Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

15th October 2025. Vol. 5. No. 1; e-ISSN: 2808-4519

RSA Algorithm Measurement Levels in Ms.Word Security

Nabila Husna Rabiulia[1]*, Achmad Fauzi[2], Marto Sihombing[3]

[1,2,3,]*Informatics Engeneering Program, STMIK Kaputama*
nabilarabiuliahusna@gmail.com[1*], fauzyrivai88@gmail.com [2], martosihombing45gmail.com[3]

Abstract

With the rapid rise of information technology in Indonesia, the risks associated with it, such as the leakage or theft of sensitive data, are increasingly apparent. Among the most frequently shared file formats, Microsoft Word documents are a crucial factor. Therefore, this study aims to implement and evaluate the performance of the RSA cryptographic algorithm to protect these types of documents. The applied methodology includes the design and implementation of a system using PHP, a data farm using MySQL, and an Xampp-based test environment. We utilized RSA to encrypt and, conversely, decrypt the Word files, with the main indicator being the processing time for each type of process. Theoretical analysis and manual calculations confirmed that RSA operates through the risk inherent in the difficulty of factoring large prime numbers. Manual simulations of encryption and decryption times verified that the RSA algorithm, when inverted, produces data in the correct format. Based on the results obtained, we conclude that the use of RSA in securing Word is feasible and appropriate, and the degree of protection can now be evaluated through the time required for each encryption and decryption operation.

Keywords: *RSA_Algorithm, Word_File,Measurement_Level*

1. Introduction

The development of security systems and information technology is growing at lightning speed [1]. In Indonesia, the surge in science and technology, reflected in government investment in the technology and start-up sectors, is driving society to increasingly rely on information and communication resources that require security and confidentiality. Although technology facilitates the flow of information, these health cases, threats are increasingly exposed, especially in Word-based documents which are statistically often the target of misuse, family up and news misuse, data that data orika data that. This research builds on previous results that provide strong justification and reference base. have shown that the RSA algorithm is capable of securing files by converting plaintext to ciphertext without losing information. Further explored document security mechanisms in web-based systems through cryptography, and then implemented and tested RSA in a web application, measuring its impact on large data sets. [2] proposed a security approach that combines the XOR method with RSA to provide an additional layer of security, while [3] applied AES-256 in the same context, showing that this symmetric algorithm is significantly faster than DES, Blowfish, in similar tests. Furthermore, the citation by in the narrative of [4] is an important reference that confirms the effectiveness of RSA with a 1024-bit key in securing relatively large objects. This study attempts to fill the remaining gap by examining RSA's performance on Microsoft Word files over a more detailed time measurement. Meanwhile, Bhakti et al. (2024) measured the performance of symmetric algorithms such as AES and found significant speedups in encryption. However, studies specifically evaluating the performance of RSA as an asymmetric algorithm in the context of securing MS Word files in terms of processing time are still limited.[5]. Meanwhile, [6] developed a web-based document security system with a cryptographic approach, highlighting the importance of security integration in cloud-based applications.[7] emphasize the urgent need for specialized cryptography to protect Word files due to the rise in sensitive data theft.

2. Research Methods

2.1. System Analysis

This is how the author intends to conduct research for the purpose of completing the final project:

1. Conduct a study on the application of data security and classical cryptographic encryption and decryption techniques through books or online resources.
2. Implement the RSA algorithm into a web programming language and other application tools. To measure the speed of MS Word file-based security using the RSA algorithm.
3. Run the program to obtain the results of the data security application interface and check for possible bugs.
4. Modify the application if the program contains errors.
5. Conduct experiments on the program to improve the results.

The problem this system aims to solve is protecting confidential Word (*.doc) files. In this approach, each document is encrypted using the RSA algorithm and a private key to prevent unauthorized reading. After encryption, the system measures the file's security level and performs tests to determine its true security against threats.

2.2. File Word

Microsoft Word is a word processing software developed by Microsoft that is used to create, prepare, and format text-based documents. It is part of the Microsoft Office suite. It has comprehensive features and is very helpful in processing documents for various purposes, both in education and work.

