

Contents lists available online at [TALENTA Publisher](#)

DATA SCIENCE: JOURNAL OF COMPUTING AND APPLIED INFORMATICS (JoCAI)

Journal homepage: <https://jocai.usu.ac.id>

Performance Analysis of Hybrid Cryptographic Algorithms Rabbit Stream and Enhanced Dual RSA

Demonius Sarumaha¹, Mohammad Andri Budiman², and Muhammad Zarlis³^{1,2}*Master of Informatics Program, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan, Indonesia*³*Department of Information System, Bina Nusantara University, Jakarta, Indonesia*

ARTICLE INFO

Article history:

Received 12 December 2022

Revised 26 January 2023

Accepted 27 January 2023

Published online 30 January 2023

Keywords:

Rabbit Stream
Enhanced Dual RSA
Cryptography
Hybrid

Email:

¹demonius213@gmail.com²mandrib@usu.ac.id³muhammad.zarlis@binus.edu

Corresponding Author:

Mohammad Andri Budiman

ABSTRACT

Cryptography is a technique for encoding data by encrypting plaintext into an unreadable (meaningless) form. Cryptographic methods have good and bad performance depending on the type of algorithm we use. Therefore, the purpose of this study is to measure speed by combining the two algorithms used. The Rabbit Stream algorithm is a stream cipher algorithm whose system security depends on the generation of a key bit stream (keystream), which only guarantees 128-bit key security but has the advantage of being fast in the encryption and decryption process, while the Enhanced Dual RSA algorithm is an asymmetric algorithm to increase data protection from the Dual RSA algorithm by utilizing the Pells equation as a substitute for public key exponents. On the other hand, the algorithm in question requires a significant amount of time to encrypt messages with a large capacity when compared to the Rabbit Stream algorithm. Nonetheless, the study's findings suggest that using a hybrid method is comparatively faster for processing substantial amounts of data.

IEEE style in citing this article:

D. Saruhama, M.A.Budiman and M. Zarlis, " Performance Analysis of Hybrid Cryptographic Algorithms Rabbit Stream and Enhanced Dual RSA", *Data Science: Journal of Computing and Applied Informatics (JoCAI)*, vol. 7, no. 1, pp. 35-43, 2023.

1. Introduction

The development of technology in this era is a very important thing. Technology is able to help various organizations, companies, or other parties in communicating and exchanging data or documents. Along with the development of the times, agencies tend to use electronic documents in exchanging data because it is very easy and fast, but in data exchange, theft is often committed by third parties for personal gain [1]. Therefore data security is very important to keep confidential and to maintain the confidentiality of the data, and cryptographic techniques are needed [2].

Data security using cryptographic techniques is one way to hide the original message in another form that cannot be accessed or modified by unauthorized persons. Cryptography can be classified into two types based on the key used, namely symmetric cryptography and asymmetric cryptography. Symmetric cryptography is a cryptographic algorithm that uses the same key in the encryption and decryption processes. The communicating entities must exchange keys so that they can be used in the decryption process. The secret key used by the sender and receiver can be a series of random letters and numbers. Examples of symmetric algorithms: the Spritz algorithm, Rabbit Stream, RC4, TwoFish, Rijndael, etc [3]. Asymmetric cryptography is a public key algorithm that uses two different keys in the encryption and decryption processes, namely one for encryption and one for decryption. The public key used for encryption can be known publicly, while the private key is not desired [4].

In asymmetric cryptography, one of the keys is published, and knowing the public key will be an opportunity for unauthorized persons to decipher all the keys, even though it requires a long process. The advantage of using an asymmetric algorithm is that it provides greater scalability than a symmetric algorithm and guarantees confidentiality and authentication, but an asymmetric algorithm works much slower than a symmetric [5]. Using a symmetric algorithm can often be penetrated by cryptanalysts easily because the security of a symmetric algorithm depends only on the secrecy of the key. If the key used can be guessed or known by irresponsible parties, then all messages can be easily decrypted [6].

The Rabbit Stream algorithm is a stream cipher cryptographic algorithm that only guarantees the secrecy of a 128-bit key and uses the same key in encrypting and decrypting messages. The encryption and decryption process is carried out by XOR the generated key with plaintext or ciphertext [7]. The Rabbit Stream algorithm has a very fast process in key generation, encryption, and decryption processes. The security of the RSA algorithm, in general, does not only lie in the large number of key characters generated but actually lies in the difficulty of factoring very large numbers into prime numbers. The purpose of factoring is to get the public key and private key [8]. The Enhanced Dual RSA algorithm is an asymmetric algorithm to increase data protection from the Dual RSA algorithm by utilizing the Pells equation as a substitute for public key exponents [9]. The Enhanced Dual RSA cryptographic algorithm is less efficient for encrypting large messages because it produces a ciphertext that increases many times, causing problems, namely where it takes a long time during the encryption process and in sending the ciphertext. So this study combined the Enhanced Dual RSA algorithm with Rabbit Stream in a hybrid scheme to solve existing problems.

