RSA Encryption Algorithm Optimization to Improve Performance and Security Level of Network Messages

Fausto Meneses, Walter Fuertes, José Sancho, Santiago Salvador, Daniela Flores, Hernán Aules, Fidel Castro

Universidad de las Fuerzas Armadas
Sangolquí, Ecuador

Jenny Torres

Escuela Politécnica Nacional
Quito, Ecuador

Alba Miranda, Danilo Nuela

Universidad Técnica de Ambato
Ambato, Ecuador

Summary
Asymmetric cryptographic algorithms are a robust technology used to reduce security threats in the transmission of messages on the network. Nowadays, one of the disadvantages are the mathematical solutions because they require a greater amount of calculation that leads to the need for increased use of computational resources. This paper aims to optimize the RSA encryption algorithm and thus improve the security, integrity and availability of information. The results show the efficiency and functionality of the RSA algorithm in terms of information security. Also, we can see that time, memory, processor and network performance when performing encryption and decryption are lower than other RSA solutions, because calculations are performed on the client and server.

Key words:
RSA; RPC; performance; security; asymmetric encryption.

1. Introduction

Encryption and decryption of information has proven to be the best way to get confidentiality and integrity of data. Nevertheless, there is a big challenge since threats and vulnerabilities are increasing with the development of technologies [1]. In order to face this problem, the scientific community has emphasized their skills in finding an alternative to improve information security by ensuring the information availability. Nowadays, different algorithms have been promoted to provide security but at the same time, generates a higher cost and consumption of computational resources. One of these mechanisms is the RSA asymmetric encryption. RSA is the most widely used worldwide algorithm, which provides security through encryption of data that transit in the Web and ensures information confidentiality and authenticity [2]. This algorithm also known as public key algorithm, became very popular due to its simplicity in calculation. However, the security of the RSA algorithm depends on the size of the prime numbers used in factorization. It is affected by the increase of computational cost [3][4] related with prime factorization, which implies bigger key length to ensure security. The development of this work includes the study of modeling techniques in order to determine which one is feasible to represent the information model. These techniques include standard modeling languages such as Unified Modeling Language (UML); frameworks such as the Model-Driven Architecture (MDA); and network management, for example, the RSA model. Subsequently, the existing approaches for modeling encryption systems are studied. With these results, we designed a model for encryption and decryption of information based on RSA. Finally, for validation, in the Application Programming Interface (API) implementation we made a library that allows us to encrypt the message in the client side and send it together with public keys through the network. The data is retrieved on the server side and through the access to the database, private keys are recovered (decryption process). To optimize the security of the model, private keys are periodically updated through a mixing process.

In order to measure the level of efficiency of the proposed model, another model was designed and implemented, called in this study Baseline RSA Model, which works with the factorization of 300 digit prime numbers. The main contribution of this research is the development of a mathematical and software optimized model that provides the following improvements: (1) the application of a mathematical model that combines modular and probabilistic calculation; (2) a matrix capable of generating encrypted messages with the same information value, but with different meanings; (3) a mixing process for updating private key; (4) the management of messages through a RPC; (5) the conversion of a deterministic basis project to a probabilistic project with the generation of random values; (6) the work with less complex structures reducing the consumption of time and resources; and (7) increases security by hiding private keys in the executable file.

The remainder of this paper has been structured in the following way. Section II describes the works found in literature related with the research. Section III describes the definition and the statement of the problem. Section IV compares the baseline RSA model with the Optimized RSA model. Section V presents the analysis of the results

Manuscript received August 5, 2016
Manuscript revised August 20, 2016

and the discussion. In Section VI, an analysis of security between baseline RSA and optimized RSA models is done. Lastly, Section VII finalizes the study with conclusions and future work lines.

2. Related Work

In the literature, there are different works that guarantee information security and increase performance efficiency, reducing the consumption of resources (memory, CPU time, encryption, and decryption time). For instance, the work proposed by Gupta and Sharma in [5], formulated an hybrid encryption algorithm based on the RSA algorithm and Diffie-Hellman key exchange algorithm, for increasing security regardless of the computing performance. Nagar in [6], presents a new method to exchange indexes that contain the values of public and private keys stored in a database. In a comparative analysis, Surbhi in [7] describes security threats in the transmission of e-mail over the Internet. This analysis includes a comparative study of different encryption algorithms and concludes choosing the best technique that deals with the problem of computational cost and security. Mahajan in [8], sets out a new solution using CUDA frameworks, which proposes a new algorithm that calculates the value of the module, processing small and large prime numbers.
Shahzadi et al. [2] presents the evaluation of asymmetric encryption algorithms: RSA, ElGamal & Pallier, which compare these algorithms in terms of encryption and decryption time, memory use and performance. A comparative assessment in [9] and [10] is performed for different commonly used symmetric-key algorithms such as DES, AES and RSA considering several parameters such as, computation time and memory use. In [8], it is proposed a method for encrypting data using images, generating a different encrypted file each time it is used to encrypt the same message. In [11], the encryption is optimized through the Miller-Rabin algorithm for determining whether a given number is prime, reducing key generation time in any algorithm. Sinjan in [12] describes an implementation of RSA encryption algorithm in C. It consists of generating two random prime numbers and a prime number (n) also called Euler function. These three numbers are used to generate a public and private key. In some cases, this calculation takes a long time. In [2] a third prime number is used, in order to make a module n difficult to decompose. In [13] the extended Euclidean theory is applied, in order to obtain the keys to solve the transmission problem. Finally, in [14] and [15] the distribution of "n" is eliminated since the finding of its factors compromise the security of the algorithm. Although previous work are concerned with the problem of RSA performance, none of them emphasizes on how to improve the security level. Not even a generic solution was achieved since these solutions focus on high consumption of resources and software costs, without setting a software engineering process. Comparing these studies with our work, we have achieved an optimized RSA model that combines modular and probabilistic computation for encryption and decryption.

