|  |  |  |
| --- | --- | --- |
| **ipn** | **INSTITUTO POLITÉCNICO NACIONAL**  **ESCUELA SUPERIOR DE CÓMPUTO** |  |

**Cryptography**

**“Digital Signature”**

Abstact

Implementation on Python of a digital signature for text files providing 4 security services: Authentication, Integrity of Data, Confidentiality and no repudiation.

**By:**

**Garduño Velázquez Alan**

**Romero Gamarra Joel Mauricio**

Professor:

MSc. NIDIA ASUNCIÓN CORTEZ DUARTE

December 2017

To validate this report it is necessary to include the corresponding seal

**Index**

Contenido

[Introduction: 1](#_Toc500968087)

[Literature review: 3](#_Toc500968088)

[Software (libraries, packages, tools): 3](#_Toc500968089)

[Procedure: 4](#_Toc500968090)

[Results 4](#_Toc500968091)

[Discussion: 4](#_Toc500968092)

[Conclusions: 4](#_Toc500968093)

[References 5](#_Toc500968094)

[Code 5](#_Toc500968095)

# Introduction:

Cryptography is the study of mathematical techniques related to the aspects of information security, such as confidentiality, integrity and availability of data, authentication of entity and origin, it doesn’t include only the media to provide information security, but to a set of techniques. [1]

For providing these services included on the principal definition there is a cryptographic mechanism that can identify the origin entity that sends the message, also, it can confirm that the message has not been modified. This, provide authentication, no repudiation and integrity of data.

In this practice, the implementation of digital signature is explained below:

**ALGORITHM FOR ENCRYPTION**

* Encrypt the original message with SHA-1 and obtain the message digest
* Encrypt again the message digest but know, with RSA using private key of transmitter, this will give us the digital signature
* Encrypt the original message with RSA using public key of receiver
* Concatenate the encrypted message with RSA with the digital signature

On Figure 1, you can observe a flowchart that describes this process.

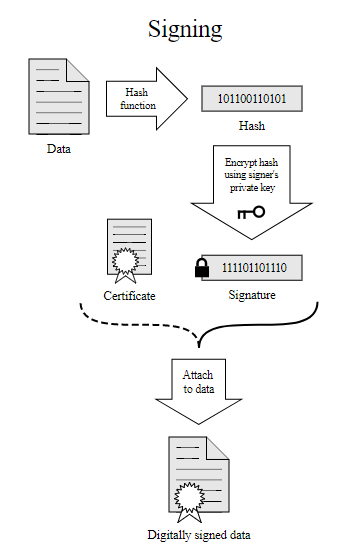


Figure 1. Signing process on digital signature

**ALGORITHM FOR DECRYPTION**

* Obtain the encrypted message (signed message)
* Decrypt the digital signature with RSA using public key of the transmitter
* Decrypt the encrypted text with RSA using private key of receiver to obtain again the original message
* Encrypt the original message with SHA-1
* Compare both message digest to verify they are the same

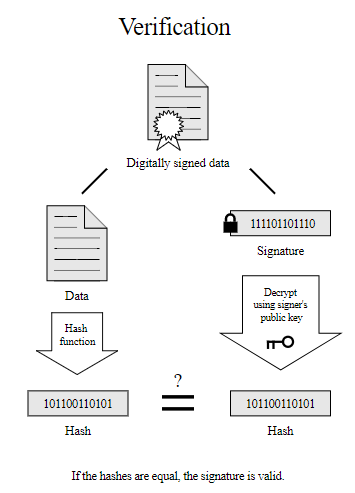


Figure 2. Verification process

It is important to remember that RSA is a cipher, so, you can encrypt a message and you can decrypt it to obtain the original one. In case of Hash functions, they are called digest functions because there is not a return, there is no way to obtain the original message, there are a lot of hash functions but in this case, we will use SHA-1, this function returns 20 bytes even if the original message has 1 letter or 200.

In the case of RSA, both parts (transmitter and receiver) needs to generate their own keys, by selecting 2 prime numbers p and q, n will be p x n and we need to calculate Euler totient function of n. Then, the ciphertext is given by the following formula: [2]

C = Me mod n

On Figure 3, you can observe RSA algorithm.

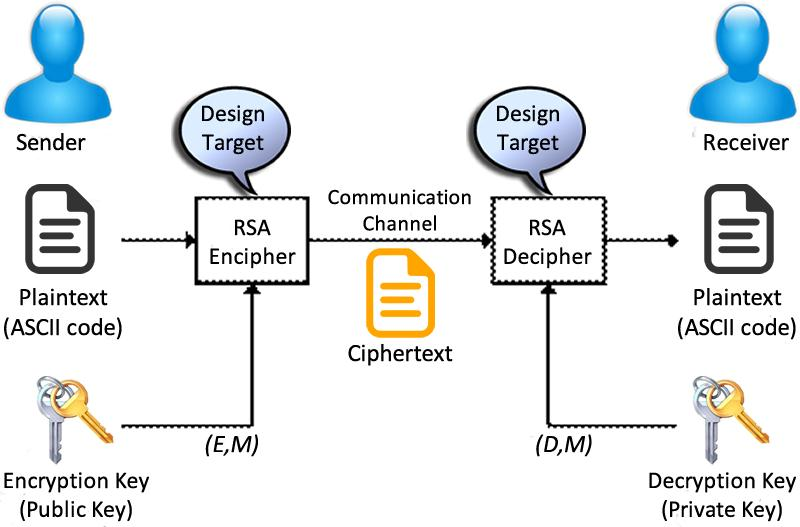


Figure 3. RSA Algorithm

**SERVICES PROVIDED:**

* **Authentication:** When the verifier validates the digital signature using public key of the transmitter, he is assured that signature has been created only by that person who possess the corresponding secret private key and no one else. [3]
* **Integrity of Data:** In case an attacker has access to the data and modifies it, the digital signature verification at receiver end fails. The hash of modified data and the output provided by the verification algorithm will not match. Hence, receiver can safely deny the message if data integrity has been breached. [3]
* **Non-repudiation:** Since it is assumed that only the signer has the knowledge of the signature key, he can only create unique signature on a given data. Thus, the receiver can present data and the digital signature to a third party as evidence if any dispute arises in the future. [3]

