	79	01001111	(79+2) mod 256 = 81	Q	01010001
T	84	01010100	(84+2) mod 256 = 86	V	01010110
A	65	01000001	(65+2) mod 256 = 67	C	01000011
K	75	01001011	(75+2) mod 256 = 77	M	01001101
R	82	01010010	(82+2) mod 256 = 84	T	01010100
A	65	01000001	(65+2) mod 256 = 67	C	01000011
H	72	01001000	(72+2) mod 256 = 74	J	01001010
A	65	01000001	(65+2) mod 256 = 67	C	01000011
S	83	01010011	(83+2) mod 256 = 85	U	01010101
I	73	01001001	(73+2) mod 256 = 75	K	01001011
A	65	01000001	(65+2) mod 256 = 67	C	01000011
A	65	01000001	(65+2) mod 256 = 67	C	01000011
D	68	01000100	(68+2) mod 256 = 70	F	01000110

Plaintext	ASCII	Binary	Encryption (M+K) mod 256	Ciphertext	Binary Results
A	65	01000001	(65+2) mod 256 = 67	C	01000011
D	68	01000100	(68+2) mod 256 = 70	F	01000110
I	73	01001001	(73+2) mod 256 = 75	K	01001011
B	66	01000010	(66+2) mod 256 = 68	D	01000100
A	65	01000001	(65+2) mod 256 = 67	C	01000011
W	87	01010111	(87+2) mod 256 = 89	Y	01011001
A	65	01000001	(65+2) mod 256 = 67	C	01000011
H	72	01001000	(72+2) mod 256 = 74	J	01001010
M	77	01001101	(77+2) mod 256 = 79	O	01001111
E	69	01000101	(69+2) mod 256 = 71	G	01000111
J	74	01001010	(74+2) mod 256 = 76	L	01001100
A	65	01000001	(65+2) mod 256 = 67	C	01000011

Mod 256 ensures that all operations remain within the range of valid values for ASCII characters. Without Mod 256, encoding can produce numbers outside ASCII. Next is the decryption calculation to return the Ciphertext to plaintext. "MQVCMTCJCUKCCFCFKDCYCJOGLC" Back to the original text, namely "SECRET BOX IS UNDER THE TABLE" using the secret key K = 2.

Table 3. Decryption using secret key K = 2

Ciphertext	ASCII	Binary	Decryption (C−K) mod 256	Results	Binary Results
M	77	01001101	(77−2) mod 256 = 75	K	01001011
Q	81	01010001	(81−2) mod 256 = 79	O	01001111
V	86	01010110	(86−2) mod 256 = 84	T	01010100
C	67	01000011	(67−2) mod 256 = 65	A	01000001
M	77	01001101	(77−2) mod 256 = 75	K	01001011
T	84	01010100	(84−2) mod 256 = 82	R	01010010
C	67	01000011	(67−2) mod 256 = 65	A	01000001
J	74	01001010	(74−2) mod 256 = 72	H	01001000
C	67	01000011	(67−2) mod 256 = 65	A	01000001
U	85	01010101	(85−2) mod 256 = 83	S	01010011
K	75	01001011	(75−2) mod 256 = 73	I	01001001
C	67	01000011	(67−2) mod 256 = 65	A	01000001
C	67	01000011	(67−2) mod 256 = 65	A	01000001
F	70	01000110	(70−2) mod 256 = 68	D	01000100
C	67	01000011	(67−2) mod 256 = 65	A	01000001
F	70	01000110	(70−2) mod 256 = 68	D	01000100
K	75	01001011	(75−2) mod 256 = 73	I	01001001
D	68	01000100	(68−2) mod 256 = 66	B	01000010
C	67	01000011	(67−2) mod 256 = 65	A	01000001
Y	89	01011001	(89−2) mod 256 = 87	W	01010111
C	67	01000011	(67−2) mod 256 = 65	A	01000001
J	74	01001010	(74−2) mod 256 = 72	H	01001000
O	79	01001111	(79−2) mod 256 = 77	M	01001101
G	71	01000111	(71−2) mod 256 = 69	E	01000101
L	76	01001100	(76−2) mod 256 = 74	J	01001010
C	67	01000011	(67−2) mod 256 = 65	A	01000001

Implementation with Python

In the implementation stage, encryption and decryption algorithms are applied using Python to ensure that this process is successful with the specified objectives.

