**Journal of Computer Networks, Architecture and High Performance Computing**
Volume 7, Number 1, Januari 2025
https://doi.org/10.47709/cnahpc.v7i1.5345

**Submitted** : Jan 17, 2025
**Accepted** : Jan 30, 2025
**Published** : Jan 31, 2025

# CRYPTOGRAPHY OF CHACHA20 and RSA ALGORITHMS for TEXT SECURITY

**Ziad Yasqi Dzahabi[1)*], Nurul Hayaty[2)], Martaleli Bettiza[3)]**
[1,2,3)]Program Studi Teknik Informatika Fakultas Teknik dan Teknologi Kemaritiman, Universitas Maritim Raja Ali Haji, Tanjungpinang, Indonesia
[1)]180155201047@student.umrah.ac.id, [2)]nurul.hayaty@umrah.ac.id, [3)]mbettiza@umrah.ac.id

## ABSTRACT

The purpose of this study is to apply the ChaCha20 and RSA cryptographic algorithms to enhance text security and safeguard data from unauthorized access, data breaches, and cyberattacks such as man-in-the-middle or replay attacks. ChaCha20, a symmetric encryption algorithm, is employed for generating efficient and secure keystreams, while RSA, an asymmetric algorithm, is used for encrypting numeric keys or messages. The integration of these two algorithms ensures robust data protection from various digital threats. The choice of this title stems from the growing urgency to prioritize data security in the digital era, especially given the increasing incidents of data leaks that often lead to significant consequences. This research focuses on analyzing the implementation of both algorithms in encryption and decryption processes, as well as evaluating their effectiveness in preserving data confidentiality and integrity. The findings of this study demonstrate that the ChaCha20 and RSA implementations effectively secure data, with the encryption and decryption processes functioning as intended. To further validate the system's robustness, simulated attacks were conducted, and the results confirmed the system's ability to prevent unauthorized access. This research not only contributes to the development of reliable data security solutions but also highlights opportunities for future improvements. Enhancing algorithm efficiency and optimizing encryption runtime are potential areas for further exploration. By addressing these challenges, the study aims to pave the way for more robust and efficient cryptographic solutions in the evolving landscape of digital security.

**Keywords:** ChaCha20 encryption; Cryptographic Implementation; Cyberattack Prevention; Data Security; RSA algorithm

## INTRODUCTION

The ChaCha20 algorithm is widely recognized for its combination of speed and robust security in encryption tasks. At its core, ChaCha20 operates by generating keystreams through the chacha20Keystream function, which serves as the backbone of the algorithm. The encryption process begins by forming an initial state, constructed by combining keys, nonces, and counters. This state undergoes a series of transformations, including rotations, spins, and bitwise manipulations, executed across 20 rounds as described in the chacha20Block function. These operations enhance the randomness and unpredictability of the keystream, ensuring that the generated ciphertext is resistant to a wide range of cryptographic attacks. A key aspect of ChaCha20's operation is the byte-by-byte XOR process, used to generate ciphertext from plaintext and to decrypt the ciphertext back into its original form. This approach is not only efficient but also reflects the symmetric nature of the algorithm, where the same key is used for both encryption and decryption.

* Corresponding author

**Journal of Computer Networks, Architecture and High Performance Computing**
Volume 7, Number 1, Januari 2025
https://doi.org/ 10.47709/cnahpc.v7i1.5345

**Submitted** : Jan 17, 2025
**Accepted** : Jan 30, 2025
**Published** : Jan 31, 2025

One notable enhancement in this study is the implementation of a ToWordArray function, which converts string keys into 32-bit word arrays. This feature adds flexibility to the algorithm, allowing it to efficiently process string inputs as cryptographic keys without compromising security. This functionality is particularly advantageous for applications requiring dynamic and diverse key management systems.

According to Barbero, etc(2022), by considering the ChaCha20 stream cipher in a related-key scenario and studying how to obtain XOR rotation pairs with a nonzero probability after applying the first quarter-round, it is possible to analyze potential vulnerabilities. The input of ChaCha20 can be viewed as a 4×4 matrix of 32-bit words, where the first row of the matrix is assigned constant values, the next two rows represent the key, and the fourth row contains some initialization values. Investigating the existence of constants different from those used in the first row of the ChaCha20 input reveals an increased probability of XOR rotations, indicating a potential weakness in certain variants of the ChaCha20 stream cipher.

