Contents lists available online at TALENTA Publisher

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

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

Signcryption with Matrix Modification of RSA Digital Signature Scheme and Cayley-Purser Algorithm

Cindy Laurent Ginting[1], Mohammad Andri Budiman[2*] and Sawaluddin[3]

[1,2]*Master of Informatics Program, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan, Indonesia*
[3]*Doctor of Mathematics Program, Faculty of Mathematics and Natural Sciences, Universitas Sumatera Utara, Medan, Indonesia*

email : [1]*cindylaurentginting@gmail.com,* [2]*mandrib@usu.ac.id,* [3]*sawal@usu.ac.id*

ARTICLE INFO

Article history:
Received 11 June 2023
Revised 13 January 2024
Accepted 29 Januari 2024
Published online 31 January 2024

Keywords:
Signcryption,
Matrix Modification of RSA Digital Signature Scheme,
Cayley-Purser Algorithm,
Encrypt-then-Sign,
Hash Function,
MD5

Corresponding Author:
mandrib@usu.ac.id

ABSTRACT

The sender must ensure the security of messages and authenticated messages in messaging communications. Additionally, the sender must guarantee the message's integrity and cannot deny its authenticity or involvement with the message. This aspect is more robust because the recipient can verify, ensuring that the message originates from an authorized sender. In addition to this crucial aspect, the Signcryption method employing the Matrix Modification of RSA Digital Signature Scheme and the Cayley-Purser Algorithm can accomplish both of the objectives of this study. Encrypt-then-sign is the Signcryption method used, and the MD5 hash function performs one-way hashing during the signing procedure to enhance message security. This study tested the message plaintext in the form of a collection of strings consisting of uppercase (capital), lowercase (small), numbers (numeric), and other punctuation characters with varying numbers of characters in each string, as well as the value of modulus n from 10 digits up to its maximum length, which is unconstrained. The test results indicate that the time required for encryption and decryption is proportional to the number of plaintext characters used

IEEE style in citing this article:
C.L. Ginting, M. A. Budiman and Sawaluddin" Signcryption with Matrix Modification of RSA Digital Signature Scheme and Cayley-Purser Algorithm," DATA SCIENCE: JOURNAL OF COMPUTING AND APPLIED INFORMATICS (JoCAI), vol. 8, no. 1, pp. 14 - 24. 2024.

1. Introduction

In the era of electronic mail, message exchange activities through open channels such as the internet, there are two essential points to note: private messages can be signed. The sender of the message must be able to guarantee and assure the recipient of the message's confidentiality, integrity, and authenticity, and the sender cannot deny his involvement in the message. In addition to being a necessary aspect and purpose in message exchange communication, this is achieved through the Public Key Cryptography and Public Key Digital Signature Scheme. In Public Key Cryptography, communicating parties can securely exchange keys over a public channel using the Diffie-Hellman Key Exchange Protocol, which addresses security concerns regarding the key exchange process. This protocol introduced the concept of a public key, which is published in an open channel and available to all parties wishing to communicate and encrypt messages [1]. As for decrypting encrypted messages, the private key will be

e- ISSN: 2580-829X | p- ISSN: 2580-6769

used. Therefore, couriers or other secure channels are not required to disseminate keys. This concept eliminates the need for channels that distribute keys securely [2], which are prohibitively expensive to build and quite challenging to manage [3].

Exchanging messages in communication must assure not only the security of messages and authenticated messages but also the integrity of the message, and the sender cannot deny the message's authenticity and involvement. This aspect is more robust because the recipient can verify and ensure that the message was sent by an authorized originator [4]. Zheng was the first to introduce the term Signcryption. The method is Public Key Cryptography, which simultaneously satisfies the requirements of digital signatures and encryption at a lower cost than the conventional method of signing and encrypting messages [5]. Jee Hea An, Yevgeniy Dodis, and Tal Rabin improved slightly from the terms introduced by Zheng in 2002, aiming to achieve greater efficiency rather than performing signature and encryption processes separately. Efficiency is only one concern (though significant) when designing secure concurrent signatures and encryption. Consequently, the term Signcryption can be applied to any scheme that aims to achieve confidentiality and authentication in the configuration of public keys. The formal definition of security developed [6] is met regardless of the performance throughout the process.

2. Method

The exchange of private messages through open channels involves ensuring the security and confidentiality of messages as well as protecting them from forgery, and the sender cannot refute sending the message. It is difficult to send confidential communications through open channels because they are not necessarily secure, whereas delivery via a secure channel will incur relatively high costs. In this instance, the symmetric cryptographic algorithm scheme is not yet qualified, making key distribution difficult. Since anyone can create encrypted communications, there is still a risk of counterfeiting crimes. Additionally, it is simple to forge or duplicate digital signatures, and using a trusted third party (the arbitrator) for digital signatures incurs relatively high costs. Similarly, symmetric digital signature schemes require a trusted arbitrator if they are to be implemented. This study, therefore, relies on asymmetric cryptographic methods (public keys) and asymmetric digital signature schemes (public keys) to enhance the security of messaging communication. The Matrix Modification of RSA Digital Signature Scheme and the Cayley-Purser Algorithm will be implemented in this investigation. The encipherment and decipherment processes are relatively quick, and the resultant file size is modest. In addition, it can guarantee the integrity and authenticity of the message in cases where the sender cannot deny his involvement. This scheme is referred to as signcryption with the encrypt-then-sign method. Figure 1. depicts the schematic of the signcryption scheme with the encrypt-then-sign method designed for this study.

