Information Encryption and Decryption Analysis, Vulnerabilities and Reliability Implementing the RSA Algorithm in Python

Rocío Rodriguez G.(✉), Gerardo Castang M., and Carlos A. Vanegas

Universidad Distrital Francisco José de Caldas, Facultad Tecnológica, Bogotá, Colombia
{rrodriguezg,gacastangm,cavanegas}@udistrital.edu.co

Abstract. The processing and transmission of information has increased its effectiveness in recent decades. From mathematical models the security and integrity of the data are guaranteed. In spite of that, interceptions in the signal, attacks and information theft can happen in the transmission process. This paper presents a RSA algorithm analysis, using 4, 8 and 10 bits prime numbers with short messages. The encryption and decryption process implemented in python allowed the computational resources use. Processing time and data security are evaluated with a typical computational infrastructure required for its operation; in order to identify vulner-abilities and their reliability level when ideal conditions are available to perform a cryptanalysis.

Keywords: Encryption · Decryption · Security · Cryptography · RSA · Cryptanalysis

1 Introduction

Information theory is based on mathematical and probabilistic concepts that allow to create a communications system. This theory aims to achieve efficient and reliable communication in the transmission of information. To be able to achieve the adequate communication. It is necessary to take into account the variables that influence the process of transmission and communication of information from the source to the destination through the communication medium where the channel is generally affected by noise. In 1928 Hartley formulated the first mathematical laws that regulate a communication system. These ideas were considered by Shannon allowing to develop the fundamental principles of the theory.

The physical model exposed by Shannon is shown in Fig. 1. It indicates how the communication can be expressed in a process of information transfer. In which a transmitter, issues a codified signal (message) that travels along a channel, the signal transits through the channel able or not able to be affected by

H. Florez and M. F. Pollo-Cattaneo (Eds.): ICAI 2021, CCIS 1455, pp. 391–404, 2021.
https://doi.org/10.1007/978-3-030-89654-6_28

noise, it finally arrives to the receiver who codifies the signal, for Shannon the quantity of contained data in a message is tantamount to a math data well defined and measurable, this quantity refers to the possibility that the message within a set will be received and not to the quantity of sent data articles [1].

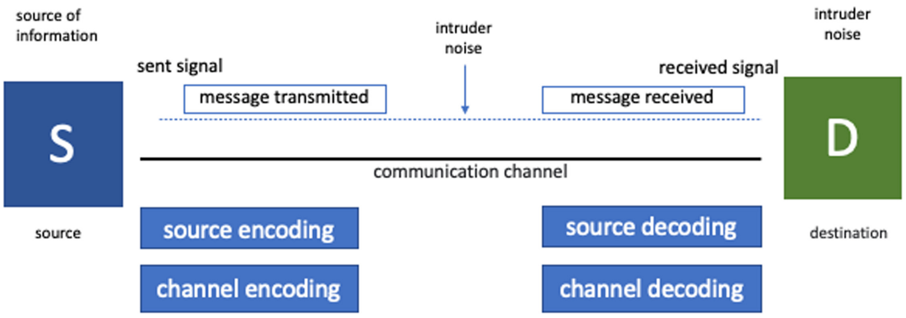


Fig. 1. Transmission and communication process of information.

The codification message (text, image, voice) makes reference to its transformation in symbols. This subsequently changes to electrical signals that provide security to the information that travels from a place to another as we can see at Fig. 1. In this case, the noise represents the effect of an attacker when intercepting and possibly decodifying the coded message (intruder).

The development of secure applications in mobile environments, or limited computational requirements, makes it necessary to create information security solutions in this area, especially for the processing and transmission of short messages.

Generally, the RSA algorithm is used for lengths of keys in high bits. The factoring of prime numbers is a complex process and requires a lot of computational time, this is the principal strength of the algorithm providing security to the information.

