

**Network Security and Resilience**

**NSR/AS Lab 4 – Public Key Cryptography**

Xuan Tuan Minh Nguyen

103819212

[Abstract 3](#_Toc167917676)

[Introduction to Public Key Cryptography 4](#_Toc167917677)

[1. *An Introduction* 4](#_Toc167917678)

[*2.* *Procedure of generating the keys* 4](#_Toc167917679)

[*3.* *Encryption procedure* 5](#_Toc167917680)

[*4.* *Decryption procedure* 5](#_Toc167917681)

[Breaking the RSA Algorithm 5](#_Toc167917682)

[Results 7](#_Toc167917683)

[Conclusion 7](#_Toc167917684)

[References 7](#_Toc167917685)

[‌ Diagrams 8](#_Toc167917686)

# **Abstract**

This report is written in order to delves into the heart of RSA algorithm, which is one of the foundation algorithm of public-key cryptography. It will discuss the fundamental procedures that created RSA, which includes the generation process of the resistant key pairs by empowering large prime numbers. While the public key, which is always available to be used publicly, serves as the gateway for encryption. Private key on the other hand, will not be used publicly, but instead it is a critical component that helps breaking the encrypted messages. To demonstrate RSA algorithm, MATLAB is used as the main programming language to showcase the process of decrypting the encrypted messages with different public keys. The code that will be used in this report unveils the interesting mathematical behind the RSA algorithm of achieving the private key from the public key. By exploring this mathematic mechanisms, we will achieve a deeper understanding of the powerful and genious procedure that made up the RSA algorithm and its essential role in safely encapsulating data communication. Furthermore, the abstract also focuses on vulnerabilities and considerations for using RSA securely in real-word scenarios. It shows the importance of using appropriate key size and employing best practices for key generation, storage and management. Finally, the abstract underscores the connection of RSA in the digital age.

# **Introduction to Public Key Cryptography**

## *An Introduction*

Public Key Cryptography is the concept that is dated back into the middle of 1970s to offer preventing public data being access prohibitively, with the massive contribution from Whitfield Diffie and Martin Hellman, who invented the asymmetric key algorithms in 1976 [1]. In 1977, a group of researchers at Massachusetts Institute of Technology (Ronald **Rivest**, Adi **Shamir** and Leonard **Adleman**) has refined this idea, brought a new standardized method for public key cryptography, which is well-known nowaday as RSA (Rivest, Shamir and Adleman) algorithm [2].

## *Procedure of generating the keys*

The Rivest, Shamir and Adleman (RSA) algorithm is an advanced asymmetric cryptogrpahic algorithm that contains two main components, which are the public key and private key. The public key, as by its name, used for encrypting publicly while private key is used for decryption [3]. The procedure belows illustrate the execution procedure of RSA Algorithm.

1. Pick two different prime numbers, which in this case and .

* Both of the prime numbers, which must be integer, must be chosen randomly and must have the same number of bits for security reasons.

1. Compute , which will be used as the modulus of both private and public keys. Moreover, will be the determination of the length of the key.

* could be calculated by multiplying the randomly picked prime number **(.**

1. Obtain the by using **Euler’s totient function** **(.**
2. Determine the **pubic exponent**  such that and **,** which and must be **co-primitive** ( have no common factors except 1).

* In most case, is commonly used with small bit length ; i.e. . However, with the use of much smaller value than mentioned, such as , could expose the vulnerability.

1. Choose the **private exponent**  that satisfies ; i.e. is the multiplicative inverse of **.**

* The **private exponent**  is usally determined using the extended Euclidean algorithm.

1. While the pair that is calculated in step 2 and step 4 will be used as the **public key** encryption. The pair **,** computed in step 2 and step 5, will be the **private key** used for decrypting.