Fig. 1: Sample Convertion Encryption and Decryption Calculations

2.3. RSA Algoritm

In cryptography, RSA is an algorithm for public-key encryption. It was the first algorithm known to be suitable for both signing and encryption, and one of the first major breakthroughs in public-key cryptography. RSA remains widely used in protocols and electronic commerce, and is considered highly secure due to its sufficiently long keys and its up-to-date implementations.

2.3.1 . Improved Properties of the RSA Algorithm

Quantities used in the RSA Algorithm:

a. p and q are prime numbers (secret)
b. n = p × q (not secret)
c. f (n) = (p – 1) (q – 1) (secret)
d. e (encryption key) (not secret)
e. d (decryption key) (secret)
f. m (plainfile) (secret)
g. c (cipherfile) (not secret)

Based on the equation, encryption and decryption are formulated as follows:

a. Encryption:

$$C(Chiperfile) = P(Plainfile)^{e} \ \textbf{Mod}\ N$$

Plainfile: The original file that can be modified
e: The public key used for encryption
N: The product of two numbers p and q

b. Decryption:

$$P(Plainfile) = C(Chiperfile)^{d} \ \textbf{Mod}\ N$$

Chiperfile:This is the message file that will be converted to its original form.
d: This is the private key used for decryption.
N: This is the product of two numbers p and q.

3. Results And Discussion

In designing this file security measurement system, the author used the *Rivest Shamir Adleman (RSA)* algorithm to solve the research problem. The system design used a flowchart to understand how the encryption and decryption processes would be implemented in the application.

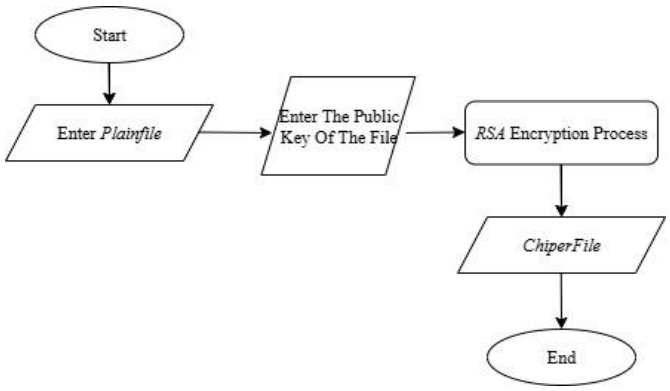

Fig. 2: Flowchart of the *RSA* Algorithm Decryption Process

The steps in the encryption process in this algorithm are as follows:

1. The *PlainFile* is encrypted with the *RSA* algorithm to produce a *ChiperFile* as a temporary *ChiperFile.*
2. The *ChiperFile* is reprocessed using the *RSA* algorithm to produce the final *ChiperFile,* which also serves as the *ChiperFile.*

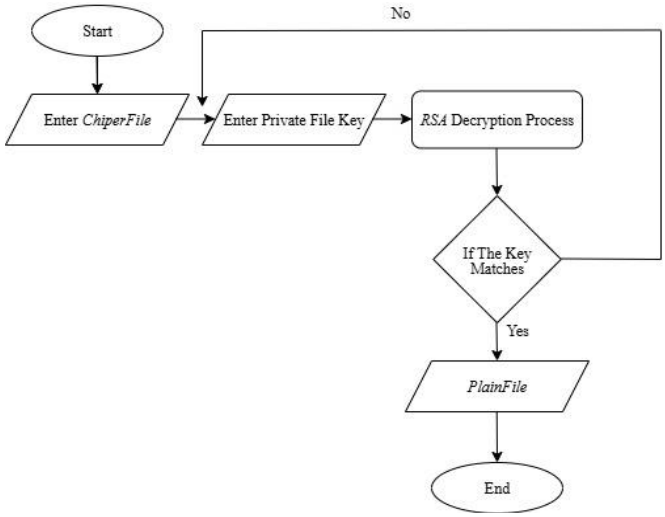

Fig. 3: *RSA* Algorithm Decryption Process Flowchart

The steps in the decryption process in this algorithm are as follows:

1. The *CiperFile* is decrypted using the *RSA* algorithm, resulting in a temporary *PlainFile* output.
2. The *PlainFile* is decrypted again using the *RSA* hexadecimal process to produce the final *PlainFile* as well as the *Plainfile* referred to at the beginning.