3. Problem Statement

During the study of the RSA algorithm, we have identified the following problems: a) the mathematical solutions of cryptographic algorithms require a large amount of calculation, which implies a higher consumption of computing resources, thus requiring greater bandwidth; b) in order to store information in a database, for example four bytes, encrypted fields of approximately 600 bytes are required; c) increasing threats and vulnerabilities, due to the development of technologies [16,17], result in the improvement of information security, which means higher cost and consumption of computational resources. This work defines a generic model that optimizes the RSA method for information encryption, combining modular and probabilistic calculation. This generic RSA model meets all the requirements and processes based on standard models accepted like cryptographic protocols.

4. Comparison between the Optimized and the Baseline Model

This section shows the comparison between the baseline algorithm and the optimized RSA model.

4.1 Baseline RSA Model

The baseline RSA model used in this work is based on the model proposed by R. Johnsonbaugh [18].

Public and private key generation:

i. Two prime numbers, p and q, are chosen. Each one must have at least 300 digits.
ii. Calculate $z=p*q$, where z is the module, which is public. For both, public and private keys. The result of this multiplication is considered the key length. The security of this model depends on this key, due to the impossibility of finding p and q.
iii. Calculate $\phi=(p-1)*(q-1)$, where ϕ is the Euler function.
iv. Choose an integer n such that $mcd(n,\phi)=1$, where n is a prime number and public key.
v. Finally, calculate s, where $0<s<\phi$ and $n*s \bmod \phi=1$, used in the decryption process.

Encryption process:

i. A sends a message M to B.
ii. For each character of the message M, its position will search into a string called Alf (ASCII alphabet), the same that will be represented with a three digit number.
iii. These numbers are concatenated to form an integer a, where $0<a<z-1$.
iv. The message is encrypted using the public key (z;n) and the integer a. So, $c = a^n \bmod z$.
v. The encrypted message c,z and n, are sent to the receiver.

Decryption process:

i. The encrypted message c, z and n are received by B.
ii. The message c is decrypted using the private key (ϕ;s) using the formula $d = c^s \bmod z$, such that $n*s \bmod \phi = 1$
iii. Resulting in d=a. Which transforms in the original message M.

Example: Using the algorithm described above, we will perform the encryption and decryption process for the message *Hello*. The following values were considered:

p:203956878356401977405765866929034577280193993314348263094772646453283062722701277632936616063144088173312372882677123879538709400158306567338328279154499698366071906766440037074217117805690872792848149112022286332144876183376326512083574821647933992961249917319836219304274280243803104015000563790123

q:531872289054204184185084734375133399408303613982130856645299464930952178606045848877129147820387996428175564228204785846141207532462936339834139412401975338705794646595487324365194792822189473092273993580587964571659678084484152603881094176995594813302284232006001752128168901293560051833646881436219

z=p*q:108479011759769393727176308457457051185868202028229648190667018102389154815514421272511788818630907941383004277918921724359698326811100798212287777156373854025160462416970207713010402684330377900249272920821196418994434689542195444008546643909734553918562196237000260811736016995830450094506601223180883688118367193617732939585810720318219660211710733808215174048484794731011578459721161876052257146687991978315725043744813520691467819521825409725421924282740640715767718923851036438509804069110541415248000720606907056103122751808376882405235966768597369796819161915929329975604623470108492326664937

ϕ=(p-1)*(q-1):

108479011759769393727176308457457051185868202028229648190667018102389154815514421272511788818630907941383004277918921724359698326811100798212287777156373854025160462416970207713010402684330377900249272920821196418994434689542195444008546643909734553918562196237000260811736016995830450094506601223180147858950956587456142088984506552341531162604414254688475101937100559489682831333211096112168725062086504041204843134019133603758846576614652942033865449245668774162405791562411624527881923723225419272555390469703102501835262272692412213406592437962333835647493323944496886794067260314259844881438596