## *Encryption procedure*

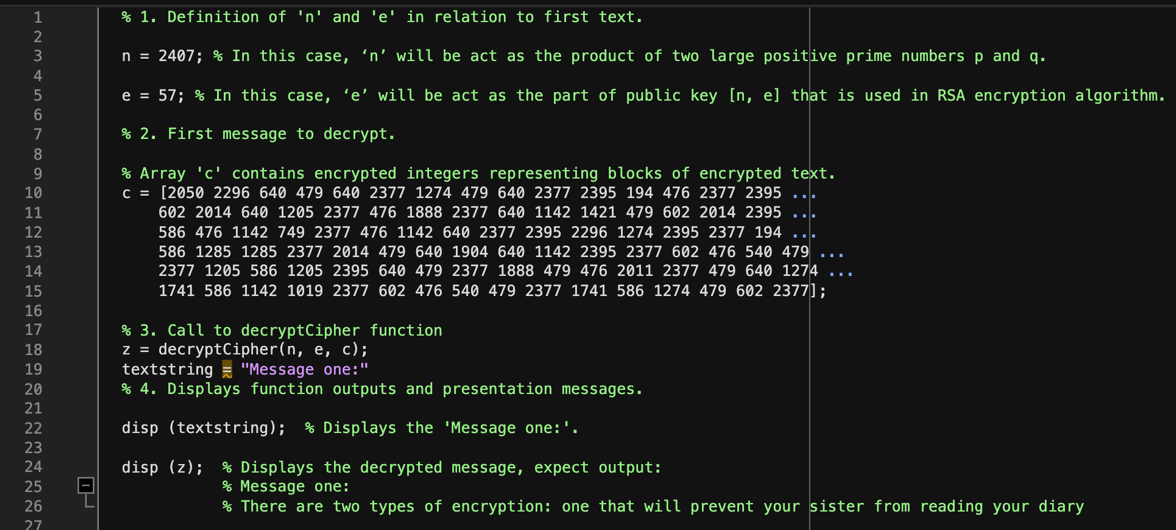
The cipher , that a message will be encrypted to using the public key described on the step 6 could be described in the given formula.

## *Decryption procedure*

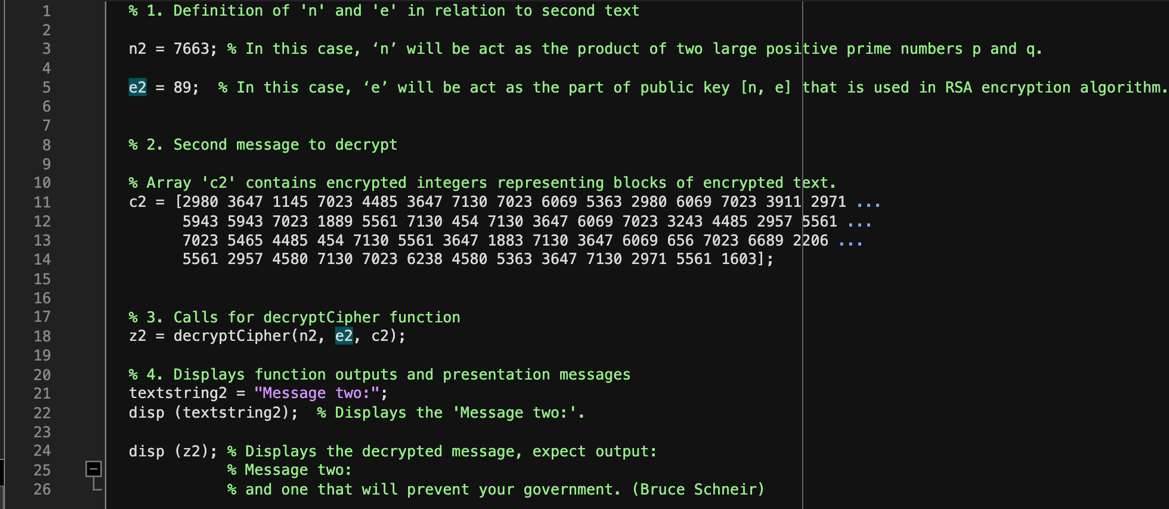
A message , which is decrypted from cipher using the private key illustrated on the step 6 could be described in the given formula.

