

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA  
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

*Mohamed El-Bachir El-Ibrahimi UNIVERSITY - Bordj Bou Arreridj*

**Faculty of sciences and technology**

*Department of Electronics*

# *Thesis*

PRESENTED TO OBTAIN :

**MASTER DIPLOMA**

**Field : Electronics**

**Specialty : Electronic industries**

by

- **Djouadi Abdelouahed**
- **Ouchene Rafik**
- **Derb Mohammed Lamdjed**

*Titled*

*Smartcard based Cryptographic E-signature*

*Soutenu le : 03/11/2024*

*In front of the following jury:*

<i>First and Last Name</i>	<i>Grade</i>	<i>Quality</i>	<i>Establishment</i>
<i>Dr. YOUSFI Abderahim</i>	<i>MCA</i>	<i>Président</i>	<i>Univ-BBA</i>
<i>Dr. BELHADAD Yehya</i>	<i>MCB</i>	<i>Supervisor</i>	<i>Univ-BBA</i>
<i>Dr. MESSAI Zitouni</i>	<i>MCA</i>	<i>Supervisor</i>	<i>Univ-BBA</i>
<i>Dr. SAIDANI Okba</i>	<i>MCB</i>	<i>Examiner</i>	<i>Univ-BBA</i>
<i>Pr. MAHDID Fatima Zohra</i>	<i>PR</i>	<i>Incubator Representative</i>	<i>Univ-BBA</i>

*Academic Year: 2023/2024*

## Abstract

Cryptography a vital pole in modern information security, it ensures the confidentiality, integrity, and authenticity of digital communications. In this thesis we explore these concepts in depth, focusing on their application in creating electronically signed diplomas through a system called EDIPLOMA. The first chapter provides a comprehensive overview of cryptographic techniques and their role in secure communication. The second chapter examines the authentication mechanisms essential for verifying identities in digital transactions. Finally, the third chapter details the implementation of the EDIPLOMA system, demonstrating how cryptography and authentication combine to enhance the reliability of electronic diplomas. This work highlights the importance of secure digital signatures in educational credentials and contributes to the ongoing discourse on improving the security of digital documents.

## المخلص

التشفير هو عمود أساسي في أمن المعلومات الحديثة، حيث يضمن سرية وسلامة وأصالة الاتصالات الرقمية. في هذه الأطروحة، نستعرض هذه المفاهيم بشكل مفصل، مع التركيز على تطبيقها في إنشاء الشهادات الإلكترونية الموقعة من خلال نظام يسمى EDIPLOMA. يقدم الفصل الأول نظرة شاملة على تقنيات التشفير ودورها في تأمين الاتصالات. بينما يتناول الفصل الثاني آليات المصادقة الضرورية للتحقق من الهوية في المعاملات الرقمية. وأخيراً، يوضح الفصل الثالث كيفية تنفيذ نظام EDIPLOMA، مظهراً كيف يجتمع التشفير مع المصادقة لتعزيز موثوقية الشهادات الإلكترونية. يبرز هذا العمل أهمية التوقيعات الرقمية الأمانة في الشهادات التعليمية ويساهم في النقاش المستمر حول تحسين أمان الوثائق الرقمية.

## Summary

Abstract.....	- 2 -
الملخص .....	- 2 -
Summary.....	- 1 -
List of figures .....	- 1 -
List of abbreviations: .....	- 2 -
Abbreviation.....	- 2 -
Description .....	- 2 -
EDIPLOMA .....	- 2 -
Electronic Diploma.....	- 2 -
PKI .....	- 2 -
Public Key Infrastructure .....	- 2 -
CA .....	- 2 -
Certificate Authority.....	- 2 -
RA .....	- 2 -
Registration Authority .....	- 2 -
HSMs.....	- 2 -
Hardware Security Modules .....	- 2 -
RF .....	- 2 -
Radio Frequency.....	- 2 -
EHRs .....	- 2 -
Electronic Health Records.....	- 2 -
UTXO.....	- 2 -
Unspent Transaction Output.....	- 2 -
TLS.....	- 2 -
Transport Layer Security.....	- 2 -
SSL .....	- 2 -
Secure Sockets Layer .....	- 2 -
API .....	- 2 -
Application Programming Interface .....	- 2 -
JWT .....	- 2 -

