American Journal of Computer Science and Technology
2022; 5(4): 198-203
http://www.sciencepublishinggroup.com/j/ajcst
doi: 10.11648/j.ajcst.20220504.11
ISSN: 2640-0111 (Print); ISSN: 2640-012X (Online)

Showcasing White-Box Implementation of the RSA Digital Signature Scheme

Colin Chibaya*, Mfundo Monchwe, Taryn Nicole Michael, Eli Bila Nimy

Department of Computer Science, Data Science and Information Technology, Sol Plaatje University, Kimberley, South Africa

Email address:

colin.chibaya@spu.ac.za (Colin Chibaya), 201726904@spu.ac.za (Mfundo Monchwe), 201800323@spu.ac.za (Taryn Nicole Michael), 201802052@spu.ac.za (Eli Bila Nimy)

*Corresponding author

To cite this article:

Colin Chibaya, Mfundo Monchwe, Taryn Nicole Michael, Eli Bila Nimy. Showcasing White-Box Implementation of the RSA Digital Signature Scheme. *American Journal of Computer Science and Technology*. Vol. 5, No. 4, 2022, pp. 198-203. doi: 10.11648/j.ajcst.20220504.11

Received: July 30, 2022; **Accepted**: August 15, 2022; **Published**: October 18, 2022

Abstract: Data security is a priority in online transactions. Data security, in this context, refers to both data confidentiality, data integrity, and data authenticity when online transactions are completed. While a lot has been done to tighten data confidentiality, algorithms to address data integrity and data authenticity are rare. The RSA digital signature scheme dominates and is often connoted when data integrity and data authenticity problems are tabled. However, the original RSA digital signature scheme is not easy to comprehend by layman. Most component units of the RSA digital signature scheme require further clarity to facilitate reproducibility and hence productivity. This study showcases the implementation of a white-box RSA digital signature scheme. In this context, a digital signature is a computational algorithm used to ensure data confidentiality, integrity, and authenticity after online transactions. It is an algorithm that ensures that data is safe, has not been tampered with, and the claimed sender is truly the sender. We build the proposed implementation from an understanding that the RSA digital signature scheme is an asymmetric model which uses two keys. One key is used to sign data such that it can only be verified using the second key. A quantitative research approach was followed in which the effectiveness of the white-box RSA digital signature scheme was evaluated with respect to the execution time and signature verification accuracy. Execution time was assessed for different values of p, q, and data lengths. Similarly, verification accuracy was also assessed with different values of p, q, and data lengths. A tradeoff between security and execution time was noted as apparent. Low accuracy was observed when the values of p and q are small. Thus, big values of p and q are recommended for better data security.

Keywords: Digital Signature Scheme, Private Key, Public Key

1. Introduction

The evolution of technology has seen an increased use of online transactions in business. However, when online transactions are completed, data security concerns are raised. Although efforts to ensure data confidentiality [10] are visible through various cryptographic algorithms in the literature, data confidentiality alone has failed to curb all probable security threats that may ensue during online transactions. The need to explore alternative mechanisms to ensure data security beyond data confidentiality alone is ostensive [7].

We consent to the notion that data integrity and authenticity create further opportunities to curb security threats in online transactions. In this context, data integrity is about accuracy, legitimacy, and consistency during transactions [8, 17]. It provides the confidence that data has not been tampered with in the process. Transacting parties would appreciate all evidence that their data has only been accessed or modified by those who are authorized to do so. On the other hand, data authenticity is the act of proving the identity of the sources of data [9]. It is about verifying that the sender is, indeed, the one claimed to be.

Although a lot of work has been reported about tightening data confidentiality [10], algorithms that bring about data integrity and authenticity are relatively few [17]. More so, the few data integrity and authenticity algorithms that exist are

purportedly hard to comprehend by layman developers for easy reproducibility. This article showcases a white-box implementation of the RSA digital signature scheme with the hope of simplifying the scheme, while also, bringing about data integrity and authenticity during online transactions [17].