# **Breaking the RSA Algorithm**

In this section, we will be going to breakdown the RSA Code using MATLAB as the main programming language. We are handed two ciphers, and , along with two pairs of private keys and described in the *Figure 1* and *Figure 2*. Our mission is to decrypt the message behind the two given ciphers. In Figure 1, the first two variables, and , are the private key of the cipher , while the matrix is the cipher . All the given variables will be passed into the *decryptCipher()* function in line 18 to process the decrypting process and retrieve the decrypted message. *Figure 2* illustrates the decryption process for cipher , which has the same initialization procedure as given in *Figure 1*. The major difference here is the private key that is distinct for cipher . It then also passes into the *decryptCipher()* function for further processing.

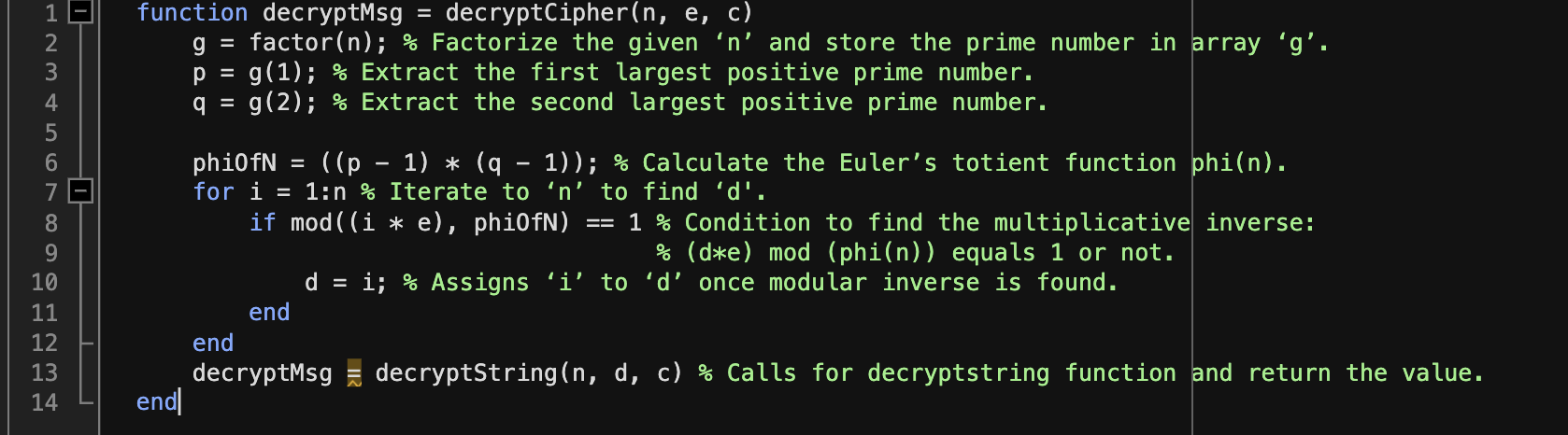
******

***Figure 1: Cipher 1 code in MATLAB***

******

***Figure 2: Cipher 2 code in MATLAB***

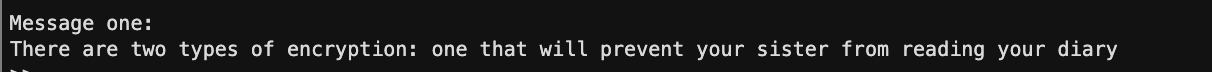
In *Figure 3*, the *decryptCipher()* function is made in order to avoid creating the same decrypting procedure for multiple ciphers and increasing the re-usability. It accepts three parameters **,** where the parameters represents the private key factor and is the cipher that need to be decrypted. In the line number 2, is being factorized, and stored into two variables and in line 3 and 4 respectively, which proofs the fact that is a result from the multiplication of and . Line 6 obtains the by using Euler’s totient function, which is a major step to generate the key as described in the section *Procedure of generating the keys.* The next line initiates a loop that iterates **[1, ],** searching for the multiplicative inverse of , such that **.** Once the value is satisfied with the condition, the value of will be assigned to the current value of , and the loop will be terminated. Finally, the function returns the *decryptMsg* value that contains the decrypted string that employed from the written fucntion *decryptString().*

