An Enhanced Cryptosystem based on Shorter Keys and New Security Component

Alaa Alslaity
School of Electrical Engineering and Computer Science
800 King Edward Ave.
Ottawa, Ontario, K1N 6N5, Canada

Thomas Tran
School of Electrical Engineering and Computer Science
800 King Edward Ave.
Ottawa, Ontario, K1N 6N5, Canada

ABSTRACT
Public-key (or asymmetric key) algorithms are the most dominant cryptographic algorithms in the cryptography era. RSA, in particular, is a good example of algorithms that belong to this category which is used for many applications and its security is beyond doubt. The key size of RSA is an important factor that determines its robustness. As the computational powers evolve, the need to increase the security of cryptosystems becomes essential to achieve more robustness against cryptanalysis attacks. In the literature, the commonly used solution to achieve this goal is to increase the key size. However, in practice, keeping increasing the key size is not feasible and not unlimited; increasing the key size demands more computational power. The increase in computational power makes it hard (or impossible) to conduct the encryption process in some environments such as smart cards. Also, it should be taken into consideration that the dramatic increase in computational capabilities of new machines improves their abilities to break encryption keys. Therefore, an alternative to increasing the key size is essentially needed. In this paper, we propose a new approach for encryption that is based on RSA main algorithm while using more encryption keys with smaller sizes in addition to extra security information component (known as the *Security Card*, *SeCa*). Our results show that the use of multiple-shorter keys with the *SeCa* component produces significant improvements in the performance of RSA algorithm in terms of increasing security and reducing the computation overhead by decreasing encryption and decryption times.

General Terms
Security, Algorithms, Encryption.

Keywords
Security, RSA, Encryption, Decryption, Cryptanalysis Attack, Cryptography, Asymmetric Key, Key Size

1. INTRODUCTION
Security is the mechanism of protecting information and services from unauthorized access, change or destruction[1]. In networking, the security is based on cryptography (a word originated in Greek, it means "hidden or secret" [2]. Cryptography is "the science or study of the techniques of secret writing, especially code and cipher systems, methods, and the like" [3]. In other words, cryptography is the science of protecting information represented as a plaintext (Original, unencrypted message) by converting it into a ciphertext in which the data is unintelligible or unreadable [4]. The main goal of cryptography is to represent the information in a format that can conceal the message from casual readers.

Due to the huge increase of information use, the need for protecting and ensuring the invulnerability of this information becomes essential. Network-based applications, like online banking and e-commerce, use sensitive information like credit/debit card numbers and personal information. Therefore, they need an end-to-end secure connection to ensure the *CIA* triad (i.e. Confidentiality, Integrity, and Availability) [1]. Cryptographic algorithms are used to encrypt/decrypt such sensitive and critical information to ensure security.

Encryption algorithms can be classified into two major groups: Symmetric key (a.k.a secret-key) such as AES [5] and DES [6], and Asymmetric key (a.k.a Public-key) encryption such as RSA [7], and DSA [7]. The security of both symmetric and asymmetric encryption algorithms depend directly on a key which can be numeric, alphanumeric, or special symbol [1]. The secrecy and length of the key play substantial role in the strength of the encryption algorithm [8].

RSA cryptosystem is extensively used in the implementation of public key infrastructures for the following reasons: it is easy to implement, easy to understand and achieves high levels of security. The strength of RSA depends solely on the length of the key used for decryption. Therefore, for the RSA to be robust against cryptanalysis attacks, long keys that are difficult to factorize are needed. In mathematics, factorization problem states that: given a very large number, it is quite impossible to find the two prime numbers whose product is the given number [9].

The 1024-bit RSA key is proved to be robust against cryptanalysis attacks due to the difficulty to factorize it into its basic prime factors. However, as computation power increases and the factorization methods improve, this will change the fact that a 1024-bit key size is really secure, and impose the need to use longer keys (e.g. 2048 bit, 3072 bit or more) to perform encryption [10].

The main question that arises in this context is: When to make a transition to a longer key size? Moreover, when will using larger key sizes be stopped? In this paper, the researchers investigate the use of multiple shorter keys along with new security component called *SeCa* to achieve more robustness against cryptanalysis attacks by increasing the workload required by the cryptanalyst to decrypt the ciphertext and get the corresponding plaintext.

