RSA Based Encryption Decryption of Medical Images

Nadjia Anane[1], Mohamed Anane[2], Hamid Bessalah[1], Mohamed Issad[1], Khadidja Messaoudi[1]

[1]*CDTA (Centre de Développement des Technologies Avancées), BP 17, Baba Hassen, Alger, Algérie*

email: anane@cdta.dz

[2]*ESI (Ecole nationale Supérieure d'Informatique), BP 68M Oued Smar, Alger, Algérie*

email: m_anane@esi.dz

***Abstract*— The use of computer networks to transmit medical information is faced to data security problems. Hence it is necessary to make these data unreadable and indecipherable during their transfer.**

The encryption and decryption of medical images are performed either by software or hardware. A software implementation has the advantage of being portable and low cost but its drawback is the slowness decryption of a huge volume of data, compared to the hardware implementation and its inability to protect private keys. This is the reason that incited us to software implement the RSA protocol to encrypt and decrypt medical images by combining MATLAB and Maple tools.

This software implementation has served as a basis comparison to the hardware implementation of the same protocol on an FPGA circuit.

Some strategies have been adopted to make this software implementation the fastest in order to permit to the user generating keys, encrypting and decrypting medical images of different sizes with different keys sizes in a reasonable time based on the influence of the RSA parameters on the delays of the encryption/decryption operations.

I. INTRODUCTION

Nowadays, digital exchanges of medical images are frequently used throughout the world in a fraction of a second via the Internet. These data can be read or modified during their transmission via a non-controlled channel [1]. Therefore, it becomes very important to protect this private information against unauthorized viewers [2] by using cryptography.

Cryptographic techniques can be divided into symmetric encryption (with a secret key) and asymmetric encryption (with private and public keys).

In symmetric cryptosystems, the same key is used for the encryption or decryption and this key need to be secure and must be shared between the emitter and the receiver. These cryptosystems are very fast and easy to use [3].

In asymmetric or public-key cryptosystems, two different keys are necessary: the public and the private keys. With the receiver's public key, the sender encrypts the message and sends it to the receiver who decrypts the message with his private key.

The most popular and most widely used public-key cryptosystem is the RSA whose security depends on the difficulty of discovering the private key in a reasonable time but not on the details of the algorithm [4].

To ensure a maximum security, RSA requires the manipulation of numbers growing in sizes which induces a long time decryption. This has let the software implementation of this protocol slow especially when the data to be encrypted are of big sizes such as medical images [5].

In this paper, we present a software implementation allowing the encryption/decryption based on the RSA protocol. We have tested our software on medical images of different sizes by encrypting and decrypting them with different key sizes. The execution delays of the encryption/decryption operations are computed in order to study the influence of the RSA parameters on these delays and to be compared to those given by the hardware implementation of the same protocol on an FPGA circuit.

Some strategies have been adopted in the reading of images to make this software the fastest in an adequate time.

The remainder of this paper is organized as follows. In section 2, the recalls on the RSA protocol are given. Section 3 is devoted to the presentation of the implementation of the RSA protocol. Section 4 deals with the influence of the RSA parameters on the execution delays of the encryption and decryption. In Section 5, some tests, using the RSA developed software, were applied to medical images of different sizes to be encrypted and decrypted with keys of different sizes. In section 6, the results are discussed. Finally, we finish with a conclusion.

II. RECALLS ON THE RSA PROTOCOL

RSA is named according to its inventors Ron Rivest, Adi Shamir and Adleman Leonard [6]. It is the most used public key algorithm. It is based on two mathematical principles: factoring large integers and congruence arithmetic. The challenge to factorize an integer large number in two prime numbers makes this cryptosystem secure [7]. Actually, keys of 1024 bits to 2048 bits are commonly used [8].

In the RSA, the public key contains the modulus n which is a large integer number, a product of two prime numbers p and q, whose bits length is the key size.

If these numbers are detected, then the private key can be detected and the RSA protocol is broken.

We denote Φ (n), the Euler function of n with:

$$\Phi(n)= (p-1) \times (q-1).$$

The public and private keys are two numbers e and d associated with n.

e is first generated randomly from 2 to $\Phi(n)$, while being prime with $\Phi(n)$.

The pair (n, e) is the public key.

978-1-4244-7534-6/10/$26.00 ©2010 IEEE

Then d is calculated such as:

$$d = e^{-1} \bmod \Phi(n).$$

The extended Euclidean algorithm can calculate this inverse instantly, even with very large numbers. The pair (n, d) is the private key.

The use of keys in the encryption/decryption is as follows:

The message M must be smaller than the modulus n otherwise it must be cut.

It is encrypted with the public key (n, e) by computing the modular exponentiation (M^e modulo n) to obtain the encrypted message $C = M^e \bmod (n)$.

