Software Engineering Department

ORT Braude College

Course 61711 – Extended Project in Software Engineering

An efficient and secure RSA–like cryptosystem exploiting Rédei rational functions over conics

In Partial Fulfillment of the Requirements for

Final Project in Software Engineering (Course 61711)

Karmiel – June 2020

Authors:

Asaf Yusufov ID: 309402451

Roni Belkin ID: 306969957

Supervisors:

Dr. Dan Lemberg

Dr. Beniamin Mounits

Table of Contents

Abstract: *In our project, we propose a realization and implementation of more efficient RSA-like cryptosystem. The original textbook RSA cryptosystem based on integer factorization problem and discrete logarithm problem. The processes of the encryption and decryption in the original textbook RSA requires a higher running time. In our implementation of the more efficient RSA-like cryptosystem we are going to use Rédei rational functions based on the conic equation and having a one-to-one correspondence between two mathematical sets. Our proposed cryptosystem becomes two times faster than the original textbook RSA cryptosystem by involving the lowest number of modular inversions with respect to original textbook RSA. Our proposed cryptosystem affords higher security in broadcast applications and equal level of security in a one-to-one communication. This is a research project that going to check if the proposed way does offer a twice faster solution for decryption. We are going to use software optimizations and mathematical equations to provide a good running time performance. Our goal is to confirm the hypothesis that the more efficient RSA-like cryptosystem two times faster than the original textbook RSA cryptosystem.*

Keywords: *Rédei rational functions, RSA problem, Public cryptography, Integer factorization problem, Modular inversions, Broadcast applications.*

1. INTRODUCTION

The RSA cryptosystem is based on one-way function that difficult to compute by her and takes a high running time. The difficulty of computing caused the fact that the function cannot convert an input number to number that can be easy to compute. For lowest running time the function needs more information.

In the RSA algorithm, the variables p, q are primary numbers and parameter e is a free number that algorithm can choose. The original algorithm chooses the parameter e to be a number that invertible, but the parameter e must belong to ring of Euler function. By using a parameter e that belongs to ring of Euler function and the multiplication of the primary parameter $N = p \cdot q$ we called it a public key. This pair – (N, e) is known to anyone. The triple – (p, q, d) is the secret key, while the parameter d is a number that can decrypt the encrypted massage *modulo* N. The N is known number in secret key, because the secret key consists parameters p, q that they are gives together the number N.

The most successful attacks on RSA algorithm happens when the parameter e and d are small. The number of options is small, and it is not a big problem to check each option. Some attacks based on leaked information that leaks to attacker and make his work to easy by eliminate a high percentage of options to possible key.

Because of the attacker success, some of upgraded RSA algorithms tried to upgrade the part of the decryption and make it faster. The upgraded RSA algorithms based on isomorphism between two groups. The first group is set of points over a conic or cubic curve.

We are going to perform an upgraded RSA algorithm and perform the RSA scheme that based on isomorphism over a conic, with the fastest performance for decryption. Furthermore, our scheme uses a different set of conics that bring the fastest running time even fastest than the upgraded RSA algorithms. The security of our RSA algorithm does not damage the security level or the efficiency that shown in original textbook RSA.

1.1. **Organization of paper**

Section 2 – we start by review the background and related mathematical equations that standing behind the RSA. Section 3 – we describe the efficient RSA-like cryptosystem algorithm step by step in details. Section 4 – we briefly discuss the expected result that we want to get. Section 5 – consist of preliminary software engineering documents – use case, initial GUI, concluded by a short text plan section.

2. BACKGROUND

2.1. **Historical overview of RSA**

With the development of computer technology, the transfer of information in digital form has rapidly increased. There are many applications, including electronic mail systems, bank systems and data processing systems, where the transferred information must pass over communications channels which may be monitored by electronic eavesdroppers. While the degree of security required may vary for various applications, it is generally important for all these examples that the substance of communications pass directly from a sender to an intended receiver without intermediate parties being able to interpret the transferred message. In addition, there are further instances where information in computer memory banks must be protected from snoopers who have access to the memory through data processing networks. In addition to these privacy requirements, authentication of the source of a message must often be insured along with the verification and security of the message content. For example, in banking applications, it is required that a signed document, such as a bank draft, be authenticated as being actually signed by the indicated signature. Furthermore, in many applications, it is desirable to further require safeguards against signature forgery by a message recipient.

In general, cryptographic systems are adapted to transfer a message between remote locations. Such systems include at least one encoding device at a first location and at least one decoding device at a second location, with the encoding and decoding devices all being coupled to a communication channel. For digital systems, the message is defined to be a digital message, M, that is, a sequence of symbols from some alphabet. In practice, the alphabet is generally chosen to be the binary alphabet consisting of the symbols 0 and 1.

Each encoding device is an apparatus which accepts two inputs: a message-to-be-encoded, M, and an encoding key or operator, E. Each encoding device transforms the message M in accordance with the encryption operator to produce an encoded version C of the message (which is denoted as the ciphertext) where $C = E(M)$. The encoding key and the ciphertext are also digital sequences.

Each decoding device is an apparatus which accepts two inputs: a ciphertext-to-be-decoded C and a decoding key or operator, D. Each decoding device transforms the ciphertext in accordance with the decryption operator to produce a decoded version M' of the ciphertext where $M' = D(C)$, or

$M' = D\big(E(M)\big)$. Like the encoding key, the decoding key and decoded message M' are also digital sequences. The encoding and decoding keys are selected so that $M' = M$ for all messages M.

In operation, a message, once encoded into ciphertext, is transmitted over the channel to a recipient who decodes the received ciphertext to obtain the original message M. Thus, a recipient sees the original message M as the output of his decoding device.

To a large degree, the quality of performance of a cryptographic system depends on the complexity of the encoding and decoding devices. Regarding the problem of ensuring privacy of communications for a system where an eavesdropper can listen to every message transmitted on the communications channel (which might, for example, be a radio link), the effectiveness of the system depends upon the ability to ensure that the eavesdropper is unable to understand any such overheard messages. In the prior art systems, the sender and recipient arrange to have corresponding encoding and decoding keys which are kept secret from the eavesdropper, so that even if the eavesdropper knows the construction of the encoding and decoding devices, he would not be able to decode the messages he hears, even after hearing a large number of messages. In practice, however, this constraint results in extremely complex and correspondingly expensive equipment. A disadvantage of the prior art systems results from the general requirement that the pre-arranged encoding and decoding keys must be delivered in a secure fashion (often by courier) to the sender and receiver, respectively, to enable communication through the systems.

In a public-key cryptosystem, each user (e.g. user *A*) places in a public file an enciphering operator, or key, E_A. User *A* keeps to himself the details of the corresponding deciphering key D_A which satisfies the equation:

$$D_A\left(E_A(M)\right) = M \qquad (1)$$

for any message M. In order for the public key system to be practical, both E_A and D_A must be efficiently computable. Furthermore, user *A* must not compromise D_A when revealing E_A. That is, it should not be computationally feasible for an eavesdropper to find an efficient way of computing D_A, given only a specification of the enciphering key E_A (even though a very inefficient way exists: to compute $D_A(C)$, just enumerate all possible messages M until one such that $E_A(M) = C$ is found. Then $D_A(C) = M$.). In a public key system, a judicious selection of keys ensures that only user *A* is able to compute D_A efficiently.

Whenever another user (e.g. user *B*) wishes to send a message M to *A*, he looks up E_A in the public file and then sends the enciphered message $E_A(M)$ to user *A*. User *A* deciphers the message by computing $D_A\left(E_A(M)\right) = M$. Since D_A is not derivable from E_A in a practical way, only user *A* can decipher the message $E_A(M)$ sent to him. If user *A* wants to send a response to user *B*, user *A* enciphers the message using user *B's* encryption key E_B, also available in the public file. Therefore, no transactions between users *A* and *B*, such as exchange of secret keys, are required to initiate private communication. The only "setup" required is that each user who wishes to receive private communication must place his enciphering key *E* in the public file.