The rest of this paper is organized as follows: Section2 states the problems and limitations that motivate us to propose our approach. Section3 provides an overview of the RSA cryptosystem. The related works are discussed in Section 4. Section 5 describes our proposed approach in details. The evaluations of the proposed approach along with the results

are illustrated in Section 6. Finally, Section 7 concludes the paper and provides future directions.

2. PROBLEM STATEMENT

As mentioned above, the increase of computation power changes the properties of which RSA keys can be considered secure. Previously, 512-bit RSA algorithms were considered reasonably secure, Ten years ago, this limit was raised to 1024–bit. RSA Laboratories [10] reported in 2003 that the recommended minimum RSA key size needed to protect data till 2030 is 2048 bits comparing to 1024 bits until the year 2010. Moreover, it should be taken into consideration that efficient machines (such as TWIRL) are continuously improving in terms of computational powers to be capable of breaking encryption keys [11]. Therefore, the 1024-bit RSA key size will eventually no longer be secure, just like the previous key sizes (512-bit RSA).

From the above discussion, it is obvious that the larger the RSA key size, the more secure the algorithm is. However, the longer the key size, the more the computational overhead. A rough approximation is that doubling the length of the key size will increase public key operations by a factor of four, private key operations by a factor of eight and key generation operations by a factor of sixteen [12]. Public key operations are less sensitive to key size increase because the public exponent is selected to be fixed, while in private key operations, the length of the private exponent increases proportionately. This is due to the fact that the public key is always generated randomly or picked to be small. Thus, the time for generating RSA key-pair is almost equal to the time of determining the RSA private key.

The limitations above motivated us to propose a more efficient solution to address the problems associated with increasing the RSA key size. The proposed solution will be discussed in Section 5.

3. AN OVERVIEW OF RSA

RSA cryptosystem proposed by Rivest-Shamir-Adleman in 1978 is considered as one of the most known public key cryptosystems [7]. It is typically used for key exchange, digital signatures, or encryption of blocks of data [1].

RSA uses a pair of different but related keys, one for encryption, called the public key, and another for decryption called private key. The public key is disclosed to the public while the private key is kept secret because it is used to decrypt the encrypted message.

The working of RSA is described as follows [7]: First, the message (known as the plaintext) should be represented as integer to make it ready for encryption. If the message is too long, it is divided into blocks, and each block is represented as integers. After that, these integers are encrypted using the encryption key (the public key) to produce the ciphertext C. At the other end of communication, i.e. at the recipient side, the ciphertext is decrypted using the recipient's private key to retransform or retrieve the message into its original presentation (i.e. plain text).

RSA involves three basic operations: Key Generation, Encryption, and Decryption. Figures 2, 3 and 4 show the major steps of these three operations, respectively [9].

***Step* 1**: Randomly generate two large prime numbers, p *and* q.

***Step* 2**: Compute $N = p * q$.

***Step* 3**: Compute the Euler's totient function φ of N, $\varphi(N) = (p-1)(q-1)$

***Step* 4**: Select a random integer e, such that $1 \leq e \leq \varphi(N)$ and $GCD(e, \varphi(N)) = 1$

***Step* 5**: Find the unique multiplicative integer d such that $1 \leq d \leq \varphi(N)$, $e.d = 1 \, mod \, \varphi(N)$

***Step* 6**: (e, N) is the public key. (d, N) is the private key.

Fig. 1: RSA Key Generation

***Step* 1**: Sender obtains recipient's public key (e, N).

***Step* 2**: Sender encrypts the message m to generate the ciphertext C, where $C = m^e \, mod \, N$

Fig. 2: RSA Encryption

***Step* 1**: Input: C, d, N

***Step* 2**: Recipient decrypts the cipher C to retrieve the original message M, where $M = C^d \, mod \, N$

Fig. 3: RSA Decryption

4. RELATED WORK

RSA is widely addressed in the literature, and many researchers have proposed different mechanisms and approaches to enhance the original RSA. In this section, we review some RSA variants that have been proposed to overcome the weaknesses of RSA.

A Hybrid Encryption RSA (HE-RSA) that is based on RSA has been proposed in [13]. The primary goal of this approach is to increase the security of data in cloud servers while minimizing time and cost consumption. HE_RSA applies dual encryption process to prevent common attacks against RSA algorithm. Moreover, it increases the number of key generation exponents to three in order to increase the security of the original RSA.