For decryption, we need the secret key (n, d), by computing once again the modular exponentiation to recover the original message $M=C^d \bmod (n)$.

$$C^d \bmod (n) = (M^e)^d \bmod (n) = M^{ed} \bmod (n) = M$$

, since $e \times d \bmod \Phi(n) = 1$

III. Software Implementation of the RSA

The software implementation of the RSA protocol was developed for encryption/decryption of medical images using MATLAB 7 and Maple 9.5 tools [9].

It is divided into three steps:

A. Generation of public/private keys (n,e)/(n,d)

It is done by generating two random numbers p and q of N/2 bits, where N is the size in bits of the modulus n, with the Maple functions "Random" and "NextPrime" by computing:

$$n = p \times q \quad \text{and} \quad \Phi(n) = (p-1) \times (q-1).$$

Then use a random generator to give a number e verifying the following two conditions:

- e must be less than 64-bits to speed-up the image encryption.
- e must be prime with $\Phi(n)$.

Finally, a number d is calculated such that:

$$e \times d = 1 \bmod \Phi(n)$$

B. Encryption of the image using the public key.

The encryption is divided into three parts which are:

1. Reading the BMP file of the image to attain its pixels whose bits are concatenated into a long message of numeric data.
2. Splitting this message into several ones M_i whose sizes are smaller than the modulus size.
3. Encryption of these messages one by one to form the list of encrypted messages $C_i = M_i^e \bmod (n)$, which are converted again to pixels then to BMP file in order to view the encrypted image.

C. Decryption of the image using the private key

It consists in:

1. Reading the BMP file of the encrypted image to attain its pixels whose values in bits are concatenated in a long message of numeric data.
2. Splitting this message into several ones C_i whose sizes are smaller than the modulus size.
3. Decryption of these messages one by one to obtain the original messages $M_i=C_i^d \bmod (n)$, which are converted to pixels then to a BMP file to recover the decrypted image.

IV. Influence of the RSA Parameters

To show the influence of each parameter of the RSA protocol on the remaining of its parameters and on the encryption/decryption time [10], two questions arise:

1. How long did it take to generate keys of hundreds of bits?
2. How long did it take to encrypt /decrypt or how long did it take to compute modular exponentiation of numbers of several hundred digits?

To answer to the first question, we need to generate large prime numbers very quickly. We must then calculate the PGCD [11].

The answer to the second question is the rapid execution of the encryption/decryption of the RSA. It is necessary to calculate powers whose exponents are very large numbers of several hundreds or even thousands of bits with a fast exponentiation algorithm [12].

Its principle is to compute successive squaring noting that:

$$M^{2k} = (M^k)^2 \quad \text{and} \quad M^{2k+1} = M \times (M^k)^2$$

For example to compute M^{37}, where 37=1+4+32, so M^{37} is divided into:

$$M^{37} = M \times M^{4} \times M^{32} = M \times (M^2)^2 \times (((((M^2)^2)^2)^2)^2)$$

In MAPLE, the operator *"&^"* uses the fast modular exponentiation algorithm. It was used for the calculation of the encryption/decryption operations of our software implementation of the RSA.

Table 1 shows the running time of generating the public and private keys and the delays of encrypting and decrypting medical images of different sizes with different sizes of the modulus.

These calculations were made with Maple, by the function ***time(.)*** on a computer based Intel Centrino Duo 1.73 GHz with a DDR2 memory of 1 G-byte.

TABLE I
ENCRYPTION/DECRYPTION DELAYS OF MEDICAL IMAGES FOR DIFFERENT SIZES OF THE MODULUS N

Modulus (bits)	Keys Generation (sec)	Image of 512x512 pixels	
		Encryption (sec)	Decryption (sec)
2048	3.993	***1.373***	***187.202***
1024	0.094	***0.969***	***59.436***
512	0.031	***0.780***	***22.043***
Modulus (bits)	Keys Generation (sec)	Image of 256x256 pixels	
		Encryption (sec)	Decryption (sec)
2048	3.869	***0.328***	***47.143***
1024	0.140	***0.249***	***15.445***
512	0.062	***0.171***	***5.772***
Modulus (bits)	Keys Generation (sec)	Image of 128x128 pixels	
		Encryption (sec)	Decryption (sec)
2048	3.885	***0.141***	***23.947***
1024	0.156	***0.094***	***3.822***
512	0.078	***0.031***	***1.497***

We noted that the keys generation delay is very fast and depends on the sizes in bits of both the modulus and the medical images.
The encryption delays are also fast, because they depend on the size of the public key e which is small, but are independent of the modulus size which is large.
We remark that for the same modulus, the encryption delays are proportional to image sizes.
Nevertheless the decryption delays of images, represented on Fig. 1 are too high especially for large images and large modulus. This is due to the size of the private key which is proportional to the size of the modulus n. Hence, a big amount of computing is induced when keys and images sizes are large.