JSON Web Tokens .....	- 2 -
RSA .....	- 2 -
Rivest–Shamir–Adleman.....	- 2 -
AES .....	- 2 -
Advanced Encryption Standard.....	- 2 -
DES .....	- 2 -
Data Encryption Standard.....	- 2 -
SLE5542.....	- 2 -
A specific type of memory card used in the project .....	- 2 -
UI.....	- 2 -
User Interface .....	- 2 -
EFT.....	- 2 -
Electronic Fund Transfers .....	- 2 -
Introduction.....	- 1 -
Chapter I: Foundations of Cryptography and Authentication.....	- 3 -
1. Introduction.....	- 3 -
.2 Cryptographic Authentication.....	- 3 -
2.1.1. Symmetric Key Cryptography .....	- 4 -
2.1.2. Asymmetric Key Cryptography .....	- 4 -
2.1.3. Uses of asymmetric cryptography.....	- 5 -
2.1.3.1. Digital signatures .....	- 5 -
2.1.3.2. Communications.....	- 5 -
2.1.3.3. Cryptocurrencies .....	- 6 -
3. Public key infrastructure .....	- 6 -
3.1. Components of a PKI .....	- 7 -
3.1.1. Certificate authority .....	- 7 -
3.1.2. Registration authority.....	- 7 -
3.1.3. Certificate database.....	- 7 -
3.1.4. Certificate administration system .....	- 7 -
3.1.5. Certificate policy:.....	- 7 -
3.1.6. Certificate (Public / private key pair).....	- 7 -
4. Smartcards:.....	- 8 -

4.1.	Classification of Smartcards.....	- 8 -
4.1.1.	Based on communication interface.....	- 8 -
4.1.1.1.	Contact smart card .....	- 8 -
4.1.1.2.	Contactless smart card .....	- 8 -
4.1.1.3.	Dual interface smart card .....	- 9 -
4.1.2.	Based on the function of the card .....	- 9 -
4.1.2.1.	Memory card .....	- 9 -
4.1.2.2.	Microprocessor smart card .....	- 9 -
4.1.2.3.	Cryptographic smart card .....	- 9 -
4.1.2.4.	Hybrid smart card .....	- 10 -
5.	Data Authentication Using Digital Signatures .....	- 10 -
5.1.	Importance of Digital Signatures.....	- 10 -
5.1.1.	Definition: .....	- 10 -
5.1.2.	Importance: .....	- 11 -
5.2.	Digital Signature Authentication mechanism.....	- 11 -
5.2.1.	Generation.....	- 11 -
5.2.2.	Verification .....	- 12 -
5.3.	Real-World Examples of Digital Signature Applications .....	- 12 -
5.3.1.	E-Government Services .....	- 12 -
5.3.2.	Financial Transactions .....	- 12 -
5.3.3.	Legal Documents .....	- 13 -
5.3.4.	Software Distribution .....	- 13 -
5.3.5.	Healthcare .....	- 13 -
6.	CONCLUTION .....	- 13 -
Chapter II: Memory card based PKI.....		- 14 -
1.	Introduction.....	- 14 -
2.	Components of a Memory Card-Based PKI System .....	- 14 -
2.1.	Memory Card.....	- 14 -
2.2.	Private Key Storage .....	- 15 -
2.3.	Overview of RSA Cryptography .....	- 15 -
2.3.1.	Introduction to RSA Algorithm .....	- 15 -
2.3.2.	Key Components of RSA .....	- 15 -

2.3.2.1.	Prime Numbers $p$ and $q$ :	15
2.3.2.2.	Modulus $n$ :	16
2.3.2.3.	Totient $\phi(n)$ :	16
2.3.2.4.	Public Exponent $e$ :	16
2.3.2.5.	Private Exponent $d$ :	16
2.3.3.	RSA Key Generation Process	17
2.3.4.	How RSA Works	17
2.3.5.	Security of RSA	18
2.3.6.	Why RSA is Suitable for PKI Systems	18
2.4.	Authentication and Key Management	19
2.4.1.	Components of Key Management	19
3.	Workflow of Memory Card-Based PKI in Your System	20
3.1.	Key Generation	20
3.2.	Data Signing Process	20
3.3.	Verification Process	21
4.	Advantages and Limitations of Memory Card-Based PKI	22
4.1.	Advantages	22
4.2.	Limitations	22
5.	Applications and Use Cases	22
5.1.	Diplomas and Certificates	22
5.2.	Identification and Access Control	22
5.3.	Business and Financial Applications	23
6.	Comparison with Cryptographic Card-Based PKI	23
6.1.	Cost	23
6.2.	Security	23
6.3.	Private Key Storage	24
6.4.	Cryptographic Operations	24
6.5.	Complexity	24
6.6.	Performance	25
6.7.	Scalability	25
6.8.	Flexibility	25
6.9.	Maintenance	26