# Literature review:

The literature needs to provide an understanding of the conceptual and theoretical and mathematical background, context and justification of your work.

You should include diagrams, formulas, algortithms, …

should be referenced using

# Software (libraries, packages, tools):

* Sublime Text 3 [4]

# Procedure:

* Generate the key (public and private part) and save both parts in a file.
* To cipher the message, we need the public part of the key of the person who we want to send the message. We obtain the public key of file previously generated and then cipher with RSA.
* To sing, we generate a digest of the message with SHA-1 and then cipher the digest with RSA with our private key.
* Then join the two parts of the complete message the ciphered message and the sing.
* The when the person receives the message, he will decypt the message using his private key and to confirm the that is the original message, he must use the public key of the person who send the message to confirm the identity,

# Results

In the figure 1 we put the name of the files that will contain the parts of the key

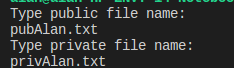


Figure 1

In the figure 2 we can see the execution of the ciphering and singing process.

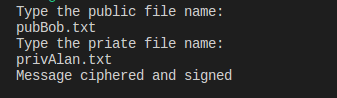


Figure 2

In the figure 3 we shoe the deciphering process

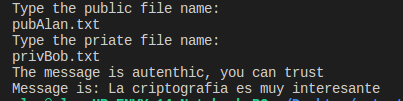


Figure 3

If something goes wrong the program output is: (See Figure 4)

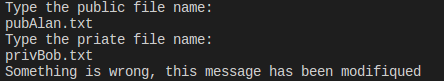


Figure 4

# Discussion:

This program works perfectly, so you can use it to send and receive messages whit the services that its provide.

As we can see in the results this program is a complete implementation of a digital signature, but is not ready to production in the side of cryptographic services is complete but in the side of UX is totally incomplete, we need more time to complete properly.

# Conclusions:

**Garduño Velázquez Alan**

It was very easy implementation, I can’t believe that is so easy to provide cryptographic services in a program it takes us less than 10 lines of code to implement it. So personally, I would like to do a more complete practice, for example to put the ciphering process in a server and the deciphering process in a client to have a real-life application and communication between two processes, also I would like to implement a complete UX so we don’t have the enough time to do such things.

**Romero Gamarra Joel Mauricio**

This practice was very interesting, because digital certificate was the safest implementation we did during the semester. It would be interesting program each algorithm used by hand, but the time was not enough due to complexity of that idea.

We know that there are 4 principal services cryptography needs to provide (excluding non-repudiation), with this practice we provide 3 of them:

* Confidentiality
* Integrity
* Authentication

Personally, I would like to finish a more complete practice for every person, but as such as I said before, the time was not enough, that would be interesting because you can be anyone sending a message to any other person and 3 services will be provided, in this case we have just 3 users and if you are another person, you can enjoy of them.

# References

|  |  |
| --- | --- |
| [1] | Menezes and Vanstone, "What is Cryptography," [Online]. Available: https://www.igi-global.com/dictionary/cryptography/6312. [Accessed 13 Diciembre 2017]. |
| [2] | M. Rouse, "RSA Algorithm (Rivest - Shamir - Adleman)," [Online]. Available: http://searchsecurity.techtarget.com/definition/RSA. [Accessed 13 December 2017]. |
| [3] | T. Point, "Cryptography Digital Signatures," [Online]. Available: https://www.tutorialspoint.com/cryptography/cryptography\_digital\_signatures.htm. [Accessed 13 December 2017]. |
| [4] | S. T. Inc, "Download," [Online]. Available: https://www.sublimetext.com/3. |
| [5] | bizagi. [Online]. Available: https://www.bizagi.com/es/productos/bpm-suite/modeler. |

# Code

1. #Generate key
2. key = RSA.generate(1024,rand)
3. keyp.write(key.exportKey('PEM'))
5. #Cipher and sing
6. mens = get\_message("m.txt")
7. hash1=SHA.new(mens).hexdigest()
8. privKey = get\_private\_key(private)
9. pubKey = get\_public\_key(public)
10. firma=privKey.sign(hash1,123)
11. enc=open("c.txt","w")
12. enc.write(str(msg\_ciph)+str(firma))
13. enc.close()
14. **print** "Message ciphered and signed\n"

17. #Decipher
18. msg=open("c.txt","r")
19. msg\_ciph=msg.readline()
20. msg.readline()
21. firma=msg.readline()
23. **if**(pubKey.verify(hash1,ast.literal\_eval(firma))):
24. **print** "The message is autenthic, you can trust"
25. **print** "Message is: "+mens
26. **else**:
27. **print** "Something is wrong, this message has been modifiqued"