Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

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

The Use of Miller-Rabin in Testing Prime Numbers in the Rsa Algorithm to Secure Files

Wanda Yohana[1*], Achmad Fauzi[2], I Gusti Prahmana[3]

[1,2,3]*Informatics Engineering, STMIK KAPUTAMA*
Wandayohana83@gmail.com [1*], fauzyrivai88@gmail.com [2], igustiprahmana4@gmail.com [3]

Abstract

The development of digital technology has facilitated data exchange, but it has also increased security risks such as wiretapping and file manipulation. One of the most widely used cryptographic methods for maintaining data confidentiality is the RSA algorithm, whose security depends on large prime numbers as keys. This study utilizes the Miller-Rabin algorithm as a probabilistic primality testing method to ensure the prime numbers used in RSA key generation. The research was conducted by building a system using the Python programming language, which included the process of testing prime numbers with Miller-Rabin, RSA key generation, and document file encryption and decryption. The test results showed that Miller-Rabin was effective in validating large prime numbers and helped speed up the RSA key generation process, so that files could be secured with ciphertext that could be returned to plaintext without data loss. Thus, the implementation of Miller-Rabin in the RSA algorithm has been proven to improve the reliability of digital file security systems.

Keywords: *Cryptography, Miller-Rabin, RSA, File Security,*

1. Introduction

The development of digital technology has facilitated the transfer of information, but it has also given rise to security threats in the form of data interception and unauthorised modification. The RSA algorithm is the primary cryptographic solution that relies on the complexity of large prime numbers for encryption strength. The use of the RSA (Rivest-Shamir-Adleman) algorithm to secure data depends on the use of prime numbers for the initial values p and q. The larger the prime numbers used, the more difficult it is to crack or intercept, but this also poses a major challenge in the process of securing data using the RSA algorithm [1]. To overcome this problem, prime number testing is required. There are several ways to test whether a number is prime or not, one of which is the Miller-Rabin prime test algorithm. Miller-Rabin itself has the advantages of being computationally light and producing high probability values [2]

2. Literatur Review

2.1. Miller-Rabin

Miller–Rabin is one of the methods or algorithms for primality testing used to check whether a number is prime or composite. The testing process involves selecting one or more random numbers called witnesses as the basis for testing. Each witness is used to test the consistency of the properties of the number being tested. If the number fails to meet certain conditions for one of the witnesses, then it can be confirmed that the number is composite. However, if it passes all tests with several witnesses, the number is very likely to be prime, although not 100% certain (because it is probabilistic). The advantage of Miller–Rabin is its ability to test very large numbers quickly, so it is often used in cryptographic systems such as RSA.

The Miller-Rabin algorithm can be calculated with 4−s, where s is the number of Miller-Rabin primality test iterations performed. For sufficiently large values of s, the probability that a number passing the primality test is a pseudoprime can be reduced[2]. The components of the Miller-Rabin algorithm are as follows:

1. Select the value y as the number to be tested, provided that the number is a random odd number.
2. Calculate $y - 1 = 2s \times d$.

3. Randomly select a number a or base, where a ∈ (2 to y−2) or (1 < a < y – 1), which will be used to find x = 1 or x = y-1. Using the formula x = ad mod y.
4. If the result x is not x = 1 or x = y-1, then the next step is to perform s-1 iterations using the formula x2 mod y. If the result of the iteration does not show x = 1 or x = y-1, then it can be confirmed that the number is not prime.

2.2. Cryptography

Cryptography comes from Greek, consisting of two words: crypto and graphia. Crypto means to hide, while graphia means writing. Cryptography is the study of mathematical techniques related to information security, such as data confidentiality, data validity, data integrity, and data authentication. However, not all aspects of information security can be addressed by cryptography [3]

2.2.1. Purpose of Cryptography

The main purpose of cryptography is to maintain information security by ensuring confidentiality, integrity, authentication, and non-repudiation so that data can only be accessed by authorized parties and is protected from alteration or misuse[4].