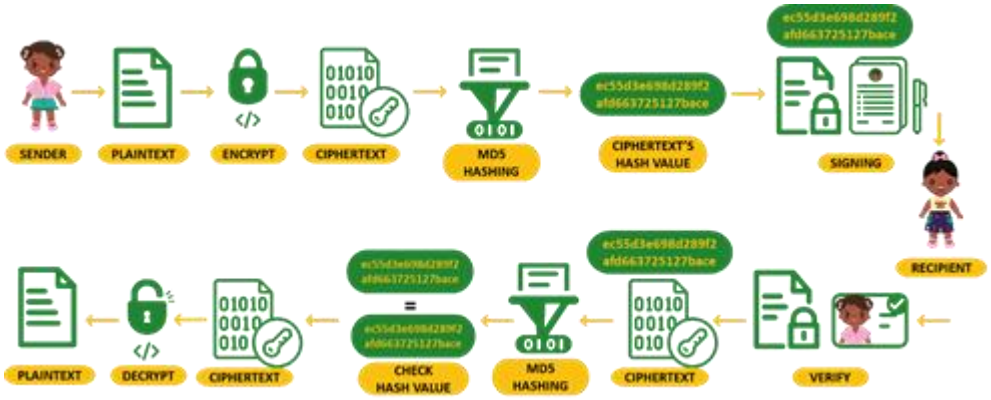

Figure 1. General Diagram of Signcryption Scheme with Encrypt-then-Sign Method

2.1 Cayley-Purser Algorithm

The Cayley-Purser algorithm is a public key cryptographic algorithm published in 1999 by 16-year-old Irish girl Sarah Flannery [7]. The research of Sarah Flannery is based on unpublished work by Michael Purser. The paper describes a novel public-key cryptographic scheme that employs non-commutative multiplication. The Cayley-Purser algorithm applies matrices to implement its schemes due to the non-commutative nature of matrix multiplication. Consequently, this algorithm will rely on multiplication, and its computational process will be significantly quicker than that of the RSA Algorithm, which employs exponential computing. The employed matrix is a square matrix with an order of 2x2. In addition, the employed matrix is derived from the multiplication group $G = GL(2,Z_n)$

1. Key Generation
   a. Randomly generate two prime numbers p and q and compute $n = p \times q$
   b. Randomly generate two matrices X and α in $G = GL(2,Z_n)$ where $X\alpha^{-1} \neq X\alpha$.
   c. Compute $\beta = X^{-1}\alpha^{-1}X$.
   d. Compute $\gamma = X^r$ where $r \in N$
   e. Publish n and the parameter α, β and γ
2. Encryption
   a. Represent messages in a matrix μ
   b. Randomly generate $t \in N$
   c. Compute $\delta = \gamma^t$
   d. Compute $\varepsilon = \delta^{-1}\alpha\delta$
   e. Compute $K = \delta^{-1}\beta\delta$
   f. Encrypt μ with $\mu^{-1} = K\mu K$
3. Decryption
   a. Compute $\lambda = X^{-1}\varepsilon X$
   b. Decrypt μ^{-1} with $\mu = \lambda\mu^{-1}\lambda$

2.2 Matrix Modification of RSA Digital Signature Scheme

S. C. Gupta and Manju Sanghi [8] discovered RSA Digital Signature Scheme with Matrix Modification in 2019 in a paper titled "Matrix Modification of RSA Digital Signature Scheme," which is a modification of RSA Digital Signature Scheme. This scheme represents the message as an h-order square matrix. This modification continues to utilize the modulus n, which is the product of two enormous prime numbers p and q. In contrast to the RSA Digital Signature Scheme, this modified scheme replaces each block of its message plaintext with a modulo n matrix of order $h \times h$. In addition, rather than relying on the Euler Function $\varphi(n)$, the RSA Digital Signature Scheme with Matrix Modification employs the exponential modulus N resulting from $(p^h-1)(q^h-1)$. It, furthermore, was obtained using the concept of an h-order general linear group.

