International Journal Of Health, Engineering And Technology (IJHET) *E-ISSN 2829 - 8683*
Volume 5, Number 2, July 2026, *Page. 2333 - 2345*
Email : editorijhess@gmail.com

Document File Security Using The Rsa Algorithm

Yusuf Ramadhan Nasution[1)], Suhardi[2)], Rahmadani Nasution[3)]
[1,2,3)] Department of Computer Science, Faculty of Science and Technology, Universitas Islam Negeri Sumatera Utara, Medan, Indonesia

*Corresponding Author
E-mail: ramadhannst@uinsu.ac.id

Abstract

The rapid development of digital technology has increased the need for secure document storage and transmission, especially for confidential information. This study aims to develop a web-based document security application using the RSA (Rivest–Shamir–Adleman) algorithm to protect document contents from unauthorized access. The research method used is Research and Development (R&D) with the Software Development Life Cycle (SDLC) Waterfall model, consisting of planning, analysis, design, implementation, and testing stages. The application was developed using PHP and implemented to secure DOCX, PDF, and XLSX document formats with a maximum file size of 2 MB. The RSA algorithm was applied through encryption using a public key and decryption using a private key. System testing was performed using Black-box Testing and Avalanche Effect analysis to evaluate application performance. The results show that the application successfully encrypts and decrypts document files while maintaining the original document contents after the decryption process. The Avalanche Effect testing obtained an average value of 49.99%, indicating that the RSA algorithm provides adequate security in protecting document contents. Therefore, the developed application can be used as an alternative solution for securing confidential digital documents.

Keywords: Anemia, Pregnant Women, Knowledge.

INTRODUCTION

The rapid advancement of technology has significantly increased the exchange of digital data through the internet, creating greater security risks during data transmission. Data security, particularly for confidential document files owned by organizations, has become an essential concern. One of the fundamental aspects of document security is ensuring the authenticity, integrity, and confidentiality of information so that the content remains consistent with the original document.

To prevent unauthorized access, modification, or misuse of sensitive information, an effective computer security mechanism is required. One of the most widely adopted approaches is cryptography, which provides data protection through encryption and decryption techniques. Cryptography transforms original information (plaintext) into unreadable ciphertext using a specific algorithm and key, and subsequently restores it to its original form through the decryption process using the corresponding key (Rambe et al., 2019).

The importance of maintaining confidentiality is also emphasized in Islam. As stated in Qur'an Surah An-Nisa verse 58, Allah commands believers to deliver trusts to those entitled to receive them and to uphold justice in all matters. This teaching highlights the obligation to safeguard confidential information and maintain the integrity of entrusted data.

To enhance document security, this study implements the RSA (Rivest–Shamir–Adleman) algorithm as a cryptographic solution. Developed by Rivest, Shamir, and Adleman in 1976, RSA is an asymmetric cryptographic algorithm that employs a pair of keys consisting of a public key for encryption and a private key for decryption. As an asymmetric encryption algorithm, RSA ensures that encrypted data cannot be interpreted without the appropriate private key (Prasetyo et al., 2018).

The security of RSA relies on the computational difficulty of factoring large prime numbers into smaller factors. During key generation, two keys are produced: a public key for encrypting plaintext into ciphertext and a private key for decrypting ciphertext back into its original plaintext. The robustness of RSA is particularly evident against brute-force attacks, as increasing the values of the prime numbers p and q increases the computational complexity required to derive the private key, thereby enhancing security.

Previous studies have demonstrated the effectiveness of RSA in securing digital information, including loan and savings data, through applications developed using Visual Basic. The implementation successfully encrypted document contents into unreadable ciphertext, which could only be restored to the original plaintext through the correct decryption process.

Based on these considerations, this research aims to develop a web-based application using PHP that implements the RSA algorithm to secure document files in DOCX, PDF, and XLSX formats with a maximum file size of 2 MB. The application performs both encryption and decryption processes using RSA and evaluates its security performance through the Avalanche Effect. The study seeks to analyze the working mechanism of the RSA algorithm in protecting document contents while providing a practical solution to prevent unauthorized access and misuse of confidential digital documents.

RESEARCH METHODS

This research was conducted at the Computer Science Laboratory, Faculty of Science and Technology, Universitas Islam Negeri Sumatera Utara, Medan, Indonesia, during the 2022–2023 academic year. The study employed a Research and Development (R&D) approach to develop and evaluate a web-based application for securing document files using the RSA cryptographic algorithm.