2. Methods

At this stage, to speed up the process of encrypting and decrypting large-capacity data, researchers use a hybrid method. Hybrid cryptography is a cryptographic technique that takes advantage of the advantages of each algorithm by combining symmetric and asymmetric algorithms. In hybrid cryptography, the sender generates a symmetric algorithm key, and then the plaintext will be encrypted using the private key to produce a ciphertext. The symmetric algorithm private key is encrypted using the asymmetric algorithm public key so as to get the cipherkey. Then the ciphertext and cipherkey will be sent to the recipient. To get the contents of the message sent, the recipient decrypts the cipherkey using the asymmetric private key algorithm and gets the symmetric private key. Furthermore, this symmetrical private key will be used to decrypt the ciphertext so as to get the original plaintext. The algorithms used in this study are the Rabbit Stream algorithm and the Enhanced Dual RSA algorithm.

2.1. Rabbit Stream Algorithm

The Rabbit Stream algorithm was first published publicly in 2003, Rabbit Stream is a stream cipher that only guarantees the security of a secret key of 128 bit. where the encryption and decryption processes are performed by XOR bit of plaintext with the key to be converted into ciphertext [10]. The steps in the process of encryption and decryption of the Rabbit Stream algorithm are as follows.

1. Key generation process

a. Key set up scheme

The first step of this algorithm is to determine and expand the key size of 128 bits which is divided into eight sub-keys: $k_0 = K^{[15...0]}$, $k_1 = K^{[31...16]}$, ..., $k_7 = K^{[127...112]}$. The state and counter variables are initialized from the subkey as follows:

$$x_{j,o} \begin{cases} k_{(j+1 \bmod 8)} \parallel k_j & \text{if } j \text{ even} \\ k_{(j+5 \bmod 8)} \parallel k_{(j+4 \bmod 8)} & \text{if } j \text{ odd} \end{cases}$$

and

$$c_{j,o} \begin{cases} k_{(j+4 \bmod 8)} \parallel k_{(j+5 \bmod 8)} & \text{if } j \text{ even} \\ k_j \parallel k_{(j+1 \bmod 8)} & \text{if } j \text{ odd} \end{cases}$$

Modify the status counter by following these steps:

$$c_{j,4} = c_{j,4} \oplus x_{(j+4 \bmod 8)}$$

b. Initialization Vector (IV))

In this scheme, the next step is to modify the status counter as an Initialization Vector function by XOR a 64-bit Initialization Vector. And iterate four times.

$$\begin{aligned}
 c_{0,4} &= c_{0,4} \oplus IV^{[31...0]} & c_{1,4} &= c_{0,4} \oplus IV^{[63...48]} \mid \mid IV^{[31...16]} \\
 c_{2,4} &= c_{2,4} \oplus IV^{[63...32]} & c_{3,4} &= c_{3,4} \oplus IV^{[47...32]} \mid \mid IV^{[15...0]} \\
 c_{4,4} &= c_{4,4} \oplus IV^{[31...0]} & c_{5,4} &= c_{5,4} \oplus IV^{[63...48]} \mid \mid IV^{[31...16]} \\
 c_{6,4} &= c_{6,4} \oplus IV^{[63...32]} & c_{7,4} &= c_{7,4} \oplus IV^{[47...32]} \mid \mid IV^{[15...0]}
 \end{aligned}$$

c. Counter system

Counter dynamics is defined as follows:

$$\begin{aligned}
 c_{0,i+1} &= c_{0,i} + \alpha_0 + \Phi_{7,i} \bmod 2^{32} \\
 c_{1,i+1} &= c_{1,i} + \alpha_1 + \Phi_{0,i+1} \bmod 2^{32} \\
 c_{2,i+1} &= c_{2,i} + \alpha_2 + \Phi_{1,i+1} \bmod 2^{32} \\
 c_{3,i+1} &= c_{3,i} + \alpha_3 + \Phi_{2,i+1} \bmod 2^{32} \\
 c_{4,i+1} &= c_{4,i} + \alpha_4 + \Phi_{3,i+1} \bmod 2^{32} \\
 c_{5,i+1} &= c_{5,i} + \alpha_5 + \Phi_{4,i+1} \bmod 2^{32} \\
 c_{6,i+1} &= c_{6,i} + \alpha_6 + \Phi_{5,i+1} \bmod 2^{32} \\
 c_{7,i+1} &= c_{7,i} + \alpha_7 + \Phi_{6,i+1} \bmod 2^{32}
 \end{aligned}$$