According to Tampubolon (2021), data theft is often carried out for criminal activities, such as the misuse of data for illicit purposes. The author aims to secure text messages using a combination of the RSA algorithm and the DES algorithm. If only the RSA algorithm is used, the security of a text message remains vulnerable. The security level of the RSA algorithm depends on the key size used; the smaller the key size, the easier it is to break using brute-force methods.

According to Aulia, etc(2019), cryptography can be defined as a technique for making text messages unreadable. In its implementation, the original message is transformed into an unrecognizable form, and upon reaching its destination, it can be restored to its original form. Cryptography is classified into three types: symmetric, asymmetric, and hash.

ChaCha20's unique combination of speed and security makes it an excellent choice for various real-time applications, such as secure communication protocols, streaming platforms, and encrypted messaging services. Furthermore, its lightweight design allows it to operate efficiently even on devices with limited resources, such as IoT devices and mobile systems, highlighting its adaptability in modern cryptographic solutions.

## LITERATURE REVIEW

### ChaCha20

ChaCha20 is a symmetric encryption algorithm belonging to the stream cipher family, developed by Daniel J. Bernstein in 2008 as a more secure and efficient alternative to AES. It utilizes 256-bit keys and 96-bit nonces, offering high-speed performance and strong resistance to cryptographic attacks, particularly on devices without specialized hardware acceleration (Nir & Langley, 2018). ChaCha20 is an improved version of Salsa20, incorporating modifications that enhance diffusion per round to strengthen security. Earlier variants, such as ChaCha8, evolved into ChaCha12 and ChaCha20, featuring 12 and 20 rounds, respectively, to enhance resilience against cryptanalysis.

Compared to Salsa20, ChaCha20 provides a more robust encryption mechanism by increasing the number of rounds, thereby improving diffusion. Bernstein also introduced key operations in ChaCha20, such as quarter-round transformations and matrix structures. While ChaCha shares similarities with Salsa20, it differs in its operational sequence. In a quarter-round, ChaCha updates four 32-bit words (a, b, c, and d) twice, whereas Salsa20 updates them only once. This modification enhances security by increasing the complexity of the encryption process.

* Corresponding author

**Journal of Computer Networks, Architecture and High Performance Computing**
Volume 7, Number 1, Januari 2025
https://doi.org/ 10.47709/cnahpc.v7i1.5345

**Submitted** : Jan 17, 2025
**Accepted** : Jan 30, 2025
**Published** : Jan 31, 2025

a += b; d ^= a; d <<<= 16;

c += d; b ^= c; b <<<= 12;

a += b; d ^= a; d <<<= 8;

c += d; b ^= c; b <<<= 7;

The advantage of quarter-round in ChaCha is that it spreads changes faster, ensuring that each input word affects the output more evenly than Salsa20.

**RSA**

RSA (Rivest-Shamir-Adleman) is an asymmetric encryption algorithm introduced by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977. These algorithms are widely used in secure communication, digital signatures, authentication, and key exchange, and are an important cornerstone in modern cryptography and information security techniques.