The research utilized both hardware and software resources. The hardware consisted of an Acer Aspire 5 A514-53-3852 laptop equipped with an Intel® Core™ i3-1005G1 processor, 8 GB DDR4 RAM, and a 14-inch display. The software environment included Windows 11 Home Single Language, Google Chrome, XAMPP, and PHP as the primary programming language for system development.

Data were collected through library research and literature studies by reviewing books, e-books, theses, and scientific journals related to cryptography, data security, and the RSA algorithm. These references served as the theoretical foundation and comparison for system development.

The system was developed using the Software Development Life Cycle (SDLC) with the Waterfall model, consisting of planning, analysis, design, implementation, and testing phases. During the planning stage, research objectives and system requirements were identified. The analysis stage examined document security issues and gathered the information required for implementing RSA encryption. In the design stage, the system architecture and workflow were modeled using flowcharts, including system flow, RSA key generation, encryption, and decryption processes.

The RSA algorithm was implemented by generating two large prime numbers (p and q) to produce a public key (n, e) and a private key (n, d). The public key was used to encrypt plaintext into ciphertext, whereas the private key restored the ciphertext to its original plaintext. The security of the algorithm depends on the computational difficulty of factoring large prime numbers, making unauthorized decryption highly impractical.

The developed application was implemented as a web-based system using PHP. The application supports the encryption and decryption of DOCX, PDF, and XLSX document files with a maximum file size of 2 MB. After implementation, system functionality was verified through Black-box Testing to ensure that every feature operated according to the system requirements. In addition, the cryptographic performance of the RSA implementation was evaluated using the Avalanche Effect to measure the effectiveness of the encryption process in producing significant output changes from small input modifications.

RESULTS AND DISCUSSION

Discussion

This section presents the research results and discusses the developed application. The study produced an application capable of securing document contents using the RSA algorithm. The discussion explains how the application addresses the research objectives and problem formulation.

Data Analysis

The data analysis describes the criteria used to implement the RSA algorithm for document security. These criteria were determined through literature studies and served as the basis for the encryption and decryption processes. The application was tested using 15 documents consisting of three file formats (DOCX, PDF, and XLSX), with five documents representing each format.

Table 1. Documents Used for Testing

No.	Document Name	Size	Document Type
1	Biodata Mahasiswa	12.7 KB	DOCX
2	Daftar Pustaka	13 KB	DOCX
3	Quiz_Rahmadani	19 KB	DOCX
4	Uts Sistem Pakar	23 KB	DOCX
5	Uas_Ik5_0701181085_Rahmadaninasution	124 KB	DOCX
6	Cover Dani	32.5 KB	PDF
7	Abstrak Dani	181 KB	PDF
8	Bab Ii Dani Praktikum Metode Numerik	495 KB	PDF
9	Praktikum Metode Numerik	982 KB	PDF
10	Bab Iv Dani	1.24 MB	PDF
11	Data VII 6	10.2 KB	XLSX
12	Pdrb Lapangan Usaha Berdasarkan Angka Berlaku Spt	10.9 KB	XLSX
13	Data Siswa Mtsn2	52.7 KB	XLSX
14	Daftar Pengeluaran Riil Padang Sidimpuan	57.8 KB	XLSX
15	Data Penjualan	112 KB	XLSX

Manual RSA Calculation

The manual calculation illustrates the RSA encryption and decryption procedures. Two prime numbers were selected, namely p = 11 and q = 23. The modulus was calculated as $n = p \times q = 253$, while Euler's totient value was $\varphi(n) = (p - 1)(q - 1) = 220$. The public exponent satisfying gcd(e,220)=1 was e = 3, and the private exponent was d = 367, resulting in the public key (3,253) and private key (367,253).

The encryption process applied the formula $C = M^e \bmod n$, where each character of the document name was converted into its ASCII value before encryption. Three representative document formats (DOCX, PDF, and XLSX) were used to demonstrate the encryption process.