Aboud et al. [14] have introduced an enhanced RSA which employs the general linear group of order h with randomly selected values from the ring of integer mod n, where n has the same definition as in RSA (i.e. the product of two large prime numbers). The main difference between RSA and Aboud's approach is in the key generation process, particularly, in calculating (n). Other steps are kept as they proposed by the original RSA. Hence, the computational cost of this approach does not vary significantly from the original RSA.

In [15] The Computational Dependent-RSA problem which is derived from the RSA assumptions has been presented. Then, a variant of this problem has been proposed, namely the Decisional Dependent-RSA Problem. Finally, the researchers applied them to provide an asymmetric encryption scheme called the Dependent-RSA (DRSA) which aims to provide semantic security to the original RSA.

LEE Public Key Algorithm[4] is an implementation of RSA. Rather than sending the public key (e) value, LEE scheme uses two public keys and some mathematical logic. The reason behind this modification is because if an attacker got the e value, she can infer the d value and hence she can decrypt the message. So, in the proposed scheme, the two public keys are sent separately as to make it more difficult for the attacker to get knowledge about the key. Although this scheme overcomes the original RSA limitations in terms of security and vulnerability to brute-force attacks, it leads to more communication overhead. Hence, it is useful only for systems in which the security is more important than speed.

Some researchers rely on algorithms combination to provide more efficient encryption algorithms by making use of the strong points and avoiding the weaknesses of these algorithms. In [16] a novel method had been proposed, it combines the most popular algorithms; RSA and Diffie-Hellman to provide more security against cryptanalysts. The two prime numbers that are generated by RSA, e and d, are used as inputs to Diffie-Hellman in order to generate the session key K which will be used for encryption and decryption. Although this algorithm increases the security, its usability is demonstrated with very few concepts.

Another hybrid algorithm had been proposed by Deshmukh and Patil [17]. It consists of two parts; one for key exchange which is based on Diffie-Hellman, while the other one depends on RSA approach for encryption and decryption. The proposed algorithm is more secure, but it needs more time in execution[18].

Al-Hamami et. al [19] proposed an enhanced version of RSA that depends basically on using multiple prime numbers instead of two, as in the case of the original RSA. The proposed algorithm is very similar to the RSA but, unlike RSA, it uses three prime numbers with smaller sizes instead of two prime numbers. The goal of this third number is to increase the factorization complexity which leads in turn to increase the security of the algorithm. The experiments results show that using more than two prime numbers with smaller sizes enhances the RSA in terms of increasing the factorization complexity and decreasing the processing time (i.e. key generation, encryption and decryption processes).

In [20] the researchers proposed a new ciphering technique which uses the concept of genetic algorithms in conjunction with RSA to provide more security against cryptanalysts. The proposed approach is a combination of symmetrical and asymmetrical systems; the symmetrical system, namely the Genetic Algorithm Inspired Cryptography, uses genetic optimization to generate the keys, while the RSA (asymmetrical system) is used to make the key more complex.

These are some of the proposed schemes that aim to enhance the security and efficiency of the conventional RSA. There are many other schemes like i-RSA [21], Quantum key Distribution [22], Modified RSA Cryptosystem Based on Offline Storage and Prime Number [23], Personal Information Protection Approach Based on RSA [24], etc.

5. PROPOSED APPROACH

This section describes the proposed encryption scheme which is called the ***Two-Key RSA*** algorithm (***2K-RSA***). The main procedures of the proposed algorithm are similar to RSA. However, 2K-RSA uses new features to enhances security; these features include dividing the message M by the sender into two parts m1 and m2, and using two keys of shorter sizes (as compared to the original RSA keys), each key is used to encrypt part of the message. In addition to other security information component referred to as *Security Card* (*SeCa*). Moreover, the message *M* is stored in a 2-dimentional array data structure. The goal of using *SeCa* is to make it harder for the cryptanalyst to decrypt the message by increasing the complexity of not only retrieving the original text but also to retrieve it in its arranged format. Table 1 shows the notations used in this paper.

Table 1: Notations used in 2K-RSA

Notation	Description
M	Original message (Plain text)
m1, m2	The two partitions of M
k1, k2	Key 1 and key 2 that are used to encrypt m1 and m2
c1, c2	The ciphertexts resulted from encrypting m1and m2
C	The Whole (Final) cipher.
SeCa	The Security Card component which contains information used along with the private key to decrypt the ciphertext.
SI	Segment Information
SIK	Segment Information key, Used to encrypt the SI
SIC	Segment Information Cipher; resulted from encrypting SI.
SIS	Segment Information Size
SICP	Segment Information Cipher Position, The position of SIC in the cipher C.
KIDi	A unique ID for the set of keys (ki)

The major contributions of *2K-RSA* encryption scheme are:

- The use of a new special secure information component called *SeCa*.
- The division of the message *M* into two parts and the use of multiple shorter keys (2 keys), one for each part of *M*.
- The use of a 2-dimentional array to represent the message *M*.