2.2.2. Components Cryptography

Cryptography has components that work together to create a perfect security system. The following are the main components in cryptography[4]:

a. Plaintext, often referred to as the original text or initial text, is a typed message that has meaning. This is the initial input that will be processed into the cryptographic system (Sir.
b. Ciphertext is a message generated from the encryption process in a cryptographic algorithm. The message in this coded text cannot be read because it consists of characters that have no meaning without the decryption process.
c. Encryption is a method of protecting transmitted data so that it remains confidential. The original message, called plain text, is converted into codes that are difficult to understand.
d. Decryption is the opposite of encryption. Encrypted messages are restored to their original form. The algorithm used for decryption is definitely different from the algorithm used for encryption.
e. A key is a value used in the encryption and decryption process. The security of modern cryptographic systems relies heavily on the secrecy of the key, not on the secrecy of the algorithm.
f. Cryptographic algorithms are methods used to perform encryption and decryption processes. In general, these algorithms are categorized into three main types. Symmetric algorithms use the same key for both encryption and decryption processes. Examples of symmetric algorithms are DES, 3DES, AES, Blowfish, etc. Meanwhile, asymmetric algorithms utilize a pair of public and private keys. Examples of asymmetric algorithms are RSA, ECC, DSA, Diffie-Hellman, etc. Another cryptographic algorithm is a hash function that works unidirectionally to ensure that data does not change. Examples of hash function algorithms are SHA-256, SHA-3, MD5, Bcrypt, etc.

2.3. RSA (Rivest-Shamir-Adleman)

RSA or Rivest Shamir Adleman is a cryptographic algorithm for data security or asymmetric encryption. The RSA algorithm itself is often used in the world of cryptography for data security and is one of the most advanced algorithms in the world of cryptography [4]. There are three main processes in using RSA for data security (encryption), namely key generation, encryption, and decryption. The prime number generation method is at the core of the key generation process, which produces a pair of keys, namely a public key and a private key [1]
Encryption is a coding process that changes a code (message) from understandable (plaintext) to incomprehensible (ciphertext). The reverse process of converting ciphertext back to plaintext is called decryption. The encryption and decryption processes require a specific mechanism and key [5]. In the data encryption process, encoding is performed using a public key (n, e). Initially, the data is in plaintext form, which is then converted into ASCII for each character. Next, the plaintext (M) is encrypted with a public key using the following calculation formula: C = Me mod n [6].
Next, in the process of decrypting previously encrypted data (ciphertext) using a public key (e, n), it will be returned to its original message (plaintext) using a private key (d, n). The first process is to enter the ciphertext data (C), then the data is decrypted into ASCII numbers for each character using the following calculation formula: M = $Cd\ (mod\ n)$. After the encryption process produces new ciphertext (C), the new ciphertext (C) is converted into ASCII form during the decryption process. Next, the data in ASCII form will be converted back into plaintext (M) as before [6].

The following are the basic components of the RSA algorithm:
1. p and q prime numbers (secret)
2. n = p × q (not secret)
3. Φ (n) = (p – 1) (q – 1) (secret)
4. e (encryption key) (public)
5. d (decryption key) (private)
6. m (plaintext) (private)
7. c (ciphertext) (public)

2.4. File Security

File security is an effort to protect data in a file, such as text files, office documents (e.g., .doc, .pdf), or digital archives, so that it cannot be accessed or modified by unauthorized parties. This protection is usually done using cryptography, where the file contents are converted into an encrypted form (ciphertext) that cannot be understood without a valid decryption key. In the context of public key cryptography such as RSA, document files are first converted into a numerical representation (encoded), then processed using a public key to generate ciphertext. The private key, which is only possessed by the recipient or owner of the file, is used to restore the ciphertext to its original form. With this mechanism, even if the encrypted file is stolen or intercepted during the transmission process, the file contents remain secure because they cannot be opened without the private key[7]. Data corruption in Word files (file contents) can be overcome by utilizing the RSA method for encryption and decryption. Therefore, a security system is needed that is capable of maintaining data confidentiality from other threats posed by irresponsible parties. By utilizing the RSA algorithm, the system will encrypt the original data inputted by the user into ciphertext using a key, then send it to other people or colleagues. To receive the original data, it is decrypted into plaintext using a key by the recipient so that the transmission of information or the use of information through the RSA algorithm security becomes easier to understand by the recipient or user of the Word file [7].