Where is the counter carry bit, $\Phi_{j,i+1}$ based on

$$\Phi_{j,i+1} = \begin{cases} 1 & \text{if } c_{0,i} + \alpha_0 + \Phi_{7,i} \geq 2^{32} \wedge j = 0 \\ 1 & \text{if } c_{j,i} + \alpha_j + \Phi_{j-1,i+1} \geq 2^{32} \wedge j > 0 \\ 0 & \text{on the contrary} \end{cases}$$

Before calculating the following state function, the system counter is updated by following the steps above:

$$\begin{aligned}
 a_0 &= 0x4d34d34d & a_4 &= 0xd34d34d3 \\
 a_1 &= 0xd34d34d3 & a_5 &= 0x34d34d34 \\
 a_2 &= 0x34d34d34 & a_6 &= 0x4d34d34d \\
 a_3 &= 0x4d34d34d & a_7 &= 0xd34d34d3
 \end{aligned}$$

d. Next state function

The essence of the Rabbit Stream algorithm is the iteration of the system, which is defined by the following equation:

$$g_{j,i} = \left((x_{j,i} + c_{j,i+1})^2 \oplus ((x_{j,i} + c_{j,i+1}))^2 \gg 32 \right) \bmod 2^{32}$$

The next step is using the formula:

$$\begin{aligned}
 x_0 &= G_0 + (G_7 \lll 16) + (G_6 \lll 16) \bmod 2^{32} \\
 x_1 &= G_1 + (G_0 \lll 8) + G_7 \bmod 2^{32} \\
 x_2 &= G_2 + (G_1 \lll 16) + (G_0 \lll 16) \bmod 2^{32} \\
 x_3 &= G_3 + (G_2 \lll 8) + G_1 \bmod 2^{32} \\
 x_4 &= G_4 + (G_3 \lll 16) + (G_2 \lll 16) \bmod 2^{32} \\
 x_5 &= G_5 + (G_4 \lll 8) + G_3 \bmod 2^{32}
 \end{aligned}$$

$$x_6 = G_6 + (G_5 \lll 16) + (G_4 \lll 16) \bmod 2^{32}$$

$$x_7 = G_7 + (G_6 \lll 8) + G_5 \bmod 2^{32}$$

e. Extraction scheme

$$S_i^{(15...0)} = x_{0,i}^{(15...0)} \oplus x_{5,i}^{(31...16)}$$

$$S_i^{(31...16)} = x_{0,i}^{(31...16)} \oplus x_{3,i}^{(15...0)}$$

$$S_i^{(17...32)} = x_{2,i}^{(15...0)} \oplus x_{7,i}^{(31...16)}$$

$$S_i^{(63...48)} = x_{2,i}^{(31...16)} \oplus x_{5,i}^{(15...0)}$$

$$S_i^{(79...64)} = x_{4,i}^{(15...0)} \oplus x_{1,i}^{(31...16)}$$

$$S_i^{(31...16)} = x_{4,i}^{(31...16)} \oplus x_{7,i}^{(15...0)}$$

$$S_i^{(111...96)} = x_{6,i}^{(15...0)} \oplus x_{3,i}^{(31...16)}$$

$$S_i^{(127...112)} = x_{6,i}^{(31...16)} \oplus x_{1,i}^{(15...0)}$$

2. Encryption and decryption process

$$\text{Encryption } c_i = p_i \oplus s_i$$

$$\text{Decryption } p_i = c_i \oplus s_i$$

The steps below will show the process of encryption and decryption of the Rabbit Stream algorithm as an example, as follows.

1. Input 128-bit Rabbit key
Key = DEMONIUSSARUMAHA
Plaintext = UNIVERSITAS SUMATERA UTARA
2. The results of the key generation process by utilizing the predetermined key are:
"cf4c1b2c4b8319dc880c5fab4fa58b52"
3. The encryption process is that the key that has been obtained from the generation process is XOR with each two-digit hexadecimal number with plaintext, resulting in:
"9a02527a0ed14a95dc4d0c8b1cf0c6139b09496d6bd64d9dda4d"
4. The decryption process, namely the ciphertext and key received, is XOR with two digits each and produces the original plaintext.

2.2. Enhanced Dual RSA Algorithm

The RSA algorithm is a public key algorithm invented by Ron Rivest, Adi Shamir, and Leonard Adleman in 1978. The RSA algorithm uses two keys to encrypt and decrypt messages [11]. Dual RSA is a development algorithm from RSA used to reduce memory usage and increase the security of the RSA algorithm [12]. The Enhanced Dual RSA algorithm is a public key algorithm to increase the security of the Dual RSA algorithm by using Pell's equation to hide exponential public keys and pseudo modulus to avoid factorization attacks. The proposed system uses the number z as the pseudo modulus, so it is very difficult to infer the value of n . The steps in the encryption and decryption process of the Enhanced Dual RSA algorithm are as follows.