The encoding key E is a pair of positive integers (e, N) which are related to the particular decoding device. The message M is a number representative of a message-to-be-transmitted and wherein

$0 \leq M \leq N - 1$ where N is a composite number of the form $N = p \cdot q$ where p and q are prime numbers.

For messages represented by numbers outside the range 0 to $N - 1$, a conventional blocking means is utilized to break the message into message block words before encoding, where each block word is representative of a number within the specified range. Following subsequent decoding, the recovered block words may be transformed back to the original message.

The transformation provided by the encoding device is described by the relation:

$$C \equiv M^e (mod\ N) \qquad (2)$$

where e is a number relatively prime to $(p-1)\cdot(q-1)$.

The decoding device is responsive to the received ciphertext word C and a decoding key to transform that ciphertext to a received message word M'. The decoding key D is a pair of positive integers (d, N). M' is a number representative of a deciphered form of C (i.e. reconstituted plaintext) and corresponds to the relation:

$$M' \equiv C^d (mod\ N) \qquad (3)$$

where d is a multiplicative inverse of e, so that:

$$e \cdot d \equiv 1\left(mod\left(lcm(p-1,\ q-1)\right)\right) \qquad (4)$$

where $lcm(p-1,\ q-1)$ is the least common multiple of numbers $p\ -\ 1$ and $q - 1$.

With these encoding and decoding devices, a message sent from the encoding device to the decoding device is transformed from M to C by the encoding device, and then back from C to M' by the decoding device, where $M'\ \equiv\ M'\ (mod\ N)$.

2.2. Security of RSA – relation to factoring

The task handled by a passive adversary is that of recovering plaintext M from the subsequent ciphertext C, given the public knowledge (N, e) of the proposed receiver A. This is called the RSA problem, known as the following: given a positive integer N that is a product of two distinct odd primes p and q, a positive integer e such that the $gcd(e, (p-1)(q-1)) = 1$, and an integer c, find an integer m such that $m^e \equiv c(mod\ N)$. In other words, the RSA problem is that of finding e^{th} roots modulo a composite integer N. There is no efficient algorithm known for this problem.

One possible approach which an adversary could employ to solving the RSA problem is to first factor N, and then compute φ and after that computing d just as shown in equation (4).

2.3. Attacks on RSA

There are several methods that have been tried over the last 20 years since the RSA method was created. No method has been successful so far, but these are all these methods.

2.3.1. Searching the message size

One of the weaknesses of public key cryptography is that one side has to give away to everybody the algorithm that encrypts the data. If the message size is small, then one could simply try to encrypt every possible message block, until a match is found with one of the ciphertext blocks. In practice this would be an insurmountable task because the block sizes are quite large.

2.3.2. Guessing d

Another possible attack is a known ciphertext attack. This time the attacker knows both the plaintext and ciphertext, one of sides simply has to encrypt something. They then try to crack the key to discover the private exponent, d. This might involve trying every possible key in the system on the ciphertext until it returns to the original plaintext. Once d has been discovered it is easy to find the factors of N. Then the system has been broken completely and all further ciphertexts can be decrypted.

The problem with this attack is that it is slow. There are an enormous number of possible options to try. This method is a factorizing algorithm as it allows us to factor N. Since factorizing is an intractable problem it is understood that this is very difficult. This method is not the fastest way to factorize N. Therefore, one is suggested to focus effort into using a more efficient algorithm specifically designed to factor N.

2.3.3. Cycle attack

This attack is very similar to the "Guessing d attack". The idea is that we encrypt the ciphertext repeatedly, counting the iterations, until the original text appears. This number of re-cycles will decrypt any ciphertext. This method is very slow and for a large key it is not a practical attack. A generalization of the attack allows the modulus to be factored and it works faster most of the time. But even this will still have difficulty when a large key is used. Also, the use of strong number p, aids the security.

Therefore, the generalized form of the cipher attack is another factoring algorithm. It is not efficient, and therefore the attack is not good enough compared with modern factoring algorithms.

2.3.4. **Common modulus**

One of the early weaknesses found was in a system of RSA where the users within an organization would share the public modulus. That is to say, the administration would choose the public modulus securely and generate pairs of encryption and decryption exponents - public and private keys, and distribute them all the users. The reason for doing this is to make it convenient to manage and to write software for. However, it allows anyone to view any messages encrypted with two keys.

2.3.5. **Faulty Encryption**

There are capitalize on the common modulus weakness due to a transient error when transmitting the public key. Consider the situation where an attacker Eve, has access to the communication channel used by Alice and Bob. In other words, Eve can listen to anything that is transmitted, and can also change what is transmitted. Alice wishes to talk privately to Bob, but does not know his public key. She requests by sending it, Bob replies. But during transmission, Eve can see the public key and decides to flip a single bit in the public exponent of Bob.

When Alice receives the faulty key, she encrypts the prepared message and sends it to Bob. Eve also gets it. But of course, Bob can not decrypt it because the wrong key was used. He lets Alice know and they agree to try again, starting with Bob re-sending his public key. This time Eve does not interfere. Alice sends the message again, this time encrypted with the correct public key.

Eve now has two ciphertexts, one encrypted with the faulty exponent and one with the correct one. She also knows both these exponents and the public modulus. Therefore, she can now apply the common modulus attack to retrieve Alice's message, assuming that Alice was foolish enough to encrypt the same message the second time.

2.3.6. **Low exponent e**

In order to improve the efficiency of encryption, it is desirable to select a small encryption exponent e such as $e = 3$. A group of entities may all have the same encryption exponent e, however, each entity in the group must have its own distinct modulus. If an entity A wishes to send the same message M to three entities whose public moduli are N_1, N_2, N_3, and whose encryption exponents are

$e = 3$, then A would send $C_i = M^3 (mod\ N_i)$, for $i = 1, 2, 3$. Since these moduli are most likely pairwise relatively prime, an eavesdropper observing C_1, C_2, C_3 can use Chinese remainder theorem to find a solution x,

$0 \leq x < N_1 N_2 N_3$, it must be the case that $x = M^3$. Hence, by computing the integer cube root of x, the eavesdropper can recover the plaintext M.

2.3.7. **Factoring the public – key**

Factoring the public key is seen as the best way to go about cracking RSA. This method for attacking RSA scheme based on the knowing public key works efficiently if the decryption key d is small enough [1].

2.4. **Conic equations usage in cryptography**

Since the detection of public-key cryptography by Diffie and Hellman, more than a few efforts have been produced to find practical public key systems dependent on the complexity of cracking some problems. There are number of core families of public key cryptosystems constructed on computational number theory. The first family comprises RSA and correlated variations. The second family is constructed on Diffie-Hellman-type schemes as ElGamal and correlated variations which take advantage of properties of exponentiation over finite cyclic groups. The schemes that belongs to the first family concern us and they are based on conic curves over ring $\mathbb{Z}_N$ [2].

Conic equations in analytic geometry are equations that defined by conic section that satisfied a set of points whose coordinates comply with a quadratic equation. This section is the curve attained as the intersection of a plane, called the cutting plane, with the surface of a double cone. There are four types of conics: the circle, ellipse, parabola, and hyperbola as Figure 1 illustrates:

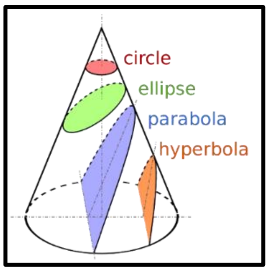


Figure 1 – Conics sections shown on one half of double cone.

The construction of public key cryptosystem using conic curves is based on the difficulty of factoring N, which is the product of two large primes. The way of the encryption defines by the type of operation between points in the conic section set. Conics are the simplest non-linear curves.

With the use of isomorphism as part of the operation between the points and use of parametrization of Pell's hyperbola equation over $\mathbb{Z}_N$ to compute those points that correspond to the message that need to be encrypted, the cryptosystem must send twice as many bits per message, with no improving the security of the system but showing that they are as secure as the original RSA.