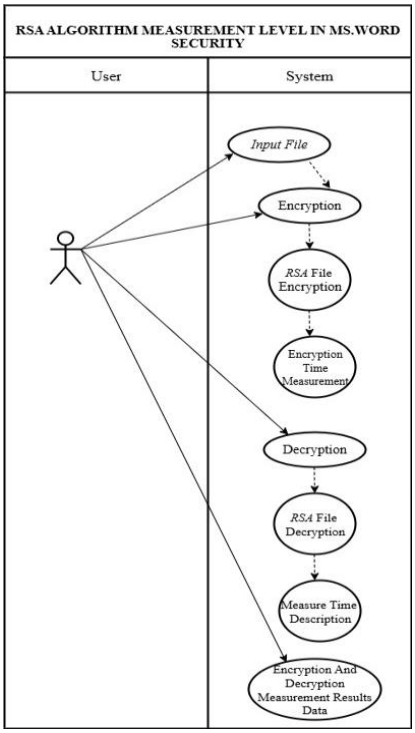

Fig. 4: Use Case System Diagram

A Use Case Diagram is a model used to describe the behavior of an application to be created. A use case diagram describes the interaction between one or more actors and the system being built. The processes depicted will occur in a structured manner. Use case diagram of a cryptographic system for file security. This use case explains how to encrypt and decrypt files using the RSA algorithm to encode a PlainFile whose confidentiality is to be maintained. It then measures the security level of the file encryption and decryption using the RSA algorithm.

3.1. Interface Design

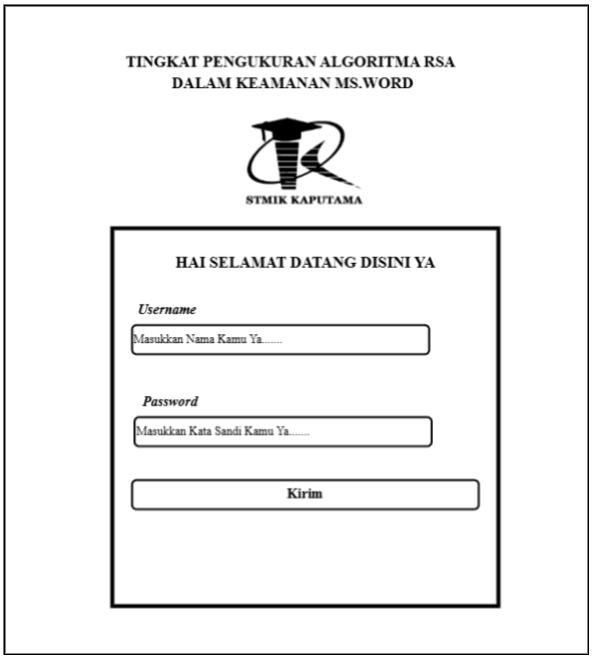

Fig. 5: *Login* Form Interface Design

This section explains the design of the login page display, where the components are two text fields with the function of inputting a username and password, and one button for the login process to the application Information:

1. Enter Username
2. Enter Password
3. Press the button to submit to the Main Menu Form

3.2. Main Page Form Design

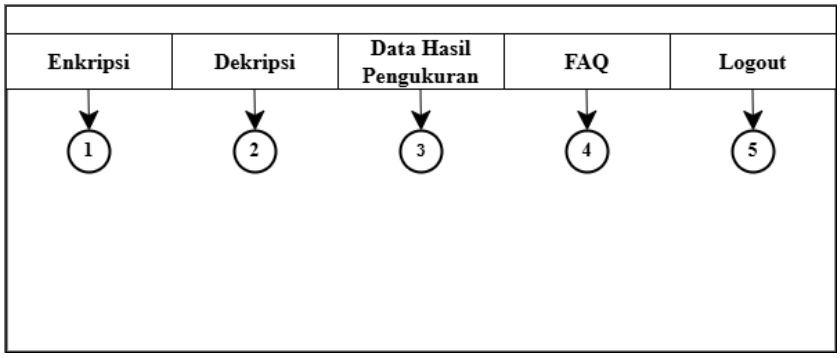

Fig. 6: Main Page Form Interface Design

The image shows the design of the first page which has many menus for testing using web-based applications.

3.3. Software Requirements

In this context, software includes all programs written in a language directly understood by the processing unit. Without instructions from software, none of the physical components in a computer machine could perform any task. Cutting-edge hardware technology is ready to operate only after receiving the appropriate set of instructions. Each instance of software is written by a human developer to translate commands into patterns that can be executed by the physical components. The software used in this lab will be Windows Server or Windows 10, depending on the specifications and purpose of each computer for the initial operating system.

3.4. Encryption Process

Before carrying out the encryption process, the original file (plainfile) is a hexadecimal number value that must be changed into a decimal ASCII value first, as follows:

Table 1: Decimal values in the table

PlainFile Hexadesimal	**6F**	**6E**	**74**	**65**	**6E**	**74**	**5F**	**54**	**79**	**70**	**65**	**73**	**5D**	**2E**	**78**	**6D**
Ascii Desimal	111	110	116	101	110	116	95	84	121	112	101	115	93	46	120	109

ASCII DESIMAL"111110116101110116958412111210111593461201 09".

Table 2: Manual RSA Algorithm Encryption Calculation Simulation

No	**Ascii Decimal**	**Process**	**Results**	
			ChipherFile	**Character**
1	111	C = P^e mod n = 111^5 mod 161 = 34	34	“
2	110	C = P^e mod n = 110^5 mod 161 = 3	3	ETX
3	116	C = P^e mod n = 116^5 mod 161 = 93	93	]
4	101	C = P^e mod n = 101^5 mod 161 = 54	54	6
5	110	C = P^e mod n = 110^5 mod 161 = 3	3	ETX
6	116	C = P^e mod n = 116^5 mod 161 = 93	93	]
7	95	C = P^e mod n = 95^5 mod 161 = 128	128	Ç
8	84	C = P^e mod n = 84^5 mod 161 = 7	7	BEL
9	121	C = P^e mod n = 121^5 mod 161 = 25	25	EM
10	112	C = P^e mod n = 112^5 mod 161 = 56	56	8
11	101	C = P^e mod n = 101^5 mod 161 = 54	54	6
12	115	C = P^e mod n = 115^5 mod 161 = 138	138	**è**
13	93	C = P^e mod n = 93^5 mod 161 = 116	116	t

14	46	C = P^e mod n = 46^5 mod 161 = 23	23	ETB
15	120	C = P^e mod n = 120^5 mod 161 = 43	43	+
16	109	C = P^e mod n = 109^5 mod 161 = 44	44	,

3.5 Decryption Process

After the encryption process, the original file (plainfile) is converted into the decrypted result. The resulting decimal number (cipherfile) is recalculated to obtain the original character (plainfile) from the encryption. The decryption process is as follows:

Table III.5: Decimal values in the table

ChiperFile	**34**	**3**	**93**	**54**	**3**	**93**	**128**	**7**	**25**	**56**	**54**	**138**	**116**	**23**	**43**	**44**
Character	“	ETX	]	6	ETX	]	Ç	BEL	EM	8	6	**È**	T	ETB	+	,

Chiperfile "343935439312872556541381162343**44"**

following is a display of the simulation of the RSA Algorithm Decryption calculation on **Table III.5:**

