

# RSA Algorithm

## Algorithm Analysis

Geovanny Burgos Retana	Anthony Leandro	Bryan Mena Villalobos
<i>Computer Engineering Student</i>	<i>Computer Engineering Student</i>	<i>Computer Engineering Student</i>
<i>Instituto Tecnológico de Costa Rica</i>	<i>Instituto Tecnológico de Costa Rica</i>	<i>Instituto Tecnológico de Costa Rica</i>
<i>San Jose, Costa Rica</i>	<i>Limon, Costa Rica</i>	<i>Heredia, Costa Rica</i>
<i>Email: geoburgosretana@gmail.com</i>	<i>Email: Anthonymyle@hotmail.com</i>	<i>Email: mena97villalobos@gmail.com</i>

**Abstract—**The abstract goes here.

### 1. Introduction

RSA algorithm, first published on 1977 by Ron Rivest, Adi Shamir, and Len Adleman, is a public-key cryptosystem, in this days, often used in various web servers, browsers and commercial system to protect web traffic among some other things like email encryption. In this type of cryptosystem the encryption key and the decryption key are different also, the encryption is public while decryption key is private. This cryptosystem is based on the difficulty of factoring to prime large numbers.

### 2. The Algorithm

Taking the words of Weisstein, E. RSA encryption is defined as "A public-key algorithm which

uses prime factorization as the trapdoor one-way function". Given the formula  $(m^e)^d \equiv m \pmod{n}$  the principle behind RSA is that is easy to find  $e, d, n$  such as the formula is true, but is very difficult even impossible, finding  $d$ , even knowing  $m, e, n$ . As mention before RSA consists of a public key and a private key, public key is used to encrypt and private key is used to decrypt the message in a reasonable time, from the formula given before, public key is represented by the integers  $e$  and  $n$ , and, the private key, is represented by  $d$  this led us with  $m$ ,  $m$  represents the message to encrypt

#### 2.1. How RSA works?

First of all, we imagine that two individuals wants to exchange an encrypted message with RSA,  $P_1$  has a public and a private key,  $P_1$  shares the public key with  $P_2$  when  $P_2$  has the key he

proceed to encrypt the message and send it to  $P_1$ , is important to clarify that only  $P_1$  has the private key which is used to decrypt the message, encryption and decryption process are described below.

**2.1.1. Encryption.** In encryption process the first step to follow is to turn  $M$  (message to send) into an integer  $m$ , such that  $0 < m < n$  in this step RSA uses a protocol such as the integer  $m$  will not be felt into a range of integers that isn't secure. After this computing  $c$  (encrypted message) will be easy, using Alice's public key  $e$  as the following:  

$$c \equiv m^e \pmod{n}$$

**2.1.2. Decryption.** Decryption is as easy as to use the private key  $d$  in the following way:

$$c^d \equiv (m^e)^d \equiv m \pmod{n}$$

This way the message is recovered with the private key

## 2.2. Before RSA

RSA, in a certain way, was the first implementation of public key encryption, but, as seen in *The history of Non-Secret Encryption* by J.H. Ellis public key encryption was long before been developed, in 1970 J. H. Ellis conceived a non-secret digital encryption (today known as public key encryption), in this time Ellis couldn't see a way to implement this type of encryption but

in 1973 an employee of GCHQ came with the basic idea of RSA encryption based on Ellis work, but, as the new techniques discovered on GCHQ that are potentially harmful, by definition, are classified information, this implementation was kept on secret.

## 2.3. Actual panorama

With the breakthrough in quantum computers is known that Shor's algorithm broke a 768-bit key (usually key sizes vary from 1024 to 4096 bits), this gives us an idea of how quantum computers may change our way of thinking and what a breakthrough it would be since almost every of our actual encryption methods are based on prime number's difficulty to be generated

## 3. Random and Pseudo-random number generation

As the title says there are different ways to find a random number, knowing that the idea of RSA is to keep secure data via public and private keys, RSA needs a complex algorithm that not only generates these keys as a random number, but also needs to be an algorithm that generates random numbers in a secure way, keeping away things like repeated random numbers or a pattern to generate them. The two basic ways to find random numbers are pseudo-random generation

and random generation, as mention before random number generation must be unpredictable but in the case of pseudo random generation knowing the seed used to generate this numbers makes easy to predict the next pseudo random number generated, so as we can see pseudo random generation is not feasible if we want a completely secure method to encrypt, next we have random generation, in random generation a non deterministic algorithm is used, as said before the number generated must be unpredictable so in random numbers generation may inputs, like keystrokes or mouse movements, may be used to ensure the randomness of the number generated. As seen in sections below this paper is base on the use of pseudo random numbers generation because the use of random generation as said in *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications* (2010) "... may still be deficient when evaluated by statistical tests. In addition, the production of high-quality random numbers may be too time consuming, making such production undesirable when a large quantity of random numbers is needed"(p.1-2)

Also the use of pseudo random generation may be a better way to get out with this problems base on the quote of *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*

Ironically, pseudorandom numbers often appear to be more random than random numbers

obtained from physical sources. If a pseudorandom sequence is properly constructed, each value in the sequence is produced from the previous value via transformations that appear to introduce additional randomness. A series of such transformations can eliminate statistical auto-correlations between input and output. Thus, the outputs of a PRNG may have better statistical properties and be produced faster than an RNG. (p.1-2)

#### 4. Methodology

Bla bla bla

#### 5. Experiments

as

#### 6. Analysis and Results

The analysis goes here

#### 7. Conclusion

The conclusion goes here.

#### References

- [1] Boneh, D (November, 1998). Twenty Years of Attacks on the RSA Cryptosystem, Retrieve from: <http://crypto.stanford.edu/dabo/pubs/papers/RSA-survey.pdf>

- [2] Ellis, J.H. (January, 1970). The possibility of secure non-secret analogue encryption, Retrieve from: <http://cryptocellar.org/cesg/possnse.pdf>
- [3] Ellis, J.H. (May, 1970). The possibility of secure non-secret analogue encryption, Retrieve from: <https://www.gchq.gov.uk/file/cesgresearchreportno3007pdf-2>
- [4] Ellis, J.H. 1987. The History of Non-Secret Encryption. Retrieve from: <https://web.archive.org/web/20130404174201/>  
<https://cryptocellar.web.cern.ch/cryptocellar/cesg/ellis.pdf>
- [5] Weisstein, Eric W. "RSA Encryption." From MathWorld—A Wolfram Web Resource. Retrieve from: <http://mathworld.wolfram.com/RSASecurity.html>
- [6] R.L.Rivest, A.Sharmir, L.Adleman: A method for obtaining digital signatures and public key Cryptosystems, Tata McGraw-Hill Retrieve from: <http://people.csail.mit.edu/rivest/Rsapaper.pdf>
- [7] Williamson, Malcolm J. (January 21, 1974). Nonsecret encryption using a finite field. Retrieve from: [https://www.gchq.gov.uk/sites/default/files/document\\_files/nonsecret\\_encryption\\_finite\\_field\\_0.pdf](https://www.gchq.gov.uk/sites/default/files/document_files/nonsecret_encryption_finite_field_0.pdf)