6.10.	Tamper Resistance .....	- 26 -
7.	Conclusion .....	- 26 -
Chapter III: EDiploma an electronic authentic diploma .....		- 27 -
1.	Introduction.....	- 27 -
2.	System Components and Architecture.....	- 27 -
2.1.	API Architecture .....	- 28 -
2.1.1.	Overview.....	- 28 -
2.1.2.	Key Components and Endpoints .....	- 28 -
2.1.2.1.	Admin API endpoints .....	- 28 -
2.1.2.2.	Assistant API endpoint.....	- 29 -
2.1.2.3.	Signer API endpoint .....	- 29 -
2.1.2.4.	Verifier API endpoint .....	- 30 -
2.1.3.	API Flow .....	- 30 -
2.2.	API Security.....	- 30 -
2.2.1.	Authentication:.....	- 31 -
2.2.2.	Encryption: .....	- 31 -
2.2.3.	Key Splitting:.....	- 31 -
2.3.	Application user interface .....	- 31 -
2.3.1.	SmartCardAdmin: .....	- 31 -
2.3.2.	SmartCardAssistant: .....	- 33 -
2.3.3.	SmartCardSigner:.....	- 34 -
2.3.4.	SmartCardVerifier:.....	- 35 -
3.	Key Management Processes.....	- 35 -
3.1.	User Creation and Authentication.....	- 35 -
3.1.1.	Request Example: .....	- 35 -
3.1.2.	Response Examples: .....	- 36 -
3.2.	Key Generation.....	- 36 -
3.2.1.	Key splitting algorithm.....	- 36 -
3.2.2.	Key Reconstruction Algorithm.....	- 38 -
3.3.	Key Distribution .....	- 40 -
4.	Data Signing and Verification .....	- 41 -
4.1.	Data Signing Process.....	- 41 -

4.2. Verification Process .....	- 42 -
5. Security Considerations.....	- 43 -
5.1. Key Security .....	- 43 -
5.2. Revocation and Recovery .....	- 44 -
6. Use Cases and Applications .....	- 44 -
7. Conclusion .....	- 45 -
Conclusion .....	- 46 -
References.....	- 47 -





## List of figures

Figure 1: System Architecture.....	- 19 -
Figure 2: Key Generation Process.....	- 20 -
Figure 3: Signing Process .....	- 21 -
Figure 4: Verification Process .....	- 21 -
Figure 5: System overview.....	- 28 -
Figure 6: Standard login UI .....	- 31 -
Figure 7: Main view of the admin.....	- 32 -
Figure 8: Add new user UI .....	- 32 -
Figure 9: Admin key manager for users .....	- 33 -
Figure 10: Assistant main view .....	- 33 -
Figure 11: Add new diploma record view .....	- 34 -
Figure 12: Signer main view .....	- 34 -
Figure 13: Verifier app view .....	- 35 -

**List of abbreviations:**

<b>Number</b>	<b>Abbreviation</b>	<b>Description</b>
<b>1</b>	<b>EDIPLOMA</b>	Electronic Diploma
<b>2</b>	<b>PKI</b>	Public Key Infrastructure
<b>3</b>	<b>CA</b>	Certificate Authority
<b>4</b>	<b>RA</b>	Registration Authority
<b>5</b>	<b>HSMs</b>	Hardware Security Modules
<b>6</b>	<b>RF</b>	Radio Frequency
<b>7</b>	<b>EHRs</b>	Electronic Health Records
<b>8</b>	<b>UTXO</b>	Unspent Transaction Output
<b>9</b>	<b>TLS</b>	Transport Layer Security
<b>10</b>	<b>SSL</b>	Secure Sockets Layer
<b>11</b>	<b>API</b>	Application Programming Interface
<b>12</b>	<b>JWT</b>	JSON Web Tokens
<b>13</b>	<b>RSA</b>	Rivest–Shamir–Adleman
<b>14</b>	<b>AES</b>	Advanced Encryption Standard
<b>15</b>	<b>DES</b>	Data Encryption Standard
<b>16</b>	<b>SLE5542</b>	A specific type of memory card used in the project
<b>17</b>	<b>UI</b>	User Interface
<b>18</b>	<b>EFT</b>	Electronic Fund Transfers



## **Introduction**

Authentication is one of the fundamental pillars in information security and cryptography and part of our ever digitally transforming world. Cryptographic authentication is what actually enables secure communication by protecting both sensitive data in transmission from unauthorized access ensuring the messages transmitted only to the intended recipient.

Cryptography provides the tools for encrypting data, while authentication mechanisms verify the identities of users and systems involved in communication. These technologies are essential in various applications, including online banking, secure messaging, and digital signatures.

The need for a secure and reliable possibility of verifying the credential on a certificates and diplomas in the education sector is ever arising. Also, the proliferation of digital diplomas and certificates has raised concerns regarding the authenticity of such certificates from fraud.

This thesis presents a solution to these challenges through the development of EDIPLOMA, a software system designed for creating electronically signed diplomas. By integrating robust cryptographic techniques and authentication methods, EDIPLOMA ensures that diplomas are both tamper-proof and easily verifiable.

This thesis is organized as follows:

The first chapter provides an overview of cryptographic methods, discussing their principles and applications in securing information. The second chapter focuses on authentication techniques, exploring various methods for verifying identities in digital environments while the third chapter details the design and implementation of the EDIPLOMA system, highlighting how cryptography and authentication work together to enhance the security of electronically signed diplomas. Finally, a conclusion concludes the thesis.