1. Key generation process
 - a. Choose four prime numbers p, q, r, s where n can be calculated by:
$$n = p \cdot q \cdot r \cdot s$$
 - b. Compute Totient n with:
$$\Phi(n) = (p - 1) * (q - 1) * (r - 1) * (s - 1)$$
 - c. Choose a prime z to replace n where $\Phi(n) < z < n$,
 - d. Find $\Phi(z)$ provided that:
$$\Phi(z) = z - 1$$
 - e. Randomly find the exponents of the public keys e and f was given the following provision:
 - odd numbers
 - $\text{GCD}(\Phi(z), e) = 1$ and $\text{GCD}(\Phi(z), f) = 1$
 - f. Find the private key component d with provision:
$$(e * f * d) \bmod \Phi(z) = 1$$
 - g. Find the partner of (x_1, y_1) and (x_2, y_2) with provision:
$$x_1^2 - e y_1^2 = 1 \text{ dan } x_2^2 - f y_2^2 = 1$$

h. Commonly shared keys are (x_1, y_1, x_2, y_2, z)

2. Encryption and decryption process

Encryption $C = (M^{(x_1^2-1)/y_1^2}) \bmod z^{(x_2^2-1)/y_2^2} \bmod z$

Decryption $M = C^d \bmod z$

The steps below will show the encryption and decryption process of the Enhanced Dual RSA algorithm an example, as follows:

1. Key generation process

- a. Choose a prime number at random $p = 89, q = 37, r = 29, s = 43$
 $n = 4106371$
- b. Calculate Totient n with:
 $\Phi(n) = 3725568$
- c. Select the prime number z to replace n where $\Phi(n) < z < n, z = 3929183$
- d. Find $\Phi(z)$:
 $\Phi(z) = 3929182$
- e. Find the exponents of the e and f public keys at random
value $e = 89$ and $f = 43$
- f. Find the private key component d
 $d = 356265$
- g. Find the partner of (x_1, y_1) and (x_2, y_2)
 $x_1 = 500001, y_1 = 53000$ and $x_2 = 3482, y_2 = 531$
- h. Commonly shared keys are (x_1, y_1, x_2, y_2, z)

2. Encryption and decryption process

Encryption

Example plaintext : $M = 127$

$$C = (127^{(500001^2-1)/53000^2}) \bmod 3929183^{(3482^2-1)/531^2} \bmod 3929183$$

$$C = ((127^{(89)}) \bmod 3929183)^{(43)} \bmod 3929183$$

$$C = (3002502)^{43} \bmod 3929183$$

$$C = 1364038$$

Decryption

$$M = C^d \bmod z$$

$$M = 1364038^{356265} \bmod 3929183$$

$$M = 127$$

3. Results and Discussion

At this stage, the researcher will show the results of combining the Rabbit Stream and Enhanced Dual RSA algorithms using the Python programming language. Where the text data to be sent is first encrypted with the Rabbit Stream algorithm, which will later produce ciphertext. Furthermore, the key from the Rabbit Stream algorithm, whose key has been determined over 128 bit, is encrypted with the Enhanced Dual RSA algorithm, which produces the cipherkey. So that the ciphertext and cipherkey will be contributed to the recipient simultaneously. To return the encrypted text data, the recipient decrypts the cipherkey using the Enhanced Dual RSA private key algorithm, which generates a Rabbit Stream key. Then the ciphertext is decrypted using the Rabbit Stream key algorithm to produce plaintext. The specifications of the laptop used are as follows:

| | |
|------------------|-------------------------|
| Processor | : 2.9 GHz Intel Core i7 |
| Memory | : 8 GB, 1600 MHz, DDR3 |
| SSD | : 500 GB |
| Operating system | : macOS Catalina |

3.1. Program view with test data

Key : DEVANKA SARUMAHA
Plaintext : Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa.
Length of character plaintext: 100