2.4.1. **Pell's equation**

Pell's equation known as Pell conic is any Diophantine equation of the form

$$x^2 - D \cdot y^2 = 1 \tag{5}$$

where D is a given positive non-square integer and integer solutions are sought for x and y. In Cartesian coordinates, the equation has the form of a hyperbola, solutions occur wherever the curve passes through a point whose x and y coordinates are both integers, such as the trivial solution with $x = 1$ and $y = 0$. Joseph Louis Lagrange proved that, if D is not a perfect square, Pell's equation has infinitely many distinct integer solutions.

The Pell conics are class of curves described by equation (5), if $\mathbb{R}$ is a domain, then

$$\mathcal{H} = \{(x,\ y) \in \mathbb{R} \times \mathbb{R} :\ x^2 - Dy^2 = 1\} \tag{6}$$

denotes the set of all points *(x, y)* on the curve with coordinates in $\mathbb{R}$. A group structure can be defined over the Pell conics by means of the resulting product between points in $\mathcal{H}$:

$$(x_1, y_1) \cdot (x_2, y_2) = (x_1 x_2 + D y_1 y_2, y_1 x_2 + x_1 y_2) \tag{7}$$

To generalize the product for any ordinary field $\mathbb{K}$ by providing a group structure over the conics:

$$C = \{(x, y) \in \mathbb{K} \times \mathbb{K} : x^2 + Hxy - Dy^2 = 1\} \tag{8}$$

When C defines our conic.

2.4.2. **Parametrization**

The parametrization in our RSA-like cryptosystem allows us to define a bijection between a conic that we use and the set of parameters. The parametrization yielding a group structure on this set. For the seek of simplicity, we only write Φ while there is no confusion. This bijection allows us to derive from $\odot\ C$ - the Rédei rational functions over conic instead of the set of parameters P_K:

$$\Phi_{H,D} : \begin{cases} \mathcal{C} & \to \mathcal{P}_{\mathbb{K}} \\ (x,y) & \mapsto \dfrac{1+x}{y} \quad \forall (x,y)\in\mathcal{C}, \quad y\neq 0 \\ (1,0) & \mapsto \alpha \\ (-1,0) & \mapsto -\dfrac{H}{2}\ , \end{cases}$$

and

$$\Phi^{-1}_{H,D} : \begin{cases} \mathcal{P}_{\mathbb{K}} & \to \mathcal{C} \\ m & \mapsto \left(\dfrac{m^2+D}{m^2+Hm-D}, \dfrac{2m+H}{m^2+Hm-D}\right) \quad \forall m\in\mathbb{K}\ . \\ \alpha & \mapsto (1,0)\ , \end{cases}$$

Figure 2 – parametrization of point on the curve
https://arxiv.org/pdf/1511.03451v2.pdf

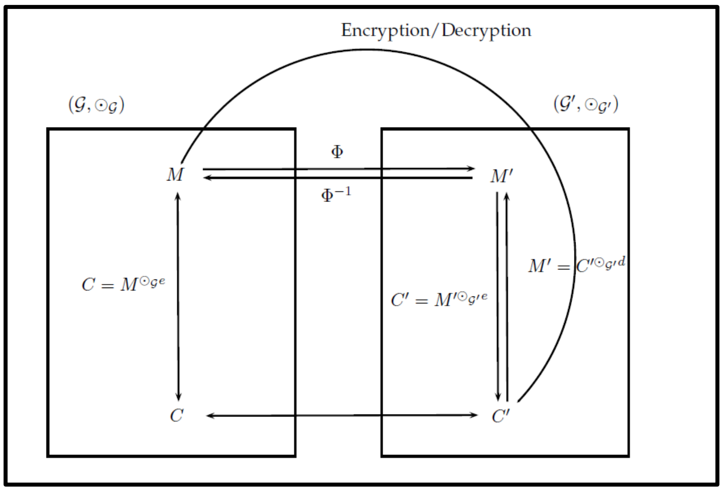
For simplicity we use instead of P_K, the next parameterization. This parametrization is only one option from the four in encryption option and one from the three in decryption option that shown in figure 1:

$$\begin{cases} a \odot_{\mathcal{P}_{\mathbb{K}}} b = \dfrac{D+ab}{H+a+b}, & a+b\neq -H \\ a \odot_{\mathcal{P}_{\mathbb{K}}} b = \alpha, & a+b=-H \end{cases}$$

Figure 3 – parametrization set of conic points
https://arxiv.org/pdf/1511.03451v2.pdf

2.5. RSA-like scheme based on isomorphism over conic

We are using a multiplicative notation for groups. For a group $\mathcal{G}$ with the operation $\odot_{\mathcal{G}}$ and an element G ∈ $\mathcal{G}$, we use the following:



Encryption/Decryption
$(\mathcal{G}, \odot_{\mathcal{G}})$
$(\mathcal{G}', \odot_{\mathcal{G}'})$
Φ
M
M'
Φ^{-1}
$M' = C'^{\odot_{\mathcal{G}'} d}$
$C = M^{\odot_{\mathcal{G}} e}$
$C' = M'^{\odot_{\mathcal{G}'} e}$
C
C'

Figure 4 – RSA – like scheme based on isomorphism

When the notation for the exponentiation as regards the group operation:

$$\boxed{G^{\odot_{\mathcal{G}} x} = \underbrace{G \odot_{\mathcal{G}} \dots \odot_{\mathcal{G}} G}_{x \text{ times}}}$$

Figure 5 – group operation notation

Such as figure 4 that can be achieved by obtaining an efficiently computable isomorphism:

$$\Phi : (\mathcal{G}, \odot_{\mathcal{G}}) \rightarrow (\mathcal{G}', \odot_{\mathcal{G}'}) \qquad (9)$$

between two groups, hence that in the decryption process, the operation of the exponentiation in $\mathcal{G}'$ and the reverse of the isomorphism is more efficient than just using the exponentiation in $\mathcal{G}$. The plaintext M is encrypted into the ciphertext C', either by first using the exponentiation in $\mathcal{G}$ and then the isomorphism, or vice versa. The decryption follows the inverse process.

Generally, $\mathcal{G}$ is the set of points of a curve over the ring $\mathbb{Z}_N$ and N is the product of two primes, and $\mathcal{G}' = \mathbb{Z}_N^*$.

2.6. **Rèdei rational function**

Let R an arbitrary commutative unitary finite Ring and let $t(x) = x^2 - ax - b$ imply a polynomial over R with the two distinct roots $\alpha, \overline{\alpha}$. There always exist unique elements $r_k, s_k \in$ R such that $\alpha^k = r_k + \alpha s_k$ and $\overline{\alpha}^k = r_k + \overline{\alpha} s_k$ for all $k \in \mathbb{N}$.

For $n \geq 1$ by the binomial theorem:

$$(x+\alpha)^n = \sum_{k=0}^{n} \binom{n}{k} \alpha^k x^{n-k} = \sum_{k=0}^{n} \binom{n}{k} r_k \cdot x^{n-k} + \alpha \sum_{k=0}^{n} \binom{n}{k} s_k \cdot x^{n-k} = g_n(x) + \alpha h_n(x) \qquad (10)$$

where $g_n(x), h_n(x) \in R[x]$ are coprime over R.

The rational function:

$$R_n(x) = \frac{g_n(x)}{h_n(x)} \qquad (11)$$

Is called Rèdei function of degree n with respect to $t(x)$ [3].

3. ALGORITHM

In our project, we have hypothesis that the running time of decryption in RSA-like cryptosystem exploiting Rédei rational functions over conics faster twice than the original textbook RSA cryptosystem. In terms of algorithm, we are going to use the scheme [4] that has the same communication between the sender and receiver. Hence, we have the same variables as int the original textbook RSA cryptosystem, only one difference that going to sustain our changing certain condition, such as parametrization of the point on the curve.

3.1. **The scheme**

In this section, we describe the key generation, the encryption and the decryption algorithms. The following steps show the key generation:

- Choose two prime numbers p, q and compute:
$$N = p \cdot q \tag{12}$$
- Choose an integer e (not Euler number) such that:
$$gcd(e, lcm(p+1, q+1) = 1) \tag{13}$$
  The pair (N, e) is called the *encryption key*.
- Evaluate:
$$d = e^{-1}\, modulo\ lcm(p+1, q+1) \tag{14}$$
  The triple (p, q, d) is called the decryption key.