# **Chapter I: Foundations of Cryptography and Authentication**

## **1. Introduction**

Cryptography is the branch of computer science that study ways to defend data and communications in all its form using mathematical complex algorithms [1]. It is based on the scrambling of vulnerable data and exchanged messages using mathematical standards and calculations, the objective is to change messages using approaches that are very challenging to reverse encrypted data to its original form if not impossible or impractical to gain real benefits from such a breach. These algorithms can be used for cryptographic key generation, digital signature, authentication and authorization [2].

Cryptography is closely related to the disciplines of cryptology and cryptanalysis. These consists of techniques such as microdots, merging phrases with images and one-of-a-kind techniques to hide information in storage or in-transit. The most common form cryptography scrambling plaintext (ordinary text, often referred to as cleartext) into ciphertext (the act of encryption), then vice versa from ciphertext to clear text (the act of decryption). Individuals who specialize in this field are known as cryptographers [2, 3].

## **2. Cryptographic Authentication**

The form of cryptography that is of interest to us is known as cryptographic authentication, a way to enable us from different parties, governments, institutions and others to verify the validity of the sender of some data. This works by transferring the data with complete confidence and honesty as the data is validated through what can be called a hash or a signature. A simple hash [4] allows us to verify the integrity of the data and that the data has not been tempered with, meanwhile, a signature ensures that only the sender which we know through special techniques in cryptography as the sole sender and source of the information exchanged.

### *2.1.1. Symmetric Key Cryptography*

Symmetric Key Cryptography additionally recognized as Symmetric Encryption is when a secret key is used for each act of encryption and decryption. This technique is to the contrary of Asymmetric Encryption the same key that is used to encrypt is used to decrypt. During this process, data (or a message) is transformed to a new form that can't be examined or inspected by any entity that does not have the secret key that was used to encrypt it [3].

The success of this method relies upon on the power of the randomness of the generator that is used to create the secret key. Symmetric Key Cryptography is extensively used in latest Internet and particularly consists of two kinds of algorithms, Block and Stream [5]. Some frequent encryption algorithms consist of the Advanced Encryption Standard (AES) [6] and the Data Encryption Standard (DES) [7]. This structure of encryption is historically quicker than Asymmetric on the other hand it requires each the sender and the recipient of the information to have the secret key.

### *2.1.2. Asymmetric Key Cryptography*

Asymmetric cryptography, additionally recognized as public key cryptography, is a manner that makes use of a pair of associated keys -- one public key and one private key -- to encrypt and decrypt a message and shield it from unauthorized access or use. A public key is a cryptographic key that can be represented as a set of characters that can be used to encrypt a message so it can solely be decrypted by the recipient with their private key. A private key - additionally recognized as a secret key - is solely kept with the secret generator. When anyone want to transfer an encrypted message, they get the recipient's public key from a public listing and use it to encrypt the message before sending it. The recipient of the message can decrypt the message using their private key. In this case the use case of such technique ensures the confidentiality in a way only the recipient of a message can know their content (even though the encryption key is public) [3, 5].



The second use case of asymmetric cryptography is if the sender encrypts the message using their own private key, the message can be decrypted solely using that sender's public key, as a consequence authenticating the source of the message which can be used in this case for the either digital signatures [8] or to test data integrity that implies beyond any doubt that no one has tempered with the original message or data.

Such techniques are done in a transparent way, this allows for many protocols to count on asymmetric cryptography, including the Transport Layer Security (TLS), Secure Sockets Layer (SSL) which in turn makes HTTPS [8] possible thus, secure internet communication.

This encryption method is additionally used in other software that require setting up a secure connection over an insecure network, such as some internet browsers due to its secure nature reducing the probabilities of a cybercriminal discovering a user's private key through all form of transmission.

### *2.1.3. Uses of asymmetric cryptography*

#### 2.1.3.1. Digital signatures

Asymmetric cryptography is normally used to authenticate information with the aid of digital signatures. A digital signature is a mathematical approach that validates the authenticity and integrity of a message, software program or digital document. It is the digital equal of a handwritten signature or stamped seal [5, 3].

Based on asymmetric cryptography, digital signatures can furnish assurances of proof to the origin, identification and reputation of a digital document, transaction or message, as effective and acknowledgeable consent by using the signer [8, 10].

#### 2.1.3.2. Communications

Asymmetric cryptography can additionally be utilized in software that allow many users to encrypt and decrypt messages, which include the following:

- Encrypted email. A public key can encrypt an e-mail message, and a personal key can decrypt it.
- SSL/TLS [9]. Establishing encrypted hyperlinks between web sites and browsers additionally makes use of uneven encryption.

#### 2.1.3.3. Cryptocurrencies

Bitcoin and other cryptocurrencies use asymmetric cryptography. Users have public keys that everybody can see and private keys that are saved secret. Bitcoin makes use of a cryptographic algorithm to make sure solely respectable proprietors can spend the funds [11].