1. Key Determination

   Before the encryption and decryption process begins, the first step is to determine the key that will be used in the algorithm. This key plays an important role in maintaining data security.

```
PS D:\pyton> & C:/Users/ACER/AppData/Local/Programs/Python/Python312/python.exe d:/pyton/diffe-helman.py
Masukkan bilangan prima (p): 23
Masukkan bilangan basis (g): 5
Masukkan private key pengguna A (a): 6
Masukkan private key pengguna B (b): 15

Proses Pembentukan Kunci Publik dan Kunci Rahasia Bersama:
+--------------+---------------+--------+
| Rumus        | Proses        | Hasil  |
+==============+===============+========+
| A = g^a mod p | = 5^6 mod 23  |      8 |
+--------------+---------------+--------+
| B = g^b mod p | = 5^15 mod 23 |     19 |
+--------------+---------------+--------+
| K = B^a mod p | = 19^6 mod 23 |      2 |
+--------------+---------------+--------+
| K = A^b mod p | = 8^15 mod 23 |      2 |
+--------------+---------------+--------+
```

Figure 2. Implementation results determine the key

2. Encryption Process

   In the encryption process and the key result is determined, the data is processed with python for encryption. This process changes the original data into ciphertext that cannot be read without the correct key.

```
Masukkan pesan yang ingin dienkripsi: KOTAKRAHASIAADADIBAWAHMEJA

Hasil Enkripsi:
+-----------+-------+----------+-------------------------+--------------+--------------+------------+
| Plaintext |  ASCII |    Biner | Enkripsi (M+K) mod 256  | Hasil ASCII  | Hasil Biner  | Ciphertext |
+===========+=======+==========+=========================+==============+==============+============+
| K         |    75 | 01001011 | (75+2) mod 256          |           77 |     01001101 | M          |
+-----------+-------+----------+-------------------------+--------------+--------------+------------+
| O         |    79 | 01001111 | (79+2) mod 256          |           81 |     01010001 | Q          |
+-----------+-------+----------+-------------------------+--------------+--------------+------------+
| T         |    84 | 01010100 | (84+2) mod 256          |           86 |     01010110 | V          |
+-----------+-------+----------+-------------------------+--------------+--------------+------------+
| A         |    65 | 01000001 | (65+2) mod 256          |           67 |     01000011 | C          |
+-----------+-------+----------+-------------------------+--------------+--------------+------------+
| K         |    75 | 01001011 | (75+2) mod 256          |           77 |     01001101 | M          |
+-----------+-------+----------+-------------------------+--------------+--------------+------------+
| R         |    82 | 01010010 | (82+2) mod 256          |           84 |     01010100 | T          |
+-----------+-------+----------+-------------------------+--------------+--------------+------------+
| A         |    65 | 01000001 | (65+2) mod 256          |           67 |     01000011 | C          |
+-----------+-------+----------+-------------------------+--------------+--------------+------------+
| H         |    72 | 01001000 | (72+2) mod 256          |           74 |     01001010 | J          |
+-----------+-------+----------+-------------------------+--------------+--------------+------------+
| A         |    65 | 01000001 | (65+2) mod 256          |           67 |     01000011 | C          |
+-----------+-------+----------+-------------------------+--------------+--------------+------------+
| S         |    83 | 01010011 | (83+2) mod 256          |           85 |     01010101 | U          |
+-----------+-------+----------+-------------------------+--------------+--------------+------------+
| I         |    73 | 01001001 | (73+2) mod 256          |           75 |     01001011 | K          |
+-----------+-------+----------+-------------------------+--------------+--------------+------------+
| A         |    65 | 01000001 | (65+2) mod 256          |           67 |     01000011 | C          |
+-----------+-------+----------+-------------------------+--------------+--------------+------------+
| A         |    65 | 01000001 | (65+2) mod 256          |           67 |     01000011 | C          |
+-----------+-------+----------+-------------------------+--------------+--------------+------------+
```

