# Digital Signature for data and documents using operating PKI certificates

*Adil* El Mane[1*], *Younes* Chihab[1], and *Redouan* Korchiyne[1]

[1]Computer Research Laboratory, Superior School of Technology, Ibn Tufail University, Kenitra, Morocco

**Abstract.** This report represents my researching work for a six-month internship in Lex Persona Enterprise. "The implementation of applications that sign data and files electronically using operating digital certificates" is the central theme of this research. This project brings together many applications related to compliance with well-specified programming constraints to sign files electronically.
The research has the goal of signing PDF documents digitally using the hash-and-sign method. The SHA-256 function as the hash function and RSA as encryption algorithm, accompanied with PKI certificates type X.509. The link between the applications and the certificates is the cryptographic API PKCS #11. This humbled work got dedicated to smartphones and computers. The use of WebView in all applications provides a taste of consistency. This article presents several processes carried out to accomplish this work.

## 1 Introduction

Developing a platform that signs data and documents digitally is the global functionality of the work. Therefore, the study focused on examining the work constraints.
The Frontend in WebView (The components seen by users got developed using HTML pages, Bootstrap Framework, and JavaScript/jQuery scripts.).
The Backend contains a set of functions programmed in Java which make it possible to list the certificates that exist in the Cryptoki. Because of that, we can implement the action of electronically signing the files by applying the RSA (Rivest – Shamir – Adleman) encryption algorithm and Check Signature validation.

The Signature function must have in parameters: the PIN that is a password of four numbers. It is essential to earn access to the Token information. The second parameter is the alias of the electronic certificate selected for signing, and each certificate has a handle. The third parameter is the Hash of the document that we demand the Signature to operate these certificates.

[*] Corresponding author: adil.elmane@uit.ac.ma

Certificates got stored in an external Token, so it is inescapable to use Hardware Security Module or HSM. It is a piece of electronic hardware that provides a security service by creating, storing, and protecting cryptographic keys. The HSM requires configuration for getting access to certificates.
The PKCS #11 standard is a platform-independent cryptographic token API. It defines all the types of cryptographic objects (RSA keys, X.509 Certificates, etc.) that got utilized. All the functions are necessary to implement, generate, modify, and delete these objects. Most certificate authority software operates the PKCS #11 standard to access the signing key, smartcards and HSMs.
The building of "Native Bridge" is necessary to link the Frontend and the Backend. For more explanation, link Java functions and JavaScript so that the components of WebView (HTML, CSS, JS ...) can achieve their objectives.

## 2 Ease of use

### 2.1 Digital Signature

The digitally signed document represents a piece of convincing evidence if the signatory accepts and adopts that all the content maintains a legal effect without forgetting that the signatory needs to respect the elements stated in the document. [1]

In other meaning, the Electronic Signature achieves approval of the content by the signatory and his acceptance that the message inside is considered trustworthy and honest. [2]

The legal value between handwritten signatures and digital signatures are pretty much the same, and even the last one offers more security guarantees. It got accepted as convincing evidence in many countries because of that.

In the security field, the Digital Signature acts as the organization for all public-key cryptography. It starts by running the key-generation algorithm. This operation provides the public key "**PK**" and the secret key "**SK**". [3]

There is another reason why the Digital Signature is more secure than the handwritten Signature: As we all recognize, the Electronic Signature of the same signatory on different documents is changing every time. That is up to the scheme of this type of signatures developed basing on mathematical functions and interactive protocols. [4]

We would not discuss the Digital Signature scheme if we did not mention those algorithms:
**Keygen($\mathbf{1^k}$)** Generate a combination of private and public keys from the parameter **k**.
**Sign $_{\text{SK}}$(m)** Provide a signature on input a message **m** and a secret key **SK**.
**Verify $_{\text{PK}}$(σ, m)** Inputs a signature **σ**, a message **m**, a public key **PK**, and outputs **1** if sigma is a valid signature on **m** concerning **PK**, and **0** otherwise.
If (**PK**, **SK**) is a valid key pair, then:
$\forall \boldsymbol{m}\colon \mathbf{Verify}_{\text{PK}}(\mathbf{Sign}_{\text{SK}}(\boldsymbol{m}), \boldsymbol{m}) = \mathbf{1}$ [5]