3. Research Methodology

Based on the methods used, this research has structured and systematic stages, in accordance with the requirements that were clearly defined at the outset. The following are the stages or workflow of this research. Based on the image above, the following is an explanation of the workflow in this study:

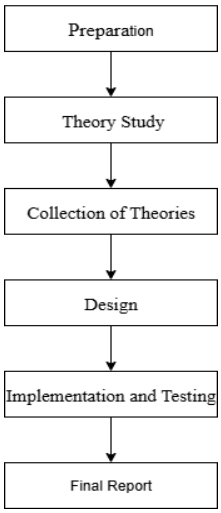


Fig.1：Research workflow

1. Preparation is the initial stage or foundation of the study. At this stage, all research requirements are usually prepared, including problem identification, problem formulation, research objectives, and preliminary studies.
2. Theoretical review is the process of gathering all the theoretical foundations needed to support the study. At this stage, the author conducts an in-depth review and analysis of concepts, theories, principles, and previous research relevant to the research topic.
3. Theory compilation is the process of compiling and synthesizing various theories that have been reviewed.
4. Design is the stage of designing the system architecture based on a theoretical review that includes the selection of research methods, research instrument design, and data analysis plan preparation.
5. Implementation and testing involve building a system using data found in theoretical studies and data collection by applying the relevant methods. In this study, a system utilizing the Miller-Rabin algorithm was created as a prime number tester to be used in the RSA algorithm to secure files so that the system cannot be easily guessed or hacked. Testing is carried out to identify problems in the system.
6. Final report: at this stage, evaluation is usually carried out, conclusions are drawn, and the results are analyzed.

3.1. Test Analysis Using Miller-Rabin

Testing numbers using Miller-Rabin as a key generation process in the RSA algorithm, because the security of RSA depends entirely on the confidentiality of two large prime numbers used to generate the modulus n = p*q. Therefore, the process of verifying that the random numbers generated are truly prime becomes crucial in this cryptographic system.
In this study, the author uses 5 random odd numbers as samples to be tested using Miller-Rabin. The first thing to do in this test is to determine the value of y (random odd number) to be tested whether the number is prime or not. This test uses bases or a 2, 3, and 5.

1. First sample Determine the value of y, y = 221
   Calculate y-1 and factor it into 2S * d
   Y – 1 = 2S * d
   221 – 1= 2S * d

220 = 22 * 55
Randomly select a number a or a base, where a ∈ [2, y−2], to be used to find x = 1 or x = y-1. Using the formula x = ad mod y
a. Test using base 2/ a = 2
x = ad mod y
= 255 mod 221
= 128 (failed, because the result is not x = 1 or x = y-1)
Iterate s-1 times using the formula x2 mod y
x2 mod y = 1282 mod 221
= 1282 mod 221
= 16384 mod 221
= 30 (failed)
x=1 or x=y-1 not found, so this is definitely not a prime number
b. Testing using base 3/ a = 3
x = ad mod y
= 355 mod 221
= 198 (failed, because the result is not x = 1 or x = y-1)
iterate s-1 times with the formula x2 mod y
x2 mod y = 1982 mod 221
= 39204 mod 221
= 87 (failed)
c. Testing using base 5/ a = 5
x = ad mod y
= 555 mod 221
= 112 (failed, because the result is not x = 1 or x = y-1)
iterate s-1 times with the formula x2 mod y
x2 mod n = 1122 mod 221
= 12544 mod 221
= 168 (failed)
Therefore, this test confirms that 221 is not a prime number