n: 29

Alf:!#$%&'()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`abcdefgh
ijklmnopqrstuvwxyz{|}~□Ã,â,¬Ã,Â□Ã,â€šÃ,Æ’Ã,â€žÃ,
â€¦Ã,â€ Ã,â€¡Ã,Ë†Ã,â€°Ã,Å Ã,â€¹Ã,Å’Ã,Â□Ã,Å½Ã,Â
Ã,Â□Ã,â€˜Ã,â€™Ã,â€œÃ,â€□Ã,â€¢Ã,â€“Ã,â€”Ã,Ëœ
Ã,â„¢Ã,Å¡Ã,â€ºÃ,Å“Ã,Â□Ã,Å¾Ã,Å¸Ã,Â¡Ã,Â¢Ã,Â£Ã
,Â¤Ã,Â¥Ã,Â¦Ã,Â§Ã,Â¨Ã,Â©Ã,ÂªÃ,Â«Ã,Â¬Ã,Â-Ã,Â
®Ã,Â¯Ã,Â°Ã,Â±Ã,Â²Ã,Â³Ã,Â´Ã,ÂµÃ,Â¶Ã,Â·Ã,Â¸Ã,
Â¹Ã,ÂºÃ,Â»Ã,Â¼Ã,Â½Ã,Â¾Ã,Â¿Ãƒâ,¬ÃƒÂ□Ãƒâ€š
ÃƒÆ’Ãƒâ€žÃƒâ€¦Ãƒâ€ Ãƒâ€¡ÃƒË†Ãƒâ€°ÃƒÅ Ãƒâ€¹Ãƒ
Å’ÃƒÂ□ÃƒÅ½ÃƒÂ□ÃƒÂ□Ãƒâ€˜Ãƒâ€™Ãƒâ€œÃƒâ€Ã
ƒâ€¢Ãƒâ€“Ãƒâ€”ÃƒËœÃƒâ„¢ÃƒÅ¡Ãƒâ€ºÃƒÅ“ÃƒÂ□Ãƒ
Å¾ÃƒÅ¸ÃƒÂ ÃƒÂ¡ÃƒÂ¢ÃƒÂ£ÃƒÂ¤ÃƒÂ¥ÃƒÂ¦ÃƒÂ§Ã
ƒÂ¨ÃƒÂ©ÃƒÂªÃƒÂ«ÃƒÂ¬ÃƒÂ-ÃƒÂ®ÃƒÂ¯ÃƒÂ°ÃƒÂ
±ÃƒÂ²ÃƒÂ³ÃƒÂ´ÃƒÂµÃƒÂ¶ÃƒÂ·ÃƒÂ¸ÃƒÂ¹ÃƒÂºÃƒÂ
»ÃƒÂ¼ÃƒÂ½ÃƒÂ¾

- Encryption:

For the message *Hello*, the sender calculates a=039068075075078

Then, $\mathbf{c = a^n \bmod z}$, being the result:

14549175903481935273327257627482397794834294736560186846820400889979925057643654082299998178301480807440625157220159030655761391001487231230955501983433332406369957571139692142691981643327833340355723383391444208965197673389827878027778474195655528915634183270055282658413366739859957185236007882922095775984701737268075528588901761776022861791997688249245723377690777452080502578270804595703808

This value is sent to the receiver, together with the values n, p and q.

- Decryption:

Based on the received valued, the receiver calculates s:

5610983366884623813474636644351226785475941484218774906413811281158059731836952824440264938894

7021348991209109268407788461912927660914205971872988184331303806117480560501831575695035871205367879439279096976480906376431735970101091728558608918828217544083894605344962488828974308188163841986173046472490271871184441787659701198882699486998877209179786907831949277810634218801464482695394540776926756251640021312849896906448415737334436179992901051999370299483848704692650808143943721318236408564872037528650242949880604397549451392627006934444364463276121886634477902325975927965824300479230111088929

Applying $\mathbf{d = c^s \bmod z}$: 039068075075078, which means that d = a, producing the decrypted message Hello.

4.2 Optimized RSA model

Consists on introducing the following variants into the baseline model:

a. Each character of the message has its own RSA value.

b. The value n is randomly generated.

c. To mix the characters in the message, a matrix (Cod) is used and the indexes of the rows in the matrix are randomly generated. Example: Assuming that we want to encrypt the message Hello; Alf and Cod are given by the table below, also indexes of the rows are generated in the order 3 1 and 2; then the encrypted message is generated as ooole.

viii.

		1	2	3	4	5
Alf:		H	e	h	l	o
Cod:	1	l	o	H	h	e
	2	e	h	l	o	H
	3	o	H	h	l	e

Fig. 1. Encryption table

The above example is fairly simple. In practice, the array has m rows by 221 columns, where m is an integer between 1 and 221 (i.e. factorial). All rows represent chains mixed randomly. Therefore, the process of generating the Encryption table has another level of complexity. In consequence, it will not be analyzed in this study due to space limitations.

Public and private key generation:

i. Two prime numbers are chosen, p and q, such that its product does not exceed the number of printable ASCII characters.

ii. Calculate z=p*q, where z is the module, (private). The result of this multiplication is considered the key length.

iii. Generate the mix of the message using the matrix Cod. The security of this system depends on this key and Alf, because they are updated periodically and the indexes of the rows are generated randomly.

iv. Calculate ϕ=(p-1)*(q-1), where ϕ is the Euler function.

v. Choose an integer n such that mcd (n,ϕ) = 1, where n is a prime number and the public key.

vi. Finally, the number s is calculated, where 0 <s <ϕ and n*s mod ϕ = 1, used in the decryption process.

vii. Repeat from step iii while there are characters available.