Table 2. Manual Encryption Calculation for DOCX Document

Text	ASCII	Encryption Process *($C = M^3 \bmod 253$)*
D	68	$68^3 \bmod 253 = 314432 \bmod 253 =$ **206**
A	65	$65^3 \bmod 253 = 274625 \bmod 253 =$ **120**
F	70	$70^3 \bmod 253 = 343000 \bmod 253 =$ **185**
T	84	$84^3 \bmod 253 = 592704 \bmod 253 =$ **178**
A	65	$65^3 \bmod 253 = 274625 \bmod 253 =$ **120**
R	82	$82^3 \bmod 253 = 551368 \bmod 253 =$ **81**
Space	32	$32^3 \bmod 253 = 32768 \bmod 253 =$ **131**
P	80	$80^3 \bmod 253 = 512000 \bmod 253 =$ **181**
U	85	$85^3 \bmod 253 = 614125 \bmod 253 =$ **94**
S	83	$83^3 \bmod 253 = 571787 \bmod 253 =$ **17**
T	84	$84^3 \bmod 253 = 592704 \bmod 253 =$ **178**

A	65	65^3 mod 253 = 274625 mod 253 = **120**
K	75	75^3 mod 253 = 421875 mod 253 = **124**
A	65	65^3 mod 253 = 274625 mod 253 = **120**
.	46	46^3 mod 253 = 97336 mod 253 = **184**
d	100	100^3 mod 253 = 1000000 mod 253 = **144**
o	111	111^3 mod 253 = 1367631 mod 253 = **166**
c	99	99^3 mod 253 = 970299 mod 253 = **44**
x	120	120^3 mod 253 = 1728000 mod 253 = **10**

Table 3. Manual Encryption Calculation for PDF Document

Text	ASCII	Encryption Process *($C = M^3$ mod 253)*
A	65	68^3 mod 253 = 314432 mod 253 = **206**
B	66	65^3 mod 253 = 274625 mod 253 = **120**
S	83	70^3 mod 253 = 343000 mod 253 = **185**
T	84	84^3 mod 253 = 592704 mod 253 = **178**
R	82	65^3 mod 253 = 274625 mod 253 = **120**
A	65	82^3 mod 253 = 551368 mod 253 = **81**
K	75	75^3 mod 253 = 421875 mod 253 = **124**
Space	32	32^3 mod 253 = 32768 mod 253 = **131**
D	68	68^3 mod 253 = 314432 mod 253 = **206**
a	97	97^3 mod 253 = 912673 mod 253 = **102**
n	110	110^3 mod 253 = 1331000 mod 253 = **220**
i	105	105^3 mod 253 = 1157625 mod 253 = **150**
.	46	46^3 mod 253 = 97336 mod 253 = **184**
p	112	112^3 mod 253 = 1404928 mod 253 = **19**
d	100	100^3 mod 253 = 1000000 mod 253 = **144**
f	102	102^3 mod 253 = 1061208 mod 253 = **126**

Table 4. Manual Encryption Calculation for XLSX Document

Text	ASCII	Encryption Process *($C = M^3$ mod 253)*
D	68	68^3 mod 253 = 314432 mod 253 = **206**
a	97	97^3 mod 253 = 912673 mod 253 = **102**
t	116	116^3 mod 253 = 1260896 mod 253 = **139**
a	97	97^3 mod 253 = 912673 mod 253 = **102**
Space	32	32^3 mod 253 = 32768 mod 253 = **131**
s	115	115^3 mod 253 = 1520875 mod 253 = **92**
i	105	105^3 mod 253 = 1157625 mod 253 = **150**
s	115	115^3 mod 253 = 1520875 mod 253 = **92**
w	119	119^3 mod 253 = 1685159 mod 253 = **179**
a	97	97^3 mod 253 = 912673 mod 253 = **102**
Space	32	32^3 mod 253 = 32768 mod 253 = **131**
M	77	77^3 mod 253 = 456533 mod 253 = **121**
t	116	116^3 mod 253 = 1260896 mod 253 = **139**
S	83	83^3 mod 253 = 571787 mod 253 = **7**
N	78	78^3 mod 253 = 474552 mod 253 = **177**
2	50	50^3 mod 253 = 125000 mod 253 = **18**
.	46	46^3 mod 253 = 97336 mod 253 = **184**
x	120	120^3 mod 253 = 1728000 mod 253 = **10**
l	108	108^3 mod 253 = 1259712 mod 253 = **25**
s	115	115^3 mod 253 = 1520875 mod 253 = **92**
x	120	120^3 mod 253 = 1728000 mod 253 = **10**

The decryption process used the formula $M = C^d \bmod n$ with d = 367 and n = 253. Each ciphertext value was converted back into its original ASCII value, successfully reconstructing the initial document names.