Now, consider the set:

$$\widetilde{\mathbb{Z}_{\mathbb{N}}^{*}} = \left\{\frac{m^2+D}{m^2-D}: \forall m, D \in \mathbb{Z}_{\mathbb{N}}, \pm D\ quadratic\ non-residues\ modulo\ N\right\}.$$

Figure 6 – Definition of set that consist message and quadratic non residue number

- If you have one message – divide the message into 2 messages $(M_x, M_y) \in \widetilde{\mathbb{Z}_{\mathbb{N}}^{*}} \times \mathbb{Z}_{\mathbb{N}}^{*}$.
- For encryption:
  - Compute:
$$D = \frac{M_x^2 - 1}{M_y^2} (mod\ N)\text{ , so that } (M_x, M_y) \in \mathcal{H}_{\mathbb{N}}^{*} \tag{15}$$

  - Compute:
$$M = \Phi(M_x, M_y) = \frac{M_x + 1}{M_y} (mod\ N) \tag{16}$$
  - Compute the ciphertext:
$$C = M^{\odot Pe} (mod\ N) = Q_e(D, M)(mod\ N) \tag{17}$$
- For decryption:
  - Compute:
$$C^{\odot Pd} (mod\ N) = M \tag{18}$$
  - Retrieve the plaintext (M_x, M_y) by means of Φ^{-1}
$$\Phi^{-1}(M) = \left(\frac{M^2+D}{M^2-D}, \frac{2M}{M^2-D}\right) (mod\ N) = (M_x, M_y) \tag{19}$$

3.1.1. **Attention**

In section 3.1, we have introduced our RSA scheme. We have considered Pell equation $X^2 - Dy^2 = 1$ in $\mathbb{Z}_{\mathbb{N}}$, where D is not quadratic residue, when in [5] the author worked on the complementary case, i.e., when D is a quadratic residue. Moreover, we have used a parametrization which allowed us to define an original $\odot\ P\mathbb{Z}_{\mathfrak{p}}$ connected to Rédei rational functions. We gave seen that in this case an analogous of the Fermat little theorem holds, i.e., $m^{\odot P\mathbb{Z}_{\mathfrak{p}}\mathfrak{p}+2} = m\ (mod\ \mathfrak{p}), \forall m \in \mathfrak{p}_{\mathbb{Z}_{\mathfrak{p}}}, \mathfrak{p}$ prime.

$\mathbb{K}$ is a finite field. For this reason, the decryption key is computed modulo $lcm(p^2-1, q^2-1)$ because it is much more efficient than calculate $lcm(p-1, q-1)$.

As we have seen the message should come in pairs in $\widetilde{\mathbb{Z}}_{\mathbb{N}}^{*} \times \mathbb{Z}_{\mathbb{N}}^{*}$. We can see that also in other RSA-like schemes exploit Pell equation in pairs in $\widetilde{\mathbb{Z}}_{\mathbb{N}}^{*} \times \mathbb{Z}_{\mathbb{N}}^{*}$ [5].

3.2. **Output type**

The output in our two cases is integer number. The first case is encryption, we are going to encrypt the message and get our ciphertext C as integer after some steps as shown in 3.1. The last step that gives us the ciphertext C is $C = M^{\odot Pe}(mod\ N) = Q_e(D, M)\ (modN)$, as we can see it must be an integer.

The second case is decryption, we are going to decrypt the ciphertext C to get the original message M. After some steps we get the last step $C^{\odot Pd}(mod\ N) = M$, as we can see it must be an integer.

3.3. **Complexity of decryption and encryption**

The original textbook RSA has the same complexity in comparison to the RSA-like cryptosystem exploiting Rédei rational functions over conics in encryption part. The difference occurs in the decryption process. The decryption of original textbook RSA costs $2\mathrm{E}_{\mathrm{Z_N}}^{d}$ when the cipher text size is $2\log N$. The RSA-like cryptosystem exploiting Rédei rational functions over conics decryption costs $\mathrm{E}_{\mathrm{Z_N}}^{d} + 3\mathrm{M} + \mathrm{I}$ when the cipher text size is $2logN$. $\mathrm{E}_{\mathrm{Z_N}}^{d}$ is the cost of Rédei rational function usage. The operations $\mathrm{E}_{\mathrm{Z_N}}^{d}$ and $\mathrm{E}_{\mathrm{P}}^{d}$ can be performed in $O(\log_2 N)$ with respect to addition, subtraction and multiplication [6]. M is message size and I is the inverse operation cost.

4. EXPECTED RESULTS

4.1. Runtime performance

Our goal is to get our algorithm of RSA-like cryptosystem exploiting Rédei rational functions over conics in two times faster decryption part than the original textbook RSA. We want to achieve this goal by using some parametrizations that bring us to use the Rédei rational functions and it is ensuring efficient and faster running time. We expect that the encryption part at our algorithm and original textbook RSA are the same, But the decryption is much faster.

4.2. Convenience

Each of mathematical operations that we use checked and compared if the operation from some libraries are faster than the implementation that we can do. Also, we try to make our algorithm even faster than twice over the original textbook RSA by making an optimization wherever as possible.

5. SOFTWARE ENGINEERING DOCUMENATION

5.1. Use case

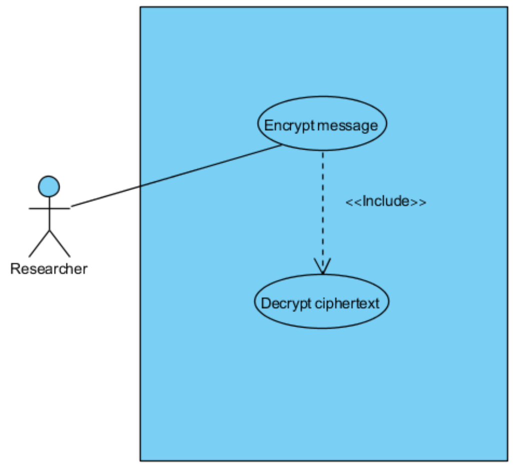


Figure 7 – An efficient and secure RSA–like cryptosystem exploiting Rédei rational functions over conics - use case

UC1: Encrypt message –

- Goal – Define all parameters that let to the encrypt function to encrypt the message successfully.
- Precondition – None.
- Possible errors – p, q are not prime numbers. Also, $\gcd\left(e, lcm(p+1, q+1)\right) \neq 1$.
- Limitations – After the use, the encryption function the parameters can not change the values of parameters that chosen before.

UC2: Decryption message –

- Goal – Decrypt the given ciphertext by using the private key d.
- Precondition – Get the public key correctly and the encryption function finished to encrypt successfully.
- Possible errors – The d value of the private key is not calculated correctly, so the decryption can not calculate the original message.
- Limitations – At the moment that the value d is not calculated correctly, the encryption scheme can not be used.