Encryption process:

i. Capture the message (msj).

ii. Generate a random prime number between 4 and 9 digits (n).

iii. Generate randomly the number of rows of the matrix Cod (nf between 0 and k-1).
Generate randomly the array of indexes of the code (alt) of nf elements.
Retrieve from the database p, q, k, Alf and Cod.

iv. For each character of the message (msj) perform:
Calculate the position of the character in the alphabet (ps), formula (1).
Apply formula (2) to calculate the basis of formula (3) (a).
Apply formula (3) to get (x).
Obtain from the alphabet, the character found at position x which is part of the encrypted message.

v. Next character of the message (msj).

vi. As final result, we have the encrypted message from the original message. (msj) to (msjc).

vii. Send to the receiver msjc, n, nf and alt.

Decryption process:

i. Receive the random number (n), the number of rows of Cod (nf), the array of indexes of the code (alt) and the encrypted message (msjc).

ii. Calculate the key (s) using formula (4)
Retrieve from the database p, q, k, Alf and Cod.

iii. For each character of the encrypted message (msjc):
Calculate the position of the character in the alphabet (a), formula (5).
Apply formula (6) to get (x).
Apply formula (7) to calculate the position (ps).
Obtain from the alphabet, the character found at position ps which is part of the decrypted message (original message).

iv. Next character of the encrypted message (msjc).

v. As final result, we have the decrypted message (original message) from the encrypted message. (msjc) to (msj).

vi. Display the encrypted message (msj).

The mathematical algorithms proposed for encryption and decryption obeys to the following mathematical expressions:

(1) $ps=Position(Alf,msj_i)$

(2) $a=Position(Alf,Codalt_{i \bmod k},ps))$

(3) $x=a^n mod\ z$

(4) $s=Calculates\ (n,\Phi)$

(5) $a=Position(Alf,msjc_i)$

(6) $x=a^s \bmod z$
(7) $ps=Position(Codalt_{i \bmod k}, Alf_x)$
Where:

Alf = Alphabet of the message
Cod = Code for encrypt and decrypt messages (matrix of characters). Alf and Cod are retrieved from a database.
i = 0,1,2,3,……, length of the message minus 1
k = number of rows of the matrix Cod
msj = original message
msjc = encrypted message
n = exponent of the bases between 4 and 9 digits; prime number randomly generated.
nf = number of rows of the matrix Cod randomly generated (between 0 and k - 1).
a = position of an element of msj in Alf.
ps = value calculated in formula (1).
alt = array of indexes randomly generated.
z = p * q = length of Alf. p and q are prime numbers.
Φ = (p-1) * (q-1)
s = inverse of n mod Φ
Position (text, car) = function that calculates the position of car within text, if not found, returns -1.
Calculates (n, Φ) = function that calculates s such that $c = a^n \bmod z$ and $d = c^s \bmod z$.
msjc, n, nf and alt are public keys.
p, q, k, Alf and Cod are private keys.

Example: Using the algorithm described above, we will perform the encryption and decryption process for the message Hello. The following values were considered:

Alf:Â Â›ÃŸlÃ®LÃ¦Ã˜kÂ«qÃ±ÂƒVÃ Ã‚8j5Ã¶}9sC_ Â ¤gÂ ¨wÂ • 4yÂ ´#Â – Â§~Ã¡ÂœxÂ…Ã§ZaÂ‚Ã Â’Â¸|Ã·Â´Ã‘UÃ³KÃ¢Â˜?Â °SÃ’Ã™1`O.ÂªÃŒÂ¦ÃŽc@Â™{NÃ• >Â ªÂ »Â Ã^Â£,$Ã‹%XÂŠpÂ¼Ã¬Â“ Ã‰Â³Â‘Â„)Â^Â²Âº^fzÃ…Â‹ÂŒÂ°ÃšHo&GÂšEPÂ¢AÃœBÂ mÂ”Â DÂ²Ã€Ã¸Â QÃ-FÃº7Ã¤ÃžrÂ¶Â•Â¾;Â¬ÂŽbÂ¼vÃ Âž3*Â»uÂ±=\Ã†Ã¹Â®Â©IÃ‰ÂŸÃ•+[Â Ã“ÃµÃ„('0Ã›J:Ã¯Â†Ã¨<Âµ]Ã«!-Â ¡TÂ ¹2Â £MÂ —Â ¥Â —Â –Â·edÂ¿"ÃŠÂ-Â¥ÃƒÂ€Ã‡6/Â¯RnÃ WhÂ½tYÂ‡Ã”Ã©i

nf: 3

Cod:

- Row 1:

Ã²Â¤Ã¥Â½|!ÂŒwaxroÃ¨Ã¶7ÃŸ;VÂ¥3JÃ£Â¿=SvÃŒ*Â‰Â¼Â¢Â°Ã¤MWÂ±seÃ±Ã“Ã¬dlÂ„Â ÂžD\Âœ%Ã´nO#Â´Ã‚",Â²Â¡9Ã¹Ã¼ Ã†}+Ã€Ã«2QFÃ»XjÃºÂ¶:0Ã™ÂªfYÂ–Â¸Ã³`Ã˜5Â¬Â…~1BÃ’Â¾Â·Ã^UÃ¦/ÃªiÃ Â€Â•Â-Â®ZuÂ“Ãºy$Ã©NmÂ‚_)Ã Ã‡zPÂ«Â‡RÃ…Â”.ÂŽÃ„Â‘?Ã HÂ˜@Ã‘8GÃšÃ cL >gÂ¹<ÃœÃŽÃ®Â†'Â -Ã›Âµ(qÃƒÃ¢Ã‰CTÂ—hÂ µkÂ Â¯Ã§ÂšÂ£Â©Ã·Ãž4Ã ÂƒÂ¦Ã¡Ã¸Â¨Â Â’Â‹^ÂŸ[IÃŠÂ›Ã—Ã”Ã‹&Â^bAKÂ»Â pÂ Â™Â§Ã¯Ã–Â Âº Ã•6Â³EÂ {tÂŠ

- Row 2:

?IÃ³@Â‘Â¬Ã…Ã¨Âœ&}Â¢Â˜JnBÃž{sÃŽ9iyÃ‘ÃŒ^Â¨ÂƒÂ¶%G_ 5Â `QÃªÃ¢Ã£rÂ¥Â«ÃŸjp"6Ã©wÂ©H2Â´Ã¸ÂŽÃ»Ã˜Ã eÃ¹Ã™Â³Ã€;TÃ†Ã¥ÃºC|Ã‹Y\]ÃšÂ¯DP<Â›fÃ¦ÂŒ#Ã›EÂ—Â Â‡Ã‰Ã’*Â†bÂ¦(Ã«d1ÃŠ[SÃ‡F-Â¸Â„Â Â½oÃ Â LUXÂ•3Ã·ctÂ¤8Â +Ã•Â–Â Ã¼WÂ”k~z=Â^$Ã^)ÃÂžÃ–Â¿VZÂ™Â‚Â Â“Â£NÃ„Ã¶RÃ°Â±ÂŸÂªMmÂšÂ²Ã±Ã ÂµqÂ€Ã Ã”Ã§Â°Â¹Â®Ã‚7Ã´Ã¡Ã—Â Ã¯ÃœAÃƒÂ .lÂ¼Â¡4Â®/Â‹vÂŠÂ…x!Ã²Ã¤uÃ¬:>hÂºÂ§ÃµÃ“KÃ·aÂ’Â¾0Â‰gOÂ»

- Row 3:

Ã¨Â±Âµ{ÂŽjÃ³vÂ¢Ã‹Â«tÂ¥gKÂšÂ¡^WÃ”Â»Ã–EÃ“Â·]fÃ‘Â¼Ã¶kÂ¯IÃ‰q$Ã’oÃŽÂ Â¶Â Ã +Â˜eÂ„w~Ã-cÃž*;?JÂ¨N0Â ZÃ‚Â”ÂƒÂ…ÃƒÃ±Â´Ã¦)ÂŠÃœÃ¸#Â^,Ã 4FÃµÃ Â†VlHAÂ•Ã‡\sÃ€S&@:Ã• Â UÂ§ÃšÃ®Ã¢-Ã ¹Â ¼ !Â £Â —/Ã^Ã„uÂ¸nÂ’ÂºÂ¦Â¾Ã˜1Â°Â ÂªMd3Â›ÂŒÂ³Ã¥<ryÂ–"9ÂŸÂ®Â™(zÃ¡D.>PÃ¯8ChÃªÃŸipRÃ¬Â¬ÃŒbÂ‰Â OÃ…Ã›BÂ¿Ã»Ã amÂ Â£Â‚_ÃŠ`Ã™6=|Ã—GLxÂ½Â¹Ã·ÂžQÃ† ‰Ã«Â€2Ã¤Ã°Ã²5Â‘Ã§7'XÃºÂ‹YÃ © Ã [Â“Ã´Â©Â¤TÃ•Â²Â‡Âœ

p: 13; q: 17; z=p*q: 221; ϕ=(p-1)*(q-1): 192

- Encryption:

The sender captures msj: Hello and generates n: 7799; alt: 2 0 1

For each value of i = 0, 1, 2, 3, 4 calculate:
ps = Position (Alf, M [i]):
110 196 3 3 111
a = Position (Alf, Cod [alt [i% 3]] [ps])
169 218 72 64 92
$x = a^n \bmod z$ (a, n, z):
169 75 149 165 105

Encrypted message: msjc: Ã„N3[Ã… This last value, with the values of n and alt, are sent to the receiver.