Table 5. Manual Decryption Calculation for DOCX Document

Ciphertext	Decryption Process *($M = C^{367} \bmod 253$)*
206	$206^{367} \bmod 253 =$ **68**
120	$120^{367} \bmod 253 =$ **65**
185	$185^{367} \bmod 253 =$ **70**
178	$178^{367} \bmod 253 =$ **84**
120	$120^{367} \bmod 253 =$ **65**
81	$81^{367} \bmod 253 =$ **82**
131	$131^{367} \bmod 253 =$ **32**
181	$181^{367} \bmod 253 =$ **80**
94	$94^{367} \bmod 253 =$ **85**
7	$7^{367} \bmod 253 =$ **83**
178	$178^{367} \bmod 253 =$ **84**
120	$120^{367} \bmod 253 =$ **65**
124	$124^{367} \bmod 253 =$ **75**
120	$120^{367} \bmod 253 =$ **65**
184	$184^{367} \bmod 253 =$ **46**
144	$144^{367} \bmod 253 =$ **100**
166	$166^{367} \bmod 253 =$ **111**
44	$44^{367} \bmod 253 =$ **99**
10	$10^{367} \bmod 253 =$ **120**

Table 6. Manual Decryption Calculation for PDF Document

Ciphertext	Decryption Process *($M = C^{367} \bmod 253$)*
120	$120^{367} \bmod 253 =$ **65**
88	$88^{367} \bmod 253 =$ **66**
7	$7^{367} \bmod 253 =$ **83**
178	$178^{367} \bmod 253 =$ **84**
81	$120^{367} \bmod 253 =$ **65**
120	$120^{367} \bmod 253 =$ **65**
124	$124^{367} \bmod 253 =$ **75**
131	$131^{367} \bmod 253 =$ **32**
206	$206^{367} \bmod 253 =$ **68**
102	$102^{367} \bmod 253 =$ **97**
220	$220^{367} \bmod 253 =$ **110**
150	$150^{367} \bmod 253 =$ **105**
184	$184^{367} \bmod 253 =$ **46**
38	$38^{367} \bmod 253 =$ **112**
144	$144^{367} \bmod 253 =$ **100**
126	$126^{367} \bmod 253 =$ **102**

Table 7. Manual Decryption Calculation for XLSX Document

Ciphertext	Decryption Process *(M = C^{367} mod 253)*
206	206^{367} mod 253 = **68**
102	102^{367} mod 253 = **97**
126	126^{367} mod 253 = **102**
139	139^{367} mod 253 = **116**
102	102^{367} mod 253 = **97**
131	131^{367} mod 253 = **32**
206	206^{367} mod 253 = **68**
102	102^{367} mod 253 = **97**
126	126^{367} mod 253 = **102**
139	139^{367} mod 253 = **116**
102	102^{367} mod 253 = **97**
131	131^{367} mod 253 = **32**
206	206^{367} mod 253 = **68**
102	102^{367} mod 253 = **97**
126	126^{367} mod 253 = **102**
139	139^{367} mod 253 = **116**
102	102^{367} mod 253 = **97**
131	131^{367} mod 253 = **32**
206	206^{367} mod 253 = **68**
102	102^{367} mod 253 = **97**
126	126^{367} mod 253 = **102**

The manual calculation confirms that the RSA algorithm successfully encrypts plaintext into ciphertext and accurately restores the original plaintext during decryption. The identical outputs before encryption and after decryption demonstrate that the implemented RSA algorithm can securely protect document contents while preserving data integrity across DOCX, PDF, and XLSX file formats.

System Design

Before implementation, the document security system using the RSA algorithm was designed to ensure that the application could properly secure document contents. The system design was created based on the required processes, including authentication, document encryption, decryption, RSA key management, and Avalanche Effect testing.

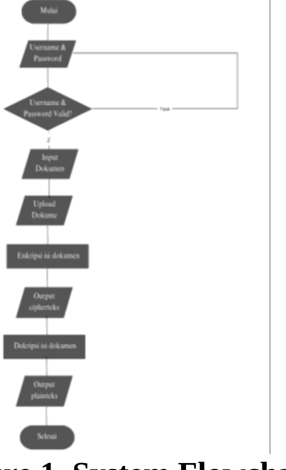

Figure 1. System Flowchart

The system flowchart describes the application process. Users must enter a valid username and password before accessing the system. After successful authentication, users can proceed to the available application features. The system design consists of interface design that describes the layout and functions of each page.