In the case of the Bitcoin ledger, every unspent transaction output (UTXO) [12] is generally attached to a public key. For example, if person X, who has an UTXO attached to their public key, wishes to send the cash to person Y, X makes use of their private key to initiate a transaction that spends the UTXO and creates a new UTXO attached to Y's public key [12].

### 3. Public key infrastructure

Some companies today, they count on the **PKI** which stands for public key infrastructure. The idea behind is based on asymmetric cryptography where each user is assigned a pair of keys (public and private) that are within a so-called certificate that is in itself issued and can be verified by a trusted third party called an issuer. These certificates can either be used by people, devices, and applications [13].

PKI safety first emerged in the Nineties to assist and govern encryption keys through the issuance and administration of digital certificates [3]. These certificates can be used to verify the proprietor of a private key and the authenticity of a possible communication going through in order to assist and preserve security [14].

One of the most common examples of PKI in used nowadays are SSL (secure socket layer) certificates on web sites so that web page site visitors understand they're

sending data to the supposed recipient (the original website not an imposter), digital signatures and authentication for Internet of Things devices [15] making sure only the ones that belong to the network are able to communicate.

### **3.1. Components of a PKI**

#### *3.1.1. Certificate authority*

The certificate authority or CA is a trusted entity that issues, stores, and verifies a digital certificate. The CA verifies the digital certificates with their very own private key and then publishes the public key that can be accessed upon request [16].

#### *3.1.2. Registration authority*

Registration authority or RA is a mediator that verifies the identity of entities before certificates are issued [13].

#### *3.1.3. Certificate database*

This database is responsible to save the certificate data in a central and accessible form to other parts of the PKI system [16].

#### *3.1.4. Certificate administration system*

This is a form of a computer that manages and transport the certificate and their access rights.

#### *3.1.5. Certificate policy:*

This coverage outlines the processes of the PKI. It can be used by means of outsiders to decide the PKI's trustworthiness [16].

#### *3.1.6. Certificate (Public / private key pair)*

The certificate containing the key pairs used in cryptographic operations are usually stored in a secure place, the private part is only kept with the user.

Some high-end use cases require users to save their private keys in a specialized hardware called hardware security modules (HSMs) that are hardened, tamper-resistant hardware units made to secure some cryptographic operations such as encryption and decryption and even digital signature and certificate generation and storage without divulging the secret keys used to do such operations [17]. Using such equipment can be found in the most secure data centers and also some advanced banking systems [17].

Other users may use smart cards details of which are presented below.

## **4. Smartcards:**

Smart cards can be classified into multiple categories, not all smart cards are equipped with the capability to either store cryptographic keys or perform cryptographic operations. Smart card can be categorized based on two main factors.

### **4.1. Classification of Smartcards**

#### *4.1.1. Based on communication interface*

There are three main types of smart cards based on the interface in which data is transferred to and from the card. Contact smart cards, contactless smart cards and dual interface smart cards.

##### 4.1.1.1. Contact smart card

These require having a physical contact that allows communications using a reader where the card must be inserted in order to perform any operations.

Common use cases for contact smart card include but not limited to: payment cards, identity cards and access control cards.

##### 4.1.1.2. Contactless smart card

Unlike contact smart cards, these can communicate with a reader without physical contact using radio frequency (RF) technology [18]. In such cards the data in the card can be transmitted wirelessly when near the card reader.

Common use cases for contactless smart card include but not limited to: contactless payment cards, public transport card and access control cards.

#### 4.1.1.3. Dual interface smart card

These combine both the technologies in the contact and contactless smart cards, common use cases of such cards can be found in payment cards that offer multiple form of payment.

#### 4.1.2. *Based on the function of the card*

Unrelated to the type of card, smart cards can be categorized based on their inner functions as follows:

##### 4.1.2.1. Memory card

Memory card smart cards allow users to store limited size data, some can be written protected using a pin while others offer more simple storage solution without any protection. Memory cards have no processing capabilities which makes them the least expensive type of smart cards. Whether contact, contactless or dual interface, some use cases of memory cards can be prepaid phone cards, prepaid gift cards and also, basic identification cards.

##### 4.1.2.2. Microprocessor smart card

Unlike memory only cards, these contain in each card a microprocessor chip that can perform some operations such as encryption, authentication and data management. Some use cases for such card can be in secure payment systems, secure identification, healthcare system and government identity cards such as passport or biometric IDs.

##### 4.1.2.3. Cryptographic smart card

These cards contain chips like their microprocessor counterpart that are designed specifically for cryptographic operations only such as encryption, decryption, digital signature, and secure key storage.

#### 4.1.2.4. Hybrid smart card

Unlike memory, microprocessor and cryptographic smart cards, these may contain multiple chips for different needs based on specific application requirements. Usually, the hardware inside this type of cards is specialized for cryptographic operations to perform such operation quicker than a simple microprocessor chip. Uses of such card is usually limited for secure login, digital signatures, and secure communications.