- Decryption:

The receiver, receives the values of msjc, n and alt, and calculate s: 91

For each value of i = 0, 1, 2, 3, 4 calculate:
a = Position (Alf, M.charAt (i)):
169 75 149 165 105
$x = a^n \bmod z$ (a, s, z1):
169 218 72 74 92
ps = Position (Cod [alt [i% 3]], Alf.charAt ((int) x)):
110 196 3 3 111

Decrypted message: Hello

5. Results and Discussion

5.1 Results Analysis

All tests were performed into two host Dell Inspiron, Intel ® Core (TM) i5-4200 CPU @ 1.60 GHz, 4 GB of RAM, with Ubuntu Server 14.0, the Java development environment JDK, the NetBeans IDE 8.0 and the database engine MySQL 5.6. To evaluate the quality of the software, 13 tests were performed, which consisted of varying the number of characters: 1,2,3,4,5,6,7,8,9,10,50,100 and 200 ciphers in a chain. The variables evaluated were memory, processor, latency and statistical reporting of the network, encryption and decryption time; and security level of the system. In the development of the tests, in order to obtain measurements of the CPU and network performance, two free software tools were used: System Activity Report (SAR), used to measure memory and processor consumption, and even statistical reporting of the network; and My Trace Route (MTR) to measure network latency. These tests were taken in real time for the client and the server at the time to encrypt and decrypt the message. To calculate encryption and decryption time, a program was developed in Java. For this analysis, the results obtained from the baseline RSA model and the optimized RSA model were compared. The techniques used for this evaluation were:

(i) Histograms of density: when comparing the medians of Fig. 2-a. Histogram of Baseline Time-Client (TBC) and Fig. 2-b Histogram of Optimized Time-Client (TOC), each one with 33.0ms and 1.00ms respectively, which corresponds to a rate of 8.94% and 9.09%; we can see that TOC>TBC (9.09%>8.94%) with a difference of 0.15% which represents the 99.85% of the observations, showing that the time was radically optimized. Furthermore, when comparing medians of Fig. 3-a Histogram of Baseline Time-Server (TBS) and Fig. 3-b Histogram of Optimized Time-Server (TOS), each one with 33090ms and 2.00ms respectively, corresponding to a percentage of 95.38% and 5.88 % respectively; we can see that the TOS<TBS (5.88%<95.38%) with a difference of 89.5%. However, the decryption time was drastically optimized. Considering the maximum times of the optimized model (11ms) and the baseline model (369ms), we can see that the time used to send messages with the proposed method is 33 times faster than the baseline. Likewise, the same analysis was performed on the rest of the variables getting a positive response with an efficiency improvement from 80% to 99%.

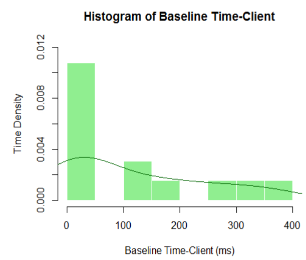


Fig. 2-a. Baseline Time-Client

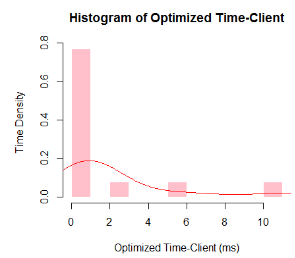


Fig. 2-b. Optimized Time-Client

(ii) Analysis of Variance (ANOVA): for this, a linear regression model was applied. As an example, the CPU Client usage (Fig. 4-a) and CPU Server usage (Fig. 4-b), as shown below. In Figs. 4-a and 4-b the results of linear regression were positive slopes ($\beta1$, $\beta2$) which means that had an increasing behavior. From Fig. 4-a the equation ① was obtained; and from Fig. 4-b the equation ②, obtaining an estimation of the regression lines:

① CPU Usage-Optimized-Client = - 1.590 + 1.096 * CPU Usage-Baseline-Client

② CPU Usage-Optimized-Server = 5.8168 + 0.6506 * CPU Usage-Baseline-Server

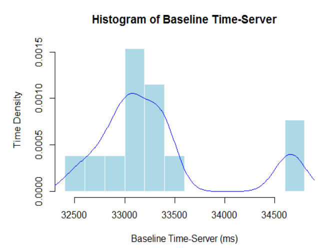


Fig. 3-a. Baseline Time-Server

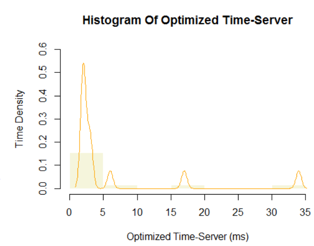


Fig. 3-b. Optimized Time-Server

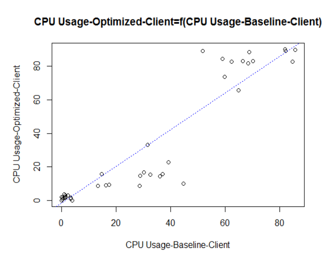


Fig. 4-a. CPU-Client usage

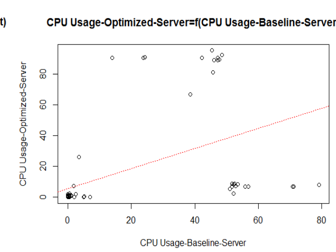


Fig. 4-b. CPU-Server usage

It was necessary to determine whether the equations obtained were the best models for the data, for this reason, to value the adjustment of these ciphers in linear regression models of Fig 4-a and Fig. 4-b, the analysis of variance of an F factor (ANOVA) was performed, which was used to perform a hypothesis test. This leads to the analysis for Fig. 4-a: (1) Null Hypothesis H0 $\beta 1=0$; (2) Alternative Hypothesis H1 $\beta 1 \neq 0$; (3) Statistic Test is F=718.39, with a significance level of $\alpha=0.05$. According to the distribution table Ft=4.84, we can deduce that the null hypothesis is rejected. Therefore, given the magnitude of the statistic test, we can deduce that the significance level of contrast is extremely low. Consequently, this study was optimal, concluding that the research is valid and reliable. On the other hand, in order to check the validity in the server side, we proceed with the same previous analysis but with Fig. 4-b: (1) Null hypothesis H0 $\beta 2=0$; (2) Alternative Hypothesis H1 $\beta 2 \neq 0$; (3) Statistic Test is F=23.23, with a significance level of $\alpha=0.05$. According to the distribution table Ft=4.84, we can deduce that the null hypothesis is rejected. Consequently, we conclude that the dispersion of Y is extremely low compared with X, therefore, the study was optimal and that research is effective.