Interface Design

The interface design was developed as a reference for building the application display. The designed interfaces include the login page, dashboard, RSA keys page, document management page, encryption page, decryption page, and Avalanche Effect page.

The login page requires administrators to enter the correct username and password before accessing the system.

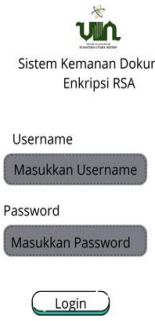

Figure 2. Login Feature Design

The dashboard page displays a summary of recent activities and security status, including recently performed encryption and decryption processes.

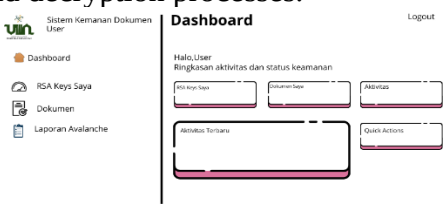

Figure 3. Dashboard Page Design

The RSA Keys page displays active and inactive RSA keys used in the encryption and decryption process.

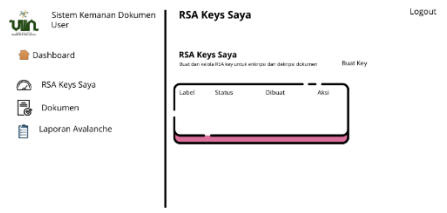

Figure 4. RSA Keys Page Design

The All Documents page displays all documents that have been processed by the system, including encrypted and decrypted documents.

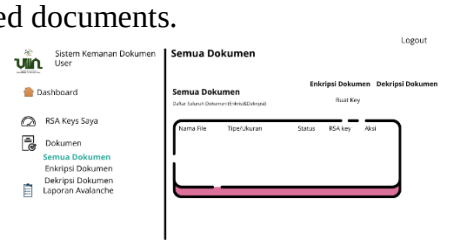

Figure 5. All Documents Page Design

The Encryption page is designed to perform document encryption by selecting and uploading files that need protection.

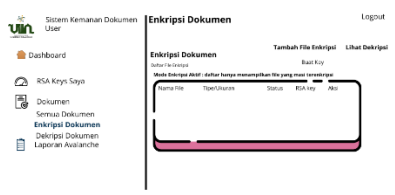

Figure 6. Document Encryption Page Design

The Decryption page is designed to restore encrypted documents by selecting the document and entering the required private key.

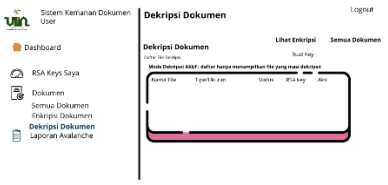

Figure 7. Decryption Page Design

The Avalanche Effect page provides information regarding the percentage results of Avalanche Effect testing performed on encrypted documents.

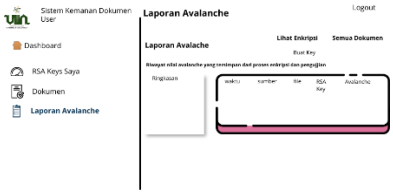

Figure 8. Avalanche Effect Page Design

Implementation Results

The implementation stage explains the results of the developed RSA-based document security application interface. The system was developed using PHP and aims to assist users in securing important documents that should not be distributed publicly. Based on the manual calculation process and system design, application testing was conducted to evaluate the performance of the developed system.

To access the application, users open a browser and access http://127.0.0.1:8000/, which displays the login page. The following section describes the implemented application interfaces.

Login Page Display

The login page is the initial interface displayed when the application is accessed. Users must enter a valid username and password to access the system.

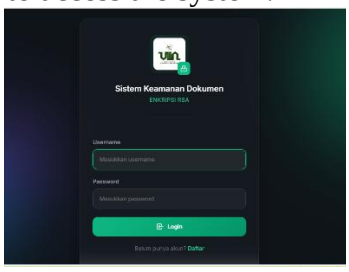

Figure 9. Login Page Display

Dashboard Page Display

The dashboard page provides information regarding recent activities performed in the system, including encryption and decryption processes, as well as the total number of documents stored in the application.