5.2. Class diagram

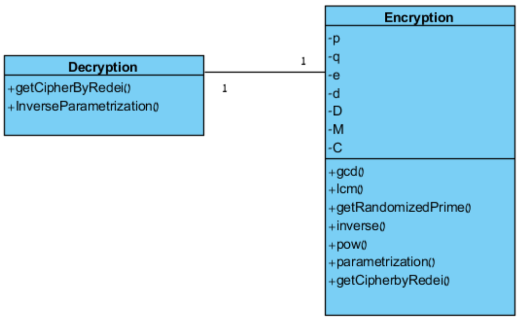


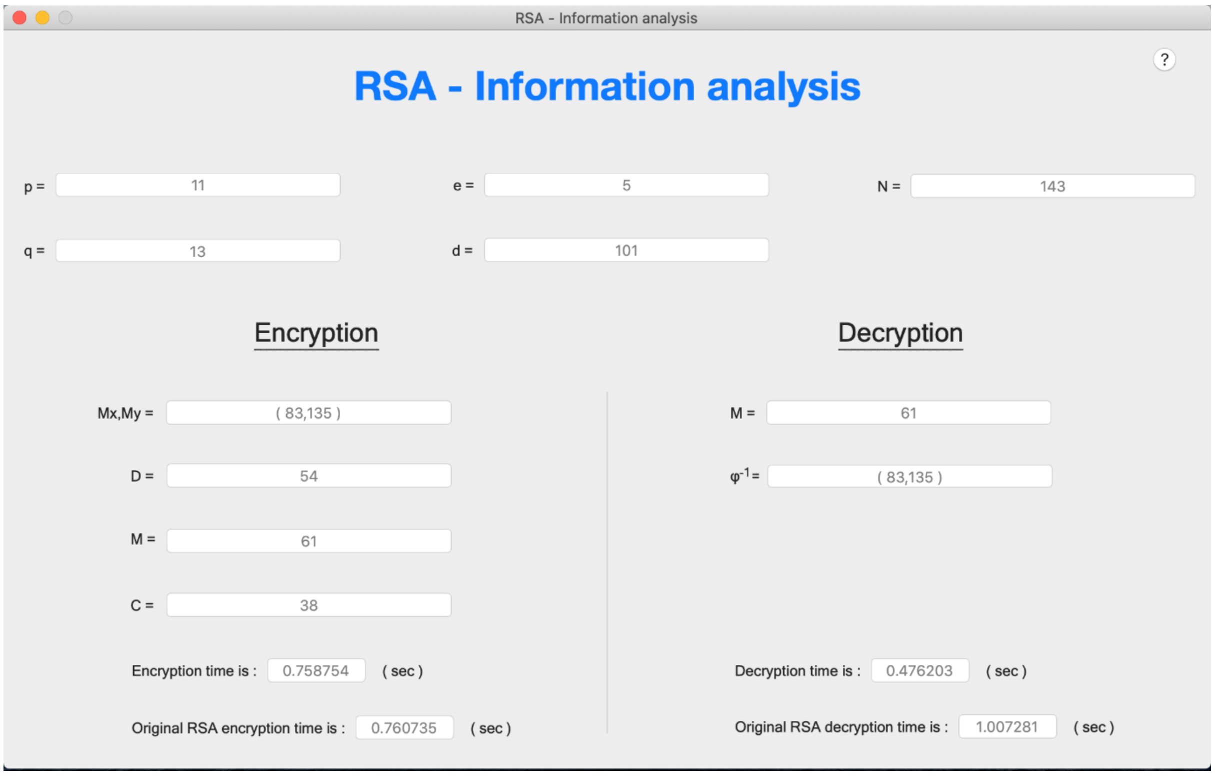
Figure 8 – An efficient and secure RSA–like cryptosystem exploiting Rédei rational functions over conics - class diagram

5.3. User interface

The RSA-like cryptosystem exploiting Rédei rational functions over conics is algorithm that works behind the Scenes. Therefore, we have chosen to create one window that going to display the information after the algorithm runs.

5.3.1. Background information window

After the original RSA and RSA-like cryptosystem runs and they are finished all calculations, this window will appear and show all values of parameters and the times of every step-in encryption and decryption algorithms for details.



RSA - Information analysis

?

RSA - Information analysis

p = 11
e = 5
N = 143

q = 13
d = 101

Encryption

Mx,My = (83,135)

D = 54

M = 61

C = 38

Encryption time is : 0.758754 (sec)

Original RSA encryption time is : 0.760735 (sec)

Decryption

M = 61

φ^{-1} = (83,135)

Decryption time is : 0.476203 (sec)

Original RSA decryption time is : 1.007281 (sec)

Figure 9 – Information analysis window

5.4. Testing plan

Because this is an algorithm, our test plan is to examine every step of the algorithm and see if the expected result is obtained.

Test #	Test subject	Expected result	Pass/Fail
1	p, q are prime numbers	The multiplication result of p, q denoted by N	Pass
2	Choosing integer e	$gcd(e, lcm(p+1, q+1) = 1$	Pass
3	Evaluate $d = e^{-1}(mod\ (lcm(p+1, q+1))$	The integer d should be positive and smaller than $lcm(p+1, q+1)$	Pass
4	Compute $D = \frac{M_x^2 - 1}{M_y^2}$	The integer D should be smaller than N and $(M_x, M_y) \in x^2 - Dy^2 = 1(modN)$	Pass
5	Compute $M = \Phi(M_x, M_y) = \left(\frac{M_x+1}{M_y}\right)(mod\ N)$	The parameterization was correct and the integer M smaller than N and positive	Pass
6	Compute $C = Q_e(D, M)(modN)$	The Ciphertext C encrypted to positive integer bigger than N	Pass
7	Compute $M = C^{\odot Pd}(modN)$	The Ciphertext C decrypted to integer M	Pass
8	Retrieve the plaintexts (M_x, M_y) by Φ^{-1} parameterization: $\Phi^{-1}(M) = \left(\frac{M^2 + D}{M^2 - D}, \frac{2M}{M^2 - D}\right)$ (mod N)	The original message (M_x, M_y)	Pass
9	Performance	The system will be able to withstand large loads of very large numbers calculations	Pass
10	Constants	Each constant will be tested for its literal value, the constant will have the same value without ambiguity	Pass

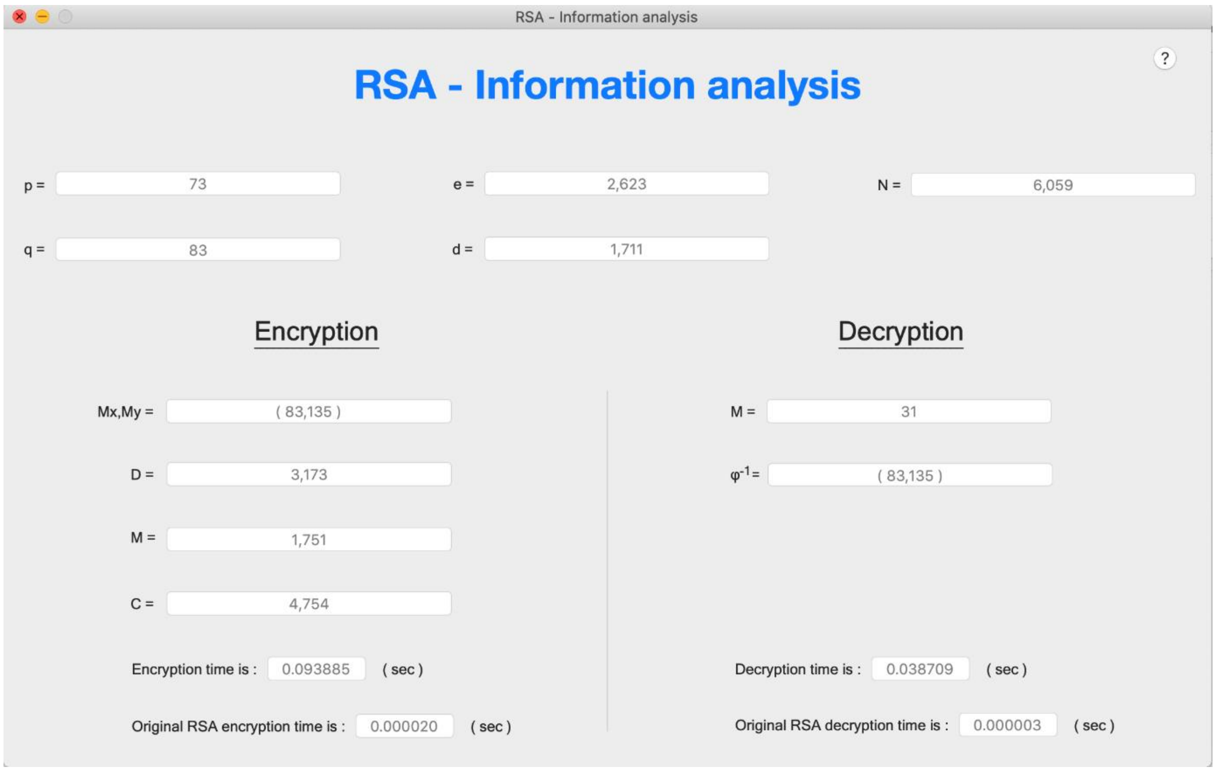
Table 1 – Testing plan

6. RESULTS AND CONCLUSIONS

6.1. Results

6.1.1. First example

Comparison between the original RSA and RSA-like cryptosystem.