The following subsections describe in details the main procedures of the *2K-RSA*.

5.1 *SeCa* Construction:

The *SeCa* is a major concept and an essential step of our proposed scheme. It contains details about the encryption process, keys, and message segments. Following are the components of *SeCa*:

- *Segments Information* (*SI*): contains the original message length (L) and the lengths of the two ciphers (originating from encrypting *m1* and *m2*) and embedded within the encrypted message.

- *Segments Information Size* (*SIS*): the size of *SI*.
- *Segments Information Cipher Position* (*SICP*): the position of the SI Cipher (*SIC*) within the encrypted message, C.
- *Segments Information Key* (*SIK*): the key used to encrypt the (*SI*).
- Finally, *SeCa* contains two unique Identifiers *KID1* and *KID2*, where each key identifier is assigned to a key pair and used to encrypt one part of the message.

The general format of the *SeCa* is defined as the following:

$$SeCa = \{SIS; SICP;\ SIK; \{KID1, KID2\}\}$$

In the context of the RSA algorithm, and before the beginning of the encryption/decryption processes, the communicating parties should agree on the key distribution mechanisms and the key size to be used for encryption and decryption. The same procedure applies in our *2K-RSA* scheme beside exchanging additional information stored in the *SeCa*. When two parties intend to communicate, they construct and exchange the *SeCa* and public keys in advance, so as to facilitate the communication later. Alternatively, the two parties can also establish a private connection over an insecure channel to send the *SeCa* in a prior. Thereafter, the process is completed as in the public-key systems[7]. Fig. **4** shows the general steps of constructing the *SeCa*.

***Step* 1**: Generate two keys ($k1$ and $k2$)

***Step* 2**: Assign a unique *ID* for each key *KID1*, *KID2*.

***Step* 3**: Generate the Segments Information key, SIK.

***Step* 4**: Determine the position of *SIC* within the whole cipher text, i.e. the value of *SICP*.

***Step* 5**: Determine the size of the Segment Information (*SIS*).

Fig. 4: SeCa Construction

5.2 *2K-RSA* Operations.

Our proposed 2K-RSA algorithm, as the case of RSA, involves three main operations: key generation, encryption, and decryption.

When a sender (Alice) intends to communicate a message M with a receiver (Bob), two pairs of keys *k1* and *k2* are generated. Alice uses Bob's both public keys to encrypt the message (as will be illustrated below), while Bob uses his two private keys as well as the *SeCa* to decrypt the resulted ciphertext and retrieve the original message M back.

5.2.1 The Encryption Operation:

At Alice's side, the encryption process is performed as the following: a message *M* to be communicated is loaded into a 2-dimentional array [r*c], where *r* is the number of rows and *c* is the number of columns. Then it is divided into two parts m_1 and m_2 where the size of each part (m_i) is computed as the following:

$$Size\ of\ mi = \frac{r}{2} \times c$$

Alice uses Bob's public keys k_1 and k_2 to encrypt m1 and m2 to get c_1 and c_2, respectively. The size of k_1 and k_2 is always smaller than the key used in RSA.

The lengths of the resulting ciphers L(c_1) and L(c_2) are computed and encrypted together along with the length (L) of M using SIK to get the Segment Information Cipher, *SIC*. Finally, the whole ciphertext *C* is constructed by concatenating c_1, c_2, and *SIC*, stored in an array according to the predefined *SICP* parameter and then sent to Bob.

Fig. **5** represents a pseudocode for the general steps of the encryption process for the *2k-RSA.*

The extra security of the encryption operation is achieved in particular by the way the whole cipher C is constructed, which depends on the *SICP* that is determined by the sender in advance and sent to the recipient as part of the *SeCa*.