In Figure 1, the results of the encryption and decryption process will be shown by combining the Rabbit Stream algorithm and the Enhanced Dual RSA algorithm. The processing time will be displayed automatically when the encryption and decryption process is complete.

```
#####  
### Hybrid Rabbit Stream and Enhanced Dual RSA ###  
#####  
----- Process encryption -----  
Original keys = DEVANKA SARUMAHA  
Key of Rabbit Stream = 9ef6c33b9e66fb2296213041b588f209  
Ciphertext = d299b15ef3b99f4db64d5926c0e49329fb91a64f6b02944ef9531e614ed9c6cf98e356  
##### Ciphertext to be sent recipient #####  
----- Results encryption key of rabbit -----  
Cipherkey = 574811998439808203132834789790087017702052500053810941473  
Cipherkey = 1157668592259035646637293054199855931249242801926559154146  
Cipherkey = 2295192521238083133808180539865775735769246394877  
Cipherkey = 88091493937224113224844218179608086220864637272213324091  
Cipherkey = 2831628973439471381973816300404801141121384557937117224043  
Cipherkey = 64411024122289211316286849805284292225828834219360464  
Cipherkey = 196621685744302127996173272223786362491373627781283744  
Cipherkey = 6188642744615825522066313765143473782584713836961787917227  
Cipherkey = 6381867572847078683975115018324029805775241065021917  
Cipherkey = 4516767324255120664789039810763217804189741093199319  
Cipherkey = 38105164582764746526350180741191707914954239557436476214768  
Cipherkey = 2377001044390520827354832325167367578144254083829055305970  
Cipherkey = 3857000317981951426529453367653713692532989517403176  
Cipherkey = 115906383148327622321662759700494784651382428617889643968  
Cipherkey = 19461306731973121079918586126349513168669041518619831148783  
Cipherkey = 317451079054413969516227780337109625143161483974763681891  
##### Ciphertext and cipherkey sent to recipient #####  
time required for the encryption process = 1.119190617  
----- Process decryption -----  
----- Result decryption cipherkey -----  
Key of Rabbit Stream = 9ef6c33b9e66fb2296213041b588f209  
----- Result decryption ciphertext with key of Rabbit -----  
Plaintext = 4c6f72656d20697073756d20646f6c6f722073697420616d65742c20636f6e7365637465747565722061646970697363696e6720656c69742e2041656e5616e20636  
fd6d6f646f206c6967756c61206567657420646f6c6f722061656e5616e206d  
Done  
time required for the decryption process = 0.00071311
```

Figure 1. Encryption and Decryption Results

Key : DEVANKA SARUMAHA
Plaintext : Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec qu
Length of character plaintext: 200

The Figure 2 shows the process of encrypting text data using the Rabbit Stream algorithm and encrypting the Rabbit key with an Enhanced Dual RSA public key. The processing time will be displayed automatically when the encryption and decryption process is complete.

```
#####  
### Hybrid Rabbit Stream and Enhanced Dual RSA ###  
#####  
----- Process encryption -----  
Original keys = DEVANKA SARUMAHA  
Key of Rabbit Stream = 9ef6c33b9e66fb2296213041b588f209  
Ciphertext = d299b15ef3b99f4db64d5926c0e49329fb91a64f6b02944ef9531e614ed9c6cf98e356  
##### Ciphertext to be sent recipient #####  
----- Results encryption key of rabbit -----  
Cipherkey = 28884833234580397729175962854852285805190036939468313867  
Cipherkey = 924249195970336583678080278910277241833474287313899928  
Cipherkey = 2534708861548406125353279740319416189974158109559011168221  
Cipherkey = 9672044698397579569003826354758819695691404226879214719  
Cipherkey = 2448063151961343277338364680815680904634328132417630452  
Cipherkey = 2698876727197031812225930858950786923681191442714872286285  
Cipherkey = 132923488206311786164026383591977805265302583826245622  
Cipherkey = 95447894195440116763394524029865338689152918909136  
Cipherkey = 191820249546826975930060188296832435945961722234700766543  
Cipherkey = 3483262167222740629898963613183807708757473683593511922  
Cipherkey = 18669881231457246399218520766625018164203918913712882458  
Cipherkey = 897336863824880094634809979079746378357366814871276901338  
Cipherkey = 10321707878083921978078844827965625672882420231022957  
Cipherkey = 3659144659789435283042445379673328195197632704471656880828  
Cipherkey = 9608334046242169751313143388759865575861979544458096205480  
Cipherkey = 6373439372934403847559494958512350940246024175110233745275  
##### Ciphertext and cipherkey sent to recipient #####  
time required for the encryption process = 1.238389012  
----- Process decryption -----  
----- Result decryption cipherkey -----  
Key of Rabbit Stream = 9ef6c33b9e66fb2296213041b588f209  
----- Result decryption ciphertext with key of Rabbit -----  
Plaintext = 4c6f72656d20697073756d20646f6c6f722073697420616d65742c20636f6e7365637465747565722061646970697363696e6720656c69742e2041656e5616e20636  
fd6d6f646f206c6967756c61206567657420646f6c6f722061656e5616e206d  
Done  
time required for the decryption process = 0.001600071
```

Figure 2. Encryption and Decryption Results

Key : DEVANKA SARUMAHA

Plaintext : Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec.