Since the RSA works efficiently for this length of keys in high bits, we intend to validate the security of the algorithm, with short key lengths. Once the investigation concluded if needed improvements will be made to the algorithm. All of this to provide protection and security of the information that is transmitted and that can be intercepted by an attacker.

2 Network Security Aspects

The topic of network security and the problems associated with it can be divided in four big interrelated areas. These areas are: confidentiality, authentication, non-repudiation and message integrity check [2].

Confidentiality consists in exchanging, delivering, sending and sharing the information, with the corresponding or appropriate user.

Authentication consists in validating, identifying or authenticating the user with whom you interact before starting the transfer or exchange process of the information.

Non-repudiation handles the verification and validation of the established terms among the users, to perform a process or a transaction.

Message integrity check validates that the received message is the same as the one that was sent, and that the sent message was not altered, replaced or modified by an intruder.

In each one of the TCP/IP model layers the concept of the information security and integrity, can be involved on each layer [3]. These qualities can be implemented logically and physically. Logically through the models and algorithms and physically, through hardware devices that perform processes such as calculations and validations, with the purpose of maintaining the information security and integrity qualities in the network.

The cryptography allows to implement math processes with a certain level of complexity to the information prior to be transmitted, processed or stored. The cryptographic algorithms and methods play a very significant role in the communication systems [4].

Encrypted messages that have been transmitted or stored can be intersected, copied and stolen with the aim of decrypt encrypted messages, it is called cryptanalysis.

The group of methods, algorithms and procedures that perform the cryptography and cryptanalysis processes is known as cryptology. Cryptology is one of the most important branches in security systems of communications networks. It allows to identify flaws in the used methods, algorithms and procedures for the information protection, storing and transmission [5].

Encryption and Decryption Processes of a Message: Cryptography is based on two fundamental processes known as encryption (E) and decryption (D). These are mathematical functions that are involved through the parameter (K), which corresponds to the set of keys used to the implementation of the procedure. The arguments of the functions are the message or plain text (P), and the cipher text (C).

The **methods** of encryption and decryption of a message can be expressed generally, through the following expressions:

$$C = E_K[P] \tag{1}$$

$$P = D_K[C] \tag{2}$$

Replacing the value of the parameter C, of the expression (1) in (2), we get:

$$P = D_K\left\{E_K[P]\right\} \tag{3}$$

Because the functions of encryption and decryption have an inverse relation, we get:

$$P = 1 \cdot P = P \tag{4}$$

This allows to obtain the original message or plain text.

Kerckhoff's Principle: One of the rules of the cryptanalysis is that the encryption and decryption methods are known, this means that the cryptanalyst knows its functioning [6]. The cryptanalyst job is to find the parameter (K) to decrypt the message. The user job is to generate or change the keys continuously each time that the information encryption process is performed. The Kerckhoff''s principle is set out as follows: all algorithms must be public just keys should be secret.

3 The Cryptology

The cryptology has as the main objective to encrypt the information, with purpose of protecting it. For that end, it employs algorithms that tend to use one or more keys, in such a way the communication between sender and receiver is made with security and privacy.

Cryptography is located at the cryptology branch along with cryptanalysis [7]. Cryptanalysis is the study of everything involved with deciphering data technics. Some of the features of cryptanalysis applied to a cryptographic system are:

- To ensure its robustness and resistance.
- To discover weaknesses to avoid possible attacks.
- To strength it in order to increase its security.

The cryptographic methods used by the cryptosystems to perform the encryption are classified in asymmetric and symmetric. The classifications are described in Table 1 [8].

Table 1. Symmetric vs asymmetric cryptosystems.