2. Second sample, y = 163
Calculate y-1 and factor it into 2S * d
Y – 1 = 2S * d
163 – 1 = 2S * d
162 = 21 * 81
Randomly select a number a or base, where a ∈ [2, y−2], to be used to find x = 1 or x = y-1. Using the formula x = ad mod y
a. Test using base 2/ a = 2
x = ad mod y
= 281 mod 163
= 162 (Pass, because the result is x = y-1)
b. Test using base 3/ a = 3
x = ad mod y
= 381 mod 163
= 162 (Pass, because the result is x = y-1)
c. Test using base 5/ a = 5
x = ad mod y
= 581 mod 163
= 162 (Pass, because the result is x = y-1)

3. Third sample, y = 127
Calculate y-1 and factor it into 2S * d
Y – 1 = 2S * d
127 – 1 = 2S * d
126 = 21 * 63
Randomly select a number a or base, where a ∈ [2, y−2] to be used to find x = 1 or x = y-1. Using the formula x = ad mod y.
a. Testing using base 2/ a = 2
x = ad mod y
= 263 mod 127
= 1 (Pass, because the result is x = 1)
b. Test using base 3/ a = 3
x = ad mod y
= 363 mod 127
= 126 (Pass, because the result is x = y-1)
c. Test using base 5/ a = 5
x = ad mod y
= 563 mod 127
= 126 (Pass, because the result is x = y-1)

From the above tests, it can be concluded that there are some numbers that are prime and some that are not. In this test, the numbers that pass the test must have a value of x = 1 or x = y-1. If not, it can be confirmed that the number is not prime. The following is a table of test results for 3 random odd numbers:

Table 1 Test Results Using Miller-Rabin

No	Y Value (Test Number)	Testing Using Base (a)		
		a=2	a=3	a=5
1	221	Failed	Failed	Failed
2	163	Passed	Passed	Passed
3	127	Passed	Passed	Passed

3.2. Analysis of the RSA Algorithm Encryption Process

Before encrypting a file, you must first select the file type. In this study, the files to be secured are Word files (.doc, .docx).

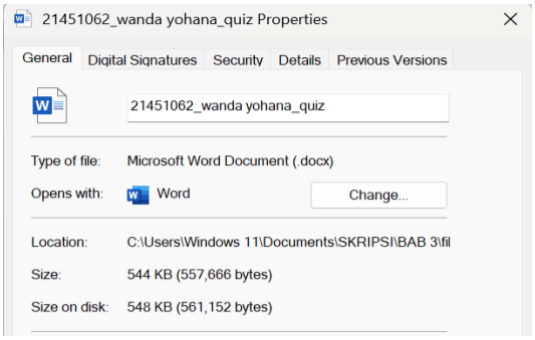


Fig.2: Files that will be secured

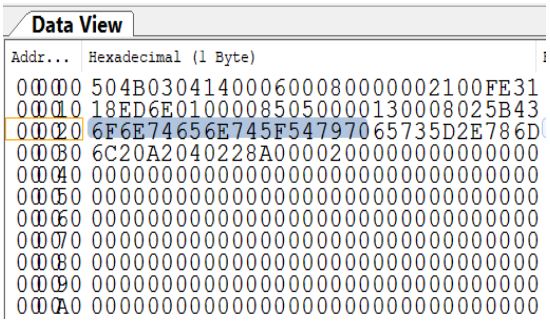


Fig.3: Converting files to hexadecimal using a binary viewer

In this encryption process, the file to be used is a docx file with a size of 544 KB, as shown in the image above. The file must be converted into hexadecimal numbers using Binary Viewer software. The purpose of this is so that the file contents can be processed mathematically and facilitate manual calculations. On the left is the file to be secured, and on the right is an image of the file that has been converted to hexadecimal, which will later be converted to decimal. After converting hexadecimal to decimal, the result of [6F, 6E, 74, 65, 6E, 74, 5F, 54, 79, 70] becomes [111, 110, 116, 101, 110, 116, 95, 84, 121, 112]. Then you can manually encrypt the file.