Length of character plaintext: 300

The Figure 3 shows an encryption and decryption process with a hybrid schema where the encoded text data can be restored to the original message. The runtime will be displayed automatically when the encryption and decryption process is complete.

```

===== Process encryption =====
Original keys = DEVANKA SARUMAHA
Key of Rabbit Stream = 9ef6c3b9e66fb2296213841b588f29
Ciphertext = d299b15ef346925e554561d1e79e6ec6db852ea469a4ff3551c61d6e79c7af95b75ee139e58b6405428c5e1816af798a41fbfb89256b8017124db
e4d367be95a5c5f3899f4db64552c6e49239f951a64f8e2944ef95316e114e9d9ccf798e356ff15843b80173348a881687d9faa48be89a56f958452495f89767f62aa9e9e15
db47c31529262e69b7abed2aa80a165a8c2542250e86629f399a4df1bf15d7f84532268cd77b0e84a3ff7058e4e3212ccbf8d295a99a5ef446807f7f4c192f4db9b
7ab2d6b657ea149241ff444361bed9125be86e57f203955f3524134d8a8977cb2db349fb129257f7b014134dcf8de29ed93ae15be288e4ef40822dae681c8e83a247beb9a51
e540183ce18129f99aa56b46b4df84436f

##### Ciphertext to be sent recipient #####

===== Results encryption key of rabbit =====
Cipherkey = 168952913152764678339688976460025764847862418281233747286
Cipherkey = 526380999495798688292771768447192437311833458190263690806
Cipherkey = 223222510545578487195908976213194011581863725361613526
Cipherkey = 6489172786541312208757169192180714814714991356407481387
Cipherkey = 18218687773618978918677523832923669959146883182944421068
Cipherkey = 849833532212598882088835534112940840868528346584
Cipherkey = 642084208426243104547655499262920444989878693808852857
Cipherkey = 76688018472226557513153298483338491134949864109672695
Cipherkey = 59197358333696424313927296468133693333280682318
Cipherkey = 1235814839346247783831520613997625188125402782018785762040
Cipherkey = 35382047657586374910485767018470898060163040109830258772
Cipherkey = 412544533484741873752541466338151774848245144568989
Cipherkey = 11427405751183081777833235236996294202077074367604057
Cipherkey = 3757454134345733394989945586651644413010250627756336
Cipherkey = 1489311869827399850783332582271871359893817175400989
Cipherkey = 2119871636538488618753858481475763738837893162519965508

##### Ciphertext and cipherkey sent to recipient #####
time required for the encryption process = 1.3180068387

===== Process decryption =====
Result decryption cipherkey
Key of Rabbit Stream = 9ef6c3b9e66fb2296213841b588f29
Result decryption ciphertext with key of Rabbit
Plaintext = 4c6f72656d2069707375620646f6c6f722073697420616d65742c20636f6e736563746574756572206164697067363696e6720656c69742c2041656e65616e20636
f6806f6467206c6967206e657420646f6c6f722073697420616d65742c20636f6e736563746574756572206164697067363696e6720656c69742c2041656e65616e20636
1676e69732064697320707375620646f6c6f722073697420616d65742c20636f6e736563746574756572206164697067363696e6720656c69742c2041656e65616e20636
4726963696573206e65632078656c6e656e746573717563206572c207072657469756d20717569732c207366562e204e756c6c120636f6e7365717561742068617373612071756
9720656e696d2e20446f6e5632e
Done
time required for the decryption process = 0.0019198311

```

Figure 3. Encryption and Decryption Results

Key : DEVANKA SARUMAHA

Plaintext : Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet

Length of character plaintext: 400

In Figure 4, the results of the encryption and decryption process will be shown by combining the Rabbit Stream algorithm and the Enhanced Dual RSA algorithm. The processing time will be displayed automatically during the encryption and decryption process.

```

===== Process encryption =====
Original keys = DEVANKA SARUMAHA
Key of Rabbit Stream = 9ef6c3b9e66fb2296213841b588f29
Ciphertext = d299b15ef346925e554561d1e79e6ec6db852ea469a4ff3551c61d6e79c7af95b75ee139e58b6405428c5e1816af798a41fbfb89256b8017124db
e4d367be95a5c5f3899f4db64552c6e49239f951a64f8e2944ef95316e114e9d9ccf798e356ff15843b80173348a881687d9faa48be89a56f958452495f89767f62aa9e9e15
db47c31529262e69b7abed2aa80a165a8c2542250e86629f399a4df1bf15d7f84532268cd77b0e84a3ff7058e4e3212ccbf8d295a99a5ef446807f7f4c192f4db9b
7ab2d6b657ea149241ff444361bed9125be86e57f203955f3524134d8a8977cb2db349fb129257f7b014134dcf8de29ed93ae15be288e4ef40822dae681c8e83a247beb9a51
e540183ce18129f99aa56b46b4df84436f

##### Ciphertext to be sent recipient #####

===== Results encryption key of rabbit =====
Cipherkey = 498601669004085086701454716898258105214634810017938670245
Cipherkey = 101844752321061999781913815764728757688208898448298771
Cipherkey = 25349678721879823482451077421193802124867327920207111962
Cipherkey = 30122041973239561173112336772077125298143467000953814540
Cipherkey = 1186791804635281636530499529529540115488955905194785137
Cipherkey = 839937335423637839725149925235146755920861064201923185680
Cipherkey = 18349574805463633331181512414398372422074074351802889
Cipherkey = 87992587830743810818560009579538200272515435731856398218
Cipherkey = 43286437998641008340681171867063062077860847029977030450
Cipherkey = 6513638741952745908835800960724249665665747725441
Cipherkey = 60981155085778013560980130422712163891446327818621281266
Cipherkey = 194517471024573998573438880824518799813139214138095
Cipherkey = 123838322719921588458417021085104863254620523870581339427
Cipherkey = 20864896238577332139084571944323271013910961134764026
Cipherkey = 29187359463710383835707804097106240443127239843809955
Cipherkey = 1416527785571588618304244599046326398578217061833380353627

##### Ciphertext and cipherkey sent to recipient #####
time required for the encryption process = 1.5302689075

===== Process decryption =====
Result decryption cipherkey
Key of Rabbit Stream = 9ef6c3b9e66fb2296213841b588f29
Result decryption ciphertext with key of Rabbit
Plaintext = 4c6f72656d2069707375620646f6c6f722073697420616d65742c20636f6e736563746574756572206164697067363696e6720656c69742c2041656e65616e20636
f6806f6467206c6967206e657420646f6c6f722073697420616d65742c20636f6e736563746574756572206164697067363696e6720656c69742c2041656e65616e20636
1676e69732064697320707375620646f6c6f722073697420616d65742c20636f6e736563746574756572206164697067363696e6720656c69742c2041656e65616e20636
4726963696573206e65632078656c6e656e746573717563206572c207072657469756d20717569732c207366562e204e756c6c120636f6e7365717561742068617373612071756
9720656e696d2e20446f6e5632e
Done
time required for the decryption process = 0.0033752239