1) Key Generation
   a. Randomly generate two prime numbers p and q Compute $n = p \times q$
   b. Compute $N = (p^h-1)(q^h-1)$ where h is a representation of the order of the matrix then $h = 2$
   c. Randomly generate e where $1 < e < N$ and e is prime relative to N
   d. Compute d^{-1} where $1 < d < N$ and $e \times d \equiv 1 \pmod N$
   e. Publish n and e
2) Signation
   a. Represent messages as an h-order square matrix
   b. Compute the digital signature $S = M^d \bmod n$
3) Verification
   a. Compute to verify the digital signature $M = S^e \bmod n$

3. Result and Discussion

Consider Cici (as the sender) wishes to send a message Kiki (as the recipient). Kiki generates both the public and private keys before transmitting the message. Then, Kiki will publish his public key, enabling

Cici to transmit the encrypted message to Kiki securely.The message is encrypted by Cici, resulting in an unreadable message (ciphertext). Then, The ciphertext is hashed to obtain the hash value that will be signed. The encrypted message (ciphertext) is then sent to Kiki along with its hash value (as the signature). When Kiki receives an encrypted message and the signature from Cici, Kiki verifies it, and if the results of the verification are successfully valid and Cici is the sender, Kiki decrypts the verified encrypted message. Therefore, Kiki can read the message.

1. Key Generation
   a. Randomly generate two prime numbers p = 23 and q = 29
   b. Compute $n = p \times q = 23 \times 29 = 667$
   c. Randomly generate two matrices X and α in $G = GL(2,Z_n)$ where $X\alpha^{-1} \neq X\alpha$

$$X = \begin{bmatrix} 609 & 369 \\ 332 & 639 \end{bmatrix}$$

   d. Compute the determinant of $X = a \times d - b \times c = 609 \times 639 - 369 \times 332 = 266643$
   e. Compute $X^{-1} = \frac{1}{\text{determinan } X} \times \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \pmod n$

$$X^{-1} \equiv \frac{1}{\text{determinan } X} \times \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \pmod n$$

$$X^{-1} \equiv \frac{1}{266643} \times \begin{bmatrix} 639 & -369 \\ -332 & 609 \end{bmatrix} \pmod{667}$$

$$X^{-1} \equiv 266643^{-1} \times \begin{bmatrix} 639 & -369 \\ -332 & 609 \end{bmatrix} \pmod{667}$$

$$X^{-1} = 650 \times \begin{bmatrix} 639 & -369 \\ -332 & 609 \end{bmatrix} \pmod{667}$$

$$X^{-1} = \begin{bmatrix} 476 & 270 \\ 308 & 319 \end{bmatrix} \pmod{667}$$

   f. Randomly generate matrix α

$$\alpha = \begin{bmatrix} 265 & 185 \\ 365 & 657 \end{bmatrix}$$

   g. Compute the determinant of $\alpha = 106580$
   h. Compute $\alpha^{-1} = \begin{bmatrix} 143 & 311 \\ 217 & 546 \end{bmatrix} \pmod{667}$
   i. Compute $\beta = X^{-1}\alpha^{-1}X$.

$$\beta \equiv X^{-1}\alpha^{-1}X \pmod{493}$$

$$\beta \equiv \begin{bmatrix} 476 & 270 \\ 308 & 319 \end{bmatrix} \times \begin{bmatrix} 143 & 311 \\ 217 & 546 \end{bmatrix} \times \begin{bmatrix} 609 & 369 \\ 332 & 639 \end{bmatrix} \pmod{667}$$

$$\beta \equiv \begin{bmatrix} 545 & 145 \\ 390 & 144 \end{bmatrix} \pmod{667}$$

   j. Randomly generate $r = 8$
   k. Compute $\gamma = X^r$

$$\gamma \equiv X^r \pmod{493}$$

$$\gamma \equiv \begin{bmatrix} 609 & 369 \\ 332 & 639 \end{bmatrix}^8 \pmod{667}$$

$$\gamma \equiv \begin{bmatrix} 400 & 112 \\ 108 & 241 \end{bmatrix} \pmod{667}$$

   l. Compute $N = (p^h-1)(q^h-1)$ where $h = 2$

$$N = (p^h-1)(q^h-1) = (23^2-1)(29^2-1) = 443520$$

   m. Randomly generate $e = 331253$
   n. Compute d^{-1} where $1 < d < N$ and $e \times d \equiv 1 \ mod \ N$

$$\Leftrightarrow e \times d \equiv 1 \pmod N$$

$$\Leftrightarrow 331253 \times d \equiv 1 \pmod{443520}$$

$$\Leftrightarrow d^{-1} \equiv 331253 \pmod{443520}$$

$$\Leftrightarrow d \equiv 115037 \pmod{443520}$$

2. Encryption
   a. Transform the ASCII code message "K" into a matrix then add 1111 to complete the 2x2 matrix's elements = 75 + 1111 = 1186
   b. Represent message in a matrix $\mu = \begin{bmatrix} 1 & 1 \\ 8 & 6 \end{bmatrix}$
   c. Randomly generate $t = 8$
   d. Compute $\delta = \gamma^t$

$$\delta \equiv \gamma^t \pmod{493}$$

$$\delta \equiv \begin{bmatrix} 400 & 112 \\ 108 & 241 \end{bmatrix}^8 \pmod{667}$$

$$\delta \equiv \begin{bmatrix} 574 & 489 \\ 543 & 386 \end{bmatrix} \pmod{667}$$