Table III.5: Manual RSA Algorithm Decryption Calculation Simulation

No	**Encryption Results**		**Process**	**Decryption Results**	
	Character	**ChiperFile**		**Hexadecimal Process**	**PlainFile Hexadesimal**
1	“	34	P = C^d mod n = 34^{53} mod 161 = 111	111÷ 16 = **6** (remainder 15**(F)**)	**6F**
2	ETX	3	P = C^d mod n = 3^{53} mod 161 = 110	110 ÷ 16 = **6** (reminber 14**(E)**)	**6E**
3	]	93	P = C^d mod n = 93^{53} mod 161 = 116	116 ÷ 16 = **7** (reminber **4**)	**74**
4	6	54	P = C^d mod n = 54^{53} mod 161 = 101	101÷ 16 = **6** (reminber **5**)	**65**
5	ETX	3	P = C^d mod n = 3^{53} mod 161 = 110	110÷ 16 = **6** (reminber 14**(E)**)	**6E**
6	]	93	P = C^d mod n = 93^{53} mod 161 = 116	116 ÷ 16 = **7** (reminber **4**)	**74**
7	Ç	128	P = C^d mod n = 128^{53} mod 161 = 95	95 ÷ 16 = **5** (reminber 15**(F)**)	**5F**
8	BEL	7	P = C^d mod n = 7^{53} mod 161 = 84	84 ÷ 16 = **5** (reminber **4**)	**54**
9	EM	25	P = C^d mod n = 25^{53} mod 161 = 121	121 ÷ 16 = **7** (reminber **9**)	**79**

10	8	56	P = C^d mod n = 56^{53} mod 161 = 112	112 ÷ 16 = **7** (reminber **0**)	**70**
11	6	54	P = C^d mod n = 54^{53} mod 161 = 101	101 ÷ 16 = **6** (reminber **5**)	**65**
12	è	138	P = C^d mod n = 138^{53} mod 161 = 115	115 ÷ 16 = **7** (reminber **3**)	**73**
13	T	116	P = C^d mod n = 116^{53} mod 161 = 93	93 ÷ 16 = **5** (reminber13**(D)**)	**5D**
14	ETB	23	P = C^d mod n = 23^{53} mod 161 = 46	46 ÷ 16 = **2** (reminber 14**(E)**)	**2E**
15	+	43	P = C^d mod n = 43^{53} mod 161 = 120	120 ÷ 16 = **7** (reminber **8**)	**78**
16	,	44	P = C^d mod n = 44^{53} mod 161 = 109	109 ÷ 16 = **6** (reminber 13**(D)**)	**6D**

4. System Implementation

In this case, after designing a system to measure customer satisfaction levels, the results of the design are as follows:

4.1 Login Form Display

Fig.7 : Login Form Display

displays the login form. The user must first enter their username and password. If the information matches the previous information and password, the user can click "Login." This allows them to access the main menu. The web program is then ready to run.

4.2 Encryption Form Display

Fig.8 : Encryption Form Display

It serves as an educational tool and a simple demo of the RSA encryption process for practical learning. Interactive, responsive, and easy-to-understand, this application makes it easy for users to enter public key values, obtain prime factors p and q, calculate key pairs, and encrypt and download Word documents. Word document encryption can be done by calculating the public and private keys. This application also calculates public and private keys. It also includes RSA public key factorization. It also includes reset, help, and logout confirmation features to make it easier for users.