Step* 1**: Load ***M into 2-D array.
Step* 2**: Divide ***M into two parts (***m*₁**, ***m*₂**).
***Step* 3**: Using k_1 and k_2, encrypt m_1 and m_2 to get c_1 *and* c_2 respectively.
***Step* 4**: Prepare *SIC* by:
a. Find Lengths of c_1 and c_2, ($L(c_1)$, $L(c_2)$).
b. Encrypt L(c1), L(c2) and L(M) using *SIK*.
***Step* 5**: Construct the Whole cipher (*C*) such that:

$$C = [c1, c2, SIC]$$

***Step* 6**: Send *C* to recipient.

Fig. 5: 2k-RSA Encryption Process.

5.2.2 The Decryption Operation:

On the recipient side (Bob), since he has the key-pairs and the SeCa, and upon receiving the whole ciphertext *C*, he first starts retrieving the *SI*. To do so, he uses the *SIP* and the *SIS* that are known a priori, so he can decrypt *SI* using the *SIK*. Having the *SI* decrypted, he can now retrieve *L*, L(c_1) and L(c_2) so as to reserve a square array and start decrypting c_1 and c_2 to get m_1 and m_2. Then, he can store m1 and m2 in the array to retrieve *M*. A step-by-step clarification of the decryption process is described in Fig. **6**.

***Step* 1**: Using SIK, Bob's retrieve the Segment Information.
***Step* 1**: Retrieve L(c1), L(c2) and L(M).
***Step* 1**: Using k_1 and k_2, decrypt c_1 and c_2 to get m_1 and m_2 respectively.
***Step* 1**: Store m_1 and m_2 in 2-D array to retrieve M.

Fig. 6: 2k-RSA Encryption Process

The cryptanalyst needs much extra time and trails to find *SI*. She does not know its position in the ciphertext nor its size, so the only way left for her is to try decrypting the ciphertext byte by byte, which is clear how much time and trails she needs. Even if she can find and decrypt *SI*, she still needs to restore *M* from both m_1 and m_2 in their correct order. This, in turn, increases the effort and time on the cryptanalyst.

To summarize, the security of the decryption operation relies on the cipher *C* itself. Since *C* now is a composite cipher (i.e.

contains several sub-ciphers: c_1, c_2, and *SIC*) and it results from applying multiple encryption keys, it cannot be decrypted using the traditional methods of attacks that the cryptanalyst uses in RSA. The cryptanalyst needs to find the position of the SIC first which takes her numerous extra steps and trials, after that she still faces the challenge of knowing c_1 and c_2 to be able to decrypt each cipher and get the whole message *M*.

6. EVALUATION AND RESULT ANALYSIS

This section discusses the experiments and results of our encryption scheme *2K-RSA*. We perform experiments using two keys of size 512-bits each to encrypt m_1 and m_2 and compare our results against using one key of size 1024-bit in the RSA scheme, in terms of encryption and decryption times. The results show that the additional processes involved in the *2K-RSA* (including the construction of the *SeCa* and the use of the 2-dim array to store the plaintext and the ciphertext) do not increase the time complexity comparing to RSA. In the meanwhile, our scheme achieves higher security and less processing time.

The next two subsections discuss the experimental environment and the experimental results, respectively.

6.1 Experimental Environment

All performance measurements are conducted on an Intel core i5 machine, with the following properties:

- Hardware Configuration:
  - 2.5 GHz CPU.
  - 6.0 GB RAM.
- Software Configuration:
  - Windows 10.
  - Java language with JDK 7.0 for the implementation of RSA and *2K-RSA* encryption, decryption and key generation algorithms.

6.2 Experiments and Results Analysis

We perform several experiments using two keys of the same size (two 512-bit keys vs. one 1024-bit RSA key) and two keys of different size (256-bit and 768-bit keys vs. one 1024-bit RSA key). The results of the first experiment are illustrated in Table 2 and plotted in Figures 5, 6 and 7.

The results show that the time required to generate 1024-bit RSA key is much longer than the time required to generate two 512-bit keys. For example, with a message of size 700 byte, the time for generating 1024-bit RSA key is four times longer than the time for generating two keys of size 512 (6495/1500 ≅ 4). This is because, in the case of 1024-bit, the modulus N, where (N = p×q) is equal to 1024-bit, and p and q are two primes of a size 512-bit each. That is, the time needed to generate 1024-bit RSA key-pairs is almost equal to the time of finding and generating two 512-bit prime numbers. However, in *2K-RSA* (where two 512-bit keys are used), the time needed to generate two keys of size 512-bit is equal to the time of generating two prime numbers of size 256-bit each. Accordingly, generating two 512-bit keys requires less time than generating one 1024-bit key.