   e. Compute the determinant of $\delta = -43963$
   f. Compute $\delta^{-1} = \begin{bmatrix} 310 & 257 \\ 224 & 499 \end{bmatrix} \pmod{667}$
   g. Compute $\varepsilon = \delta^{-1}\alpha\delta$

$$\varepsilon \equiv \delta^{-1}\alpha\delta \pmod{667}$$

$$\varepsilon \equiv \begin{bmatrix} 310 & 257 \\ 224 & 499 \end{bmatrix} \times \begin{bmatrix} 265 & 185 \\ 365 & 657 \end{bmatrix} \times \begin{bmatrix} 574 & 489 \\ 543 & 386 \end{bmatrix} \pmod{667}$$

$$\varepsilon \equiv \begin{bmatrix} 371 & 175 \\ 648 & 41 \end{bmatrix} \pmod{667}$$

   h. Compute $K = \delta^{-1}\beta\delta$

$$K \equiv \delta^{-1}\beta\delta \pmod{493}$$

$$K \equiv \begin{bmatrix} 310 & 257 \\ 224 & 499 \end{bmatrix} \times \begin{bmatrix} 545 & 145 \\ 390 & 144 \end{bmatrix} \times \begin{bmatrix} 574 & 489 \\ 543 & 386 \end{bmatrix} \pmod{667}$$

$$K \equiv \begin{bmatrix} 391 & 550 \\ 17 & 254 \end{bmatrix} \pmod{493}$$

   i. Compute $\mu^{-1} = K\mu K$

$$\mu^{-1} \equiv K\mu K \pmod{493}$$

$$\mu^{-1} \equiv \begin{bmatrix} 391 & 550 \\ 17 & 254 \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 8 & 6 \end{bmatrix} \times \begin{bmatrix} 391 & 550 \\ 17 & 254 \end{bmatrix} \pmod{667}$$

$$\mu^{-1} \equiv \begin{bmatrix} 394 & 112 \\ 276 & 272 \end{bmatrix} \pmod{667}$$

   j. Hashing μ^{-1} and ε as the ciphertext message with MD5 using the python standard library for cryptographic services
   (source library: https://docs.python.org/3/library/hashlib.html)

$$\left[\begin{bmatrix} 394 & 112 \\ 276 & 272 \end{bmatrix}, \begin{bmatrix} 371 & 175 \\ 648 & 41 \end{bmatrix}\right] = \text{f9792d589cc5a7f4fb23e50e9f28e80f}$$

   k. Convert hash value in ASCII Code to hash value 2×2 matrix = [[1, 2, 1, 3], [1, 1, 6, 8], [1, 1, 6, 6], [1, 1, 6, 8], [1, 1, 6, 1], [1, 2, 1, 1], [1, 1, 6, 4], [1, 1, 6, 7], [1, 1, 6, 8], [1, 2, 1, 0], [1, 2, 1, 0], [1, 1, 6, 4], [1, 2, 0, 8], [1, 1, 6, 6], [1, 2, 1, 3], [1, 1, 6, 3], [1, 2, 1, 3], [1, 2, 0, 9], [1, 1, 6, 1], [1, 1, 6, 2], [1, 2, 1, 2], [1, 1, 6, 4], [1, 1, 5, 9], [1, 2, 1, 2], [1, 1, 6, 8], [1, 2, 1, 3], [1, 1, 6, 1], [1, 1, 6, 7], [1, 2, 1, 2], [1, 1, 6, 7], [1, 1, 5, 9], [1, 2, 1, 3]]

Char	Decimal (ASCII Code)	Decimal (ASCII Code) + 1111
f	102	1213
9	57	1168
7	55	1166
…	…	…
f	102	1213

l. Compute the hash value that has been converted into a matrix with the Matrix Modification of RSA Digital Signature Scheme $S = M^d \bmod n$

$$S = M^d \bmod n = \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}^{115037} \bmod 667 = \begin{bmatrix} 3 & 182 \\ 91 & 185 \end{bmatrix}$$

$$S = M^d \bmod n = \begin{bmatrix} 1 & 1 \\ 6 & 8 \end{bmatrix}^{115037} \bmod 667 = \begin{bmatrix} 73 & 517 \\ 434 & 357 \end{bmatrix}$$

$$S = M^d \bmod n = \begin{bmatrix} 1 & 1 \\ 6 & 6 \end{bmatrix}^{115037} \bmod 667 = \begin{bmatrix} 422 & 422 \\ 531 & 531 \end{bmatrix}$$

…

$$S = M^d \bmod n = \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}^{115037} \bmod 667 = \begin{bmatrix} 3 & 182 \\ 91 & 185 \end{bmatrix}$$