## 5. Data Authentication Using Digital Signatures

This section explores digital signatures, their role in ensuring both data authenticity and integrity and known use cases within a public key infrastructure. Such signatures which are based on mathematically proven algorithms are a vital tool for both secure communications and the verification of either digital document or transactions authenticity in a cryptographic system.

### 5.1. Importance of Digital Signatures

In the cryptographic world in order to prove the source of a data (message, document, a communication ...etc.) and just like a handwritten signature, a digital signature serves as proof of both the identity and consent of the data creator or the one responsible for its authorization, in order to ensure that the data has not been modified either during transmission or after it was saved on a storage server. Before we delve into the importance of digital signature, below is a definition of what is a digital signature.

#### 5.1.1. Definition:

A signature that is cryptographically calculated is based on a mathematical scheme in order to demonstrate the authenticity of a digital data [8]. The idea takes use of asymmetric cryptography as explained before, the private key is used to generate a signature. Given the data, signature and the public key as input the recipient can therefore verify the origin of the data since we know the owner of the public key.

### 5.1.2. Importance:

Digital signatures are crucial for secure communication in today's digital world.

They ensure that:

- **Authenticity:** The sender's identity is verified, providing assurance that the message or document is from the legitimate source.
- **Integrity:** The content of the message has not been tampered with since it was signed.
- **Accountability:** Once signed, the sender cannot deny having signed the document, ensuring accountability. As no one other than the one in possession of the private key can perform the signature.

Digital signatures are an essential part of secure email communication, financial transactions, software distribution, and legal documentation in various industries.

## 5.2. Digital Signature Authentication mechanism

Digital signatures work using cryptographic processes in order to ensure the integrity and authenticity of some data being transmitted.

### 5.2.1. Generation

- A **hash function** is applied to the original data [4]. This results in a unique hash value (also called a digest) that represents the data. A hash function is usually a mathematically one-way function that allows the generation of a unique set of characters based on an input (the data) whenever a single bit changes the resulting hash will eventually change.
- After getting the hash of the data, the sender will employ their **private key** to encrypt the hash value thus, creating the final digital signature. And since the private key is only known to the sender it ensures no one else can generate such a signature.

### 5.2.2. Verification

- When a recipient receives a signed data (or in our case a document), they also receive the digital signature.
- The recipient decrypts the digital signature using the sender's **public key**, obtaining the hash value that was originally created.
- The recipient then applies the same hash function applied by the data sender. If the two hash values match this automatically proves that the document has not been altered, ensuring its **integrity**.
- Since the public key corresponds only to the sender's private key mathematically, this also confirms the **authenticity** of the sender.

Through this process, digital signatures play a vital role in maintaining trust in digital communications, preventing both forgery and data tampering.

## 5.3. Real-World Examples of Digital Signature Applications

Digital signatures are used in a wide variety of industries and applications, ensuring secure and authenticated communication in many critical systems:

### 5.3.1. E-Government Services

As an integral part of multiple modern digitalized governments secure communications, digital signature makes for a corner stone for such systems as it allows not just signing but secure signing and non-repudiation and avoid signees to sign forms, legal documents and even contracts ensuring a high level of both security and authenticity.

### 5.3.2. Financial Transactions

Some banking and finance institutions opt for using digital signatures in their electronic fund transfers (EFT) and also for online banking transactions. A digital signature is used to ensure the origin of the transaction, and also to determine the rightful account holder.



### *5.3.3. Legal Documents*

In both legal and corporate sectors, some institutions tend to use digital signatures in order to sign contracts, agreements, and even other legal documents. The binding nature of such digital signatures depends greatly on the laws governing the place and the way it was issued but technically speaking it's still mathematically binding.

### *5.3.4. Software Distribution*

Whether to verify the source and integrity of an update or a downloaded software package by an already installed, digital signatures are used. Another example is operating systems to verify a software or an update being installed and that it comes from a trusted source and that it has not been altered during transfer from a server to the client's computer.

### *5.3.5. Healthcare*

EHRs short for Electronic Health Records, are part of the modern health systems including prescriptions. In order to ensure the patient information is securely transferred and verified between medical professionals, digital signatures are used widely.

In all of the above cases, the use of digital signatures provides a way to verify the authenticity of the source of data, and also to ensure the data integrity preventing tampering during digital exchanges.

## **6. CONCLUSION**

Cryptography and validation are crucial for modern-day cybersecurity. Cryptography protects facts from unauthorized access, whilst validation ensures its authenticity and integrity. By grasp and enforcing these technologies, companies can efficaciously guard their touchy information.

## Chapter II: Memory card based PKI

### 1. Introduction

A Public Key Infrastructure (PKI) is an essential framework for securing communication and managing digital identities. Memory cards, despite being non-cryptographic, can play a vital role in storing private keys and supporting the signing process in a PKI system.

