Rodrigo Sibaja Villarreal A01023121
Fernando Garrote De la Macorra A01027503
Rodrigo Benavente García A01026973
Karen Isabel Morgado Castillo A01027446
Ana Paola Minchaca García A01026744
Computational Mathematics

FINAL PROJECT ヽ༼ຈل͜ຈ༽ﾉ
RSA Cryptosystem

1. Topics to be developed:

a) Mathematical foundations of the RSA cryptosystem
b) Vulnerable areas of the RSA cryptosystem
c) Two specific applications of the RSA

The RSA public-key cryptosystem was invented at MIT in 1977 by Ronald Rivest, Adi Shamir and Leonard Adleman. And basically under this system, messages are encrypted with a code called the “public key”, which can be shared openly. Once a message has been encrypted with the public key, it can only be decrypted by another key, called the “private key”. Each RSA user has a key pairing consisting of their public and private keys. RSA is useful for communicating in situations where there has been no opportunity of distributing safely the keys beforehand.

RSA Mathematic Fundamentals

The implementation of RSA makes heavy use of modular arithmetic, Euler’s theorem, and Euler’s totient function. Each step of the algorithm involves multiplication, so it makes it easy for a computer to perform:

An RSA public-key/private-key pair can be generated by the following steps, as said in The Mathematics of the RSA Public-Key Cryptosystem by Burt Kaliski (all of this is made by the receiver):

1. Generate a pair of large, random primes p and q.
2. Compute the modulus n as $n = pq$. This product will be half of the public key.
3. Select an odd public exponent e between 3 and n-1 that is relatively prime to p-1 and q-1. e will be the other half of the public key.
4. Calculate the modular inverse d of e modulo $\phi(n)$. In other words, $de \equiv 1(\mathrm{mod}\phi(n))$. d is the private key. The triple bar (≡) here denotes modular congruence.
5. Distribute poth parts of the public key: n and e. d is kept a secret.

Now, to transmit a message, the steps are as described below:

1. Convert the message into a number m. It is important that $m<n$, as otherwise the message will be lost when taken modulo n, so if n is smaller than the message, it will be sent in pieces. One common conversion process uses the ASCII alphabet.
2. Calculate $c \equiv m^e \pmod{n}$. c is the ciphertext, or the encrypted message.
3. The receiver computes $m \equiv c^d \pmod{n}$, thus retrieving the original number m.
4. The receiver translates m back into letters, retrieving the original message.

Rodrigo Sibaja Villarreal A01023121
Fernando Garrote De la Macorra A01027503
Rodrigo Benavente García A01026973
Karen Isabel Morgado Castillo A01027446
Ana Paola Minchaca García A01026744
Computational Mathematics

So, taking that information into consideration, we can conclude that in order to do the encryption operation with a message *m* and a public key (*n*, *e*), we need the following formula:

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

Therefore, in order to do the decryption operation with a private key (*n*, *d*), we need this formula:

$$m = c^d \pmod{n}$$

<u>First Proof:</u> Now, to prove this, the Euler-Fermat Theorem is needed. It works for messages *m* that are relatively prime to the modulus *n*, that is where c^d (*m*, *n*) = 1.

So, first, Euler's theorem tells us that $m^{\phi(n)} \equiv 1 \pmod{n}$. From the receiver's 4th step on the first part, we choose *d* so that $de \equiv 1 (\bmod \phi(n))$. This means that there exists an integer *k* satisfying $de = k\phi(n)+1$. Then:

$$\begin{aligned} c^d &\equiv m^{de} \\ &\equiv m^{k\phi(n)+1} \\ &\equiv m^{k\phi(n)} \cdot m \\ &\equiv (m^{\phi(n)})^k \cdot m \\ &\equiv 1^k \cdot m \\ &\equiv m \pmod{n} \end{aligned}$$

Hence, m = c^d (mod *n*) is a unique integer in the range $0 \leq m < n$

For example, let's suppose the receiver selected the primes *p* = 11 and *q* = 17, along with *e* = 3, and following the algorithm:

1. *n* = *pq, n* = (11)(17) = 187 (half of the public key)
2. $\phi(n) = (p-1)(q-1)$, $\phi(n)$ = (11-1)(17-1) = 160. *e* = 3 was chosen before.
3. $de \equiv 1(\bmod \phi(n))$, *d* = 3^{-1} mod ((11-1)(17-1)) = 107.
4. So the public key (*n*, *d*) is (187, 3)

And now, suppose the message to send is "HI". The sender will have to send the message character by character.

1. "H" in ASCII is 72, so the message *m* = 72.
2. $c \equiv m^e \pmod{n}$, $c \equiv 72^3 \pmod{187}$ = 183.
3. $m \equiv c^d \pmod{n}$, $m \equiv 183^{107} \pmod{187}$ = 72.
4. The receiver translates 72 as "H".
5. "I" in ASCII is 73, so the message *m* = 73.
6. $c \equiv m^e \pmod{n}$, $c \equiv 73^3 \pmod{187}$ = 57.
7. $m \equiv c^d \pmod{n}$, $m \equiv 57^{107} \pmod{187}$ = 73.

Rodrigo Sibaja Villarreal A01023121
Fernando Garrote De la Macorra A01027503
Rodrigo Benavente García A01026973
Karen Isabel Morgado Castillo A01027446
Ana Paola Minchaca García A01026744
Computational Mathematics

8. The receiver translates 73 as "I". Having now the entire message encrypted and decrypted.

RSA Vulnerable Areas

One of the safest ways to send messages through the Internet is to use the RSA algorithm because it encrypts and decrypts the messages so that only the sender and receiver will know its contents. The strength of the algorithm lies on the size of the key which is $n = pq$.