$p = 127 \qquad q = 43 \qquad n = p \times q$

$= 127 \times 43 = 5461$

$\varphi(n) = (p-1) \times (q-1)$

$= (127-1) \times (43-1) = 126 \times 42 = 5292$

$e = 163$

Prime numbers at values p and q are numbers that have passed the prime number test using the Miller-Rabin algorithm. Next is the RSA encryption process with the formula **$C = m^e \pmod n$.**

1. $C1 = m^e \pmod n$

   $= 111^{163} \bmod 5461 = 2030$

2. $C2 = m^e \pmod n$

   $= 110^{163} \bmod 5461 = 2504$

3. $C3 = m^e \pmod n$

   $= 116^{163} \bmod 5461 = 2598$

4. $C4 = m^e \pmod n$

   $= 101^{163} \bmod 5461 = 3543$

5. $C5 = m^e \pmod n$

   $= 110^{163} \bmod 5461 = 2504$

6. $C6 = m^e \pmod n$

   $= 116^{163} \bmod 5461 = 2598$

7. C7 = m^e (mod n)
   = 95^{163} mod 5461 = 1389
8. C8= m^e (mod n)
   = 84^{163} mod 5461 = 4519
9. C9 = m^e (mod n)
   = 121^{163} mod 5461 = 5396
10. C10 = m^e (mod n)
    = 112^{163} mod 5461 = 2480

After encrypting the file using 10 bytes, the resulting cipherfile is 111, 110, 116, 101, 110, 116, 95, 84, 121, 112].

3.3. Process Analysis Description of the RSA Algorithm

In the process of decrypting previously encrypted data (cipherfile) using the system key (e, n), it will be returned to the original message (plainfile) using the private key (d, n). The first process performed is to enter the cipherfile data. In this study, the private key d is 487, where d*e mod φ(n) = 1, so 487*163 mod 5292 = 1. The decryption process aims to restore encrypted data to its original form using a private key so that the information can be read and understood by the relevant system. In other words, decryption serves to unlock data that has been scrambled through the encryption process, so that messages or system files can be restored to their original format without losing their meaning or original content. The following is the RSA decryption process using the formula M = C^d ($mod\ n$):

1. M1= $C^d\ mod\ n$
   = $2030^{487}\ mod$ 5461 = 111
2. M2= $C^d\ mod\ n$
   = $2504^{487}\ mod$ 5461 = 110
3. M3= $C^d\ mod\ n$
   = $2598^{487}\ mod$ 5461 = 116
4. M4= $C^d\ mod\ n$
   = $3543^{487}\ mod$ 5461 = 101
5. M5= $C^d\ mod\ n$
   = $2504^{487}\ mod$ 5461 = 110
6. M6= $C^d\ mod\ n$
   = $2598^{487}\ mod$ 5461 = 116
7. M7= $C^d\ mod\ n$
   = $1389^{487}\ mod$ 5461 = 95
8. M8 = $C^d\ mod\ n$
   = $4519^{487}\ mod$ 5461 = 84
9. M9= $C^d\ mod\ n$
   = $5396^{487}\ mod$ 5461 = 121
10. M10= $C^d\ mod\ n$
    = $2480^{487}\ mod$ 5461 = 112After decryption, it can be confirmed that the plainfile has returned to its original state, meaning that the decryption process was successful. If the decryption results are incorrect, there was an error during the calculation, so repeat the process to ensure that everything is correct. The next step is to return the decrypted value to its hexadecimal value or original value, commonly referred to as conversion, in the following manner.

    111 = 111: 16 = 6 sisa 15 (F)

    = 6 : 16 = 0 sisa 6

= 6F

1. 110 = 110 : 16 = 6 sisa 14 (E)
   = 6 : 16 = 0 sisa 6
   = 6E
2. 116 = 116 : 16 = 7 sisa 4
   = 7 : 16 = 0 sisa 7
   = 74
3. 101 = 101 : 16 = 6 sisa 5
   = 6 : 16 = 0 sisa 6
   = 65
4. 110 = 110 : 16 = 6 sisa 14 (E)
   = 6 : 16 = 0 sisa 6 = 6E
5. 116 = 116 : 16 = 7 sisa 4
   = 7 : 16 = 0 sisa 7
   = 74
6. 95 = 95 : 16 = 5 sisa 15 (F)
   = 5 : 16 = 0 sisa 5
   = 5F
7. 84 = 84 : 16 = 5 sisa 4
   = 5 : 16 = 0 sisa 5
   = 54
8. 121 = 121 : 16 = 7 sisa 9
   = 7 : 16 = 0 sisa 7
   = 79
9. 112 = 112 : 16 = 7 sisa 0
   = 7 : 16 = 0 sisa 7 = 70Detailed submission guidelines can be found on the journal web pages. All authors are responsible for understanding these guidelines before submitting their manuscript.