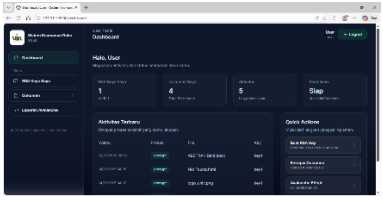

Figure 10. Dashboard Page Display

RSA Keys Page Display

The RSA Keys page displays the active keys currently used by the system and inactive keys that are no longer in use.

Figure 11. RSA Keys Page Display

All Documents Page Display

The All Documents page displays all documents processed by the application, including both encrypted and decrypted documents.

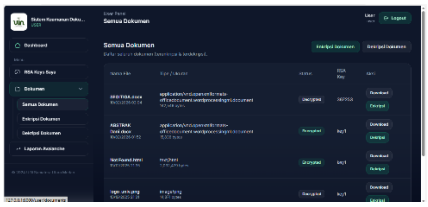

Figure 12. All Documents Page Display

Encryption Page Display

The Encryption page displays documents that have been uploaded and successfully encrypted. Users can view the encryption results during the process or download the encrypted document.

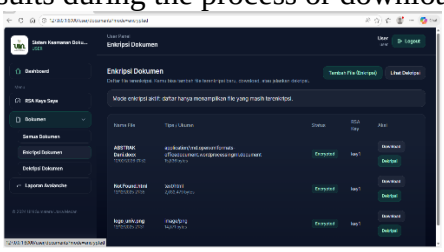

Figure 13. Encryption Page Display

Decryption Page Display

The Decryption page displays documents that have completed the decryption process using the RSA private key. The restored documents can be accessed through this page.

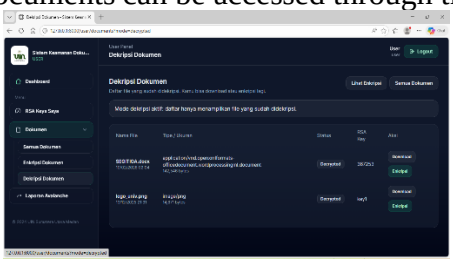

Figure 14. Decryption Page Display

Avalanche Report Page Display

The Avalanche Report page presents the results of the Avalanche Effect evaluation based on the encrypted documents tested in the system.

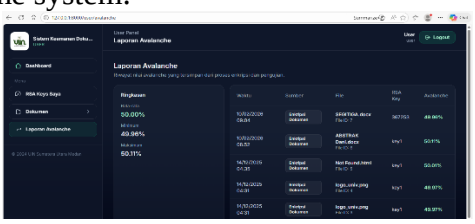

Figure 15. Avalanche Report Page Display

Program Testing

After the implementation stage, testing was conducted to evaluate the performance of the RSA algorithm and the developed application system. The testing process was performed on DOCX, PDF, and XLSX documents by applying encryption and decryption procedures. During the encryption

process, PDF documents required slightly longer processing time compared to DOCX and XLSX documents due to their larger data structure.

Encryption Process

Encryption is a process that converts the original document contents (plaintext) into ciphertext so that the information cannot be accessed without the appropriate key. The process utilizes a password and the RSA cryptographic algorithm to secure document contents.

The encryption process begins with the initial page before uploading the document to be protected.

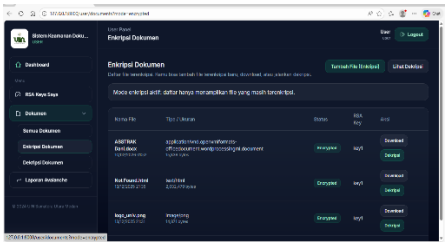

Figure 16. Initial Page Before File Upload

The next stage is selecting the document that will be encrypted through the application.

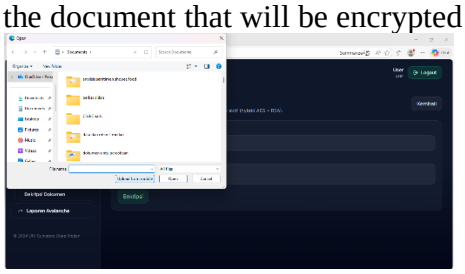

Figure 17. File Selection for Encryption

After selecting the document, the system performs the encryption process to transform the original document into an encrypted format.