	Public key	Private key
key management	It is only necessary to memorize the sender's private key and the receiver's public key	you have to memorize a high keys number
length and key space	key is in the order of thousands of bits	the order key is of hundreds of bits
key's life	key duration is usually long	the key duration is short, it usually ends when the session is over.
Authentication	By having a public and a private key, the sender and the mes-sage can be authenticated	It is possible authenticate the message only
Speed cipher	Speed encryption is slow	Speed encryption is high
Use	They are used for key exchanges and digital signatures	They are algorithms used for encryption

4 Cryptosystem Analysis RSA

Rivest, Shamir and Adleman developed an asymmetric cryptosystem [9][10], from the product of two prime numbers (p and q) previously selected. These numbers multiplied allow to get an n number.

$$n = p \cdot q \tag{5}$$

Using the *Euler function(*φ*)* the size of the integers multiplicative group is generated.

$$\varphi\left(n\right) = \left(p-1\right)\left(q-1\right) \tag{6}$$

Subsequently it is proceeded to calculate two parameters. One is used as public key(e) and the other as private (d). They are used for encryption and decryption respectively [11]. To make this calculation, one of the parameters is chosen and the other one is calculated.

$$e \cdot d = 1 + k \cdot \varphi\left(n\right) \tag{7}$$

For an integer k is relevant that k is less than $\varphi(n)$ and coprime. The following equation must be verified:

$$e \cdot d = 1\left(mod\varphi\left(n\right)\right) \tag{8}$$

Therefore

$$d = e - 1\left(mod\varphi\left(n\right)\right) \tag{9}$$

$$e = d - 1\left(mod\varphi\left(n\right)\right) \tag{10}$$

From this algorithm the public key is obtained:

$$K_{pb} = (e, n) \tag{11}$$

From this algorithm the private key is obtained:

$$K_{pr} = (d, n) \tag{12}$$

The resulting key is obtained from the two components:

$$< K_{pb}, K_{pr} > \tag{13}$$

The RSA algorithm works with the premise of calculating n by multiplying p and q. It is difficult to factorize the product since it prevents to obtain the private key K_{pr} from public key K_{pb}.

The Python code of the implemented RSA algorithm is described as follows:

```
Begin
import time
import random
def getD(e, z):
    for x in range(1, z):
        if (e * x) % z == 1:
            return x
def values E(a):
    list_obj = []
    for x in range(2, a):
        if mcd(a, x) == 1 and encrypt_function(x,z) !=
None:
            list_obj.append(x)
    for x in list_obj:
        if x == encrypt_function(x,z):
            list_obj.remove(x)
    return list_obj
def mcd(a, b):
    while b != 0:
        c = a % b
        a = b
        b = c
    return a
def encrypt_function(e, z):
   return e%z
def encrypt_block(x):
    encryption = encrypt_function (x**e, n)
    return encryption
def decrypt_block(encryption):
    decrypted_message =encrypt_function(encryption**d, n)
    return decrypted_message
def encrypt_message(message):
    return ''.join([chr(encrypt_block(ord(x))) for x in
list(message)])
def decrypt_message(message):
    return ''.join([chr(decrypt_block(ord(x))) for x in
list(message)])
p=int(input('Enter prime number P: '))
q=int(input('Enter prime number Q: '))
print("\nSelected prime numbers: p=" + str(p) + ", q=" +
str(q) + "\n")
n=p*q
print("Value of (n = p * q) = " + str(n) + "\n")
z=(p-1)*(q-1)
print("Euler's function [Z(n)]: " + str(z) + "\n")
print("Possible values for number 'e':\n")
print(str(valuesE(z)) + "\n")
e = random.choice(valuesE(z))
print ("Random selected value:",e)
d=getD(e,z)
```

```
 print("\nPublic key e,n (e=" + str(e) + ", n=" + str(n)
+ ").")
 print("Private key  d,n (d=" + str(d) + ", n=" + str(n)
+ ").")
 message = input("Plain text message: ")
 begin=time.time()
 encryption = encrypt_message (message)
 print("\nEncrypt message: " + encryption + "\n")
 decrypted = decrypt_message (encryption)
 print("Decrypt message: " + decrypted + "\n")
 final=time.time()
 print("Processing time in seconds:",round(final - begin,
10))
 end
```

4.1 RSA Algorithm Source Code's Analysis

At first, the random and time libraries are imported. Time is a Python library that provides a set of functions to work with dates and/or time, and Random contains different functions related with random values.