We can distinguish that the signed document got altered using the verification procedure. If the verification result returns true, that means the paper is unmodified and holds its legal effects. [6]

## 2.2 The RSA algorithm

In the past, we provoked problems in the distribution of the key. In another way, if the attacker obtained the private key in the message canal, he can decrypt the messages. So, the public key cryptography got created to correct the previous problem starting with dividing the key into two divisions, the secret key and the public key. The public key can encrypt information, and the private one stills secret and works when we want to decrypt messages. [7]
The RSA cryptosystem begins with a key generator to generate the pair keys of the asymmetric encryption algorithm.
To construct the key, **side X** chooses two prime numbers, ***p*** and ***q***. The goal in this step is to enhance security.

**Side X** then calculates ***n=p*q***. **Side X** then computes **ø(*n*)**, which is ***(p-1)(q-1).***
**Side X** then chooses an integer ***e*** with ***1< e < ø(n)*** and ***gcd( e, ø(n)) =1***. **Side X** then computes ***d=*** $e^{-1}$***(mod ø(n)).***
The public key consists of ***n*** and ***e****.* The secret key is ***d****.*
Using the key, **Side Y** can transmit messages to **Side X**, and **Side X** can decrypt them. [8]

Before talking about RSA encryption, we need to define the process of cryptography. Its starts by indicating the original message (plaintext), after that, we encrypt the text using the encryption key, and as a result, we get ciphertext. The opposite operation is called decryption; this operation got activated using the corresponding decryption key.

Cryptography ***(def)*** Encryption $\oplus$ Decryption

I already mention that asymmetric cryptography divides the key into public key and private key. We cannot find any shared key between the different sides. For example, **side X** uses the first key to encrypt messages and makes it incomprehensible for the people who do not have the second key. **Side Y** possesses the private key, so he can easily decrypt the ciphertext and discover the content. [9]

## 2.3 The SHA-256 hash function

The SHA-256 (Secure Hash Algorithm 256) is the most popular hash function. The number 256 means that the Hash consists of 256 bits, so there are many possibilities. [10]
The several cryptanalytic attacks targeting hash function let many products and services depend on SHA-256 instead of the previous hash functions. The evaluation of SHA carries many improvements in probability procedures of collision from $2^{-66}$ to $2^{-38}$.
In the end, Mendel finishes the research, and the obtained results are a practical 18-step collision and differential characteristic for 19-step SHA-224 collision. In addition, a model of a pseudo-near-collision for 22-step SHA-256 got produced. [11]
To carefully specify the number of possibilities, the calculation suggests that to guess a unique password randomly by an attacker, half of the time would take at least: $\mathbf{ln(2) \times 2^{256} \approx 8 \times 10^{76}}$ guesses.
Those estimates would take about $\mathbf{2.5 \times 10^{66}}$ years if we utilized a modern computer that makes 1000 guesses/second. The obtained result is more than the age of earth itself ($\mathbf{1.4 \times 10^{10}}$ years old).
The results of the calculation make the hacker not encouraged to start the guessing game. [12]

In a general sense, a hash function uses mathematical algorithms to convert binary data of any size into a fixed-size hexadecimal text. It offers message authentication, data integrity, and a slew of additional data security features. [13]

## 2.4 The PKI Certificates

The goal of Software Certification is to produce certificates that maintain the security and the protection of data sending or receiving between sides. It utilizes a software validation approach to make sure the document saves all the functionality of Digital Signature. [14]

The identity certificates in PKI (Public Key Infrastructure) produced many applications like X.509, SPKI, SDSI, etc. [15]

Using the X.509 Certificate develops encrypted communication between sides. Consequently, the transporting of data became encrypted. [16]

The PKI achieves both security and scalability. Other approaches have been either scalable but not secure or secure but not scalable. The X.509v3 is just representing the Internet PKI standard. It is built on a hierarchical model and got described in RFC-5280. The PEM (Privacy Enhanced Mail) encoding represents certificates in text format. It can be identified through the headers and trailers to reveal the content. [17]