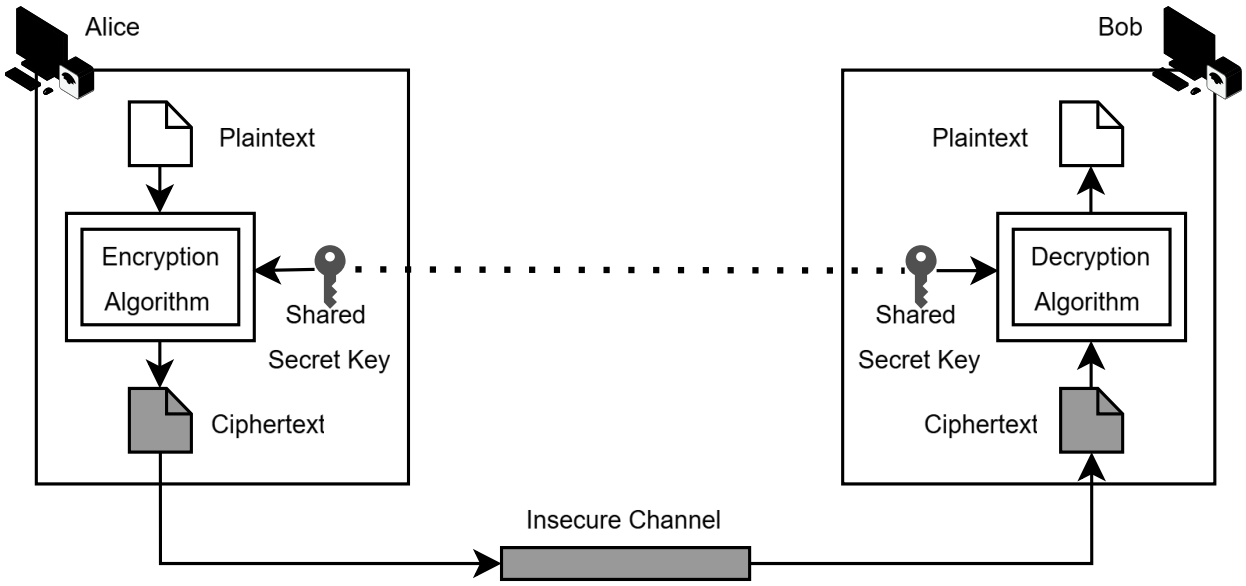


*Fig 1.* RSA encryption and decryption algorithms(Forouzan, 2008)

The original message from Alice to Bob is called plain text, while the sent message is called passtext. Alice uses encryption algorithms and secret keys to convert plain text into ciphertext. Bob uses the same decryption algorithm and key to return it to plain text. Forouzan refers to these encryption and decryption algorithms as ciphers, while keys are the values used in password operations.

* Corresponding author

**Journal of Computer Networks, Architecture and High Performance Computing**
Volume 7, Number 1, Januari 2025
https://doi.org/ 10.47709/cnahpc.v7i1.5345