4.3 Decription Form Display

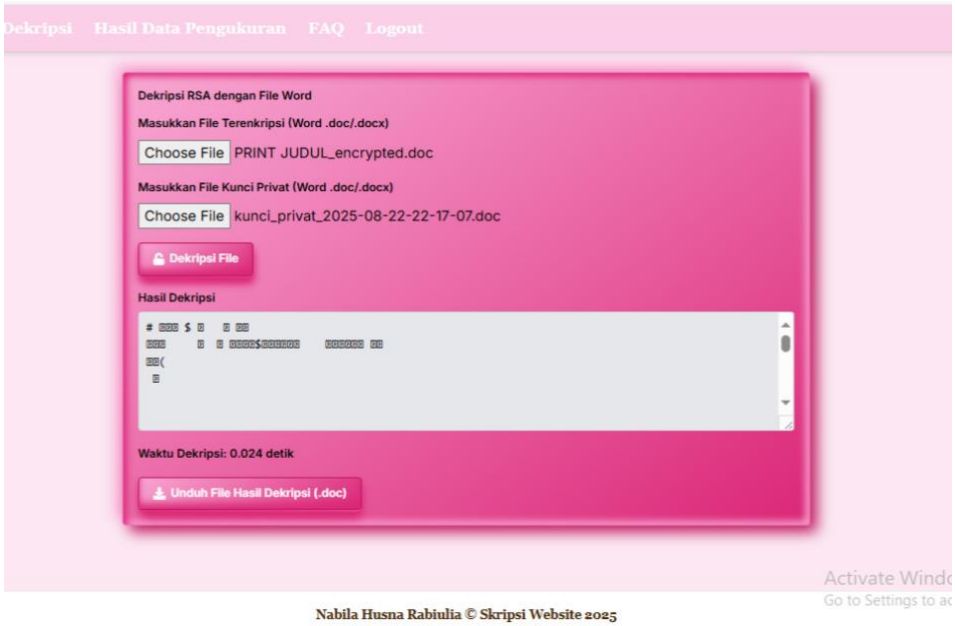

Fig.9 : Decription Form Display

This program is a web application that offers the ability to decrypt Word documents (files with the .doc and .docx extensions) encrypted using the RSA algorithm. Users can upload either the encrypted file or the private key file for decryption directly in the browser. Users can view and download the decrypted results in Word format. Responsive and interactive buttons make decrypting and downloading files easier.