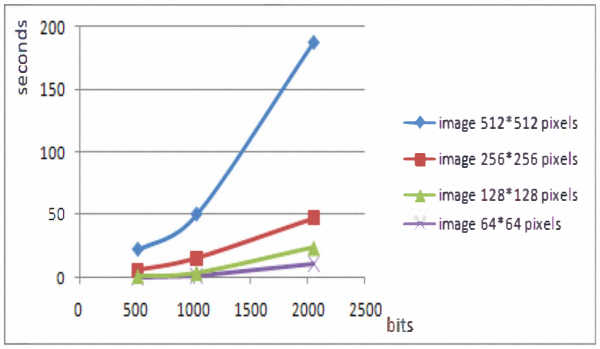


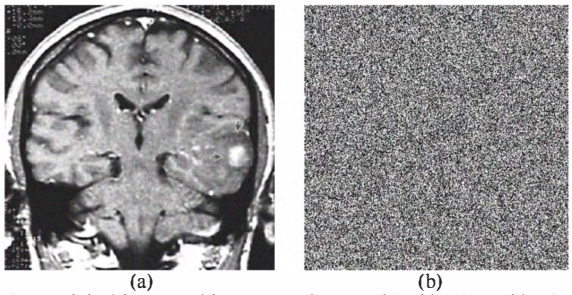
Fig. 1 Execution delays of images decryption for different sizes in bits of the modulus n.

V. RSA ENCRYPTION/DECRYPTION OF MEDICAL IMAGES

We have tested our software on 256 grey levels medical images concerning their encryption/decryption with different keys sizes. Each pixel is represented by 8 bits and it is possible to define 2^8 values of pixels intensity varying from 0 (black) to 255 (white) and the intermediate values represent the gray levels.
The encryption of the image requires reading it: pixel by pixel, then converting the values of these pixels to binary values which have been concatenated into a long message. This latter has been cut in messages or blocs of data whose sizes are smaller than the modulus size n.
Our encryption/decryption programs were developed with Maple 9.5. The RSA protocol applied to blocks of data requires the reading of the BMP files of medical images which was done by MATLAB 7 to obtain the numerical data of pixels.

A. Image Encryption

Images were converted from BMP format to attain their pixel values which have been translated to binary values hence to a list of messages M_i. The values of the pixels given in decimal must be verified if they are represented on eight bits. If not, they need adding zeros to the most significant values in order to be represented by 8 bits in the binary form. Once all the pixels are represented in binary with 8 bits, we need to concatenate all the values in a large message then divide it in blocks of data smaller than the size of the modulus. These blocks of data are encrypted one by one to provide a list of encrypted messages that can be reconstituted as a matrix of pixels to create the encrypted image to be viewed by MATLAB. From the 256 grey levels original medical image of 512×512 pixels, represented on the Fig. 2(a), we applied the RSA encryption with a key length of 1024 bits to obtain the encrypted image shown on the Fig. 2(b).

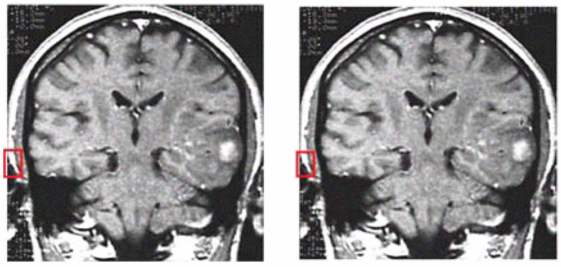


(a) (b)

Fig. 2 a) An original image and its encrypted Image (b) with RSA, with a key of 1024 bits

B. Image Decryption

To decrypt the image, we still need to read the BMP file of the encrypted image in order to convert it to pixels whose decimal values are transformed to binary values of 8 bits, then to a large encrypted message of data which must be cut in messages Ci smaller than the modulus. Hence these messages are decrypted one by one to finally reconstruct the matrix of pixels of the decrypted image which can be viewed.
After decryption, we can see that the decrypted image, shown on Fig. 3(b), is identical to the original image shown in 3(a).