**Submitted** : Jan 17, 2025
**Accepted** : Jan 30, 2025
**Published** : Jan 31, 2025

## METHOD

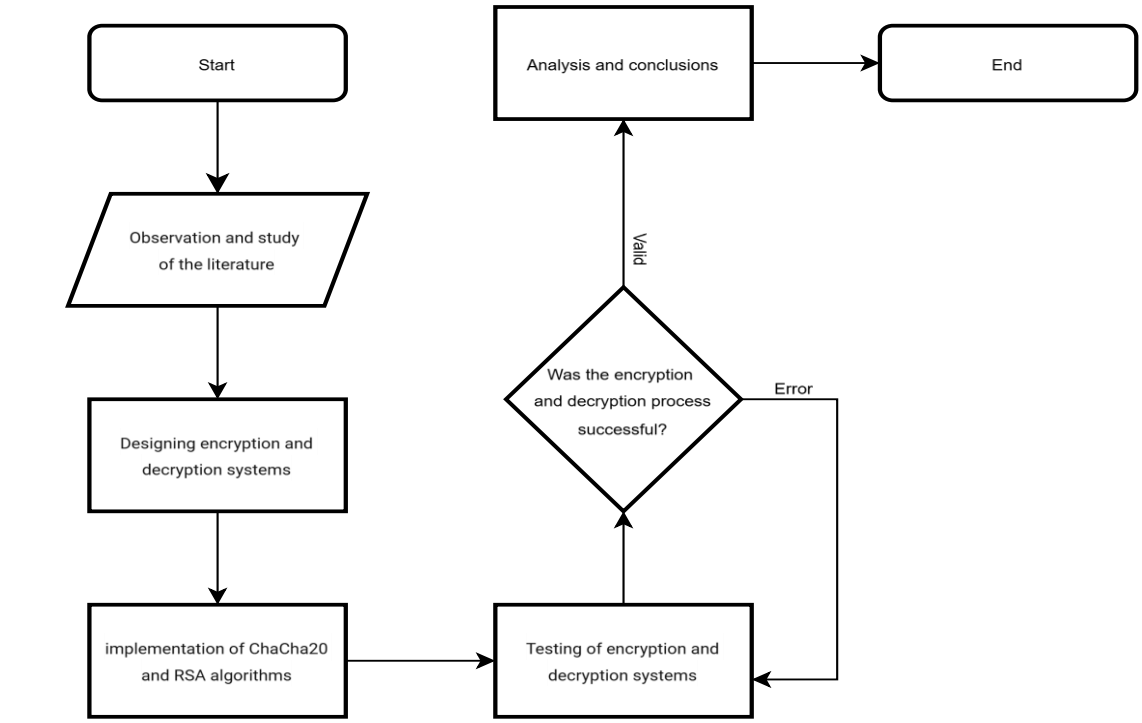


*Fig 2.* Research stages

The research method used in this study is designed to achieve the goals that have been formulated. Data collection techniques are one of the main components in this research process, where various approaches are used to obtain relevant information and support the analysis. The approach used is adjusted to the needs of the research, so that the results obtained can answer the problems studied comprehensively. The data collection technique is carried out by combining primary and secondary data sources. Primary data were obtained through direct observation and experiments, which allowed researchers to evaluate the application of the ChaCha20 algorithm in hybrid cryptography. Meanwhile, secondary data is obtained from literature studies, such as scientific journals, reference books, and other credible sources, in order to enrich the theoretical basis and provide a broader context for the problem being researched. The data analysis process is carried out in stages to ensure that each step of the research runs systematically. The analysis begins with the grouping of data based on their type and relevance to the research objectives. The collected data is then processed and interpreted to produce meaningful information. The analysis also involves comparing the results of encryption and decryption using ChaCha20, in order to assess the level of security and efficiency. In addition, this study uses an experimental approach to evaluate the performance of the applied algorithm. Experiments are conducted with specific scenarios that represent real use, such as encryption and decryption of sensitive text files. The results of the experiment were analyzed to see the effectiveness of the algorithm in maintaining data confidentiality and the time efficiency required during the cryptographic process. Through a combination of systematic data collection and analysis techniques, this study aims to contribute to understanding the application of the ChaCha20 algorithm in hybrid cryptography systems. With a

* Corresponding author

**Journal of Computer Networks, Architecture and High Performance Computing**
Volume 7, Number 1, Januari 2025
https://doi.org/ 10.47709/cnahpc.v7i1.5345

**Submitted** : Jan 17, 2025
**Accepted** : Jan 30, 2025
**Published** : Jan 31, 2025

structured approach, the results of this study are expected to provide a clear picture of the benefits, advantages, and limitations of the proposed method.

**Diagram System**

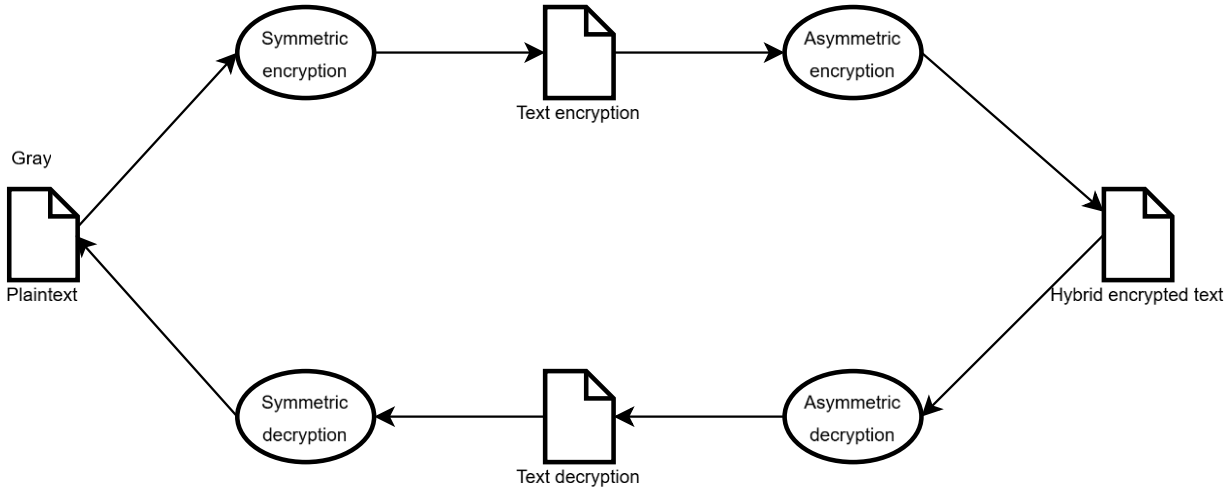


*Fig 3.* Combination of ChaCha20 and RSA

Figure 3 above illustrates in general how hybrid cryptography works, the process starts with a gray file that contains sensitive information that needs to be protected from unauthorized access. To improve the security of text files, Gray implements a layer of symmetric encryption. It involves using a symmetric encryption algorithm, such as ChaCha20, to encrypt the entire text file. Gray chose a strong and secret password. This password will serve as a key to unlock the symmetric encryption and decrypt the text file later. Using the chosen password, Gray applies a symmetric encryption algorithm to encrypt text files. It creates an encrypted version of the original text file. Gray generates a pair of asymmetrical keys: a public key and a private key. The public key will be used for encryption, and the private key will be used for decryption. Gray encrypts the symmetric encryption key (the password used for text files) using the recipient's public key (Green). This ensures that only Green, who has the associated private key, can decrypt the symmetric encryption key. The receiver, "Green", who has the private key that corresponds to the public key used for encryption, receives an encrypted symmetric encryption key. Green used their private key to decrypt the encrypted symmetric encryption key, revealing the original symmetric encryption key that Gray used to encrypt text files. Armed with the original symmetric encryption key, Green uses it to decrypt symmetrically encrypted text files. This process returns the original plaintext content of the file. By using hybrid cryptography in this way, Gray ensures that sensitive text files remain secure during transmission or storage. The combination of symmetric encryption for file content and asymmetric encryption for encryption key exchange improves overall data security and enables secure sharing between Gray and Green.