Subsequently, the following functions are implemented:

– ***getD(e,z)*:** This function receives two parameters (e, z). A *for* cycle is executed from an x value equals *1* to the z parameter value. The if structure verifies if the $e * x$ modulus (%) product of z is equal to *1*. If the condition is met, the x value is re-turned.
– ***valuesE(a)*:** This function receives a parameter (a). First, an empty list (*list_obj*) is declared. The *for* cycle iterates from x equal two, to *a value*. The *if* structure verifies that the *mcd(a, x)* function is equal to *1* and the *encrypt_function(x, z)* function is different to *empty*. If the condition is met, it is added to the list (*list_obj*) the value of the x variable. Furthermore, in a second *for* cycle, the function iterates from one of *x's* values equal to *0* to the quantity of the list (*list_obj*) elements. With the *if* structure it is verified that the value of the x variable is equal to the value that the *encrypt_function(x, z)* function returns. If the condition is met, the x variable value is deleted from *list_obj*. Finally, the list *(list_obj)* elements are sent back.
– ***mcd(a,b)*:** The function receives two parameters (a, b). The *While* cycle iterates while the value b variable is different from *zero*, the modulus of a and b are assigned to a variable called c, the b value is assigned to the a variable and the c value to the b variable. When the cycle is done, the value of a variable is returned.
– ***encrypt_function(e, z)*:** The function receives two parameters (e, z). It returns the modulus result among the *e, z* variables.
– ***encrypt_block(x)*:** This function receives one parameter (x), the *encryption* variable is assigned the value that returns the *encrypt_function(x**e, n)* function. This function returns the value of the *encryption* variable.

- ***decrypt_block(encryption)*:** This function receives one parameter (*encryption*), the *decrypted_message* variable is assigned the value that returns the *en-crypt_function(encryption**d, n)* function. This function returns the value of the *decrypted_message* variable.
- ***encrypt_message(message)*:** The function receives one parameter (*message*). The *encrypt_block()* function returns the ascii value from x variable that belongs to the list (*list_obj*). The function receives one parameter when performing the loop.
- ***decrypt_message(message)*:** The function receives one parameter (*message*). The *decrypt_block()* function returns the ascii value from x variable that belongs to the list (*list_obj*). The function receives one parameter when performing the loop.

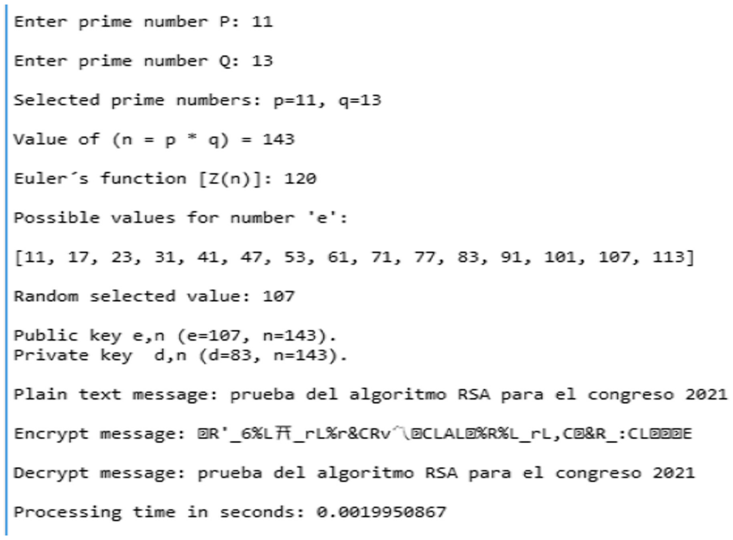
In addition, the program carries out the process to obtain the input and output values of the information. First, the p and q variables are declared, and a prime number is assigned to them. The prime number is captured from the keyboard with the *input()* function and is converted to an integer with the *int function*, in order to be printed with *print()* function. Then, a variable called n is declared, and the product of $p * q$ is assigned to it, that also is printed. The z variable is declared as well, and the product of $(p - 1) * (q - 1)$ is assigned to it and it is printed.