Signed hash value matrix: [[3, 182, 91, 185], [73, 517, 434, 357], [422, 422, 531, 531], [73, 517, 434, 357], [400, 267, 268, 400], [22, 416, 208, 22], [197, 400, 399, 63], [51, 340, 39, 90], [73, 517, 434, 357], [299, 300, 150, 149], [299, 300, 150, 149], [197, 400, 399, 63], [1, 270, 0, 279], [422, 422, 531, 531], [3, 182, 91, 185], [217, 651, 571, 185], [3, 182, 91, 185], [1, 148, 0, 593], [400, 267, 268, 400], [333, 167, 335, 500], [248, 496, 248, 496], [197, 400, 399, 63], [224, 615, 407, 475], [248, 496, 248, 496], [73, 517, 434, 357], [3, 182, 91, 185], [400, 267, 268, 400], [51, 340, 39, 90], [248, 496, 248, 496], [51, 340, 39, 90], [224, 615, 407, 475], [3, 182, 91, 185]]

3. Verification

a. Compute to verify the digital signature $M = S^e \bmod n$

$$M = S^e \bmod n = \begin{bmatrix} 3 & 182 \\ 91 & 185 \end{bmatrix}^{331253} \bmod 667 = \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}$$

$$M = S^e \bmod n = \begin{bmatrix} 73 & 517 \\ 434 & 357 \end{bmatrix}^{331253} \bmod 667 = \begin{bmatrix} 1 & 1 \\ 6 & 8 \end{bmatrix}$$

$$M = S^e \bmod n = \begin{bmatrix} 422 & 422 \\ 531 & 531 \end{bmatrix}^{331253} \bmod 667 = \begin{bmatrix} 1 & 1 \\ 6 & 6 \end{bmatrix}$$

…

$$M = S^e \bmod n = \begin{bmatrix} 3 & 182 \\ 91 & 185 \end{bmatrix}^{331253} \bmod 667 = \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}$$

Verification hash value matrix = [[1, 2, 1, 3], [1, 1, 6, 8], [1, 1, 6, 6], [1, 1, 6, 8], [1, 1, 6, 1], [1, 2, 1, 1], [1, 1, 6, 4], [1, 1, 6, 7], [1, 1, 6, 8], [1, 2, 1, 0], [1, 2, 1, 0], [1, 1, 6, 4], [1, 2, 0, 8], [1, 1, 6, 6], [1, 2, 1, 3], [1, 1, 6, 3], [1, 2, 1, 3], [1, 2, 0, 9], [1, 1, 6, 1], [1, 1, 6, 2], [1, 2, 1, 2], [1, 1, 6, 4], [1, 1, 5, 9], [1, 2, 1, 2], [1, 1, 6, 8], [1, 2, 1, 3], [1, 1, 6, 1], [1, 1, 6, 7], [1, 2, 1, 2], [1, 1, 6, 7], [1, 1, 5, 9], [1, 2, 1, 3]]

b. Convert hash value matrix to hash value in ASCII

Decimal (ASCII Code) - 1111	Decimal (ASCII Code)	Char
1213	102	f
1168	57	9
1166	55	7
…	…	…
1213	102	f

Hash value = f9792d589cc5a7f4fb23e50e9f28e80f

4. Decryption
   a. *Compute* $\lambda = X^{-1}\varepsilon X$

$$\lambda \equiv X^{-1}\varepsilon X$$

$$\lambda \equiv \begin{bmatrix}476 & 270\\ 308 & 319\end{bmatrix}\begin{bmatrix}371 & 175\\ 648 & 41\end{bmatrix}\begin{bmatrix}609 & 369\\ 332 & 639\end{bmatrix}(\text{mod } 667)$$

$$\lambda \equiv \begin{bmatrix}458 & 295\\ 379 & 621\end{bmatrix}(\text{mod } 667)$$

   b. Decrypt μ^{-1} with $\mu = \lambda\mu^{-1}\lambda$

$$\mu \equiv \lambda\mu^{-1}\lambda$$

$$\mu \equiv \begin{bmatrix}458 & 295\\ 379 & 621\end{bmatrix}\times\begin{bmatrix}394 & 112\\ 276 & 272\end{bmatrix}\times\begin{bmatrix}458 & 295\\ 379 & 621\end{bmatrix}(\text{mod } 667)$$

$$\mu \equiv \begin{bmatrix}1 & 1\\ 8 & 6\end{bmatrix}(\text{mod } 667) \equiv 1186\text{-}1111 = 75 \text{ (ASCII Code)} = \text{K (ASCII Character)}$$

The test results represent the key generation, encryption, and decryption processes using the signcryption method with the Cayley-Purser algorithm, as well as the signing and verification processes of signatures using the Matrix Modification of RSA Digital Signature Scheme. This test was conducted on text messages consisting of a compilation of strings containing uppercase (capital), lowercase (small), numeric, and punctuation characters with varying numbers of characters. In addition to using the modulus n value from 10 to an unlimited number of integers, whose length is not constrained. Table 4 displays the results of a comparison of the length of the modulus to the processing time based on the results of the experiments conducted.