(iii) Coefficient of determination (R^2): this value depends on: (1) SCYY known as the sum of the squares around the mean of Y; and (2) SCR is designated as the sum of squares due to regression. In Fig. 4-a we get values of SCR = 65841 and SCYY=72806, obtaining a result of R2=SCR/SCYY=0.9043, which means that the 90.43% of the variability data is collected by the regression line. With this analysis we can deduce that the equation obtained is optimal for a good data model. This study was conducted with all the variables achieving a positive result, concluding that the research about RSA is integral and recommended.

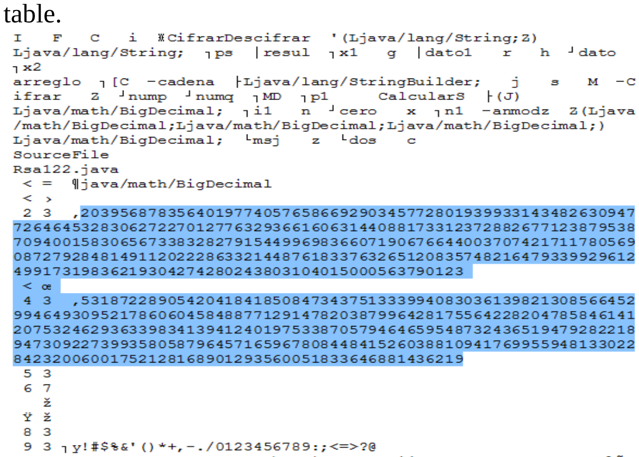
5.2 Discussion

In the present study the optimization of RSA encryption algorithm is confirmed through a generic model, able to encrypt and decrypt information which has increased the efficiency and security of messages transmitted over the network. When comparing the results obtained with the baseline model with the optimized model, we detected differences concerning the mathematical model, the development tools and the algorithms used. The base project has basic features such as the use of large prime numbers p and q, which in this case were found on the Internet. Being public, it can attempt the security of this method. On the other hand, these numbers limit the message, because its length depends on the digits comprising p and q. However, the proposed project was radically optimized because it allows to send encrypted and decrypted messages whose length is limited to the width of the channel and the processor, thus ensuring that the restriction is not due to the algorithm but to the infrastructure. Finally, we can see that the optimized RSA model responds favorably to three technical and legal measures which prove the level of security: confidentiality, availability and integrity. Confidentiality was achieved by not disclosing the parameters that are essential for the encryption and decryption. For this, we used a database that is accessed via username and password, and specifying the computer that can connected to it according to an IP or a range of IPs. Integrity was achieved by generating the Dynamic Link Library (DLL) to which only the author of the new RSA encryption method will be allowed to manipulate and modify. Availability refers to the accessibility of the user to send messages that require a higher degree of security to destination.

6. Analysis of security between Baseline RSA Model and Optimized RSA Model

The main vulnerability of the baseline RSA model is that anyone can access to private keys (i.e. large prime numbers) when editing the executable code (i.e. jar), as shown in Figure 5. In the optimized RSA model it does not happen because private keys are encrypted.

While the baseline RSA model generates encrypted messages completely asymmetric, the optimized RSA model is semi-asymmetric because it generates encrypted message consisting of 2-byte characters where 1 byte come from the original message and vice versa. Being considered the printable characters, the 255 of the ASCII table.



```
I   F   C   i  ¤CifrarDescifrar  '(Ljava/lang/String;Z)
Ljava/lang/String;  ┐ps  |resul  ┐x1   g  |dato1   r   h  ┘dato
┐x2
arreglo  ┐[C  -cadena  ├Ljava/lang/StringBuilder;   j   s   M  -C
ifrar   Z  ┘nump  ┘numq  ┐MD  ┐p1     CalcularS  ├(J)
Ljava/math/BigDecimal;  ┐i1   n  ┘cero   x  ┐n1  -anmodz  Z(Ljava
/math/BigDecimal;Ljava/math/BigDecimal;Ljava/math/BigDecimal;)
Ljava/math/BigDecimal;  └msj   z  └dos   c
SourceFile
Rsa122.java
 < =  ¶java/math/BigDecimal
 < >
 2 3  ,2039568783564019774057658669290345772801939933143482630947
72646453283062722701277632936616063144088173312372882677123879538
70940015830656733832827915449969836607190676644003707421711780569
08727928481491120222863321448761833763265120835748216479339929612
49917319836219304274280243803104015000563790123
 < œ
 4 3  ,5318722890542041841850847343751333994083036139821308566452
99464930952178606045848877129147820387996428175564228204785846141
20753246293633983413941240197533870579464659548732436519479282218
94730922739935805879645716596780844841526038810941769955948133022
84232006001752128168901293560051833646881436219
 5 3
 6 7
   ž
 Ÿ ž
 8 3
 9 3 ┐y!#$%&'()*+,-./0123456789:;<=>?@
```

Fig. 5. Fragment of executable code (jar). Baseline RSA Model

The probability that the optimized RSA model is violated is inversely proportional to the function f (u,v,w). Where u is a function of the number of characters in the message, v is a function of the number of secret keys whose maximum value is the factorial number of 221 and w is a function of the power of the formula RSA (n).