The X.509 contains much information like the owner's name, the encryption algorithm used, the hash function used, and of course, a period of validity. [18]

The PKI is a kit of authorizations, policies, and processes for handling public-key mechanisms and their associated identities. [19]

## 2.5 The PKCS #11 API

The industry universally supported RSA Laboratories standard PKCS #11's cryptographic token as known as Cryptoki. Created as API Cryptographic for smartcard security tokens and HSMs (Hardware Security Modules). The process of use includes functions that linked smartcards and other programs that accept this concept. [20]

The HSM characterizes by the existence of memory and a processor inside. Any breakthrough effaced the memory immediately. In addition, it contains storage space for the host system. [21]

The PKCS #11 is a cryptographic API. It composes many functions that typically got used in scenarios that contain crypto secrets. Also, it can make sure that data are secured.

The HSMs obtain physical computing tools that can get added to a server through Ethernet or other sets. [22]

It is a way to store keys, PKI Certificates, and their applications. It maintains other cryptographic functions, for example, making private keys unreadable and encrypting data. [23]

# 3 Content

## 3.1 Previous works

One of the previous works, represented in the article "Digital Certificate Management for Document Workflows in E-government Services." The paper compares the PKI deployment (OpenSSL) with MOSS (Microsoft Office SharePoint Services). The OpenSSL is open source and free to use. Its goal is to regenerate updates to improve the service. The OpenSSL has no User Interface because it optimizes for SSL and TSL protocols. In contrast, MOSS is straightforward to implement with good workflow management. Unfortunately, MOSS is unfree, and it works exclusively for documents.

Moreover, the article took up a group of applications created by multiple companies. For example, there are "CertSign certificates.". It is a trusted PKI that offers digital certificates and produces signatures in well-known formats (PKCS #7). The CertSign Certificates can offer certificates on USB Tokens. Unlikely the services are expensive to implement, and there is no control over issued certificates.

In addition, the CryptoBOT e-Workflow represents a commercial application that can sign and integrate all types of documents. It can insert explanation comments to the Signature using the e-Crypt application. The User Interface is complex with no attractive look. It is necessary to buy the entire package to receive the right to use the e-Crypt application.

Another PKI Certificate Authority based on J2EE technology is the "EJBCA." It is free to use and comes with a lot of great features. Those characteristics allow some major enterprises to implement the application as PKI. On the other hand, "EJBCA" is considered a web application, so it depends on the perfection of the webserver.

The suggested PKI design is composed of a two-tier Certification Authority architecture. This structure includes one Root CA and one subordinate CA.

The analysis revealed that maintaining security policies for the CA's activities necessitates regular monitoring. Also, the existence of a User Interface will make the architecture way better. [24]

Some other researches are focused on reducing the client's signature generation time. As a model, the requests for concert tickets can be high at the beginning.

"The online/offline scheme" is an excellent technic to sign documents digitally. It starts by performing a calculation that is independent of the paper. When data got delivered to the server, additional work is required to produce a valid Signature.

Furthermore, "the short signature scheme" generated DSA Signature with the same degree of security and protection using particular elliptic arches. [25]

Electronic commerce may necessitate handling digital contracts. These contracts call for generating digital signatures using private keys. These types of keys got saved in some tamper-Proof digital Token like smartcards. It is possible to add a tool that controls access to the user's private key, like Fingerprint technology.

The procedure of Fingerprint is straightforward. Preprocessing is the first step, and if the extracted minutiae almost match the template, then the user of the smartcard can get access to content.

Unfortunately, the researchers find that the fingerprint matching process demands a significant amount of time, and the size of the template requires much space storage. [26]

### 3.2 The research

This work starts with selecting a file imported from a server to sign it later. We need to verify the contents of the document before signing it. Also, it is necessary to make sure this is the document we want to sign. The following interface contains a list of electronic certificates obtained from an external smartcard security token accompanied by a zone to enter the correct PIN of the Token. We apply those instructions to be able to pursue the digitally signing operation (*Fig. 1*).