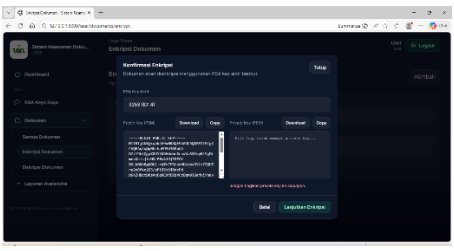

Figure 18. Encryption Process Display

The final encryption result displays the document in an unreadable encrypted form along with the Avalanche Effect testing results.

Figure 19. Encryption Result Display

Decryption Process

Decryption is the reverse process of encryption, where encrypted documents are restored to their original condition. This process converts ciphertext back into the original document using the RSA private key.

After a document has been encrypted, users can restore the document by selecting the encrypted file on the encryption page and choosing the decryption option.

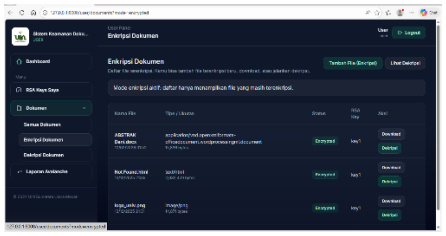

Figure 20. Document Selection for Decryption

After selecting the document, the system provides an option to use an automatic or manual private key. Users select the desired key method and continue the decryption process.

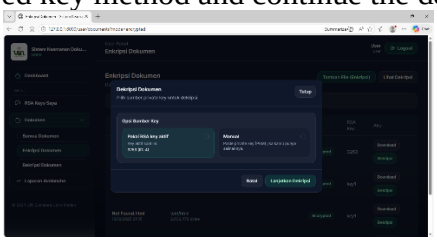

Figure 21. Decryption Key Display

After the decryption process is completed, successfully restored documents are displayed on the decryption document page.

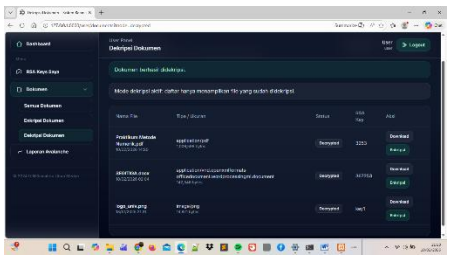

Figure 22. Successfully Decrypted Document Display

The decrypted document can then be downloaded and opened to verify that the document contents have returned to their original state.

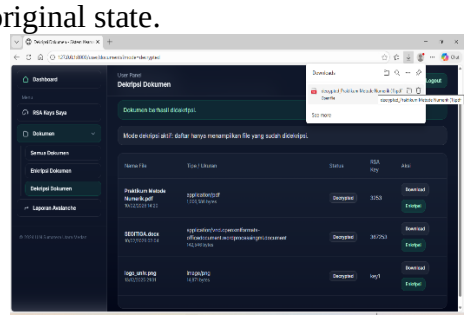

Figure 23. Downloaded Decrypted Document Result

Avalanche Effect Testing

Avalanche Effect is an important property in cryptography that describes how a small change in input can produce a significant change in output. This effect improves encryption security because it makes ciphertext appear random and difficult to predict.

A minor change in plaintext, such as changing one character or bit, should produce a significant difference in the resulting ciphertext. The Avalanche Effect is closely related to confusion and diffusion concepts, where the relationship between keys and ciphertext becomes complex and small plaintext changes produce major output changes.

The percentage calculation of the Avalanche Effect is represented by Equation (4.1):

$P\ \frac{D}{T}$ .....(4.1)

Where:

D = Different Character

T = Total Character

Table 8. Avalanche Effect Testing Results

No	Document Name	Document Type	Avalanche Effect
1	DAFTAR PUSTAKA.docx	DOCX	49.94%
2	DISKUSI 7 AKP.docx	DOCX	50.18%
3	Surat Lamaran Kerja.pdf	PDF	50.13%
4	Praktikum Metode Numerik.pdf	PDF	49.99%
5	Data VII 6.xlsx	XLSX	49.99%
6	Tabel referensi.xlsx	XLSX	49.99%

Based on the testing results in Table 8, the document security application achieved an average Avalanche Effect value of 49.99%. This result indicates that the RSA algorithm implemented in the application is capable of providing adequate document protection. A higher Avalanche Effect value indicates better cryptographic security because changes in plaintext produce more significant changes in ciphertext, making the encrypted data more difficult to analyze.