3.4. Implementation

Built using the Python programming language, utilizing standard libraries for cryptographic calculations and a simple user interface. The testing process was carried out directly by running the application and trying out each available feature, starting from prime number testing with Miller-Rabin, the RSA encryption process, and the RSA decryption process. The files used in this test were document files in .docx format, but they can also be used with documents in .pdf and .xlsx formats.

4.1 Program Testing

The test was conducted by directly operating the system using prepared test data. The objectives were to assess system performance, verify process accuracy, and ensure that every feature was working properly. The following are the steps that must be taken to secure files using the RSA and Miller-Rabin algorithms as prime number testing algorithms.

1. Main Form

Fig.4: Main form display

The initial testing step begins with the Main Form, which is the first screen that appears when the application is launched. Users are greeted with three main options: Miller-Rabin Test, Encryption Process, Decryption Process, and Close.

2. Miller-Rabin Test Form

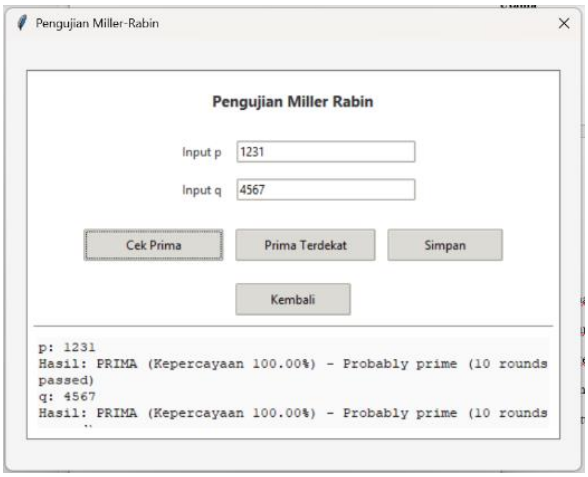


Fig. 5: Miller-Rabin Test Form Display

The next step is to test prime numbers using the Miller-Rabin algorithm, as shown in the image below. On this form, users can enter the number to be tested. If the “Check Prime” function finds a composite number or a non-prime number, users can immediately click the “Nearest Prime” button, then save the prime number by clicking the “Save” button, as shown in the image below.

3. RSA Algorithm Encryption Process Form

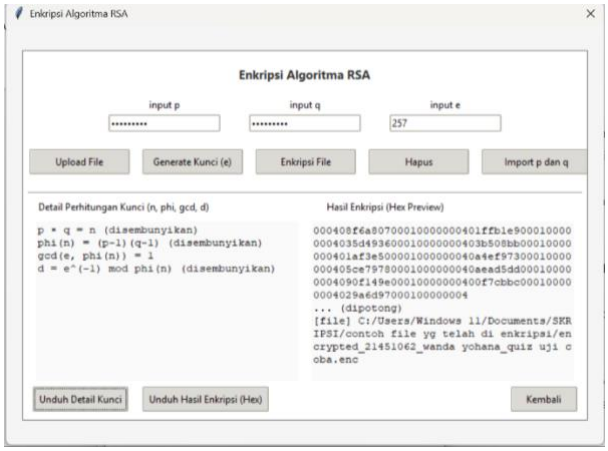


Fig.6: RSA Algorithm Encryption Process Form Display

The next step is to test the Encryption Form, which is used to secure digital files. At this stage, the previously tested and stored p and q input values are automatically imported by clicking the "Import p and q" button. To obtain the input value e, click the "Generate e key" button to generate a random number as the encryption key. Once the file has been uploaded, you can immediately click the "Encrypt File" button, which will then automatically save the file. Don't forget to save or "download the key" to be able to decrypt the file or return it to its original form, as shown in the image below.