Digital signatures are computational algorithms used to verify that the sender of data is truly the claimed sender [9]. They are mathematical procedures [1] used to prove the originality of data [2] by allowing the receiver of data to confirm the sender [2]. That way, data impersonation and forgery are minimized. However, implementation of practical digital signature schemes is impeded by the complexity of the mathematics behind signing and verifying the signatures. Precisely, mathematics is pivotal and core in the implementation of digital signature schemes [4]. In fact, much of the mathematics involved in this process is intractable in polynomial times [11, 16]. We seek to showcase the white-box implementation of a simplified RSA digital signature scheme towards replicability and reproducibility of the algorithm, and hence improved productivity.

Three services are delivered by digital signature schemes, namely, non-repudiation, data integrity, and data authenticity [1, 3]. Contextually, non-repudiation ensures that the senders of data do not deny their actions [3]. We indicated that data authenticity is the act of ensuring that data came from valid sources [1]. Often, digital signatures incorporate transaction time, date, and the details of the sender. We said that data integrity verifies that data was not altered in the process [3]. The proposed white-box implementation of the RSA digital signature scheme showcases the design of these three aspects.

1.1. Problem Statement

The proposed white-box implementation of the RSA digital signature scheme is based on asymmetric crypto systems [2, 5]. An asymmetric crypto system utilizes two keys, one to sign data on the sender side, and another used by the receiver to confirm the sender [1, 6]. Three objectives summarize the problem addressed in this study as follows:

1) We want to showcase the white-box implementation of the RSA digital signature scheme.
2) We will then evaluate the implemented white-box RSA digital signature scheme using different input parameters.
3) We want to be able to pinpoint the key issues in this proposed white-box implementation of the RSA scheme.

Achieving the three objectives is the key deliverable sought.

1.2. Overview

The rest of the article proceeds as follows: Section 2 reviews related work, emphasizing attempts made in the past to improve the RSA digital signature scheme. In section 3, we present the methods followed in addressing the problem, focusing on the components of the white-box RSA digital scheme. The results yielded from evaluating the scheme in terms of execution time [16] and verification accuracy are presented in section 4. Conclusions are drawn in section 5, highlighting the contributions and future direction of the study.

2. Related Work

Attempts to improve the RSA digital signature scheme have been mainly about varying the hash function [1]. In most cases, the SHA-256 protocol [1] was considered, especially when biometric studies that involved facial recognition digital scheme were proposed [1]. In their case, the sender's face served as the digital signature to lock and unlock access to data [1]. Such improvements to the RSA digital signature scheme where faces served as digital signatures are inspiring.

An enthralling analog digital signature scheme was also proposed in the work of Gil, S. K. [10]. Here, the RSA digital signature scheme was modified with optical phase-shifting digital holography [10]. In their work, an analog signature was generated to ensure both data confidentiality [10], data integrity [17], and data authenticity [10]. The holographic encryption technique that was applied to a hash value showed analog pseudo-random patterns in the emanating analog signature scheme. Enticing was that the public key and private key that were required to validate the analog digital signature were found using the same holographic encryption [10]. Also, some double encryption was required when creating the digital signature, which tremendously improved the strength of the RSA digital signature scheme. As a result, very resilient digital signatures emanated, which further motivated the undertaking of this study.

Additionally, the original RSA digital signature scheme uses a minimum of *1024* bits as the minimum size of the modulus required for secure data transmission [11]. But developments in technology call for a rise in the modulus size to accommodate large integer factorization. However, such an increase will, in turn, increase the time and computational complexity of the RSA digital signature scheme. Improvements of the RSA digital signature scheme along these lines have already been proposed [11], where instead of representing messages in integer format, data is represented as square matrices. That way, the key size was reduced, as well as the storage space required. In their case, blocks of text were represented using $n \times n$ matrices. The totient functions were changed to the exponentiation denoted as $m = (p^h - 1)(q^h - 1)$ [11]. The value of the exponentiation modulus to replace the totient function was relatively large, making the digital signature scheme thereof far much secure, yet minimizing the storage space.

The potential opportunity to modify the RSA digital signature scheme and guarantee reproducibility is appealing [15]. This article showcases the white-box implementation of the RSA digital signature scheme to, hopefully, provide explainable procedures. These white-box implementation will likely create a baseline avenue for more detailed research.