CONCLUSIONS

Based on the research conducted on document content security using the RSA algorithm, it can be concluded that the developed computer security application is capable of securing document contents effectively. The RSA algorithm works by uploading the document to be protected, selecting the public key to perform encryption, and storing the encrypted document in the application. To restore the original document, users can perform the decryption process using the private key, after which the document can be downloaded and accessed again. The implementation results of the web-based RSA algorithm are consistent with the previous manual calculations, indicating that the application functions properly. Furthermore, the Avalanche Effect testing achieved an average value of 49.99% or higher, demonstrating that the application provides adequate security for protecting document contents.

For future research, document security can be further developed by implementing other cryptographic algorithms or combining multiple algorithms to improve file protection and achieve stronger security performance.

REFERENCES

Ananda, R. I., Fauziah, & Hayati, N. (2020). Keamanan Email Menggunakan Metode Pretty Good Privacy Dengan Algoritma RSA. *Jurnal Ilmiah Informatika Komputer*. 25(3): 213–224. https://doi.org/10.35760/ik.2020.v25i3.3118

Anwar Badrul, N. B., Nugroho, J., & Azanuddin. (2019). Implementasi Algoritma RSA Terhadap Keamanan Data Simpan Pinjam. *Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika Dan Komputer)*. 18(1): 30–34.

Apdilah, D., & Swanda, H. (2018). Penerapan Kriptografi RSA Dalam Mengamankan File Teks Berbasis PHP. *Jurnal Teknologi Informasi*. 2(1): 45. https://doi.org/10.36294/jurti.v2i1.407

Gunawan, I. (2018). Kombinasi Algoritma Caesar Cipher Dan Algoritma RSA Untuk Pengamanan File Dokumen Dan Pesan Teks. *InfoTekJar (Jurnal Nasional Informatika Dan Teknologi Jaringan)*. 2(2): 124–129. https://doi.org/10.30743/infotekjar.v2i2.266

Harbani, A., & Fahreza, M. A. (2019). Aplikasi Keamanan Data Gambar Menggunakan Algoritma RSA (Rivest Shamir Adleman) Berbasis Desktop. *Teknois: Jurnal Ilmiah Teknologi Informasi Dan Sains*. 9(1): 1–9. https://doi.org/10.36350/jbs.v9i1.1

Noviyantono, E., & Fadlan, M. (2024). Studi Perbandingan Avalanche Effect Pada Algoritma Kriptografi Transposisi Untuk Meningkatkan Keamanan Data. 9: 1–5.

Prasetyo, Y., Triandi, B., & Hardianto, H. (2018). Perancangan Aplikasi Pengamanan File Teks Dengan Skema Hybrid Menggunakan Algoritma Enigma Dan Algoritma RSA. *IT (Informatic Technique) Journal*. 6(1): 46–55. https://doi.org/10.22303/it.6.1.2018.46-55

Purwanto, E. W., & T, S. S. S. (2018). Algoritma Kriptografi El-Gamal Untuk Pengamanan Pesan Pada Steganografi Citra Domain Discrete Cosine Transform Dengan Teknik Penyisipan Least Significant Bit. *E-Proceeding of Engineering*. 5(1): 116–123.

Putri, G. G., Setyorini, W., & Rahayani, R. D. (2018). Analisis Kriptografi Simetris AES Dan Kriptografi Asimetris RSA Pada Enkripsi Citra Digital. *ETHOS (Jurnal Penelitian Dan Pengabdian)*. 6(2): 197–207. https://doi.org/10.29313/ethos.v6i2.2909

Rambe, M. R., Haryanto, E. V., & Setiawan, A. (2019). Aplikasi Pengamanan Data Dan Disisipkan Pada Gambar Dengan Algoritma RSA Dan Modified LSB Berbasis Android. *IT (Informatic Technique) Journal*. 7(1): 51–62. https://doi.org/10.22303/it.7.1.2019.51-62

Suhandinata, S., Rizal, R. A., Wijaya, D. O., Warren, P., & Srinjiwi. (2019). Analisis Performa Kriptografi Hybrid Algoritma RSA. *Jurteksi*. VI(1): 1–10.

Sutejo, S. (2021). Implementasi Algoritma Kriptografi RSA (Rivest Shamir Adleman) Untuk Keamanan Data Rekam Medis Pasien. *INTECOMS: Journal of Information Technology and Computer Science*. 4(1): 104–114. https://doi.org/10.31539/intecoms.v4i1.2437