Eventually, an obtained random value with the random library's choice method is assigned and printed to a variable called e. Such value is returned by the *valuesE(z)* function.

Subsequently, a value returned by the *getD(e, z)* function is assigned to a variable called *d*, and the variable *e* value is printed as public key and the variable *d* value as private key. In a variable called *message*, the text to be encrypted is saved, and the initial processing *time* (time.time()) is assigned to a variable called *begin*. Other variables are also declared: *encrypted*, to this variable is assigned the value that returns the *encrypt_message(message)* function. The *decrypted* variable is assigned the value that returns the *decrypt_message (message)*. These variables are also printed.

The final processing time is assigned to a variable called *final*, and the result of the *final-begin* subtraction operation is printed. The *roud()* method is used to show the result with ten decimals maximum.

The algorithm execution can be seen in Fig. 2.



```
Enter prime number P: 11

Enter prime number Q: 13

Selected prime numbers: p=11, q=13

Value of (n = p * q) = 143

Euler´s function [Z(n)]: 120

Possible values for number 'e':

[11, 17, 23, 31, 41, 47, 53, 61, 71, 77, 83, 91, 101, 107, 113]

Random selected value: 107

Public key e,n (e=107, n=143).
Private key  d,n (d=83, n=143).

Plain text message: prueba del algoritmo RSA para el congreso 2021

Encrypt message: ⊡R'_6%L开_rL%r&CRv^\⊡CLAL⊡%R%L_rL,C⊡&R_:CL⊡⊡⊡E

Decrypt message: prueba del algoritmo RSA para el congreso 2021

Processing time in seconds: 0.0019950867
```

Fig. 2. Transmission and communication process of information.

In order to perform tests on the implemented algorithm we used: Intel (R) Core (TM) i5-10210U Processor CPU @ 1.60 GHz 2.11 GHz, 8.00 GB RAM and Windows 10 $\times$ 64 b Operating System.

4.2 Length of Encrypted Message Test

A series of tests carried out with values for p and q, using prime numbers of 4, 8 and 10 bits. Using short message lengths (Typically 2 KB). This allowed establishing the values for e and the processing time shown on Table 2.

Table 2. Values and data of tests results on the RSA algorithm

Number of bits to generate the prime numbers	Values for P	Values for Q	Values of E	Processing time (In seconds)
4	11	13	23	0.003
8	191	227	42529	20.467
10	937	1019	952787	1697.762

The tests results allowed establish the following analysis:

– To the extent that small keys in number of bits are used, the processing time is very small.

- To go from four to eight bits, the time spent in processing is multiplied by an approximated factor of 1000.
- To go from eight to ten bits, the time spent in processing is multiplied by an approximated factor of 10.
- To the extent that increases the number of bits used to generate the prime numbers p and q, the processing time to code the message increases exponentially and presents a relation by ten times as shown above.
- We attempted to perform tests using prime numbers of 12 bits, where the e generation was about 3 h and 30 min and the message could not be encrypted because a memory overflow was presented.

4.3 Cryptanalysis Test

This section proposes to carry out the cryptanalysis to validate the ability to decrypt the information, in order to find the user's private key with an ideal environment for the attacker [13]. That is, knowing part of the transmitted message, the length in bits of the prime numbers used to encrypt and decrypting the message and the user's public key [14,15].

A decryption algorithm developed by the authors was designed. First described with a natural language and then implemented with python's programming language [16]. In purpose to verify the processing time and the reliability of the decrypted in-formation.

The aim goes to validate the strength of the encryption algorithm implemented and the speed or time required to decrypt an encrypted message, using typical computational capabilities of a user.