**Encryption Design**

In the process of designing ChaCha20 and RSA encryption, it will be applied using an input in the form of a plaintext that will be encrypted first. This process will go through four important steps, namely Encrypt Key to create a Stream key(ChaCha20 Key), Prime Number p, Prime Number q and the Encryption process.

* Corresponding author

**Journal of Computer Networks, Architecture and High Performance Computing**
Volume 7, Number 1, Januari 2025
https://doi.org/ 10.47709/cnahpc.v7i1.5345

**Submitted** : Jan 17, 2025
**Accepted** : Jan 30, 2025
**Published** : Jan 31, 2025

Here the plaintext to be used is "ZIAD" with its mandatory encryption key of 32 bytes or 64 hexadecimals, namely "00112233445566778899AABBCCDDEEFF".

**ChaCha20 Encryption**

Encryption with the ChaCha20 algorithm is the process of securing data by converting it into an unreadable (encrypted) format using an encryption key known only to authorized parties. Here is a step-by-step explanation of how to encrypt "Ziad" messages using ChaCha20:

**Step 1: Understanding Constants and Variables**

**A. Locks and Nonce**

Keys and nonces are important components used in ChaCha20 to ensure the security of the encryption results.

Table 1. Locks and Nonce

| Variable | Heksadesimal Value | Explanation |
|---|---|---|
| Key | 00112233445566778899AABBCCDDEEFF | 256-bit (32 bytes) encryption key |
| Nonce | 11223344 | 64-bit values used as nonce to ensure uniqueness |

**Step 2: Keystream block formation (Key Initialization and Initial Block)**

ChaCha20 utilizes initialized encryption keys and nonces to form a keystream through the initialization process.

**Step 3: (Permutation Round)**

In ChaCha20, this operation describes how an algorithm works in a single round or "permutation round" using input data.

**Step 4: (Keystate Block)**

**Keystate Block Results**

The result of the last round of permutation is a keystate block. This keystate block is used as the "internal key" for the encryption process on the ChaCha20 algorithm.

**Step 5: (Encrypt data)**

In this step, the data to be encrypted is converted to binary format and encrypted by performing an XOR operation using the keystream that has been generated.

**Data Conversion**

The "Ziad" data in UTF-8 format is converted into a binary representation:
"Z" → 01011010
"i" → 01101001
"a" → 01100001
"d" → 01100100
So, the "Ziad" data in the binary is: 01011010 01101001 01100001 01100100

* Corresponding author

**Journal of Computer Networks, Architecture and High Performance Computing**
Volume 7, Number 1, Januari 2025
https://doi.org/ 10.47709/cnahpc.v7i1.5345

**Submitted** : Jan 17, 2025
**Accepted** : Jan 30, 2025
**Published** : Jan 31, 2025

**RSA Encryption**
To encrypt the results of ChaCha20 encryption with RSA, we first need to create a strong private key and a public key and then take the next steps to encrypt it. Here are the steps to encrypt the results of ChaCha20 with RSA:

**Step 1: Generate Keys**
Select two prime number (p dan q)
Calculate the Modulus Value (n)(1)
Formula for calculating Modulus Value:

$$n = p \times q \qquad (1)$$

Calculate the Totien Function(2)
Since p and q are prime numbers, ($\Phi(\boldsymbol{n})$)they are calculated as:

$$\Phi(n) = (p - 1) \times (q - 1) \qquad (2)$$

Select Encryption Exponent (e)
Choose a relatively prime one with $\Phi(n)$. The common values for e are 3, 17, or 65,537.
Calculate Decryption Exponents (d)
The decryption exponent d is the modular inverse of the e modulo $\Phi(n)$. It is calculated using the Euclidean algorithm:

$$e \; x \; d \; mod \Phi n \; = \; 1 \qquad (3)$$

Initialization of calculations(4)
Each iteration uses the formula:

$$q \; = \; a \div b$$
$$(a, b) \; = (b, a \; mod \; b)$$
$$(x, y) \; = (y, x \; - \; q \; x \; y) \qquad (4)$$