```

Figure 4. Encryption and Decryption Results

From the appearance of the encryption program above that uses the same key with different plaintext character lengths, it can be seen that the resulting ciphertext, if converted into text, will show results that cannot be read or understood, and the Rabbit key encryption using the Enhanced Dual RSA algorithm produces a cipherkey that is increased many times. In the display of the decryption program, it can be seen that ciphertext that cannot be understood can be returned to the original plaintext without reducing or adding to the contents of the message characters. From the results of the encryption and decryption processes, it is concluded that the time required during the encryption process is longer when compared to the time required during the decryption process.

From the appearance of programs that use the same key with different plaintext character lengths, it shows the difference in the time needed during the encryption and decryption processes. The difference in time required is shown in Table 1.

Table 1. Encryption and Decryption Time

| Key | Plaintext Character Length (Character) | Encryption Time (Second) | Description Time (Second) |
|---------------------|--|--------------------------|---------------------------|
| DEVANKA SARUMAHA | 100 | 1.1199190617 | 0.00071311 |
| | 200 | 1.2383890152 | 0.0016000271 |
| | 300 | 1.3180060387 | 0.0019190311 |
| | 400 | 1.5302689075 | 0.0033252239 |

Note: The key used is in accordance with the provisions of the Rabbit stream algorithm, which is 128 bits or 16 characters long.

Based on the time required for data encryption and decryption, a relationship between length of character and runtime speed can be seen in Figure 5.