**

***Figure 3: decryptCipher() function***

# **Results**

This section focuses on inspecting the results of code execution. Upon executing the code for the first cipher, which is illustrated in *Figure 1*, a message containing 93 characters is shown, as illustrated in *Figure 4*. *Figure 5* depicts the workspace of the corresponding executed cipher, where the given cipher is shown as a 93 elements vector. With the public key of 2407, the two obtained factors that stored respectively in and are 29 and 83, which are two different prime factors that satisfies **.** These variables are then used for calculates , which is an essential component for deriving the value of . The value of indicates that the condition is met after looping 1289 times, thus assigning the value of that will be used as the private key for decrypting process.

Figure 4: Output of cipher

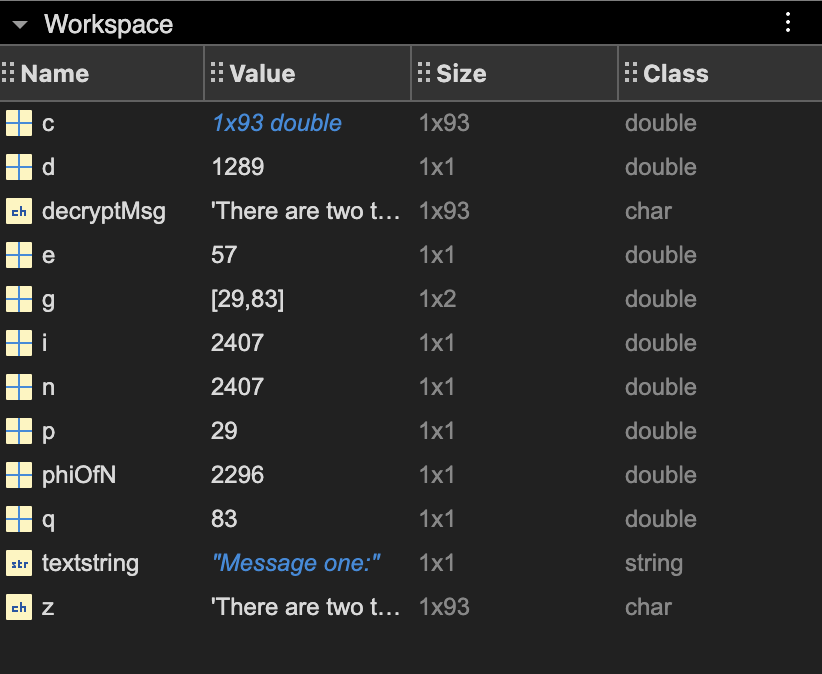


Figure 5: Workspace of cipher

The outcome for cipher is illustrated in *Figure 6*, resulting in a different and shorter value compared to cipher . In *Figure 7*, it is clear that the length of cipher only contains 58 elements. Although using the same function *decryptCipher(),* cipher prompts different values to all variables within the function, as described in *Figure 7*. With the new public key in cipher , the value of private key is also changed, with the variable set at 3113.



Figure 6: Output of cipher

A screenshot of a computer

Description automatically generated

Figure 7: Workspace of cipher

# **Conclusion**

To conclude, this report is written to provide an extensive overview of the procedure that made up the RSA encryption algorithm. With the use of MATLAB, this report has shown the successful of executing the encryption and decryption processes for two different ciphers. The received message content from both ciphers and has illustrated the efficacy and reliability of the RSA algorithm in the context of transmitting data securely. By utilizing the magic of prime factorization and modular arithmetic, the encryption and decryption functions could accurately processed the ciphers, safeguarding the confidentiality and integrity of the information. Through this implementation, it underscores the applicability of RSA encryption in real-world scenarios, marking its status as a strong foundation of nowadays cryptography.