Table 2: two 512-bit keys (2k-RSA) vs. One 1024-bit key (RSA)

Message Size (Byte)	Time (millisecond)					
	Key Gen.		Encryption		Decryption	
	RSA	2K-RSA	RSA	2K-RSA	RSA	2K-RSA
700	6495	1500	19	11	2134	539
1000	7668	1645	23	14	2874	779
1500	7890	1732	36	20	4359	1182
2000	8970	1763	47	26	5772	1658

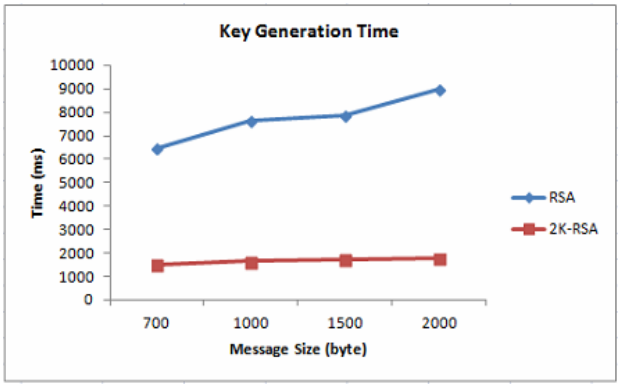

Fig. 7: Key Gen. Time for RSA vs. 2K-RSA

Moreover, the results show that doubling the key size leads to an increase of the encryption time by a factor of four (19 ms for1024-bit compared to about 5 ms for each 512-bit key). These results support the feasibility of our *2K-RSA* proposed approach which uses more, but shorter keys to reducing the computational overhead associated with encryption and decryption operations. Our results are consistent with the results of other researches [10].

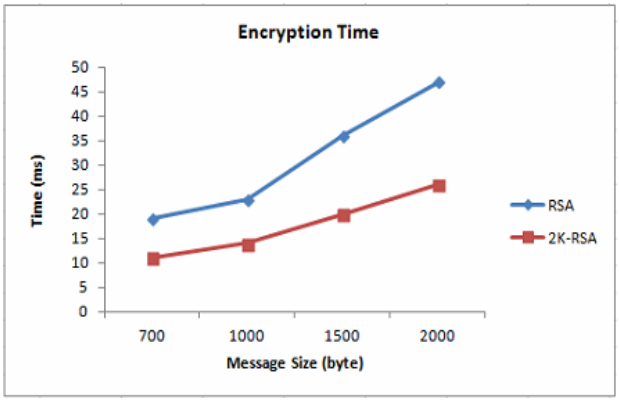

Fig. 8: Encryption Time for RSA vs. 2K-RSA

By examining the results of Table 2, it can also be inferred that the public-key (encryption) operations take much shorter time than the private-key (decryption) operations; That is, 19 ms compared to 2134 ms for a 1024-bit key, and 11 ms compared to 539 ms for a 512-bit key for a 700-byte message size. This is because the encryption exponent e is selected to be small (e = 3, 17, or 65537 are commonly used) and remain fixed when the modulus *N* is increased, whereas the private exponent d increases proportionally. In this case, d is roughly the same size as *N*.

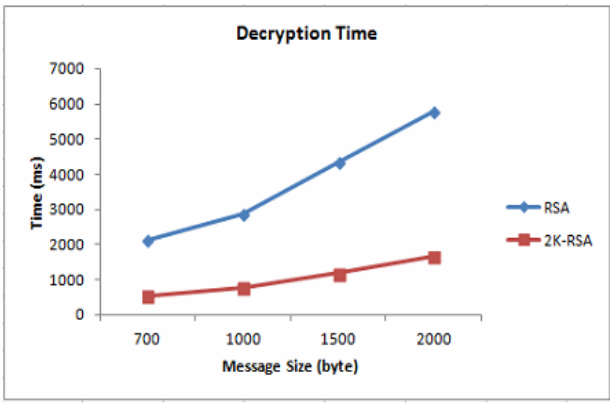

Fig. 9: Decryption Time for RSA vs. 2K-RSA

Other experiments are applied on different file sizes (1000, 1500 and 2000 bytes) using two keys of size 512 bit each. The results are all consistent with what has mentioned above. In addition, as the file size increases, the time required for key generation, encryption and decryption increase as well.