Table 1. Running Time of 100 Characters

Running Time of 100 Characters			
Modulus Length (decimal digits)	**Key Generation (seconds)**	**Encryption (seconds)**	**Decryption (seconds)**
10	0.0011	0.0067	0.0040
110	0.0233	0.0487	0.0385
210	0.0825	0.1385	0.1291
310	0.3192	0.3155	0.3102
410	0.9054	0.6045	0.6078
510	1.7854	1.0449	1.0785

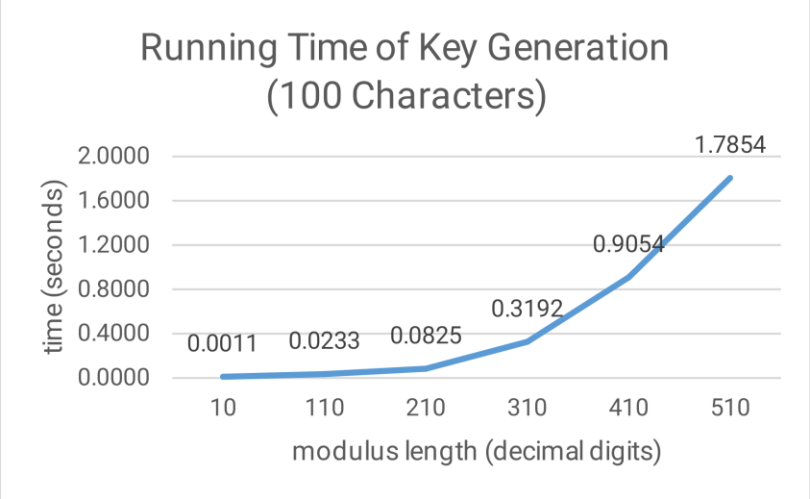

Figure 1. Running Time of Key Generation (100 Characters)

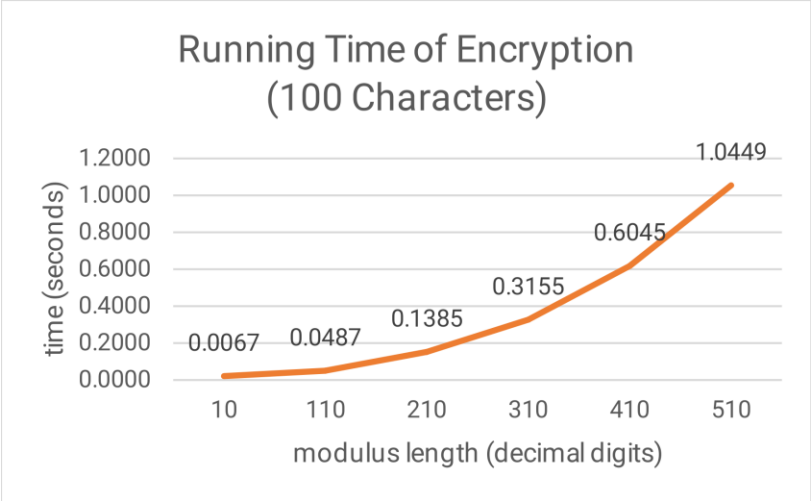

Figure 2. Running Time of Encryption (100 Characters)

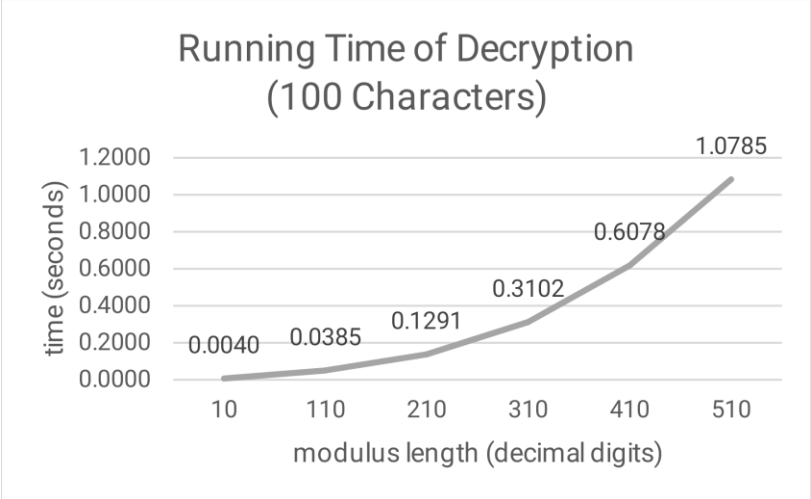

Figure 3. Running Time of Decryption (100 Characters)

Table 2. Running Time of 100 Characters

Running Time of 1000 Characters			
Modulus Length (decimal digits)	**Key Generation (seconds)**	**Encryption (seconds)**	**Decryption (seconds)**
10	0.0014	0.0605	0.0301
110	0.0143	0.3805	0.3405
210	0.0656	1.1637	0.9948
310	0.2043	2.5969	2.4138
410	0.3272	5.3579	5.2393
510	0.8997	8.9527	8.5678