In this chapter, we explore the integration of memory cards into a PKI, focusing on key storage, authentication, and digital signatures. We will as well discuss the practical use of memory cards in secure communication, particularly in environments where cost and ease of use are primary considerations. By examining the components, workflows, advantages, and limitations, we present memory cards as a cost-effective solution in PKI implementations.

### 2. Components of a Memory Card-Based PKI System

#### 2.1. Memory Card

A memory card is a storage device that can hold data such as private keys in a secure manner. Unlike cryptographic smart cards, memory cards like the SLE5542 used in our project do not perform any cryptographic operations but only serve as key storage.

In our system **SLE5542** will only be used to store key parts in its memory for retrieval during the signing process.

- **Key characteristics of SLE5542:**
  - **Memory Type:** EEPROM with 256 bytes.
  - **Protection:** PIN-based write protection, offering a basic security layer.
  - **Applications:** Typically used in access control and low-security applications.

## 2.2. Private Key Storage

In our memory card-based PKI, private keys are split between the server (API) and the memory card. This splitting method ensures that no single entity (the server or the card) has complete control over the key, thus lowering the cost while at the same time enhancing security.

## 2.3. Overview of RSA Cryptography

### 2.3.1. Introduction to RSA Algorithm

RSA (Rivest–Shamir–Adleman) algorithm as it was named after its mathematicians' creators [19] is one of the first and most used public-key cryptosystems. Unlike symmetric encryption algorithms, and as explained in chapter 1 RSA uses a pair of keys: a public key and a private key. These pair of keys are generated through a mathematical process that ensures that data encrypted with the public key can only be decrypted by the private key, and vice versa.

In any modern PKI, RSA can play an integral role for the two following functionalities:

- **Encryption:** Protecting the confidentiality of messages.
- **Digital Signatures:** Ensuring the authenticity and integrity of messages.

### 2.3.2. Key Components of RSA

The RSA cryptosystem revolves around the following core components:

#### 2.3.2.1. Prime Numbers $p$ and $q$ :

- The RSA algorithm begins by selecting two large prime numbers,  $p$  and  $q$ . These primes are kept secret, and their product forms the modulus  $n$ , which is used in both the public and private keys.

- The security of RSA relies on the difficulty of factoring this product  $n$  back into  $p$  and  $q$ , a process that is computationally infeasible for sufficiently large primes.

#### 2.3.2.2. Modulus $n$ :

- The modulus  $n$  is calculated as the product of  $p$  and  $q$ :  $n=p \times q$
- The modulus is a part of both the public key and the private key, and it defines the size of the RSA key. Typically,  $n$  is a very large number, usually 2048 or 4096 bits long, which ensures the strength of the encryption.

#### 2.3.2.3. Totient $\phi(n)$ :

- The totient function  $\phi(n)$  is calculated as  $\phi(n)=(p-1)(q-1)$ . It represents the number of integers less than  $n$  that are coprime to  $n$ . This value is used during the key generation process to create the private key.

#### 2.3.2.4. Public Exponent $e$ :

- The public exponent  $e$  is an integer that is typically small and chosen such that  $1 < e < \phi(n)$ , and  $e$  is coprime with  $\phi(n)$ . A common choice for  $e$  is 65537 because it provides a good balance between security and computational efficiency.
- The public key is comprised of  $e$  and  $n$ , and it is used to encrypt messages or verify digital signatures.

#### 2.3.2.5. Private Exponent $d$ :

- The private exponent  $d$  is calculated such that it satisfies the equation  $d \times e \equiv 1 \pmod{\phi(n)}$ . This ensures that  $d$  is the modular inverse of  $e$  with respect to  $\phi(n)$ .
- The private key is made up of  $d$  and  $n$ , and it is used to decrypt messages encrypted with the public key or to generate digital signatures.

### 2.3.3. RSA Key Generation Process

The key generation process for RSA is as follows:

- **Step 1:** Select two large prime numbers  $p$  and  $q$ .
- **Step 2:** Compute  $n=p \times q$ .
- **Step 3:** Compute the totient  $\phi(n)=(p-1)(q-1)$ .
- **Step 4:** Choose a public exponent  $e$  such that  $1 < e < \phi(n)$  and  $\gcd(e, \phi(n))=1$ .
- **Step 5:** Compute the private exponent  $d$ , where  $d \times e \equiv 1 \pmod{\phi(n)}$ .
- **Step 6:** Publish  $(e, n)$  as the public key and keep  $(d, n)$  as the private key.

Once generated, the public key can be distributed openly, while the private key must be kept secret.

### 2.3.4. How RSA Works

#### 1. Encryption:

- To encrypt a message  $M$  using RSA, the sender first obtains the recipient's public key  $(e, n)$ . The message  $M$  is then converted into an integer  $m$ , such that  $0 \leq m < n$ .
- The encryption process is performed by computing the ciphertext  $c$  as:  
 $c = m^e \pmod{n}$ .
- The sender transmits  $c$  (the encrypted message) to the recipient.