To examine the impact of using two keys to encrypt a message *M* that is subdivided into two parts, the experiments are extended as the following: *M* is divided into two unequal size parts, p_1 and p_2, where $p1 = \frac{1}{3}$ L and $p2 = \frac{2}{3}$ L and two keys, k_1 and k_2, of different sizes is generated, where k1= 256-bit and k_2=768-bit, then p_1 and p_2 are encrypted using $k_1$and k_2, respectively. The results of this experiment are illustrated in Table 3. From these results, it can be concluded that using two keys of different sizes still achieves better results compared to using a single key in terms of decreased encryption and decryption times. Furthermore, comparing Table 2 to Table 3, it can be noticed that with the same message size (i.e. 1000 byte), the use of two 512-bit keys is proved to be superior to the use of 256-bit and 768-bit keys in terms of reducing key generation, encryption and decryption time.

Table 3: Two keys of different sizes (768-bit and 256-bit 2k-RSA) vs. One 1024-bit key (RSA)

	Time (millisecond)			
Message Size (Byte)	**Key Size** (bit)	**Key Gen.**	**Encryp.**	**Decryp.**
1000	1024	7668	23	2874
666	768	4925	16	1230
333	256	241	6	86

7. DISCUSSION

This paper addresses the issue of increasing key size as a solution to enhance the security of cryptosystems. It discusses the impact of using two keys instead of one key for the encryption and decryption processes. As shown in the previous section, the results of the experiment show noticeable benefits gained from using multiple keys in terms of time consumptions and security enhancement.

We can conclude that, In general, using two keys instead of one will enhance the performance of the RSA by:

- Decreasing processing time (i.e. encryption, decryption and key generation time).
- Increasing the security against cryptanalysts.

These inferences are approved by other researchers. AlHamami et. al. [10] found that using three prime numbers in the composition of the private and public keys will enhance the encryption method by increasing the difficulty of the factorization process and decreasing the processing time [19].

In addition to the previous points, it can be inferred from the results that using two smaller keys with the same size is better than using two keys with different sizes in term of time complexity.

Our results show that using the additional component in conjunction with the basic components of RSA will increase the algorithm's security and will not increase the time complexity of RSA.

The contribution of this work can be summarized in the following points:

1. Enhancing the performance of RSA in term of time consumption by using two smaller keys instead of one larger key.
2. Improving the security of RSA by applying new component (*SeCa*) as well as dividing the message into two parts.
3. Showing that using two smaller keys with the same size overcomes the use of two keys with different sizes. However, using two keys with different sizes is better than using one key.

It is worthwhile to mention here that although our results show that the use of additional component as well as two pairs of keys will increase the performance of the RSA in terms of increasing security and decreasing time complexity, more experiments are still needed to make sure whether or not these results are true in all cases.

8. CONCLUSION AND FUTURE WORK

The most important factor that determines the security of the public (asymmetric) key cryptosystem, such as RSA, is its key size. So as to achieve more robustness against cryptanalysis attacks, the available solution is to increase the key size. Increasing the key size, however, has a drawback on the computational power. This paper proposes a new cryptographic approach, the *2K-RSA*, which uses two keys to perform the encryption where the size of each key is less than the key used in the conventional RSA. In addition to using two shorter keys, our approach uses an additional security component, *SeCa*, which makes the encryption/decryption processes more secure. The *2K-RSA* achieves more robustness against cryptanalysis attacks and reduces the encryption, decryption and key generation times, thus, reduces the computational overhead.

As a future work, we will conduct more experiments using more than two keys (3, 4 or more) of equal (or different) sizes, in addition to different formats of message segmentations. For example, instead of dividing the message into two equal parts, we will divide it into several parts of different sizes and/or arrangements. Moreover, we will generalize our results to generate mathematical models for the encryption/decryption times and the overall complexity on the cryptanalyst.

9. ACKNOWLEDGMENTS

We thank Sanaa Alwidian, a Ph.D. candidate at the University of Ottawa, for sharing her pearls of wisdom with us during this research and for her comments that greatly improved the manuscript.

10. REFERENCES

[1] G. Singh and S. Supriya, "A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security", International Journal of Computer Applications, Volume 67– No.19, (2013), pp. 33-38.

[2] Wikipedia, "Cryptography", [Online] Available: https://en.wikipedia.org/wiki/Cryptography, [Accessed, 12 April 2016]

[3] Dictionary.com, "Cryptography", [Online] Available: http://www.dictionary.com/browse/cryptography%20?s=t. , [Accessed 19 April 2016]

[4] A. A. Ayele and V. Sreenivasarao, "A Modified RSA Encryption Technique Based on Multiple public keys", International Journal of Innovative Research in Computer and Communication Engineering Vol. 1, (2013).