# **References**

[1] J. Schneider, “A brief history of cryptography: Sending secret messages throughout time,” *IBM Blog*, Jan. 05, 2024. <https://www.ibm.com/blog/cryptography-history/>

[2] “Understanding the RSA algorithm,” *ar5iv* (accessed May 20, 2024).

<https://ar5iv.labs.arxiv.org/html/2308.02785>

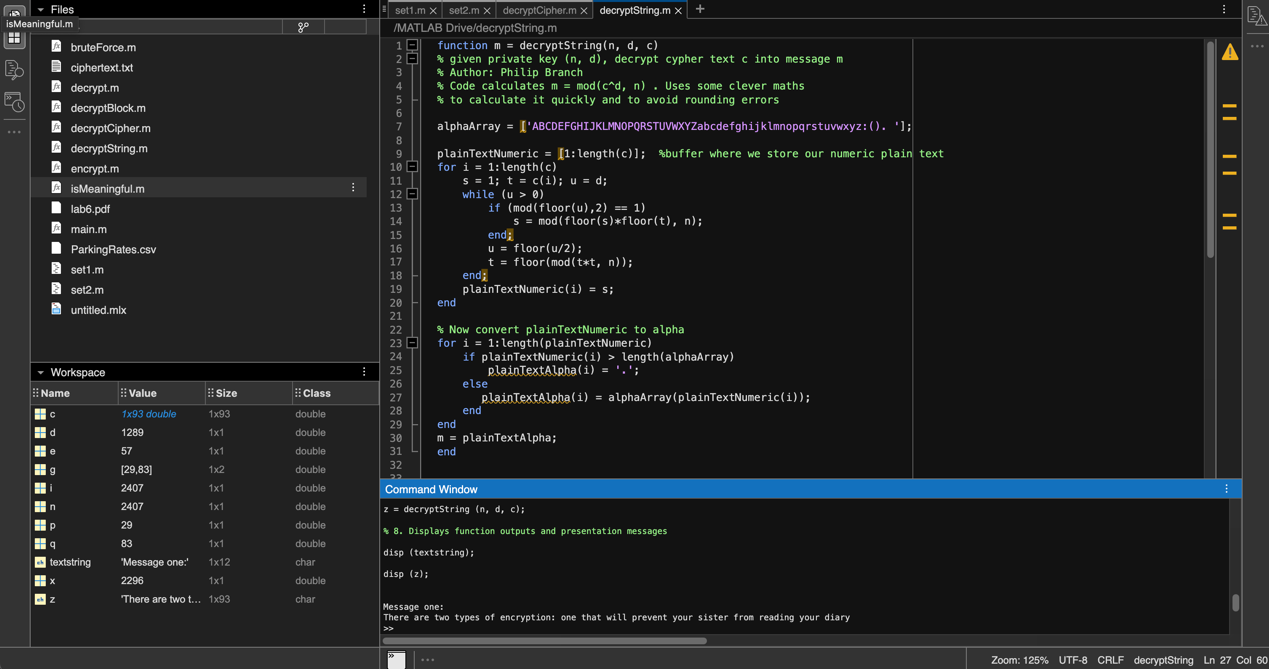
[3] S. Wickramasinghe, “RSA Algorithm in Cryptography: Rivest Shamir Adleman Explained,” *Splunk-Blogs*, May 15, 2023. <https://www.splunk.com/en_us/blog/learn/rsa-algorithm-cryptography.html>

# ‌ **Diagrams**

A computer screen with green and white text

Description automatically generated

***Figure 8: decryptString() function (Written by Philip Branch)***



***Figure 9: Execution of cipher***

‌A screenshot of a computer

Description automatically generated

***Figure 10: Execution of cipher***

‌