3. Methods

This is a quantitative study where a model is created and evaluated. The performance of the proposed white-box version of the RSA digital signature scheme will be assessed and analyzed both descriptively and inferentially. The work befits the design science research theoretical framework [12] because of its emphasis on replicability, repeatability, and the likely spiral improvements to the implementation of the component units of the digital signature scheme. Much focus was invested on understanding the steps taken to complete the white-box implementation. The algorithmic procedure that we followed in the implementation of the key generation routine was the first task completed. Incorporation of the digital signature was the second task, which was the theme of the work. Signature verification culminated the tasks completed. Precisely, three routines were connoted in this white-box implementation of the RSA digital signature scheme. These routines can be rephrased as; (a) the key generation routine, (b) the signing sub-system, and (c) the signature verification component. Figure 1 summarizes the features of each routine and how the routines are interlinked. Subsequent sub-sections of this section provide detailed descriptions of how each routine was tackled in computational perspectives.

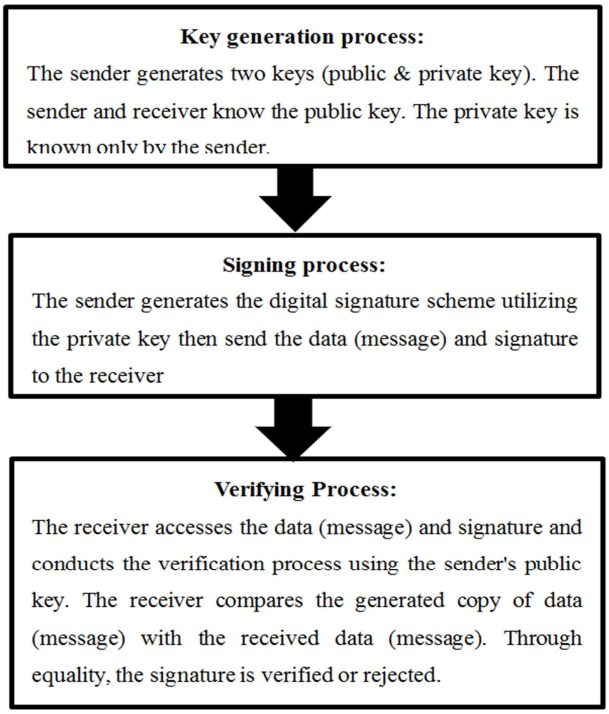

Figure 1. *Units of the rsa digital signature scheme.*

3.1. Key Generation

This routine is mainly based on the use of two prime numbers p and q. Preferably, these prime numbers should be very large, with a length of more than three digits. The routine multiplies these two prime numbers to get an integer n. This n is the modulo value used in subsequent computations. A totient function, $\varphi(n)$, is also generated from multiplying $(p-1)$ and $(q-1)$. The totient function is a useful parameter when we get to generating the required keys. It is also useful in the determination of another integer, e, which is chosen such that $1 < e < \varphi(n)$ and the $gcd\ (\varphi(n), e)$ should be 1. The process of finding e is quite technical. That way, another integer, d, can be determined such that $e \times d \bmod \varphi(n) = 1$. The public key of the digital signature scheme would be denoted as $K_1 = (n, e)$ while the private key would be denoted as $K_2 = (n, d)$ [13].

Algorithm 3.1 summarizes the steps we follow when completing the key generation process. Note that the primality testing algorithm (algorithm 3.2) is used to tell whether the chosen p and q are prime numbers. Smaller values of p and q are discouraged because it will be easy to factorize the resultant n [13]. Algorithm 3.3 is used to find the greatest common divisor of two numbers. It can also be used to compute the coefficients of the Bezout's identity, which uses x and y such that $ax + by = gcd\ (a, b)$.

Algorithm 3.1: The RSA and key generation

```
void pickPrimes ()
{
  Pick two large prime numbers p and q
  Verify primality of p and q (see algorithm 3.2).
  Find n = p × q
  Find totient function φ(n) = (p − 1) × (q − 1)
  While (K1 = null or K2 = null)
    Pick e and find gcd (φ(n), e) (algorithm 3.3)
    if (gcd (φ(n), e) = 1)
      Select that e if 1 < e < φ(n)
      Find d such that e × d mod φ(n) = 1
      Store the public key as K1 = ( n , e )
      Store the private key as K2 = ( n , d )
    End if
  End while
}
```