Figure 3. Encryption Process

```
| D |  68 | 01000100 | (68+2) mod 256 |  70 | 01000110 | F |
| A |  65 | 01000001 | (65+2) mod 256 |  67 | 01000011 | C |
| D |  68 | 01000100 | (68+2) mod 256 |  70 | 01000110 | F |
| I |  73 | 01001001 | (73+2) mod 256 |  75 | 01001011 | K |
| B |  66 | 01000010 | (66+2) mod 256 |  68 | 01000100 | D |
| A |  65 | 01000001 | (65+2) mod 256 |  67 | 01000011 | C |
| W |  87 | 01010111 | (87+2) mod 256 |  89 | 01011001 | Y |
| A |  65 | 01000001 | (65+2) mod 256 |  67 | 01000011 | C |
| H |  72 | 01001000 | (72+2) mod 256 |  74 | 01001010 | J |
| M |  77 | 01001101 | (77+2) mod 256 |  79 | 01001111 | O |
| E |  69 | 01000101 | (69+2) mod 256 |  71 | 01000111 | G |
| J |  74 | 01001010 | (74+2) mod 256 |  76 | 01001100 | L |
| A |  65 | 01000001 | (65+2) mod 256 |  67 | 01000011 | C |
```

Figure 4. Encryption Process

3. Description Process

   The decryption process is to return the message to its original text using a predetermined key. This process means that only interested people have the key and can access the secret message.

```
Mengembalikan ke Teks Asli:
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
| Ciphertext | ASCII |    Biner | Dekripsi (C-K) mod 256  | Hasil ASCII  | Hasil Biner  | Plaintext |
+============+=======+==========+=========================+==============+==============+===========+
| M          |    77 | 01001101 | (77-2) mod 256          |           75 |     01001011 | K         |
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
| Q          |    81 | 01010001 | (81-2) mod 256          |           79 |     01001111 | O         |
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
| V          |    86 | 01010110 | (86-2) mod 256          |           84 |     01010100 | T         |
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
| C          |    67 | 01000011 | (67-2) mod 256          |           65 |     01000001 | A         |
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
| M          |    77 | 01001101 | (77-2) mod 256          |           75 |     01001011 | K         |
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
| T          |    84 | 01010100 | (84-2) mod 256          |           82 |     01010010 | R         |
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
| C          |    67 | 01000011 | (67-2) mod 256          |           65 |     01000001 | A         |
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
| J          |    74 | 01001010 | (74-2) mod 256          |           72 |     01001000 | H         |
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
| C          |    67 | 01000011 | (67-2) mod 256          |           65 |     01000001 | A         |
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
| U          |    85 | 01010101 | (85-2) mod 256          |           83 |     01010011 | S         |
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
| K          |    75 | 01001011 | (75-2) mod 256          |           73 |     01001001 | I         |
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
| C          |    67 | 01000011 | (67-2) mod 256          |           65 |     01000001 | A         |
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
| C          |    67 | 01000011 | (67-2) mod 256          |           65 |     01000001 | A         |
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
| F          |    70 | 01000110 | (70-2) mod 256          |           68 |     01000100 | D         |
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
| C          |    67 | 01000011 | (67-2) mod 256          |           65 |     01000001 | A         |
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
| F          |    70 | 01000110 | (70-2) mod 256          |           68 |     01000100 | D         |
+------------+-------+----------+-------------------------+--------------+--------------+-----------+
```

Figure 5. Process Description

F	70	01000110	(70-2) mod 256	68	01000100	D
K	75	01001011	(75-2) mod 256	73	01001001	I
D	68	01000100	(68-2) mod 256	66	01000010	B
C	67	01000011	(67-2) mod 256	65	01000001	A
Y	89	01011001	(89-2) mod 256	87	01010111	W
C	67	01000011	(67-2) mod 256	65	01000001	A
J	74	01001010	(74-2) mod 256	72	01001000	H
O	79	01001111	(79-2) mod 256	77	01001101	M
G	71	01000111	(71-2) mod 256	69	01000101	E
L	76	01001100	(76-2) mod 256	74	01001010	J
C	67	01000011	(67-2) mod 256	65	01000001	A

Figure 6. Process Description

With testing carried out with Python, it is hoped that this coding system will provide an adequate level of security to protect sensitive data.(Romindo & Ferawaty, 2021). To ensure reliabilityand encryption strength under various circumstances.