A vulnerability that the RSA has is that the entire algorithm can be bypassed if an attacker decides to use the Euclidean Algorithm on two public keys. The Euclidean Algorithm is used in order to calculate the greatests common divisor of two specific numbers. If the greatest common divisor of both keys is not equal to one, then the attacker has successfully found a prime number that breaks both public keys.

For example, let's suppose that two public keys are "239149" and "166381", by using the Euclidean algorithm we can see that:

$$\begin{gathered} 239149 = 1 \times 166381 + 72768 \\ 166381 = 2 \times 72768 + 20845 \\ 72768 = 3 \times 20845 + 10233 \\ 20845 = 2 \times 10233 + 379 \\ 10233 = 27 \times 379 + 0 \end{gathered}$$

Since our last non-zero remainder is 379 that is the greatest common divisible number and now the attacker is able to break both keys.

The RSA algorithm has another vulnerability that happens when the two random primes "p" and "q" are too close to each other. For example, if: $p \approx q$ then we can assume that $N^2 \approx p$, and if this is the case, then N can be factorized using Fermat's Factorization Method. If we're using the Fermat's method then we can assume that , in which case N is factorable as $(a+b)(a-b)$, so assuming that "p" and "q" are close then "a" would be close to $\sqrt{N}$ and since "b" is small that would make it easy to find both prime numbers, which would break the RSA algorithm.

Common Modulus is another simple, but not so common in a real scenario, vulnerability that the RSA algorithm can have. RSA keys that use the same modulus are exposed to this threat. Eavesdroppers with access to a communication channel will wait for an opportunity where either a computer fault causes a bit to flip on the public exponent or the eavesdropper themself is able to flip it, causing a decryption problem on the receiving end. The eavesdropper now has access to two ciphertexts, a correct one and the faulty one, which share a public modulus, the Common Modulus attack can now be applied to extract the plain text.

$$\begin{aligned} C_1^x * C_2^y &= (M^{e_1})^x * (M^{e_2})^y \\ &= M^{e_1 x} * M^{e2y} \\ &= M^{e_1 x + e_2 y} \\ &= M^1 \\ &= M \end{aligned}$$

RSA Applications

One of the multiple applications for the RSA algorithm is in Ecommerce, as it secures communication between the web browser and the eCommerce website, the reason for this is the resistance it has to possible attacks. The connection made by this algorithm uses a secure socket layer (SSL) certificate, this certificate is created from the public and private keys that are created as a part of the encryption and decryption process. This results in a pseudo.random number which forms the basis for the certificate and it is installed at both ends of the connection to ensure protected communication.

Another application is to create a digital signature, in order to explain this application, we have to take in consideration the formulas:

$$f(m) = m^e \pmod{n}$$

$$g(m) = m^d \pmod{n}$$

The first one is the encryption function (public) and the second is the decryption function (private) then we can say that:

$$f(g(m)) = m \text{ and } g(f(m)) = m$$

The idea is that *f* is a function known by everyone, but only the owner knows the decryption function. In order for the owner to sign a message *(m),* the owner has to send *g(m)* with an indication that the message comes from him. When the other person receives the message and applies the encryption function of the owner *f(m) to the g(m)* sent then the other person will get *m.* If a stranger sends a false message *m',* supposedly from the owner to the receiver, the stranger wouldn't be successful as he doesn't know g(m).

Finally, another application, the protocol called Secure Socket Layer (SSL), provides data security between TCP/IP and application protocols such as HTTP, Telnet, etc. This protocol authenticates each end of the connection and it uses RSA for the authentication steps. First, the client requests the server`s certificate and its ciphers preferences. Then it generates a master key and proceeds to encrypt it with the server's public key and sends it to the server. The server decrypts the key with its private key and then it returns a message encrypted with the master key in order to authenticate itself to the client. Finally the server challenges the client, and the client responds by returning the client's digital signature with its public key certificate.

Rodrigo Sibaja Villarreal A01023121
Fernando Garrote De la Macorra A01027503
Rodrigo Benavente García A01026973
Karen Isabel Morgado Castillo A01027446
Ana Paola Minchaca García A01026744
Computational Mathematics

Bibliography:

- Kats, A. (n.d.). *RSA Encryption.* Retrieved from Brilliant.org: https://brilliant.org/wiki/rsa-encryption/
- Ireland, D. (2019). *RSA Theory.* Retrieved from DI Management: https://www.di-mgt.com.au/rsa_theory.html
- Kaliski, B. (n.d.). *The Mathematics of the RSA Public-Key Cryptosystem.* Retrieved from RSA Laboratories. [PDF provided by the professor].
- Lake, J. (2010). *What is RSA encryption and how does it work?* Retrieved from comparitech: https://www.comparitech.com/blog/information-security/rsa-encryption/
- Bhowmick, S. (2017). *What is the Role of RSA in Ecommerce.* Retrieved from Apps E Connect: https://www.appseconnect.com/what-is-the-role-of-rsa-in-ecommerce/
- Levasseur, K. (2013). *Digital Signatures using RSA.* Retrieved from Faculty UML: http://faculty.uml.edu/klevasseur/math/RSA_Signatures/RSA_Signatures.pdf
- Simpson, S. (2007). *Cryptography in Everyday Life.* Retrieved from University of Texas: https://www.laits.utexas.edu/~anorman/BUS.FOR/course.mat/SSim/life.html
- Pogiatzis, A. (2018). *RSA Attacks: Common Modulus.* Retrieved from Medium: https://medium.com/bugbountywriteup/rsa-attacks-common-modulus-7bdb34f331a5