4. RSA Algorithm Decryption Process Form

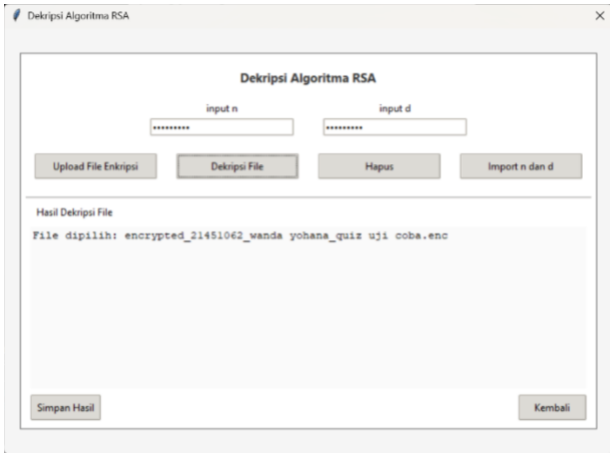


Fig.7: RSA Algorithm Decryption Process Form Display

This is the final stage, which is the testing stage on the Decrypt Form. On this form, the user selects the previously encrypted file via the "Upload Encrypted File" button, then imports the private values n and q using the "Import n and d" button, which will automatically fill in the input values n and d. Next, the user can click the "Decrypt File" button, then download the results and check the encrypted file, as shown in the image below.

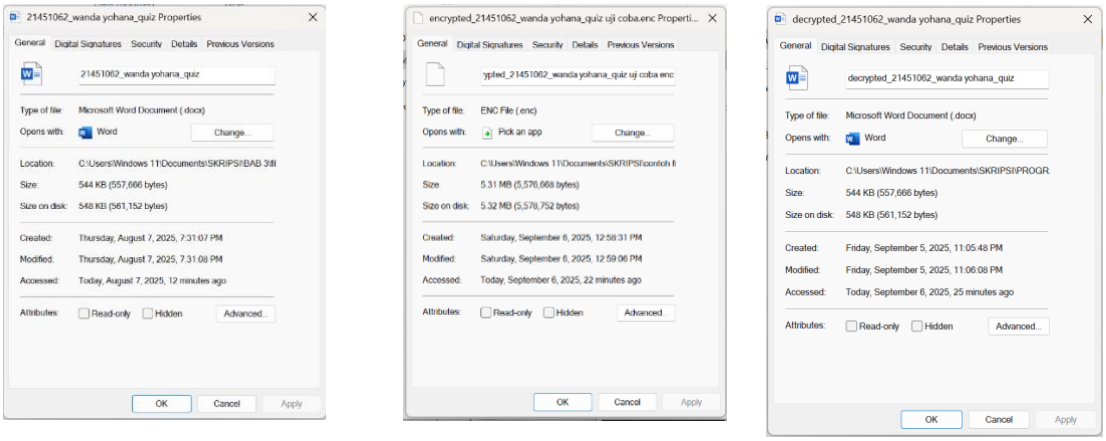
4.2. Test Results

The following is a table of file encryption test results using the RSA algorithm with Miller-Rabin as the prime number test.