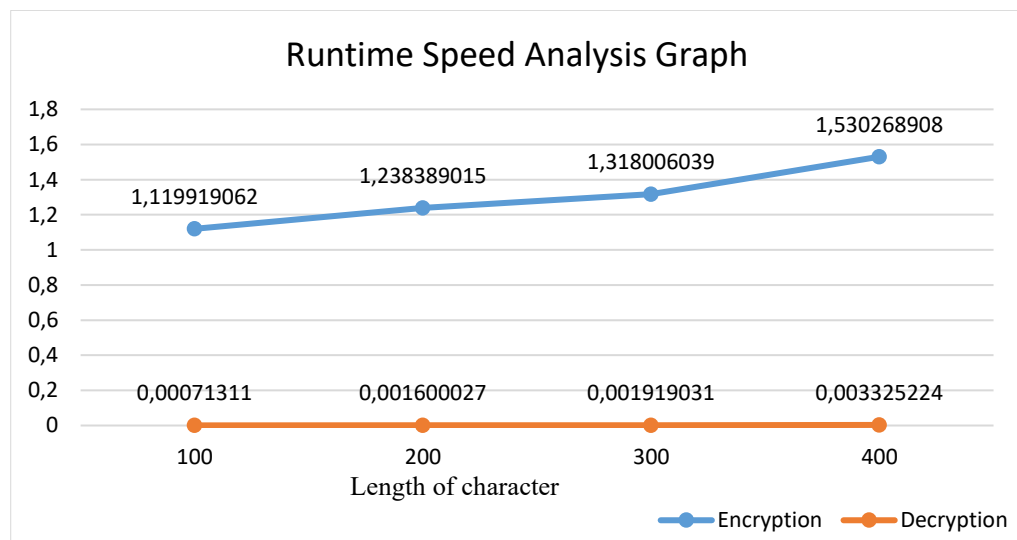


Figure 5. Processing Time Speed Graph

4. Conclusions

The study's findings suggest that using a combination of two algorithms in the encryption process generates a very large and random ciphertext that differs from the previous key. Consequently, without knowledge of the passwords for these algorithms, a hacker will need more time to compute the original message. Additionally, the Enhanced Dual RSA algorithm used in conjunction with the Rabbit Stream Algorithm generates a cipher key that is significantly more complex and challenging for cryptanalysts to decode. The encryption process requires more time compared to the decryption process, as it involves several stages. Finally, the decryption process of the ciphertext is successful in returning the original message without modifying its character content.

References

- [1] D. Kusumaningsih, A. Pudoli, And I. Rahmadan, “Steganografi Metode End Of File Untuk Keamanan Data,” *J. Telemat. Mkom*, Vol. 9, No. 1, Pp. 47–55, 2017.
- [2] S. Yakoubov, V. Gadepally, N. Schear, E. Shen, And A. Yerukhimovich, “A Survey Of Cryptographic Approaches To Securing Big-Data Analytics In The Cloud,” *2014 Ieee High Perform. Extrem. Comput. Conf. Hpec 2014*, 2014, Doi: 10.1109/Hpec.2014.7040943.
- [3] F. Maqsood, M. Ahmed, M. Mumtaz, and M. Ali, “Cryptography: A Comparative Analysis for Modern Techniques,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 6, pp. 442–448, 2017, doi: 10.14569/ijacsa.2017.080659.
- [4] D. P. Joseph and M. Krishna, “Cognitive Analytics and Comparison of Symmetric and Asymmetric Cryptography Algorithms,” vol. 6, no. 3, pp. 51–56, 2015.
- [5] M. A. Habib, M. Ahmad, S. Jabbar, S. H. Ahmed, And J. J. P. C. Rodrigues, “Speeding Up The Internet Of Things: Leaiot: A Lightweight Encryption Algorithm Toward Low-Latency Communication For The Internet Of Things,” *Ieee Consum. Electron. Mag.*, Vol. 7, No. 6, Pp. 31–37, 2018, Doi: 10.1109/Mce.2018.2851722.
- [6] M. A. Budiman, M. Y. Saputra, And Handrizal, “A Hybrid Cryptosystem Using Vigenère Cipher And Rabin-P Algorithm In Securing Bmp Files,” *J. Comput. Appl. Informatics*, Vol. 4, No. 2, Pp. 2016–2017, 2018, [Online]. Available: https://www.data-science.ruhr/about_us/.
- [7] F. Akhyar, S. M. Nasution, And T. W. Purboyo, “Rabbit Algorithm For Video On Demand,” *Apwimob 2015 - Ieee Asia Pacific Conf. Wirel. Mob.*, Pp. 208–213, 2016, Doi: 10.1109/Apwimob.2015.7374978.
- [8] B. S. Muchlis, M. A. Budiman, And D. Rachmawati, “Teknik Pemecahan Kunci Algoritma Rivest Shamir Adleman (Rsa) Dengan Metode Kraitichik,” *J. Penelit. Tek. Inform. E-Issn 2541-2019, P-Issn 2541-044x*, Vol. 2, No. 2, Pp. 49–64, 2017, [Online]. Available: <http://jurnal.polgan.ac.id/index.php/sinkron/article/view/75>.
- [9] Y. Funabashi, A. Shibata, S. Negoro, I. Taniguchi, And H. Tomiyama, *A Dynamic Programming Algorithm For Energy-Aware Routing Of Delivery Drones*, Vol. 13. 2020.
- [1] F. Maqsood, M. Ahmed, M. Mumtaz, and M. Ali, “Cryptography: A Comparative Analysis for Modern Techniques,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 6, pp. 442–448, 2017, doi: 10.14569/ijacsa.2017.080659.
- [2] D. P. Joseph and M. Krishna, “Cognitive Analytics and Comparison of Symmetric and Asymmetric Cryptography Algorithms,” vol. 6, no. 3, pp. 51–56, 2015.
- [3] M. Boesgaard, M. Vesterager, and E. Zenner, “The rabbit stream cipher,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4986 LNCS, pp. 69–83, 2008, doi: 10.1007/978-3-540-68351-3_7.
- [11] O. G. Abood And S. K. Guirguis, “A Survey On Cryptography Algorithms,” *Int. J. Sci. Res. Publ.*, Vol. 8, No. 7, 2018, Doi: 10.29322/Ijsrp.8.7.2018.P7978.
- [12] S. Verma And D. D. Garg, “Efficient Rsa Variants For Resource Constrained Environment,” *Secur. Commun. Networks*, No. August, Pp. 1–22, 2012.