What we extract from the smartcard are these four elements:

- Name: Basically, the name of the certificate. It is essential to indicate the certificate that the user wants to use to complete the signing operation.
- Alias: Certificate aliases intended for encryption and signature verification.
- Usage: There are different certificate stores where certificates need to get imported into depending on use.
- Chain: The list of SSL certificates, from the root certificate to the end-user certificate, represents the chain of SSL certificates.

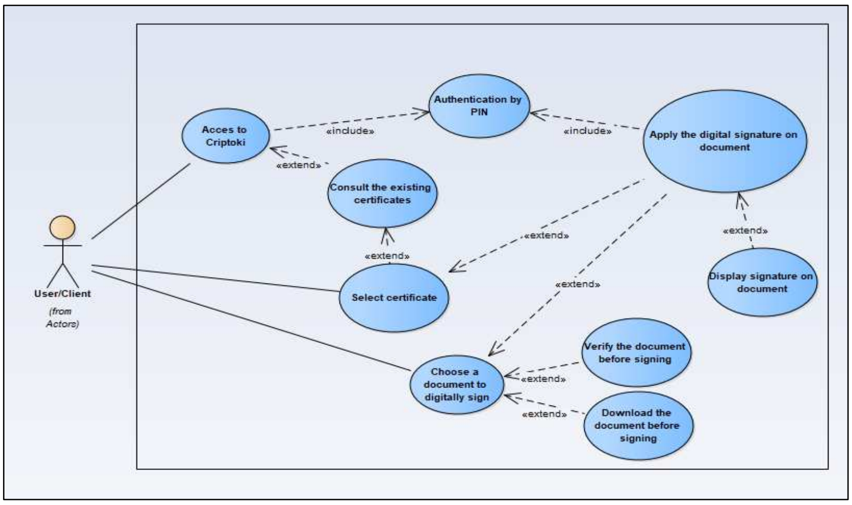
Authentication by PIN
Acces to Criptoki
«include»
«include»
Apply the digital signature on document
«extend»
Consult the existing certificates
«extend»
«extend»
«extend»
User/Client
(from Actors)
Select certificate
«extend»
Display signature on document
Verify the document before signing
Choose a document to digitally sign
«extend»
«extend»
Download the document before signing

**Fig. 1.** The use case template for the Digital Signature system

The application system transforms the document to hash format.
Then, it implements the RSA algorithm for encryption by a private key of the certificate. We display the Signature result, and finally, we confirm the validity of this Signature (verification by public key).

The changes observed after the signing operation:
The PDF reader displays the certificate name, the validity of this Signature, the date of the Signature, and a rectangle that contains the information about the signing certificate.

We can find more details like the reason for this Signature, the location of the Signature, and the certificate owner (*Fig. 2)*.

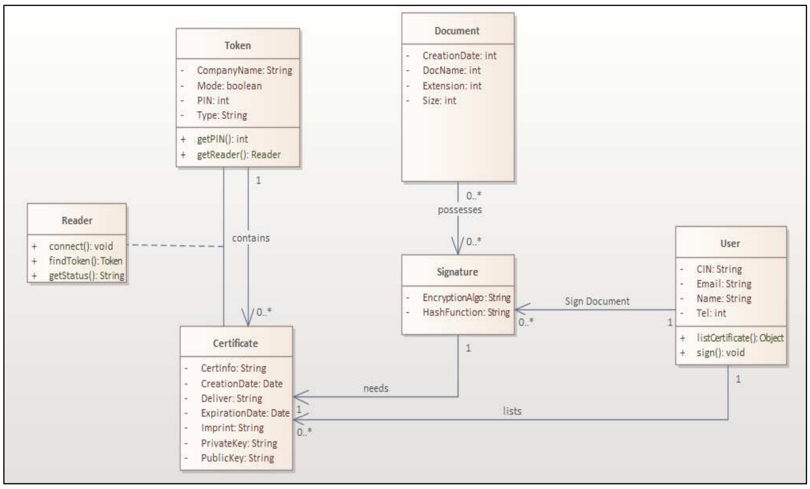

**Fig. 2.** The class diagram that explains the Digital Signature system

+ Each Token contains none or many Certificates.
+ To make the Signature, it is necessary to have at least a Digital Certificate.
+ Each User can list the Certificates from the Cryptoki.
+ A User can sign the document using the Electronic Signature.
+ A Document can have several Electronic Signatures.
+ To access the Token and view the existing Certificates, it is necessary to configure the associative Reader class.