We performed a test with the same available hardware infrastructure, taking into account the following ideal conditions for the attacker:

- Intercepts and knows the encrypted message.
- Knows part of the original message.
- Knows the bit number of P and Q.
- Knows the public key of the sender (e, n).
- Lacks to determinate the private key (d, n) for deciphering the message.

The following is the algorithm in natural language proposed to obtain the private key and decipher the message by an attacker.

1. Knows the public key (e, n).
2. Knows the number of bits to generate the prime numbers.
3. Generates the set of primes for that given number of bits.
4. Performs operations in order to find both prime numbers that multiplied give the value of n, to get p and q values subsequently.
5. Finds $(p-1)$ and $(q-1)$ and multiply them to get the Euler function *(called Z)*
6. Deduces a d number, that is within range from *1* to z, when the remainder of the operation is equal to *1*, d will be the result.

7. With the d number generates the private key (d, n), thus proceeds to decipher the message.

The previously algorithm was implemented in python:

```
 Begin
 import time
 encrypted_message="#"
 bit=4
 public_key=[23,143]
 z=0
 d=0
 def decrypt_message(encrypted_message):
   return ''.join([chr(decrypt_block(ord(x))) for x in
list(encrypted_message)])
 def encrypt_function(e, n):
   return e%n
 def decrypt_block(x):
     decrypted_message = encrypt_function(x**d,
public_key[1])
     return decrypted_message
 def prime(i):
   x=2
   flag=0
   while (x<i):
     if (i%x==0):
       flag=1
       return flag
     else:
         x=x+1
   return flag
 nro_primes=[]
 for i in range(2,15,1):
     if(prime(i)==0):
      nro_primes.append(i)
 for j in range(len(nro_primoes)):
   for y in range(len(nro_primes)):
     values=nro_primes[j]*nro_primes[y]
     if(values==public_key[1]):
       z=(nro_primes[j]-1) *(nro_primes[y]-1)
       print ("z",z)
 for account in range(z):
    value_d=(account*public_key[0])%z
    if(value_d==1):
      d=account
      print ("private key:",d)
 begin=time.time()
 decrypted = decrypt_message(encrypted_message)
 print ("The original message is:",decrypted)
 final=time.time()
```

```
  print("processing time in seconds:",round(final - begin,
 10))

  end
```

With the implementation of cryptanalysis, the values for *p, q, e* and the *time of processing* were established as shown on Table 3.

The tests results allowed to establish the following analysis:

– For 4 bits the decryption time is: 0.0019946098 milliseconds. To generate the e parameter, the time processing was less than a millisecond.
– For 8 bits the decryption time is: 6.6627941132 s. To generate the *e* parameter, the approximate time processing was 0.4679889679 millisecond.
– For 10 bits the decryption time is greater than 11 min. To generate the *e* parameter the approximate time processing was 173.8833482265 min.
– For 12 bits with values of $P = 2059, Q = 4067, E = 8366779$ spent about 3 h to get the possible values of *e*.
– To go from 4 to 8 bits the time spent in processing is multiplied by an approximated factor of 1000.
– To go from 8 to 10 bits the time spent in processing is multiplied by an approximated factor of 100.
– To go from 10 to 12 bits the estimated time for processing is multiplied by an approximated factor of 10.

Table 3. Values and data of tests results on the RSA algorithm

No of bits to generate the prime numbers	Values for P	Values for Q	Values of E	Time to generate E	Processing time (in seconds) to decipher the message
4	11	13	17	millisecond	0.0019946098
8	191	227	47737	millisecond	6.6627941132
10	937	1019	205157	10 min	689.0595903397

5 Conclusions

When the number of bits used to generate the primes *p* and *q* increases, the time processing to decipher the message gets exponentially higher whit a deduced relation of the ten times factor.

When working with larger key sizes, the processing time increases causing the computational requirements to be stronger.

For internal messaging applications with functional levels of security the RSA algorithm for low-bit lengths is useful.

After reviewing the specialized literature, we can conclude that one way to improve the security of the RSA algorithm against possible cyberattacks is obtaining a suitable bit length for the encryption and decryption keys. Unfortunately, there are not procedures that can accurately determine the length of the keys.