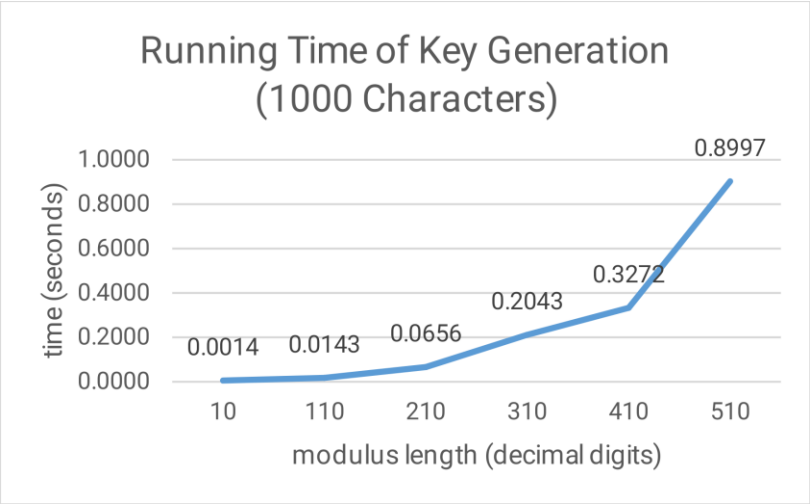

Figure 4. Running Time of Key Generation (1000 Characters)

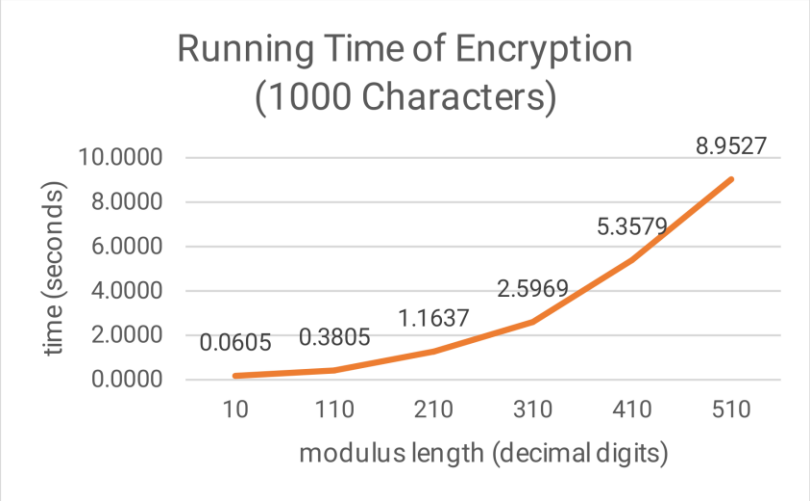

Figure 5. Running Time of Encryption (1000 Characters)

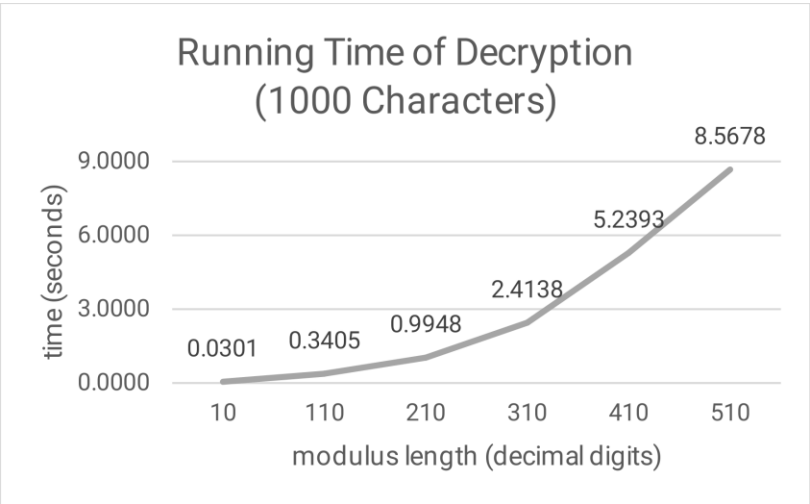

Figure 6. Running Time of Decryption (1000 Characters)

Table 3. Running Time of 10000 Characters

Running Time of 10000 Characters			
Modulus Length (decimal digits)	**Key Generation (seconds)**	**Encryption (seconds)**	**Decryption (seconds)**
10	0.0014	0.4998	0.3004
110	0.0089	3.8796	3.3482
210	0.0889	11.3292	10.5405
310	0.3294	26.3025	24.7352
410	0.1681	51.4552	49.8270
510	1.5487	91.8697	88.7776

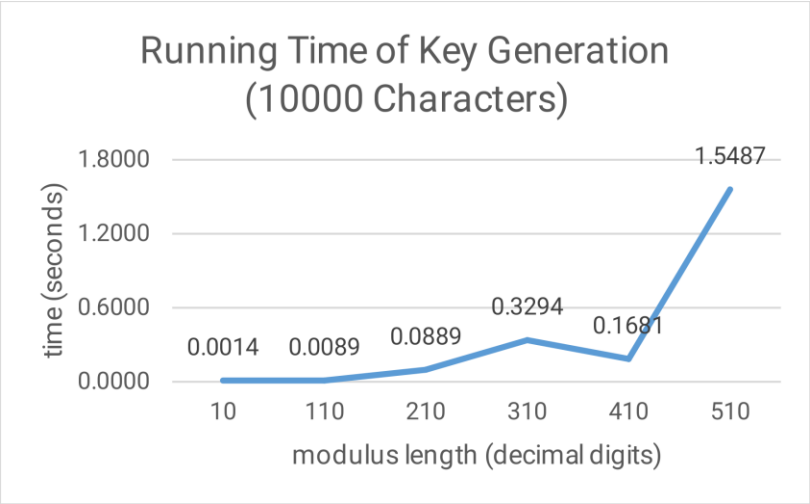

Figure 7. Running Time of Key Generation (10000 Characters)