Algorithm 3.2: Primality test

```
boolean primeNumber (int p)
{
  if (p = 2)
      return true,
  else if (p< 2) or (p mod 2 = 0)
      return false
   else
      for (k = 3 to √p )
        if (k mod p ≠ 0) and (p mod 2 = 1)
            return true
        else if (p mod k = 0)
            return false
       end for
   end if
}
```

Algorithm 3.3: Extended Euclidean algorithm

```
int gcd (φ(n) , e)
{
  Set x = max (φ(n) ,   e)      //bigger of the two
  Set y = min (φ(n) , e )
  Find m = int (x/y )
  Find r = x mod y
  r = x – my
  If (r=0)
        return y
  else
      return gcd (y, r)
  }
```

3.2. Digital Signature Generation

This sub-system is about signing and verifying the authenticity of the signed data [1]. It uses both the private and public keys [3], where the private key is used by the sender (for signing) and the public key is used by the receiver (for verifying) the data received. Thus, the private key is important during the signing of data while the public key is important during the verification of the signature. As a result, data integrity [17], data authenticity, as well as non-repudiation services are guaranteed [3]. Algorithm 3.4 describes the data signing and signature verification process. In this routine, H represents the SHA-256 hash function.

Algorithm 3.4: Digital signature

```
String sign (String Data)
{
   S =  H × (d  mod n)
   return S
}

boolean Verify (String S, String Data)
{
   D1 = S^e mod n
   If (D1=Data)
         return true
   Else
         return false
}
```

3.3. Integrated RSA Digital Signature Scheme

The three parts of the white-box implementation of the RSA digital signature scheme are illustrated in algorithms 3.1, 3.2 and 3.3 [3, 9]. Although no specific hash function is required, we adopt the commonly used H = SHA-256 for its security [1, 14]. In this context, hashing is about transforming data into a secure unreadable format. It maps data of any arbitrary size to data of a fixed size. In the case of SHA-256 hashing, data is converted to 256-bit strings [14]. At the receiving end, hashed data is compared to the original data to verify its integrity.

The white-box implementation was completed in Python 3.8 on an HP ProBook 450 GS with an Intel® Core™ i5-8250U CPU with 8GB RAM, running Windows 10 Home operating system. The sample space used consisted of randomly generated data (simulated). The data provided as input was of varying lengths. Different upper and lower bounds for the prime numbers were also provided as input. In the program, the user is prompted to input the range in which the program should choose the p and q before the key generation routine is executed and before data signing. We mentioned earlier on that the program was tested for execution time and the efficiency of the signature verification process when different p, q, and data lengths were provided as inputs.

4. Results

Table 1 reports the execution time for different data length, and varying p, and q values. In each case, the program was run for 100 iterations. Key in this testing was the use of the *timeit* library in python. Data was randomly generated using the *string* and *random* libraries. The reported execution times were rounded to 3 decimal places.

Notably, there is a very small change in execution time when the p and q lengths are not changed, even when data length is changed. Thus, for a constant p and q, the time complexity of the program remains consistent (see Figure 2). However, this execution time significantly goes up when large p and q value are selected (see Figure 3). This is purportedly due to the increase in the complexity of the key generation procedure. Nevertheless, higher p and q values guarantee stronger security. An apparent tradeoff between time complexity and security, thus, emanated.