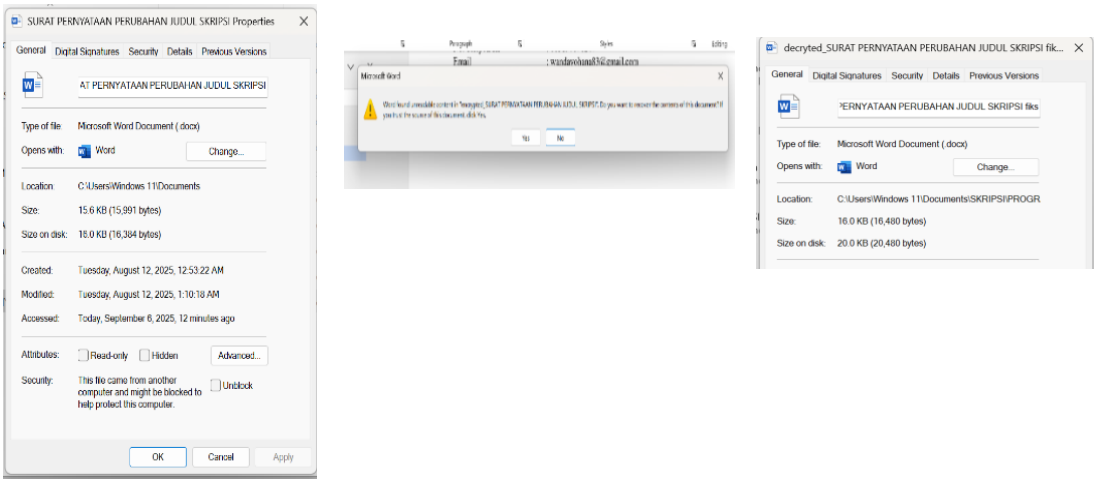
Table.2File Security Test Results

No	Original File	RSA Algorithm Encryption	RSA Algorithm Decryption

1



2



4. Conclusion

Based on the discussion, design, and testing in this study on the Use of Miller-Rabin in Prime Number Testing in the RSA Algorithm for File Security, the following conclusions can be drawn from this study:

1. The Miller-Rabin algorithm has been proven effective in verifying a number to be used as the p, q value in RSA or prime number testing, especially large prime numbers. When tested using a standard basis, this method was able to identify prime and composite numbers, thereby ensuring that the numbers to be used in RSA generation were indeed prime.
2. The implementation of Miller-Rabin as a validation stage prior to RSA key generation is a systematic and integrated workflow for building a file security system. This research developed a file security system by building a desktop application using Python-based Visual Studio Code (VS Code) software equipped with a form interface, making it easier for users to test prime numbers and perform file encryption and decryption. The designed system is capable of generating secure ciphertext and returning it to its original plaintext without any data loss.

References

[1] M. Mulya, “Perancangan Perangkat Lunak dengan Metode RSA dengan Pembangkit Bilangan Prima Secara Opsional,” JUPITER (Jurnal Penelitian Ilmu dan Teknologi …, vol. 7, no. 1, pp. 1–7, 2015, [Online]. Available: https://www.jurnal.polsri.ac.id/index.php/jupiter/article/view/705

[2] N. Liem, “Penerapan Teori Bilangan pada Algoritma Pembangkit Bilangan Prima untuk Kriptosistem RSA,” Penerapan Teori Bilangan pada Algoritma Pembangkit Bilangan Prima untuk Kriptosistem RSA, pp. 1–7, 2023.

[3] M. M. Amin, “IMPLEMENTASI KRIPTOGRAFI KLASIK PADA KOMUNIKASI BERBASIS TEKS,” vol. III, no. September, pp. 129–136, 2016.

[4] S. J. Siregar, N. B. Nugroho, and H. Sigalingging, “Implementasi Algoritma Kriptografi RSA (Rivest Shamir Adleman) Dalam Pengamanan Data Gaji Karyawan Di Kantor BSPJI,” Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika dan Komputer), vol. 22, no. 2, p. 528, 2023, doi: 10.53513/jis.v22i2.9409.

[5] M. Amin, “Komunikasi Berbasis Teks,” M. Miftakul Amin, vol. III, no. September, pp. 129–136, 2016.

[6] C. Christian, S. H. Sitorus, and I. Nirmala, “Implementasi Algoritma Rsa Dan One Time Password (Otp) Untuk Pengamanan Data Pengguna Dan Proses Transaksi Pada Website E-Commerce,” Coding Jurnal Komputer dan Aplikasi, vol. 11, no. 1, p. 62, 2023, doi: 10.26418/coding.v11i1.58684.

[7] U. Indriani, O. Alfina, and N. Syahputri, “Penerapan Algoritma RSA Dalam Keamanan File Ms Word,” ulfah indriani, vol. 01, no. 02, pp. 95–100, 2021, [Online]. Available: http://repository.potensi-utama.ac.id/jspui/handle/123456789/5074