(a) (b)

Fig. 3 a) Original image, b) Decrypted Image based on RSA, with a key of 1024 bits,

To speed up the image encryption delay, we have chosen the size of the message block inferior to the size of the image line ($size_{Line}$) in bits and equal to size of (N-1) where N is the size of modulus in bits. Hence we don't considerate some bits in the decryption process that we recover just after. The number of the altered bits is computed as follows:

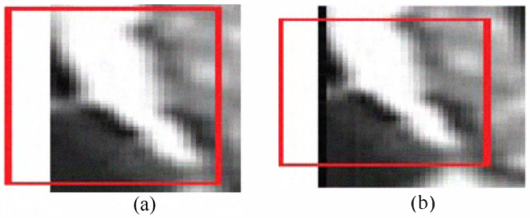
$$A_{bits} = size_{Line} \; mod \; (N\text{-}1)$$

Table 2, summarizes the number of bits altered in each column of the images of different sizes decrypted with different keys sizes.

TABLE II
NUMBER OF ALTERED BITS IN THE DECRYPTED IMAGE

Images sizes (pixels)	Modulus sizes in bits		
	2048	1024	512
512 × 512	2 bits	4 bits	8 bits
256 ×256	1 bit	2 bits	4 bits
128 ×128	0 bits	1 bit	2 bits

Fig. 4 shows the zooming of the red parties in the original and in the decrypted images. We remark, on Fig. 4(b), that the first column is black while the first column of Fig. 4(a) is grey, sometimes white. A block of 4 bits has not been encrypted and has been taken as zero (black) in the decryption process.



(a) (b)

Fig. 4 a) Original image, b) RSA decrypted Image with a key of 1024 bits,

This difference is located on the first column of the image and has been programmed during the reading and the visualization of the image in order to reduce the delays of the encryption/decryption operations.

VI. CONCLUSION

In this paper, we have presented a software implementation for encryption/decryption based on the RSA cryptosystem and tested on medical images. This implementation has a graphical interface that allows the user to generate keys of 128 to 2048 bits and encrypt or decrypt medical images with sizes ranging from (128×128) pixels to (512×512) pixels while viewing them after the encryption or decryption. We studied the influence of the RSA parameters and the image size on the delays of encryption/decryption operations. As the generation keys and the encryption processes are fast, we have adopted some strategies to speed up the delay of the decryption operation. However, this software implementation of the RSA protocol remains slow because the medical images are large and the sizes of the keys are in the range of (1024-2048) bits and tend to increase in the future. Therefore, it requires an improvement that was oriented towards a hardware solution [13], which is the only way to optimize the encryption/decryption delays of medical images in one part and to ensure greater security in the other part.

REFERENCES

[1] A. José Marconi, M. Rodrigues, "Transfert sécurisé d'images par combinaison de techniques de compression et cryptage", thèse de doctorat de l'université de Montpellier II, Octobre 2006.

[2] Bruce Schneier, *Cryptographie appliquée*, 2eme Edition de Vuibert, 2001.

[3] Richard A. Mollin, *RSA and Public-Key Cryptography (Discrete Mathematics and Its Applications*, Chapman and Hall/CRC; 1st edition, November 12, 2002.

[4] Renaud Dumont, *Introduction à la Cryptographie et à la Sécurité informatique*, Notes de cours provisoires, 2006–2007, Université de Liège, Faculté des Sciences Appliquées.

[5] Naveed Islam, William Puech, and Robert Brouzet, "A Homomorphic Method for Sharing Secret Images", in Proc. IWDW'09, LNCS 5703, Springer (Eds), Springer, pp. 121-135,2009.

[6] Alfred J. Menezes, Paul Van Oorschot and Scott Vanstone, *Handbook of Applied Cryptography (Discret Mathematics and its Applications*, CRC Press 1996.

[7] R.L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems" Communications of the ACM, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[8] Joppe W. Bos, Marcelo E. Kaihara, Thorsten Kleinjung, Arjen K. Lenstra and Peter L. Montgomery, "On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography", September 2009, Available: http://eprint.iacr.org/2009/389.pdf

[9] Richard E Klima, Neil Sigmon, Ernest Stitzinger, *Applications of Abstract Algebra with Maple and MATLAB*, Chapman and Hall/CRC; 2nd edition (July 12, 2006).

[10] Allam Mousa "Sensitivity of Changing the RSA Parameters on the Complexity and Performance of the Algorithm", Journal of Applied Sciences, Volume 5, Issue 1, pp. 60-63, 2005.

[11] Jean Guillame Dumas, "Casser le code RSA", Sciences et info HS n° 2 : Composition et décomposition, 2001.

[12] Ridha Ghayoula, El Amdjed Hajlaoui, "FPGA Implementation of RSA cryptosystem", Proceeding of World Academy of Science Engineering and technology, Vol. 14 august 2006.

[13] Nadia Nedjah, Luiza de Macedo Mourelle, "High-Performance Hardware of the Sliding-Window Method for Parallel Computation of Modular Exponentiations". International Journal of Parallel Programming 37(6): pp. 537-555 (2009).