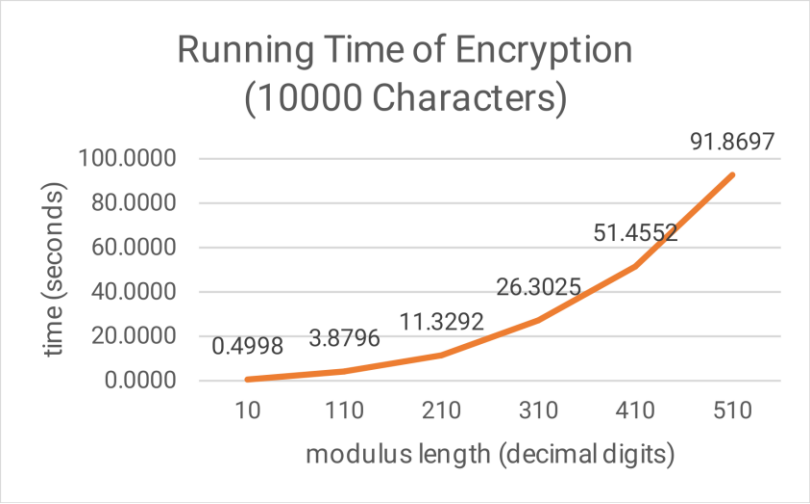

Figure 8. Running Time of Key Generation (10000 Characters)

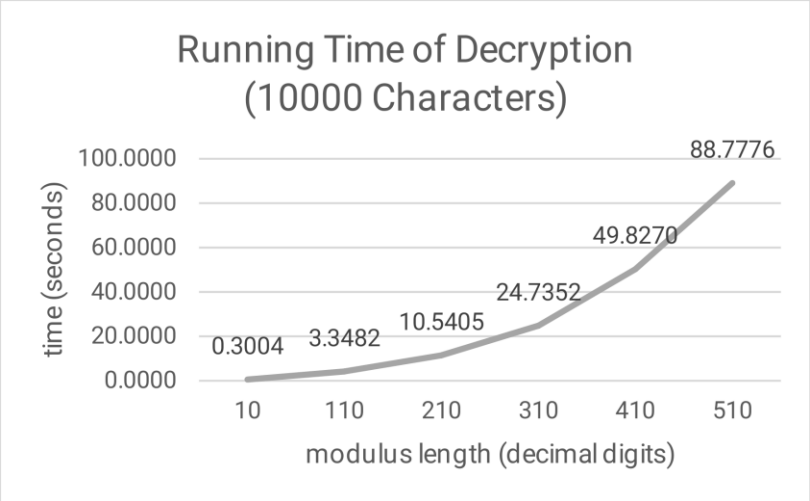

Figure 9. Running Time of Key Generation (10000 Characters

4. Conclusion

Analysis and testing of the signcryption method, namely encrypt then sign and verify then decrypt using the RSA Digital Signature Scheme with Matrix Modification and the Cayley-Purser Algorithm, was conducted successfully with plaintext in the form of text in the form of a collection of strings consisting of uppercase letters (capital), lowercase letters, numbers (numeric), and other punctuation characters with varying numbers of characters in each string as well as using the value of the modulus n from 10 digits to the maximum whose length is not limited. The time of the encryption and decryption process using the Cayley-Purser Algorithm is directly proportional to the number of plaintext characters, where the greater the number of characters, the longer the real running time in the encryption and decryption process.

REFERENCES

[1] Diffie, W., & Hellman, M. E. (1976). New directions in cryptography. IEEE Transactions on Information Theory, 22(6).

[2] Merkle, R. C. (1978). Secure Communications Over Insecure Channels. Communications of the ACM, 21(4).

[3] Rachmawati, D., & Budiman, M. A. (2020). On Using the First Variant of Dependent RSA Encryption Scheme to Secure Text: A Tutorial. Journal of Physics: Conference Series, 1542(1).

[4] Rivest, Shamir, & Adleman. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2).

[5] Zheng, Y. (1997). Signcryption and its applications in efficient public key solutions. International Workshop on Information Security.

[6] An, J. H., Dodis, Y., & Rabin, T. (2002). On the security of joint signature and encryption. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2332, 83–107.

[7] Flannery, S. (1999). Cryptography: An investigation of a new algorithm vs. the RSA. In Collected From Http:\\ Cryptome. Org.

[8] Gupta, S. C., & Sanghi, M. (2019). Matrix Modification of RSA Digital Signature Scheme. Journal of Applied Security Research, 16(1).