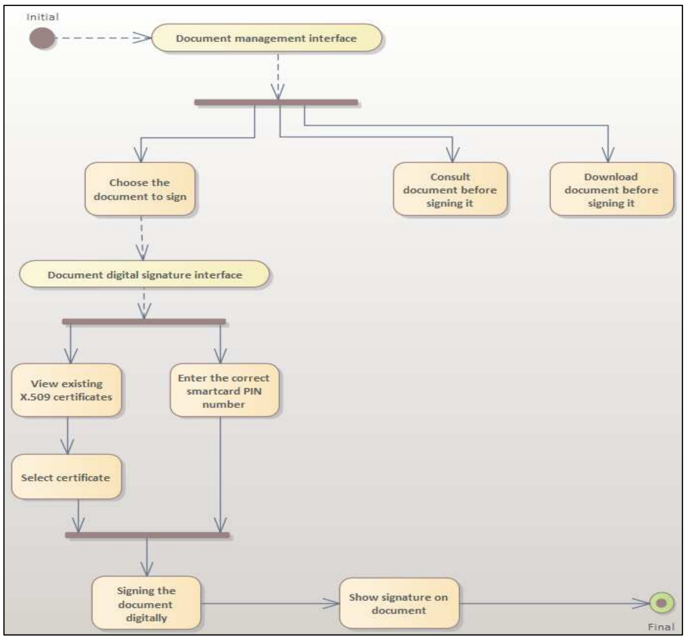
Initial
Document management interface
Choose the document to sign
Consult document before signing it
Download document before signing it
Document digital signature interface
View existing X.509 certificates
Enter the correct smartcard PIN number
Select certificate
Signing the document digitally
Show signature on document
Final

**Fig. 3.** The activity diagram for the Digital Signature generation

The activity diagram (*Fig. 3)* above shows all the actions users must follow to obtain a document electronically signed according to the proposed scheme.

The user can choose a document to sign and consult it before signing it. Next, he selects the certificate that he wants to operate for signing. After that, he becomes forced to enter the correct PIN to complete the operation. This PIN is in the format of a four-digit password.
The user lunches the signed document in a PDF Reader. He examines the validity of the Digital Signature, assigning to the characteristics of certificates. If the certificate passes the certification policies (Egg: issue by the certification authority, unexpired certificate, etc.), the electronic Signature becomes valid, and the acquired document possesses a legal value.
The scheme must deal with the case of absence of the Token or the case of a disconnected smartcard. The application detects the entrance of Tokens (Smartcard became connected) and lists the certificate that exists inside it at the same time.

## 4 Conclusion

Using the Digital Signature became famous. Some people start using it because of other services that we can find in it comparing to a handwritten one. The most considerable advantage is offering more security and protection because the Electronic Signature of the same signatory is changing every time. If all people start operating it, we will see no more archives of printed documents handwritten signed, and no more loss of many on papers. The proposed strategy contains the hash-and-sign method, SHA-256 as the hash function, and RSA as the encryption algorithm. This strategy allows us to display a Digital Signature on a document in a polished manner by relying on the latest means of protection and encryption. The result is promising to carry out more research in a way that we can apply a more robust encryption algorithm. Plus, the work can improve by Signing the compressed files as well. The library that allows connecting to smartcards and read the content is still not yet perfect on mobile. It lacks some valuable features for the process. Regarding the adaptation with new library updates, the scheme will be easy to manage.