4.4 Measurement Result Data Form Display

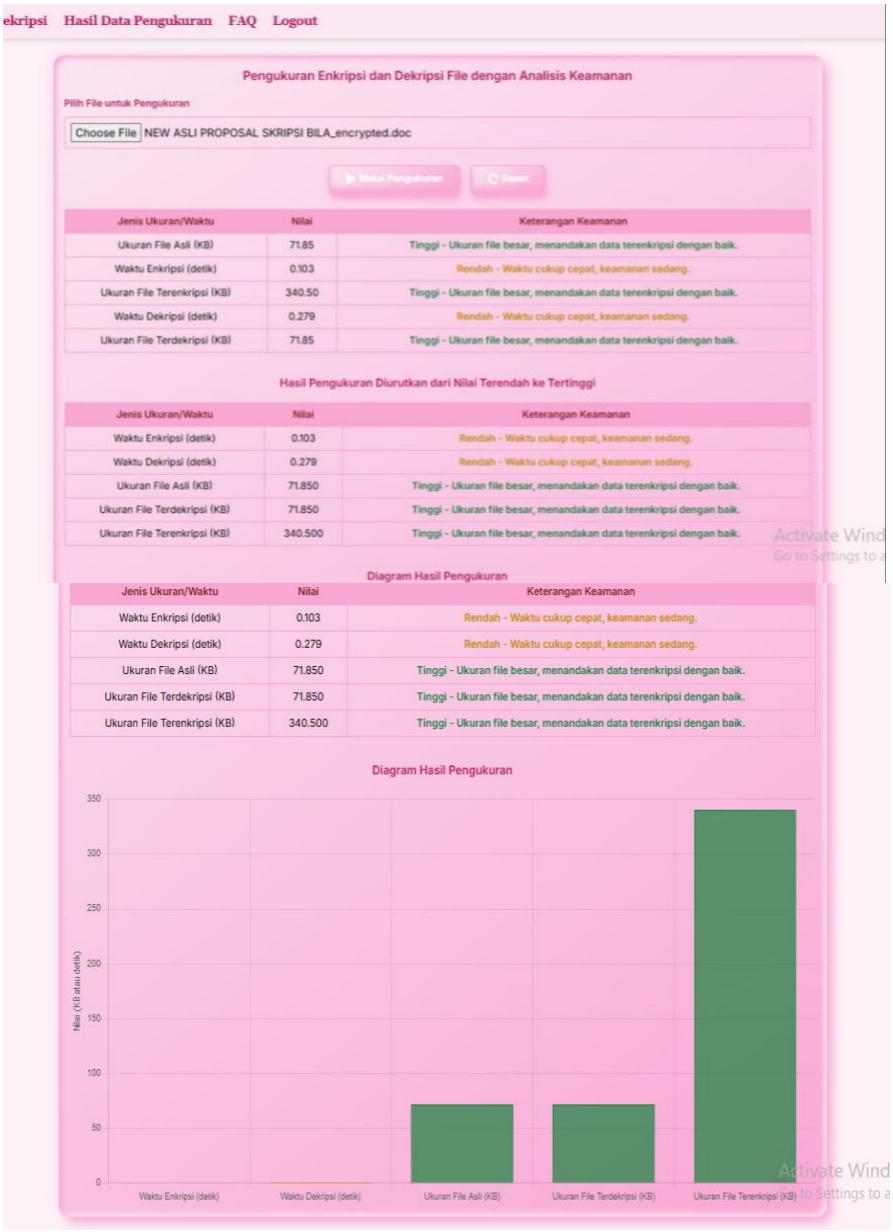

Fig.10 : Measurement Result Data Form Display

File encryption and decryption performance measurements using a simple RSA algorithm are presented interactively. Users can select a file, and the system automatically calculates various parameters such as the original file size, encryption time, encrypted file size, decryption time, and decrypted file size. Each measurement result is then analyzed graphically and in tables, with complex security estimates based on the encryption and decryption time and file size. Measurements can be repeated using the reset feature. Users can easily, visually, and concisely understand the efficiency and security of the encryption-decryption system thanks to the responsive design and appropriate use of color in the measurement results.

5. Conclusion

The RSA algorithm has been proven to be effective in encrypting and decrypting Microsoft Word files in .docx format, thus maintaining the confidentiality of sensitive data stored within them. Performance testing shows that the time required for encryption and decryption is directly proportional to the file size; larger files obviously require more time, reflecting the inherent computational nature of RSA. Implementing this layer of security provides effective protection for users to prevent unauthorized access to Word documents, provided that the private key must be stored in a highly secure location and out of the reach of unauthorized parties.

6. Sugesstion

InIntegration with Additional Algorithms for Increased Efficiency: RSA can be beneficial when combined with symmetric cryptography such as AES. Here, RSA doesn't handle the entire message, but rather generates and encrypts the symmetric key that will be used to encrypt large files. Proper code optimization is also crucial; savings are achieved by simplifying and speeding up modular exponentiation operations, especially when handling large prime numbers. Security Testing: Future research, for robustness, requires conducting a thorough security audit, mapping the entire environment, and verifying every point and process. These findings will help eliminate any missed vulnerabilities. Continue exploring other platforms, with a focus on leveraging platforms beyond the web. Prototype desktop

applications and native mobile versions to expand reach. Users are not tied to large screens and institutions, but rather use it on their everyday smartphones, adapting to existing habits.

References

[1] Indriani, et al, 2021, "Application of the RSA Algorithm in Ms. Word file security".
[2] Refialy 2022, "Design and Construction of Document Delivery Application Using a Combination of the RSA Algorithm and the XOR
[3] Algorithm".
[4] Febrianto et al. 2023, "Implementation of Advanced Encryption Cryptographic Algorithm".
[5] Darip et al. 2025, "Analysis and Implementation of the RSA Algorithm for Security of Payment Transactions in the System E-Commerce at Bangun Jaya Store".
[6] Bhakti et al, 2024, "Implementation of plaintext cryptography using RSA".
[7] Agustina, 2017, "Document security using a combination of RSA and Vigenere Cipher methods.
[8] Siddik et al, 2020, "Design and Construction of a POS (Point of Sale) Information System for Cashiers Using Object-Oriented Programming Language Concepts".