This process is repeated until b=0, and the last x value is d. After the iteration, it was found:

Table 2. d value

| Variable | Value |
|---|---|
| d | 62,486,071,073 |

With the above steps, the RSA key has been successfully generated:
Public Key (e, n): (17, 531,133,135,307)
Private Key (d, n): (62,486,071,073, 531,133,135,307)

**Step 2: Convert ChaCha20 Results to RSA Messages**
Binary Encryption Results
The results of the binary encryption provided are:

01110001 00101001 10001110 11111101

* Corresponding author

**Journal of Computer Networks, Architecture and High Performance Computing**
Volume 7, Number 1, Januari 2025
https://doi.org/ 10.47709/cnahpc.v7i1.5345

**Submitted** : Jan 17, 2025
**Accepted** : Jan 30, 2025
**Published** : Jan 31, 2025

Steps to Convert to Decimals
Each binary block is converted to a decimal:
Where is the binary value in the ith position, calculated from the right (0 position).

1. blok 1: $11111101_2$

$11111101_2 = ((1 \times 2_7)+(1 \times 2_6)+(1 \times 2_5)+(1 \times 2_4)+(1 \times 2_3)+(1 \times 2_2)+(1 \times 2_1)+(1 \times 2_0) = 128 + 64 + 32 + 16 + 8 + 4 + 0 + 1 = 253$

Conversion Results
The result of binary encryption in decimal format is:

253 142 41 113

**Step 3: Encrypt Messages with RSA Public Key**
In this step, we will encrypt the numeric message "253 142 41 113" using the RSA public key that has been provided:
Encryption Exponent (e): 29
Modulus (n): 531133135307
The formula for RSA encryption is(5):

$$C = M^e \ mod \ n \quad (5)$$

Where:
- C is the ciphertext (the result of encryption).
- M is the numeric message that you want to encrypt.
- e is an encryption exponent.
- n is the modulus.

The steps are as follows:
Numerical Message (M): 253 142 41 113
Encryption Exponent (e): 29
Modulus (n): 531133135307
To calculate the operation correctly, it is necessary to calculate separately, namely 253, 142, 41, 113. Here are the calculations(6):

$$C = 253^{29} \bmod 531133135307 = 392169214730 \quad (6)$$

So the results of all calculations from 253, 142, 41 and 113 are 392169214730, 412260330458, 46740195788, 213243971870.

**RSA Decryption**
The steps to decrypt an RSA ciphertext with its private key are as follows:

**Step 1: Get the RSA private key**
Because at the beginning you have received the RSA private key in the form of (d, n), which is as follows:
C (encrypted message) = 392169214730 412260330458 46740195788 213243971870
d (decryption exponent) = 238093477709
n(modulus) = 531133135307

* Corresponding author

**Journal of Computer Networks, Architecture and High Performance Computing**
Volume 7, Number 1, Januari 2025
https://doi.org/ 10.47709/cnahpc.v7i1.5345

**Submitted** : Jan 17, 2025
**Accepted** : Jan 30, 2025
**Published** : Jan 31, 2025

**Step 2: Description of RSA ciphertext**

RSA ciphertext descriptions involve using a private key (d,n) to describe a message that has been previously encrypted with a public key (e,n). Just like before this description process is processed separately, the steps are as follows:
Using the private key (d,n), you can use the RSA description formula(7):

$$M = C^d \bmod n \quad (7)$$

Enter the values C, d, and n into the formula and then do the calculation(8):

$$M = 392169214730^{238093477709} \bmod 531133135307 \quad (8)$$

So the results of all calculations from 92169214730, 412260330458, 46740195788 and 213243971870 are 253, 142, 41, 113.