[5] A. Alhasib and A. L.Haque, "A Comparative Study of the Performance Issues of the AES and RSA Cryptography",in Proc. 3rd International Conference on Convergence and Hybrid Information Technology (ICCIT), Busan, (2008), pp.505-510.

[6] Thakur, Jawahar, and N. Kumar. "DES, AES and Blowfish: Symmetric key cryptography algorithms simulation based performance analysis.", International journal of emerging technology and advanced engineering Volume 1.2 (2011), pp. 6-12.

[7] R. Rivest, A. Shamir, and L. Adleman, "A Method For Obtaining Digital Signatures and Public Key Cryptosystems", ACM Transactions on Communications, Vol. 21, (1978), pp. 120-126.

[8] E. Thambiraja, G. Ramesh and R. Umarani, "A Survey on Various Most Common Encryption Techniques", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 7, (2012), pp. 226-233.

[9] R. Biswas, S. Bandyopadhyay and A. Banerjee, "Fast Implementation of the RSA Algorithm Using the GNU MP library", in Proc. National workshop on cryptography, (2003), pp. 1–15..

[10] RSA Laboratory, "What key Size Should Be Used?", EMC, [Online]. Available: http://www.emc.com/emc-plus/rsa-labs/historical/twirl-and-rsa-key-size.htm, [Accessed, 1 March 2016]

[11] S. Adi and E. Tromer. "Factoring large numbers with the TWIRL device", Advances in Cryptology-CRYPTO. Springer Berlin Heidelberg, (2003), pp. 1-26.

[12] Vocal, "RSA Key Size Selection," [Online]. Available: http://www.vocal.com/cryptography/rsa-key-size-selection/#. [Accessed, 3 March 2016].

[13] F. Fatemi, M. Maen, T. Alrashdan and O. Karimi, "A Hybrid Encryption Algorithm based on Small-e and Efficient RSA for Cloud Computing Environments", Journal of Advances in Computer Network, Vol. 1, No. 3, (2013). Available Online at http://www.jacn.net.

[14] S. J. Aboud, M. A. Alfayoumi, M. Alfayoumi and H. Jabbar, "An Efficient RSA Public Key Encryption Scheme", in Proc. ITNG, (2008), pp.127-130.

[15] D. Pointcheval, "New Public Key Cryptosystem based on the Dependent-RSA Problem", in Proc. LNCS, Springer-Verlag, Berlin-Heidelberg, (1999), pp.239-254.

[16] Gupta, Swastik, and Jaibir Sharma. "A hybrid encryption algorithm based on RSA and Diffie-Hellman." Computational Intelligence & Computing Research (ICCIC), 2012 IEEE International Conference on. IEEE, 2012.

[17] Deshmukh, Shyam, and Rahul Patil. "Hybrid cryptography technique using modified Diffie-Hellman and RSA."

[18] Marwaha, Mohit, et al. "Comparative analysis of cryptographic algorithms."Int J Adv Engg Tech/IV/III/July-Sept 16 (2013): 18.

[19] Al-Hamami, Alaa Hussein, and Ibrahem Abdallah Aldariseh. "Enhanced method for RSA cryptosystem algorithm." Advanced Computer Science Applications and Technologies (ACSAT), 2012 International Conference on. IEEE, 2012.

[20] Hassan, Abdel-karim SO, Ahmed F. Shalash, and Naglaa F. Saudy. "MODIFICATIONS ON RSA CRYPTOSYSTEM USING GENETIC OPTIMIZATION." International Journal of Research and Reviews in Applied Sciences 19.2 (2014): 150.

[21] N. Muhammadi, J. M. Zaini and M. Y. Saman, "Loop-based RSA Key Generation Algorithm using String Identity", in Proc. 13th International Conference on Control, Automation and Systems, ICCAS (2013).

[22] A. Odeh, K. Elleithy, M. Alshowkan and E. Abdelfattah, "Quantum Key Distribution by Using Public Key Algorithm(RSA)", IEEE, (2013)

[23] R. Patidar and R. Bhartiya, "Modified RSA Cryptosystem Based on Offline Storage and Prime Number", IEEE, (2013).

[24] L. Wang and Y. Zhang, "A New Personal Information Protection Approach Based on RSA Cryptography", IEEE, (2011).