RSA - Information analysis

RSA - Information analysis

p = 73
e = 2,623
N = 6,059
q = 83
d = 1,711

Encryption

Mx,My = (83,135)
D = 3,173
M = 1,751
C = 4,754
Encryption time is : 0.093885 (sec)
Original RSA encryption time is : 0.000020 (sec)

Decryption

M = 31
φ^{-1} = (83,135)
Decryption time is : 0.038709 (sec)
Original RSA decryption time is : 0.000003 (sec)

Figure 10 – Output result between the original RSA and RSA-like cryptosystem

In this experiment, we take two different algorithms – the original RSA and RSA-like cryptosystem and define the parameters – p, q double-digit to be $p = 73, q = 83$, and let to both algorithms to run. All the rest parameters are been calculated by each algorithm.

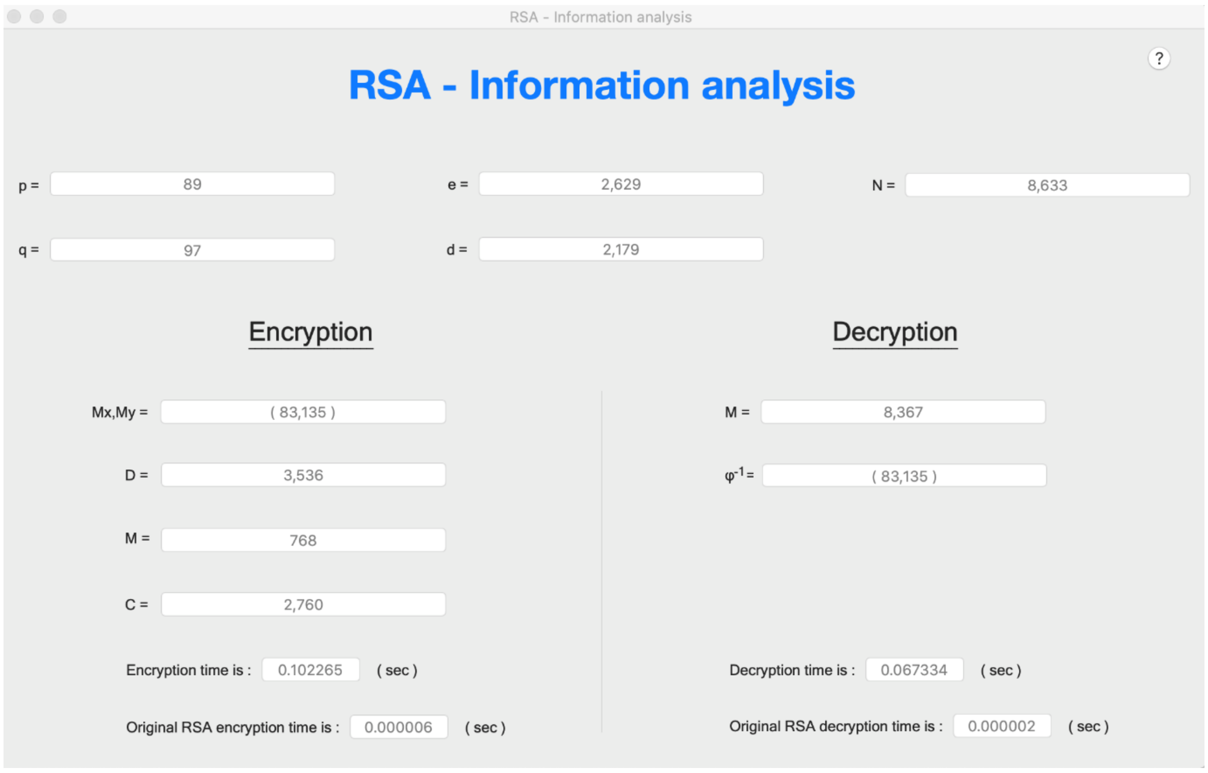
In this output result, we can see on the left side of the window, the encryption parameters calculation, and the time it took to calculate the encryption in the RSA-like cryptosystem and original RSA. In this experiment the original RSA was faster than the RSA-like cryptosystem, the original RSA encryption time is 0.000020 seconds, and the time for the RSA-like cryptosystem is 0.093885 seconds.

On the right side of the window, we can see the decryption parameters calculation and the time it took to calculate the decryption in the RSA-like cryptosystem and original RSA. In this experiment the original RSA was faster than the RSA-like cryptosystem, the original RSA encryption time is 0.000003 seconds, and the time for the RSA-like cryptosystem is 0.038709 seconds.

For a summary, in this example, the decryption of original RSA is faster than the RSA-like cryptosystem approximately by 1,290,200%.

6.1.2. Second example

Another comparison between the original RSA and RSA-like cryptosystem.


RSA - Information analysis

RSA - Information analysis

p = 89 e = 2,629 N = 8,633

q = 97 d = 2,179

Encryption

Mx,My = (83,135)

D = 3,536

M = 768

C = 2,760

Encryption time is : 0.102265 (sec)

Original RSA encryption time is : 0.000006 (sec)

Decryption

M = 8,367

φ^{-1} = (83,135)

Decryption time is : 0.067334 (sec)

Original RSA decryption time is : 0.000002 (sec)

Figure 11 – Output result between the original RSA and RSA-like cryptosystem

In this experiment, we take two different algorithms – the original RSA and RSA-like cryptosystem and define also the parameters - p, q double-digit to be $p = 89, q = 97$, but different than the example in 6.1.1. All the rest parameters are been calculated by each algorithm.

In this output result, we can see on the left side of the window, the encryption parameters calculation, and the time it took to calculate the encryption in the RSA-like cryptosystem and original RSA. In this experiment the original RSA was faster than the RSA-like cryptosystem, the original RSA encryption time is 0.000006 seconds, and the time for the RSA-like cryptosystem is 0.102265 seconds.

On the right side of the window, we can see the decryption parameters calculation and the time it took to calculate the decryption in the RSA-like cryptosystem and original RSA. In this experiment the original RSA was faster than the RSA-like cryptosystem, the original RSA encryption time is 0.000002 seconds, and the time for RSA-like cryptosystem is 0.067334 seconds.

For a summary, in this example, the decryption of original RSA is faster than the RSA-like cryptosystem approximately by $5{,}113{,}250\%$.