5. CONCLUSION

The Diffie–Hellman (DH) algorithm is an important method in cryptographic key exchange, allowing two parties to share a secret key over an insecure channel, with a complex mathematical basis, namely the discrete logarithm problem. The security of this algorithm is highly dependent on the size of the modulus and group numbers used, which makes it secure if the applied parameters are large enough. However, this algorithm is vulnerable to Man-in-the-Middle (MITM) attacks if not equipped with adequate authentication mechanisms. Based on testing, Diffie-Hellman can generate secure keys if the used parameters are strong enough, but its implementation must be accompanied by additional security measures, such as digital certificates or other authentication methods, to avoid potential attacks. Therefore, although this algorithm is effective for key exchange, its implementation must be done carefully and equipped with additional protection, such as TLS, to ensure the optimal level of communication security.

REFERENCE

*7.ahmad_*. (n.d.).

Febrianingsih, R., Hafiz, A., & Informatikan, M. (2019). Jurnal Informasi Dan Komputer Vol : 7 No : 2 Thn .: 2019 IMPLEMENTASI KRIPTOGRAFI BERBASIS CAESAR CHIPER UNTUK Jurnal Informasi Dan Komputer Vol : 7 No : 2 Thn .: 2019. *Analisa Infrastruktur Teknologi Informasi Menggunakan Framework Cobit 4.1*, *Vol :7*, 81–86.

Gunardi, T. (n.d.). *ANALISIS DAN STUDI ALGORITMA PERTUKARAN KUNCI DIFFIE-HELLMAN*.

Limbong, T., Katolik, U., Thomas, S., Utara, S., & Silitonga, P. D. P. (2017). *Testing the Classic Caesar Cipher Cryptography using of Matlab Parasian Silitonga Testing the Classic Caesar Cipher Cryptography using of Matlab*. https://doi.org/10.17605/OSF.IO/PEMA5

Ng, F. (2017). Pertukaran kunci Diffie-Hellman dengan Pembangkit Bilangan Acak Linear Congruential Generator (LCG). *Semantika (Seminar Nasional Teknik Informatika)*, *1*(1), 25–29.

Pratiwi, R., Utami, L. C., & Sakti, R. B. (2022). *Bulletin of Information Technology (BIT) Perancangan Keamanan Data Pesan Dengan Menggunakan Metode Kriptografi Caesar Cipher*. *3*(4), 367–373. https://doi.org/10.47065/bit.v3i1

Purwanto, H., & Informatika, M. (n.d.). *PENERAPAN KEAMANAN E-MAIL DENGAN MENGGUNAKAN GNU PRIVACY GUARD (GNUPG)*.

Romindo, & Ferawaty. (2021). Penerapan Algoritma Hybrid RSA Terhadap Pembangkit Kunci Diffie Hellman untuk Sistem Keamanan. *SATIN - Sains Dan Teknologi Informasi*, *7*(2), 92–101. https://doi.org/10.33372/stn.v7i2.783

Saepulrohman, A., & Negara, P. (2021). *IMPLEMENTASI ALGORITMA TANDA TANGAN DIGITAL BERBASIS KRIPTOGRAFI KURVA ELIPTIK DIFFIE-HELLMAN*. *18*(1), 22–28. https://asecuritysite.com/encryption/js08.

Thahara, A., & Siregar, I. T. (2021). Implementasi Kriptografi untuk Keamanan Data dan Jaringan menggunakan Algoritma DES. *JURTI*, *5*(1).

Wahyuni, A. (2011). Keamanan Pertukaran Kunci Kriptografi dengan Algoritma Hybrid : Diffie-Hellman dan RSA. *Majalah Ilmiah INFORMATiKA*, *2*(2), 15–23.