## References

1. S. Mason, *Electronic Signatures in Law, chapter 1: The Signature*, 9 (2003)
2. S. Mason, *Electronic Signatures in Law, chapter 2: International initiatives*, 101-102 (2003)
3. J. Katz, *Digital Signatures, Digital Signatures: Background and Definitions*, 3 (2010)
4. B. Pfitzmann, *Digital Signature Schemes: General Framework and Fail-Stop Signatures, Requirements on digital signature schemes*, 2 (1996)
5. L. El Aimani, *Verifiable Composition of Signature and Encryption: A Comprehensive Study of the Design Paradigms, Preliminaries*, 3-4 (2017)
6. S. Mason, *Electronic Signatures in Law, chapter 3: The practical issues in using electronic signatures in different jurisdictions*, 127 (2003)
7. S. Katzenbeisser, *Recent Advances in RSA Cryptography, chapter: Public Key Cryptography and RSA-Type Cryptosystems*, 25 (2001)
8. S. Rubinstein-Salzedo, *Cryptography, Chapter: The RSA Cryptosystem*, 113 (2018)

9. S. Y. Yan, *Cryptanalytic Attacks on RSA, Chapter: RSA Public-Key Cryptography*, 55-56 (2008)
10. A. A. Smith, U. Whitcher, *Making a Hash of Things,* Math Horizons, **23**, 5-9 (2015)
11. R. Avanzi, L. Keliher, F. Sica, *Selected Areas in Cryptography, Chapter Collisions and Other Non-random Properties for Step-Reduced SHA-256*, 277 (2009)
12. A. A. Smith, U. Whitcher, *Making a Hash of Things,* Math Horizons, **23**, 5-9 (2015)
13. D. I. Nassr, *Secure Hash Algorithm-2 formed on DNA*, Journal of the Egyptian Mathematical Society*,* (2019)
14. E. Damiani, C. A. Ardagna, N El Ioini, *Open-Source Systems Security Certification, Chapter: Test-based security certifications*, 27-28 (2009)
15. B. Crispo, B. Christianson, J. A. Malcolm et M. Roe, *Security Protocols, Chapter Review and Revocation of Access Privileges Distributed with PKI Certificates*, 100 (2001)
16. W. R. Simpson, *Enterprise Level Security: Securing information systems in an uncertain world, Chapter8: Claims-Based Authentication*, **1**, 103 (2016)
17. A. Karamanian, F. Dessart, S. Tenneti, *PKI Uncovered: Certificate-Based Security Solutions for Next-Generation Networks*, Cisco Press*,* 19-23 (2011)
18. S. Mason, *Electronic Signatures in Law 3rd edition, chapter 14: Digital signatures*, 283 (2012)
19. Y. C. Elloh Adja, B. Hammi, A. Serhrouchni, S. Zeadally, *A blockchain-based certificate revocation management and status verification system*, (2021)
20. P. Degano, L. Viganò, *Foundations and Applications of Security Analysis, Chapter: Analysing PKCS#11 Key Management APIs with Unbounded Fresh Data*, 92-93 (2009)
21. P. Degano, L. Viganò, *Foundations and Applications of Security Analysis, Chapter: Towards a Type System for Security APIs*, 174 (2009)
22. R. Focardi, A. Myers, *Principles of Security and Trust, Chapter: Automated Backward Analysis of PKCS#11 v2.20*, 219 (2015)
23. A. Karamanian, F. Dessart, S. Tenneti, *PKI Uncovered: Certificate-Based Security Solutions for Next-Generation Networks*, Cisco Press*,* 34 (2011)
24. F. Pop, C. Dobre, D. Popescu, V. Ciobanu, V. Cristea, *Digital Certificate Management for Document Workflows in E-Government Services*, International Conference on Electronic Government EGOV, (2010).
25. C. Chou, W. C. Cheng, L. Golubchik, *Performance study of online batch-based digital signature schemes*, Journal of Network and Computer Applications*,* **33**, 98-114 (2010)
26. Y. Lin, X. Maozhi, Z. Zhiming, *Digital signature systems based on smart card and fingerprint feature*, Journal of Systems Engineering and Electronics*,* **18**, 825-834 (2007)