7. Conclusions and Future Work

This research focused on optimizing the RSA encryption algorithm. To achieve this, we designed and implemented a generic solution capable of encrypt and decrypt information, increasing efficiency and security of messages transmitted over the network. This solution included the review of the model, optimization of the mathematical expression model, which has improved the encryption method, and implementation of algorithms for secure transmission of messages on the network. We used Netbeans 8.0 which is free software that allowed the development of a new API with Java. Within this API we declared the method of the new RSA algorithm. The results show the functionality, security and usability of our study, but especially show quantitatively that the algorithm has been optimized.
As future work we planned to complete this algorithm in a DLL multiplatform.

Acknowledgments

This work has been partially funded by Ecuador Contest of Research Projects in Advanced Network CEDIA CEPRA IX-2015-01-RSA, under the “RSA encrypted algorithm optimization to improve performance and security level of Web messages" project.

References

[1] X. Zhou and X. Tang, “Research and implementation of RSA algorithm for encryption and decryption”, in 6th International Forum on Strategic Technology (IFOST), 2011,1118-1121.
[2] R. Patidar and R. Bhartiya, “Modified RSA cryptosystem based on offline storage and prime number” in: IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), 2013, 1-6.
[3] P.S. Yadav, P. Sharma and D.K. Yadav, “Implementation of RSA algorithm using Elliptic Curve algorithm for security and performance enhancement” in International Journal of Scientific & Technology Research 1(4), 2012, 102-105.
[4] G. Singh and A. Supriya, “A study of encryption algorithms (RSA, DES, 3DES and AES) for information security” in International Journal of Computer Applications 67(19), 2013, 33-38.
[5] Q. Liu, Y. Li, T. Li and L. Hao “The research of the batch RSA decryption performance” in Journal of Computational Information Systems 7(3), 2011, 948-955.
[6] ISCI, “Enhancing security features in RSA cryptosystem” in: Computers Informatics (ISCI), 2012 IEEE Symposium on. 2012, 214-217.
[7] L. Dongjiang and W. Yandan, “An optimization algorithm of RSA key generation in embedded system” in Journal of Theoretical and Applied Information Technology 7(3), 2012, 948-955.
[8] S. Mahajan and M. Singh, “Analysis of RSA algorithm using GPU programming” CoRR abs/1407.1465, 2014.
[9] M. Gadelha, C. Costa Filho and M. Costa, “Proposal of a cryptography method using gray scale digital images,” in International Conference for Internet Technology and Secured Transactions, 2012, 331-335.
[10] H. J. Wang, “Key generation research of RSA public cryptosystem and matlab implement,” in IEEE International Conference on Sensor Network Security Technology and Privacy Communication System (SNS & PCS), 2012, 639-642.
[11] S. Nagar and S. Alshamma, “High speed implementation of RSA algorithm with modified keys exchange,” in 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), 2012, 639,642.
[12] R. Minni, K. Sultania, S. Mishra and D. Vincent “An algorithm to enhance security in RSA,” in Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), 2013, 1-4.
[13] L. Wang and Y. Zhang, “A new personal information protection approach based on rsa cryptography,” in IT in Medicine and Education (ITME), 2011 International Symposium on. Volume 1, 2011, 591-593.
[14] P. Yellamma, C. Narasimham and V. Sreenivas “Data security in cloud using RSA,” in Computing, Communications and Networking Technologies (ICCCNT), 2013, 1-6.
[15] R. Dhakar, A. Gupta and P. Sharma, “Modified RSA encryption algorithm,” in Advanced Computing Communication Technologies (ACCT), 2012, 426-429.
[16] S. Beniwal and E. Yadav, “An effective efficiency analysis of random key cryptography over RSA,” in Computing for Sustainable Global Development (INDIACom), 2015, 267-271.
[17] M. Rahman, I. Rokon and M. Rahman “Efficient hardware implementation of RSA cryptography,” in 3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication, 2009, 316-319.
[18] R. Johnsonbaugh, “Matemáticas discretas,” Pearson Educación, 2005.
[19] J. Ramió, “Libro electrónico de seguridad Informática y Criptografía,” Manual docente de libre distribución, Universidad Politécnica de Madrid, 2006.
[20] A.E. Cohen and K. Parhi, “Architecture optimizations for the RSA public key cryptosystem,” A tutorial. IEEE Circuits and Systems Magazine 11(4), 2012, 24-34.
[21] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, “Handbook of applied cryptography,” CRC press, 1996.