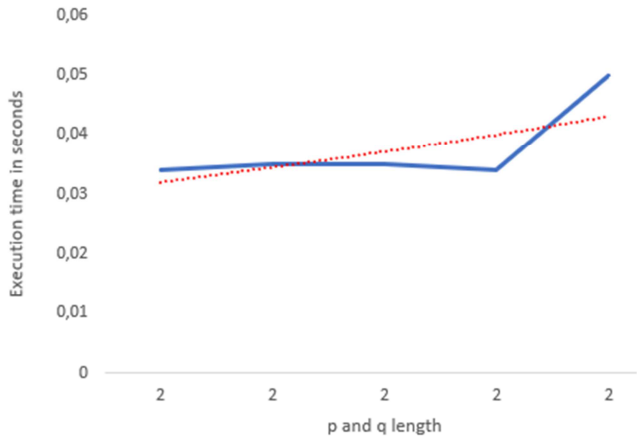

Figure 2. *Execution time for constant p and q length.*

Table 1. *Comparison of execution time.*

Test	data length	*p* and *q* length, range	Iteration	Execution time in seconds
		2 (10, 99)		0.034s
		3 (100, 999)		0.104s
1	10	4 (1000, 9999)	100	0.397s
		5 (10000, 99999)		3.447s
		6 (100000, 999999)		26.774s
		2 (10, 99)		0.035s
		3 (100, 999)		0.101s
2	25	4 (1000, 9999)	100	0.407s
		5 (10000, 99999)		3.727s
		6 (100000, 999999)		28.059s
		2 (10, 99)		0.035s
		3 (100, 999)		0.112s
3	50	4 (1000, 9999)	100	0.431s
		5 (10000, 99999)		3.573s
		6 (100000, 999999)		28.343s
		2 (10, 99)		0.034s
		3 (100, 999)		0.104s
4	100	4 (1000, 9999)	100	0.421s
		5 (10000, 99999)		3.708s
		6 (100000, 999999)		26.468s
		2 (10, 99)		0.050s
		3 (100, 999)		0.104s
5	200	4 (1000, 9999)	100	0.412s
		5 (10000, 99999)		3.270s
		6 (100000, 999999)		29.628s

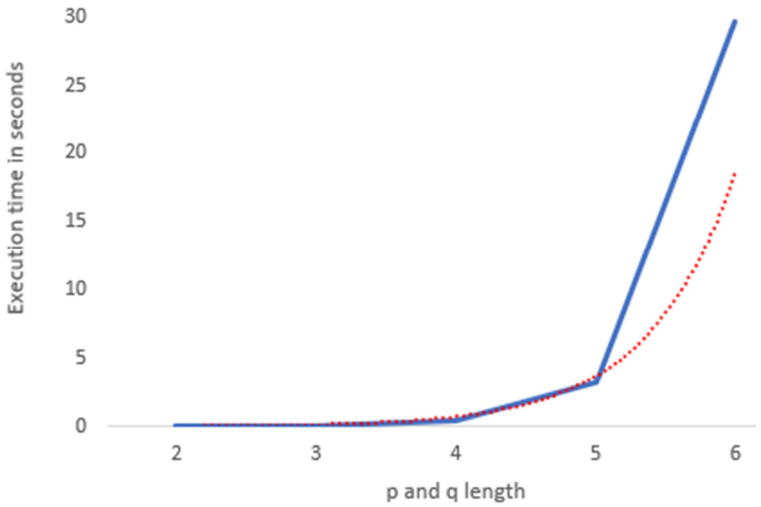

Figure 3. *Execution time for increasing p and q length.*

Table 2. *Comparison of verification outcome.*

Test	*p* value	*q* value	Msg length	Verification
1	3	5	2	Failed
2	3	5	20	Failed
3	7	11	2	Failed
4	7	11	20	Failed
5	13	17	2	Passed
6	13	17	20	Passed
7	23	31	2	Passed
8	23	31	20	Passed
9	823	101	2	Passed
10	823	101	20	Passed
11	1709	1699	2	Passed
12	1709	1699	20	Passed
13	10453	10459	2	Passed
14	10453	10459	20	Passed

Signature verification also relied on p, q, and data lengths. In this case, we determine how well the program performs given different p, q, and data. Table 2 summarizes the performance. Results indicate that the verification process is outstanding for a variety of p, q, and data length. However, verification fails when p and q values are below *11*.

Thus, higher p and q values are preferred. That weakness of the program failing when smaller p and q values are chosen is historic. Although the program will take longer when p and q values are larger, the tradeoff for stronger security is desirable.

5. Conclusion

We described the white-box implementation of the RSA digital signature scheme based on the SHA-256 hash function. Large prime numbers are heavy for processors. However, they bring about stronger security. Execution time stays consistent when the p and q values remain constant [16]. Contrary, time complexity increases when the p and q value are increased. Although the verification effectively works for a wide range of p and q, it should be noted that values below *11* are undesirable.

Two contributions are noted as follows:

1) This white-box implementation of the RSA digital signature scheme is opensource. Hopefully, this may trigger practical applications.
2) The work opens a dialogue on the implementation of better digital signature schemes. Ideally, more research should surface focusing on the simplification of hybrid digital signature scheme [15].