6.1.3. Comparison between first and second examples

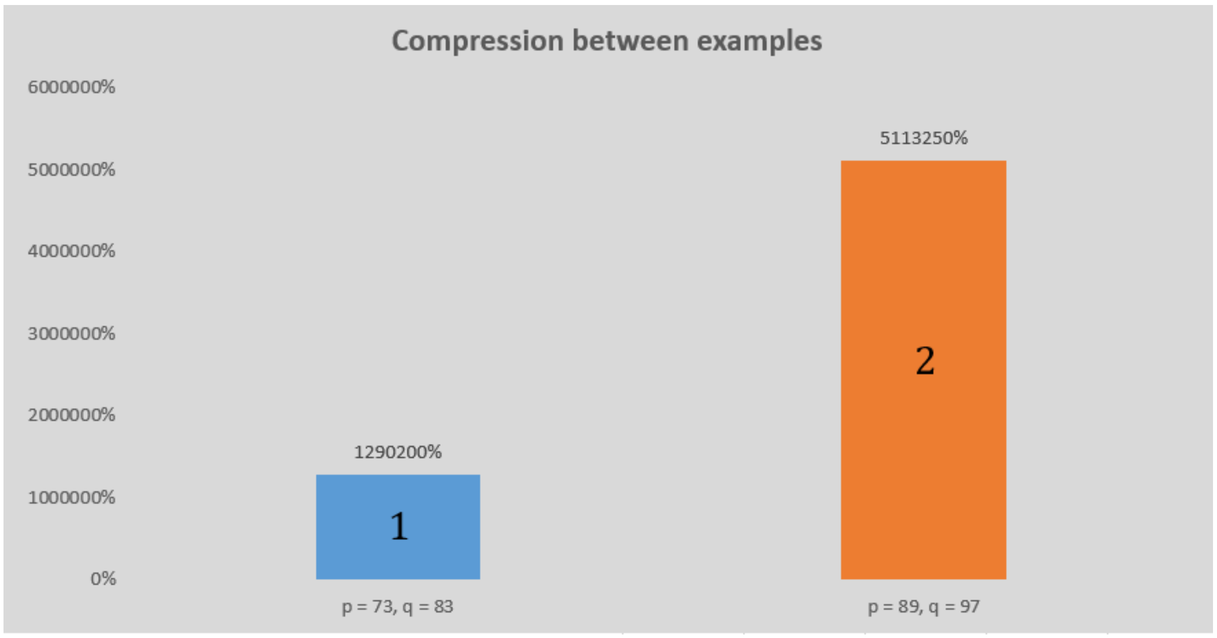
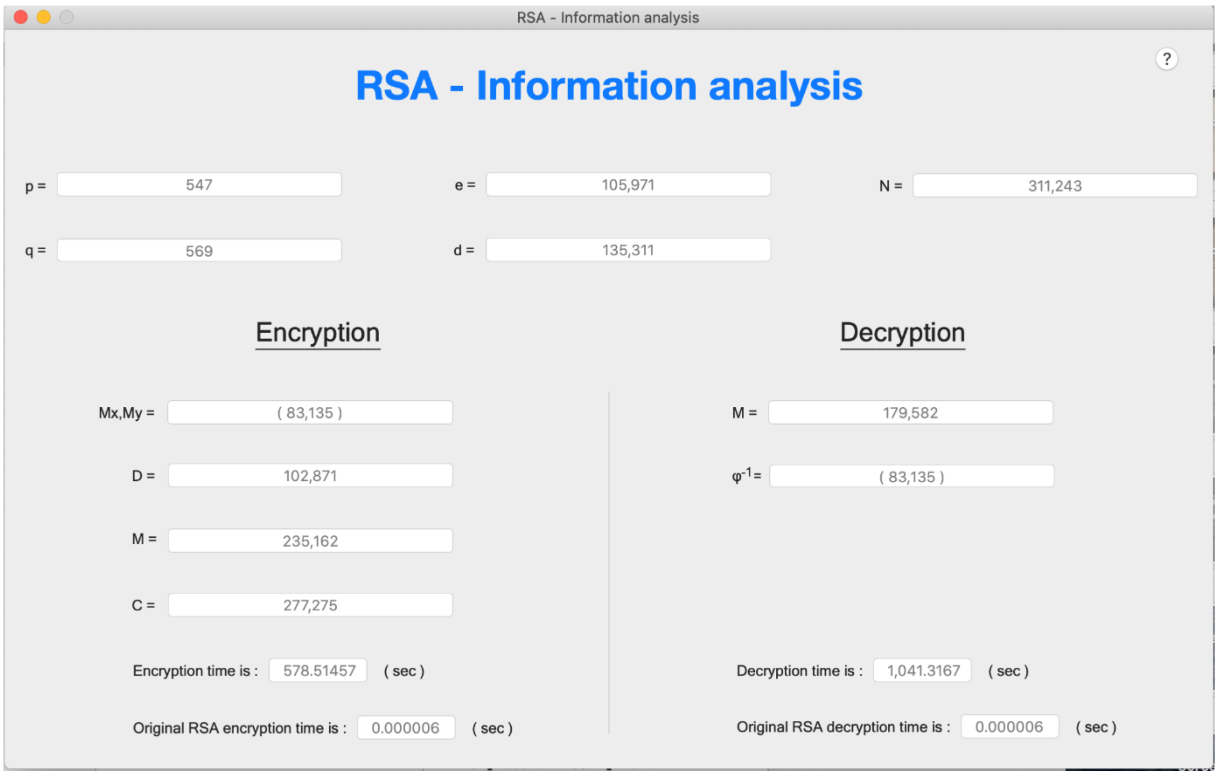


Figure 12 – Comparison between the decryption results of example one and example two

6.1.4. Third example

Comparison between the original RSA and RSA-like cryptosystem for 3-digit p and q.


RSA - Information analysis

RSA - Information analysis

p = 547
e = 105,971
N = 311,243
q = 569
d = 135,311

Encryption

Mx,My = (83,135)
D = 102,871
M = 235,162
C = 277,275
Encryption time is : 578.51457 (sec)
Original RSA encryption time is : 0.000006 (sec)

Decryption

M = 179,582
φ^{-1} = (83,135)
Decryption time is : 1,041.3167 (sec)
Original RSA decryption time is : 0.000006 (sec)

Figure 13 – Output result between the original RSA and RSA-like cryptosystem for 3-digit case

In this experiment we let both algorithms – the original RSA and RSA-like cryptosystem to run with 3-digit p and q parameters, $p = 547, q = 569$. We let both algorithms to run and we could see that the total running time that took to RSA-like cryptosystem much longer in comparison to the original RSA. The original RSA total time was 14.7 seconds and the total time for the RSA-like cryptosystem was 1,619.8 seconds, it is almost 27 minutes. The most of the time was spent on the decryption process.

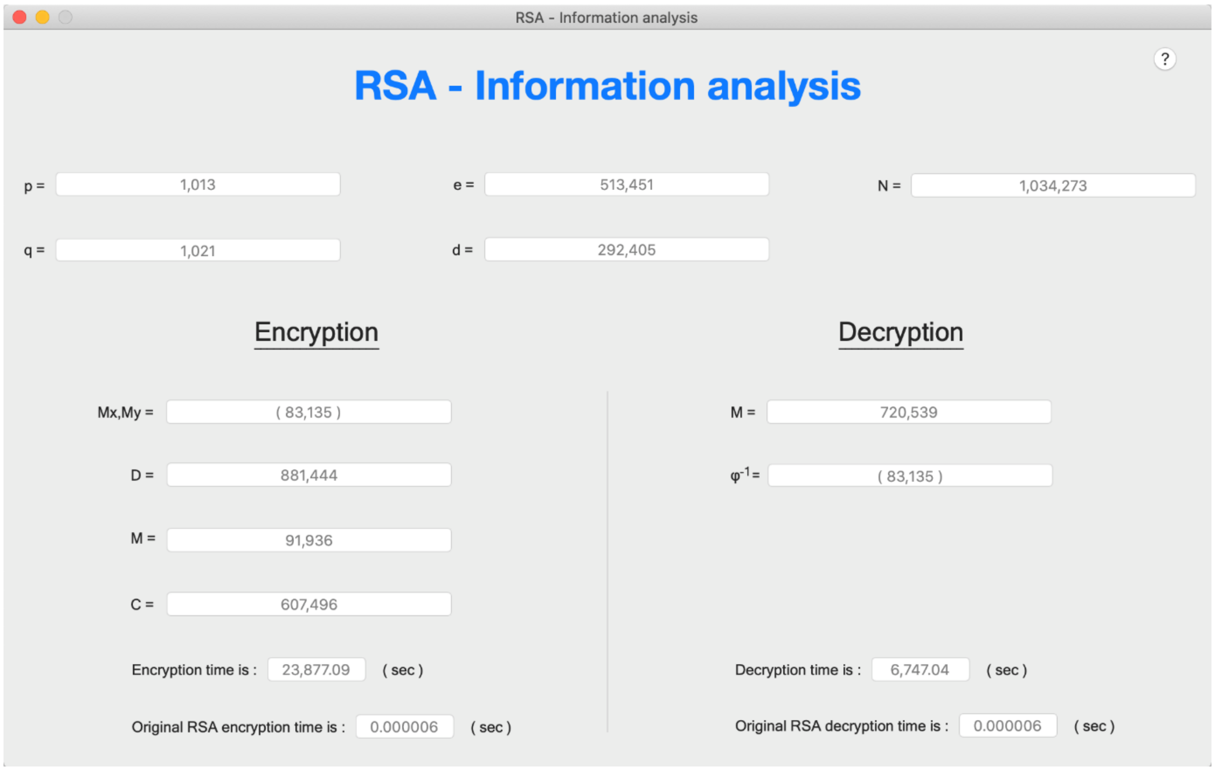
In this output result, we can see on the left side of the window, the encryption parameters calculation, and the time it took to calculate the encryption in the RSA-like cryptosystem and original RSA. In this experiment the original RSA was faster than the RSA-like cryptosystem, the original RSA encryption time is 0.000006 seconds, and the time for the RSA-like cryptosystem is 578.51457 seconds, it is almost 10 minutes.