**ChaCha20 Decryption**

To decrypt the ciphertext generated with ChaCha20, it is only necessary to perform a decryption process which is the opposite of the encryption process. It uses the same keystream that is used for encryption and performs XOR operations between the ciphertext and this keystream. Here are the steps:

**Step 1: Convert decimal to binary**

253 = 11111101
142 = 10001110
41 = 00101001
113 = 01110001
Now we have these values in binary format.
Previous ciphertext results:
01110001 00101001 10001110 11111101

**Step 2: Find the keystream to use**

Use the same keystream used when encrypting. In this case, the keystream of the same key state block is several, namely, as follows:
Word 0: 01101011 01000000 11100011 11110101
Word 1: 01001111 00010110 00101111 00001001
Word 2: 01000110 11010100 01001100 00111111
Word 3: 00111001 11000010 00010001 00001000
Word 4: 00000011 00000010 00000001 00000000
Word 5: 00110011 00100000 01100100 01101110
Word 6: 01111001 01100010 00101101 00110010
Word 7: 00010001 00100011 00110011 01000100

Perform XOR operations between ciphertext and keystream. This will return the original plaintext(9):

01110001 00101001 10001110 11111101 (Ciphertext)
01101011 01000000 11100011 11110101 (Keystream) $\oplus$
(9)

* Corresponding author

**Journal of Computer Networks, Architecture and High Performance Computing**
Volume 7, Number 1, Januari 2025
https://doi.org/ 10.47709/cnahpc.v7i1.5345

**Submitted** : Jan 17, 2025
**Accepted** : Jan 30, 2025
**Published** : Jan 31, 2025

The result of the XOR operation is the original plaintext:
01011010 01101001 01100001 01100100
This is the original message that you have successfully decrypted from the ciphertext. In the decryption process, it is important to use the same keystream used during encryption to get the correct result. Thus, both the receiver and the sender must have access to the same keystream and the encryption key used during the encryption process.
Changing this utf-8 binary format to characters:
Here is the "Ziad" character conversion from the UTF-8 binary format:
Character 'Z' (01011010 in binary)
Character 'i' (01101001 in binary)
Character 'a' (01100001 in binary)
Character 'd' (01100100 in binary)
So, in UTF-8 binary format: 01011010 01101001 01100001 01100100 is "Ziad"

## RESULT

Based on the results of the research that has been conducted, it can be concluded that the integrated application of ChaCha20 and RSA cryptographic algorithms is an effective solution to improve the security of digital texts. The ChaCha20 algorithm, with its lightweight design and high efficiency, is able to provide a fast encryption process without sacrificing a level of security. Meanwhile, the RSA algorithm offers an edge in key management, particularly in public and private key exchange mechanisms, which ensures data remains protected even in environments that are vulnerable to threats. The combination of these two algorithms results in an encryption system that is not only strong but also flexible in various usage contexts, whether for personal communications, business applications, or other security systems. This study shows that the integration of ChaCha20 and RSA is able to answer the challenges of modern encryption needs that demand speed and high security.

Thus, this research has succeeded in achieving its main goal, which is to contribute to the development of reliable and practically implementable data protection methods. In addition, these findings open up opportunities for further research, such as the exploration of other supporting algorithms or applications to non-textual data, to further expand the benefits of the solutions that have been developed.

## DISCUSSIONS

This study uses ChaCha20 and RSA algorithms as a combination of cryptography for text security. Further research can explore other symmetrical and asymmetric algorithms to compare efficiency and safety in different scenarios. In addition, the application of non-text data, such as images, videos, or audio, is relevant for the protection of multiformat data. The system can also be tested on more complex environments, such as cloud applications or distributed systems, to evaluate performance at scale. Algorithm optimization in terms of execution time and resources is important, especially for IoT or mobile devices with limited power. More in-depth security testing, such as simulated brute force or man-in-the-middle attacks, is also needed to ensure the system's resilience to cyber threats.

## CONCLUSION

Based on the results of the research that has been conducted, it can be concluded that the integrated application of ChaCha20 and RSA cryptographic algorithms is an effective solution to improve the security of digital texts. The ChaCha20 algorithm, with its lightweight design and high efficiency, is able to provide a fast encryption process without sacrificing a level of security. Meanwhile, the RSA algorithm offers an

* Corresponding author

**Journal of Computer Networks, Architecture and High Performance Computing**
Volume 7, Number 1, Januari 2025
https://doi.org/ 10.47709/cnahpc.v7i1.5345

**Submitted** : Jan 17, 2025
**Accepted** : Jan 30, 2025
**Published** : Jan 31, 2025

edge in key management, particularly in public and private key exchange mechanisms, which ensures data remains protected even in environments that are vulnerable to threats. The combination of these two algorithms results in an encryption system that is not only strong but also flexible in various usage contexts, whether for personal communications, business applications, or other security systems. This study shows that the integration of ChaCha20 and RSA is able to answer the challenges of modern encryption needs that demand speed and high security. Thus, this research has succeeded in achieving its main goal, which is to contribute to the development of reliable and practically implementable data protection methods. In addition, these findings open up opportunities for further research, such as the exploration of other supporting algorithms or applications to non-textual data, to further expand the benefits of the solutions that have been developed.