We observe two directions for future work as follows:

1) This implementation may require further scrutiny. It serves as a baseline upon which more detailed analyses may ensue. In future, we will emphasize practical use of hybrid digital signature scheme.
2) More metrics may be considered for evaluating the validity of the improved hybrid RSA digital signature schemes. Currently, time complexity and message length are limited measures.

Acknowledgements

This article was funded through the CAIR (Centre for Artificial Intelligence Research) grant agreement number: CSIR/BEI/HNP/CAIR/2020/10, supported by the Department of Science and Innovation (DSI), South Africa. We are grateful for the mentorship provided by Prof Deshen Moodley, Computer Science Department, University of Cape Town.

References

[1] Ahmed, A., Hasan, T., Abdullatif, F. A, S. T. M and Rahim, M. S. M, 2019. A Digital Signature System Based on Real Time Face Recognition, 2019 IEEE 9th International Conference on System Engineering and Technology (ICSET), pp. 298-302, doi: 10.1109/ICSEngT.2019.8906410.

[2] Lutkevich, B., 2021. What is a Digital Signature? [online] SearchSecurity. Available at: https://searchsecurity.techtarget.com/definition/digital-signature [Accessed 11 November 2021].

[3] Khalique, A., Vishnoi, S., & Shrivastava, V. 2014. Modified RSA Digital Signature Scheme for Data Confidentiality. International Journal of Computer Applications 106 (13), pp. 13-16.

[4] Seo, J., 2020. Efficient digital signatures from RSA without random oracles. Information Sciences, 512, pp. 471-480.

[5] Sari, I., Zarlis, M. and Tulus, T., 2020. Optimization of Data Encryption Modeling Using RSA Cryptography Algorithm as Security E-Mail Data. Journal of Physics: Conference Series, 1471, p. 012068.

[6] Zhou, A. and Tang X., 2011. Research and implementation of RSA algorithm for encryption and decryption, Proceedings of 2011 6th International Forum on Strategic Technology, pp. 1118-1121.

[7] Das, S. B., Mishra, S. K, and Sahu, A. K, 2019. A New Modified Version of Standard RSA Cryptography Algorithm. Samrt Computing Paradigms: New Progresses and Challenges, pp 281-287.

[8] Ganbaatar, G., Nyamdorj, D., Cichon, G. and Ishdorj, T., 2021. Implementation of RSA cryptographic algorithm using SN P systems based on HP/LP neurons. Journal of Membrane Computing, 3 (1), pp. 22-34.

[9] Yang, T., Zhang, Y., Xiao, S. and Zhao, Y., 2021. Digital signature based on ISRSAC. China Communications, 18 (1), pp. 161-168.

[10] Gil, S. K., 2020. Proposal for Analog Signature Scheme Based on RSA Digital Signature Algorithm and Phase-shifting Digital Holography. Current Optics and Photonics, 4 (6), pp. 483-499.

[11] Gupta, S. C. and Sanghi, M., 2021. Matrix Modification of RSA Digital Signature Scheme. Journal of Applied Security Research, 16 (1), pp. 63-70.

[12] Ostrowski Ł., Helfert M., Hossain F. (2011) A Conceptual Framework for Design Science Research. In: Grabis J., Kirikova M. (eds) Perspectives in Business Informatics Research. BIR 2011. Lecture Notes in Business Information Processing, vol 90. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-24511-4_27

[13] Mansour A. H. (2017). Analysis of RSA Digital Signature Key Generation using Strong Prime. In: International Journal of Computer (IJC - Global Society of Scientific Research and Researchers, GSSRR).

[14] N-ABLE (2019). SHA-256 Algorithm Overview. Available at: https://www.n-able.com/blog/sha-256-encryption.

[15] Takehiro M., Masayuki A., Tatsuaki O., 2015. Secure Signature Scheme with Tight Reduction to the RSA Assumption from Indistinguishability Obfuscation. The 32nd Symposium on Cryptography and Information Security.

[16] Rania S., Moustafa A., Mohammad A., 2008. An Efficient Signature System Using Optimized RSA Algorithm. In the International Journal of Computer Science and Network Security, Vol. 8 No. 12.

[17] Fang W., Chen W., Zhang W., Pei J., Gao W., Wang G., 2020. Digital signature scheme for information non-repudiation in blockchain: a state of the art review. Journal on Wireless Communications and Networking. Vol. 56.