On the right side of the window, we can see the decryption parameters calculation and the time it took to calculate the decryption in the RSA-like cryptosystem and original RSA. In this experiment the original RSA was faster than the RSA-like cryptosystem, the original RSA encryption time is 0.000006 seconds, and the time for the RSA-like cryptosystem is 1,041.3167 seconds, it is almost 17 minutes.

For a summary, in this example, the decryption of original RSA is faster than the RSA-like cryptosystem approximately by 17,355,278,334%.

6.1.5. Fourth example

Comparison between the original RSA and RSA-like cryptosystem for 4-digit p and q.


RSA - Information analysis

RSA - Information analysis

p = 1,013
q = 1,021
e = 513,451
d = 292,405
N = 1,034,273

Encryption

Mx,My = (83,135)
D = 881,444
M = 91,936
C = 607,496
Encryption time is : 23,877.09 (sec)
Original RSA encryption time is : 0.000006 (sec)

Decryption

M = 720,539
φ^{-1} = (83,135)
Decryption time is : 6,747.04 (sec)
Original RSA decryption time is : 0.000006 (sec)

Figure 14 – Output result between the original RSA and RSA-like cryptosystem for 4-digit case

In this experiment we let both algorithms – the original RSA and RSA-like cryptosystem to run with 4-digit p and q parameters, $p = 1{,}013, q = 1{,}021$. We let both algorithms to run and we could see that the total running time that took to RSA-like cryptosystem much longer in comparison to the original RSA. The original RSA total time was 9.34 seconds and the total time for the RSA-like cryptosystem was 30,624.04 seconds, it is almost 8.5 hours. The most of the time was spent on the encryption process.

In this output result, we can see on the left side of the window, the encryption parameters calculation, and the time it took to calculate the encryption in the RSA-like cryptosystem and original RSA. In this experiment the original RSA was faster than the RSA-like cryptosystem, the original RSA encryption time is 0.000006 seconds, and the time for the RSA-like cryptosystem is 23,877.09 seconds, it is almost 6.6 hours.

On the right side of the window, we can see the decryption parameters calculation and the time it took to calculate the decryption in the RSA-like cryptosystem and original RSA. In this experiment the original RSA was faster than the RSA-like cryptosystem, the original RSA encryption time is 0.000006 seconds, and the time for the RSA-like cryptosystem is 6,747.04 seconds, it is almost 1.8 hours.

For a summary, in this example, the decryption of original RSA is faster than the RSA-like cryptosystem approximately by 112,450,666,667%.

6.1.6. Comparison between third and fourth examples

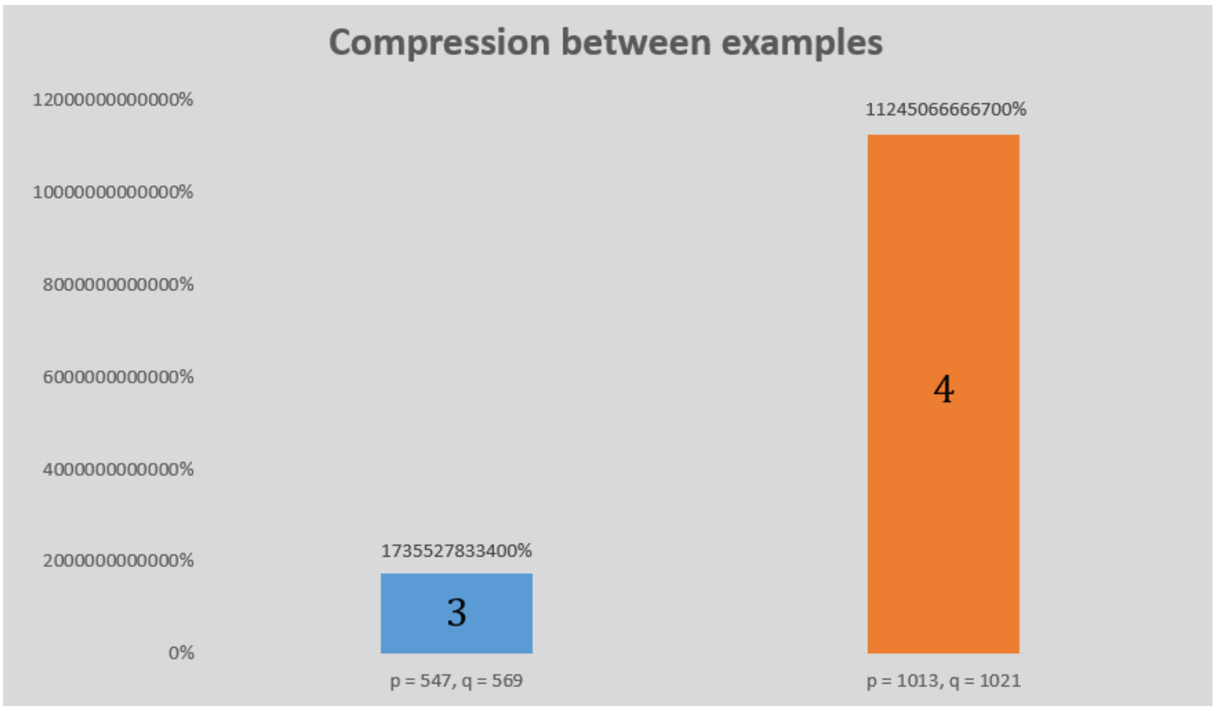


Figure 12 – Comparison between the decryption results of example three and the result of example four

6.2. **Conclusions**

At the experiment stage, we have created many cases with different parameters, trying to understand better how the original RSA and the RSA-like cryptosystem responds to the various parameter's values. We learned that the RSA-like cryptosystem affected by the size of the parameters $p, q,$ as much as this parameter will bigger the decryption running time will be longer. After deep thinking, we concluded that the original RSA decryption time will be faster than the RSA-like cryptosystem. The number of iterations effects on the RSA-like cryptosystem running time and this number can not be reduced. The number of iterations depends on the value of e.

The weak point of the encryption and decryption procedures is the calculation of the factorial. The calculation of the factorial of huge numbers takes a lot of time and does not has a shortcut to calculate it. The calculation of factorial appears in Rédei rational functions that used in the encryption and decryption procedures.

Another weak point is that in every one iteration of the decryption process, the Original RSA needs one operation of multiplication as part of the raising to the power, and the RSA-like cryptosystem needs to calculate five complicated operations to be an alternative to the raising to the power of the Original RSA.

Our assumption can not be achieved with the time results that we saw in hundreds of running tests of the Original RSA and RSA-like cryptosystem.

7. REFERENCES

[1] D. Coppersmith, M. Franklin, J. Patarin and M.Reiter, *Low – exponent RSA with related messages*, Advances in cryptology – EUROCRYPT'96, Springer, 1996, p. 1-9.

[2] Z. Chen, X. Song, j. Li, *A public - key cryptosystem scheme on conic curves over the ring* z_n, Computer Science and Information Technology College Zhejiang Wanli University, 2006, p. 156 - 160.

[3] S. Barbero, U. Cerruti and N. Murru, *Solving the Pell equation via Rédei rational functions*, Fibonacci Quarterly, 2010.

[4] N. Lal, A.P Singh, S. Kumar, *Modified Trial Division Algorithm Using KNJ-Factorization Method To Factorize RSA Public Key Encryption*, Wireless Communication and Computing Indian institute of Information Technology, arXiv:1501.02365.

[5] E. Bellini and N. Murru, *An efficient and secure RSA-like cryptosystem exploiting Rédei rational functions over conics*, arXiv:1511.03451v2 , 2016.

[6] S. Padhye, *A public key cryptosystem based on Pell equation*, IACR cryptology ePrint archive, 2006, p. 191.