When the attacker has the perfect conditions to perform the cryptanalysis, there is a greater risk in the interception of the transmitted information. This is due to a short-er processing time to discover the message, according to the computational capabilities used by the attacker.

6 Future Work

We propose to modify the operations performed in the RSA algorithm to use it with short key lengths that can be mathematically proven by including more elements that increase the security level.

We propose to experiment with key lengths for public and private keys with twelve, fourteen, sixteen, twenty and thirty-two bits. To validate the security and strength of the encryption and decryption process information. The challenge is to use any hardware available for the majority of the community, without the processing time being quite high and expensive, slowing down the processing of the keys.

One challenge is to increase the security levels in the message processing when the bit sizes of the primes p and q are small to generate the public and private keys.

References

1. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**(3), 379–423 (1948). https://doi.org/10.1002/j.1538-7305.1948.tb01338.x
2. Redes de Computadoras, Andrew S. Tanenbaum; cuarta edición, editorial: Pearson-Prentice Hall
3. Mateti, P.: Chapter 1 Security Issues in the TCP/IP Suite (2007). https://doi.org/10.1142/9789812770103_0001
4. Satish, G., Raghavendran, dr. Ch., Varma, dr.: Secret key cryptographic algorithm. Researchgate.net (2012). https://www.researchgate.net/publication/266389826_secret_key_cryptographic_algorithm
5. Awad Al-Hazaimeh, O.: A new approach for complex encrypting and decrypting data. Int. J. Comput. Networks Commun. **5**(2), 95–103 (2013)
6. Petitcolas, F.: Kerckhoffs Principle (2011). https://doi.org/10.1007/978-1-4419-5906-5
7. Shamir, A., Rivest, R.L., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems Mag. Commun. ACM **21**, 120–126 (1978). https://doi.org/10.1145/359340.359342
8. Gabriel, E.M.: Clúster de alto rendimiento en un cloud: ejemplo de aplica-ción en criptoanálisis de funciones hash. Universidad de Almería, p. 60 (2011). http://repositorio.ual.es/bitstream/handle/10835/1202/PFC.pdf?sequence=1
9. Asjad, S.: The RSA Algorithm. Researchgate.net, pp. 5–15 (2019). https://www.researchgate.net/publication/338623532_The_RSA_Algorithm

10. Fonseca-Herrera, O.A., Rojas, A.E., Florez, H.: A model of an information security management system based on NTC-ISO/IEC 27001 standard. IAENG Int. J. Comput. Sci. **48**(2) (2021)
11. Goldreich, O.: Modern Cryptography, Probabilistic Proofs and Pseudorandomness (Second Edition - author's copy), pp. 1–2. Springer (2000). http://www.wisdom.weizmann.ac.il/~oded/PDF/mcppp-v2.pdf
12. Castro Lechtaler, A., Cipriano, M., García, E., Liporace, J., Maiorano, A., Malvacio, E. and Tapia, N.: Estudio de técnicas de criptoanálisis.XXI Workshop de Investigadores en Ciencias de la Computación.Sedici.unlp.edu.ar (2021). http://sedici.unlp.edu.ar/handle/10915/77269
13. Al-hazaimeh, O.: A new approach for complex encrypting and decrypting data. Int. J. Comput. Networks Commun. **5**, 95–103 (2013). https://doi.org/10.5121/ijcnc.2013.5208
14. Tiwari, G., Nandi, D., Mishra, M.: Cryptography and cryptanalysis: a review. Int. J. Eng. Res. Technol. **2**, 1898–1902 (2013)
15. Bokhari, M., Alam, S., Masoodi, F.: Cryptanalysis tools and techniques (2014)
16. Rodríguez, G.R., Vanegas, C., Castang, G.: Python a su alcance. EditorialUD, 2020. Páginas, pp. 100–120 (2020). ISBN 978-958-787-181-4