## ACKNOWLEDGMENT

I would like to express my gratitude to Mrs. Nurul Hayaty, S.T., M.Cs., as the first supervisor, and Mrs. Martaleli Bettiza, S.Si, M.Sc., as the second supervisor as well as the Dean of the Faculty of Maritime Engineering and Technology, Raja Ali Haji Maritime University, for their guidance, direction, and support that is very meaningful in this research. I am also grateful to Ahmad Zulfikar, my good friend, who has helped and encouraged me so that this research can be completed properly.

## REFERENCES

Forouzan, B. A. (2008). *Introduction to cryptography and network security*. New York, America: McGraw-Hill Higher Education.

KEBANDE, V. R. (2023). Extended-Chacha20 Stream Cipher With Enhanced Quarter Round Function, 11. doi: 10.1109

Enung, N. (2016). APLIKASI ENKRIPSI DAN DEKRIPSI MENGGUNAKAN ALGORITMA RSA BERBASIS WEB

Campbell, T. R. (2020). Daence: Salsa20 and ChaCha in deterministic authenticated encryption with no noNCEnse. Cryptology ePrint Archive. Retrieved from https://eprint.iacr.org/2020/067

Siregar, S. J., Nugroho, N. B., & Sigalingging, H. (2023). Implementasi algoritma kriptografi RSA (Rivest Shamir Adleman) dalam pengamanan data gaji karyawan di kantor BSPJI. Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika dan Komputer), 22(2), 528–538.

Chyquitha, D. (2018). Pengamanan Data Melalui Cloud Computing Dengan Integrasi Steganografi LSB Dan Kriptografi Vigenere Key Berbasis Android.

Liu, Y., Gong, W., & Fan, W. (2018). Application of AES and RSA hybrid algorithm in e-mail. In Proceedings of the 2018 IEEE International Conference on Information Security (ICIS). IEEE. https://doi.org/10.1109/ICIS.2018.00000

Muhammad, A. (2019). Implementasi Algoritma Enkripsi Blowfish pada Sistem Revisi Anggaran Berbasis Online (studi kasus : perpustakaan nasional republik indonesia).

* Corresponding author

**Journal of Computer Networks, Architecture and High Performance Computing**
Volume 7, Number 1, Januari 2025
https://doi.org/ 10.47709/cnahpc.v7i1.5345

**Submitted** : Jan 17, 2025
**Accepted** : Jan 30, 2025
**Published** : Jan 31, 2025

Novita P. D, dkk. (2022). pengamanan data dengan kriptografi hibrida algoritma HILL CIPHER dan Algoritma LUC Serta Steganografi Chaotic LSB.

Fadhil, F. A., Alhilo, F. T. A. H., & Abdulhadi, M. T. (2024). Enhancing data security using Laplacian of Gaussian and ChaCha20 encryption algorithm. Journal of Intelligent Systems, 33, 20240191.

Cai, W., Chen, H., Wang, Z., & Zhang, X. (2021). Implementation and optimization of ChaCha20 stream cipher on Sunway TaihuLight supercomputer. The Journal of Supercomputing. https://doi.org/10.1007/s11227-021-04023-9

Minggu, D., Nurpratama, R. S., & Setiawan, R. B. (2024). Pengembangan kunci enkripsi asimetris pada Vigenère cipher. JATI (Jurnal Mahasiswa Teknik Informatika), 8(6), [article number]. https://doi.org/10.36040/jati.v8i6.11968

Dola R, dkk.(2024). Strategi Perlindungan Data Menggunakan Sistem Kriptografi Dalam Keamanan Informasi

Ramalinda, D., Jayadi, & Raharja, A. R. (2024). Strategi perlindungan data menggunakan sistem kriptografi dalam keamanan informasi. Journal of International Multidisciplinary Research, 2(6), 665–671

.

Pramudita, K. E. (n.d.). Brute force attack dan penerapannya pada password cracking. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

Mondal dan Primajaya(2024). THE IMPLEMENTATION OF RSA ALGORITHM IN TEXT MESSAGES ENCRYPTION AND DECRYPTION

Ariel H, dkk.(2015). Cryptography for Big Data Security Book Chapter for Big Data: Storage, Sharing, and Security (3S)

* Corresponding author