#### 2. Decryption:

- To decrypt the ciphertext  $c$ , the recipient uses their private key  $(d, n)$ . The decryption process is performed by computing:  $m = c^d \pmod{n}$ .
- The recipient then converts  $m$  back into the original message  $M$ .

#### 3. Digital Signatures:

- To sign a message  $M$ , the sender uses their private key  $(d, n)$ . The message  $M$  is first hashed using a secure hash function to create a message digest  $H(M)$ .
- The sender then computes the signature  $S$  as:  $S = H(M)^d \bmod n$ .
- The signature  $S$  is sent along with the message to the recipient.

#### 4. **Signature Verification:**

- To verify the signature, the recipient uses the sender's public key  $(e, n)$  to compute:  $H'(M) = S^e \bmod n$
- The recipient compares  $H'(M)$  to the hash of the original message  $H(M)$ . If they match, the signature is valid, and the message is authenticated.

#### 2.3.5. *Security of RSA*

The security of RSA relies on the difficulty of the **integer factorization problem**. Given a modulus  $n$  that is the product of two large primes  $p$  and  $q$ , it is computationally infeasible to factor  $n$  into its prime factors. This ensures that an adversary cannot derive the private key  $d$  from the public key  $(e, n)$ .

RSA's security also depends on choosing large prime numbers and ensuring that the public and private exponents are chosen correctly. If poorly chosen or too small, these values can lead to vulnerabilities such as **low-exponent attacks** or **factorization attacks**.

#### 2.3.6. *Why RSA is Suitable for PKI Systems*

RSA is widely used in PKI systems because:

- It supports both encryption and digital signatures, enabling secure communication and data integrity.
- It allows for **asymmetric key management**, meaning that private keys can be kept secret, while public keys can be distributed openly.

- RSA’s flexibility in key lengths (e.g., 2048-bit, 4096-bit) allows it to provide scalable security, making it suitable for systems that require long-term data protection.

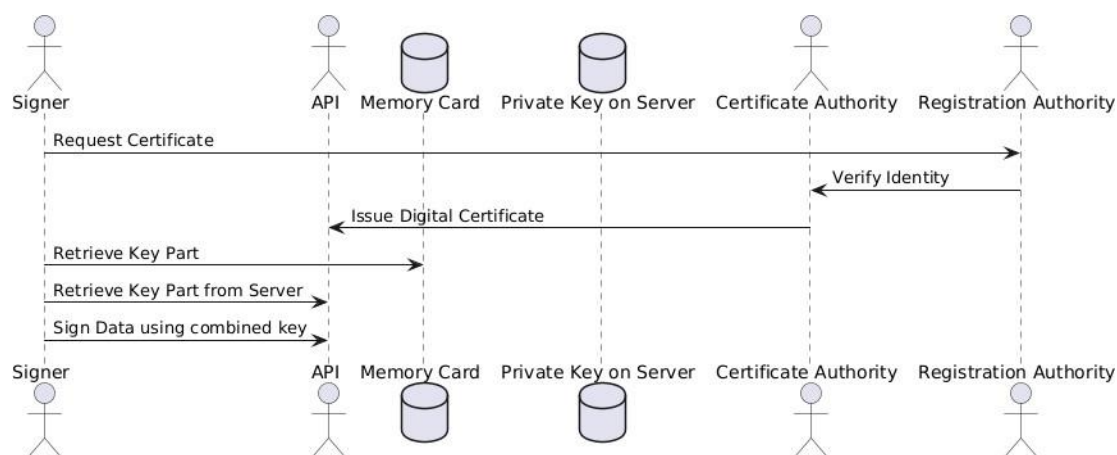
In the context of this thesis, RSA forms the foundation of the proposed key-splitting mechanism, where the private key is divided between the API server and the memory card. This splitting adds an additional layer of security by ensuring that no single entity has access to the entire private key, thus mitigating risks associated with key exposure.

## 2.4. Authentication and Key Management

Key management in memory card-based PKI systems is essential for secure communication. The memory card does not perform encryption but serves as secure storage. When a signer retrieves the private key part from the memory card, it is combined with the key part from the API to sign the data.

### 2.4.1. Components of Key Management

- **Certificate Authority (CA):** Issues and manages digital certificates.
- **Registration Authority (RA):** Verifies user identities before certificates are issued.
- **Memory Card:** Stores part of the private key.
- **API:** Stores the other part of the private key and manages user authentication.



**Figure 1: System Architecture**

### 3. Workflow of Memory Card-Based PKI in Your System

#### 3.1. Key Generation

In this system, private keys are generated for users by the administrator (SmartCardAdmin), who splits the key between the memory card and the API. This ensures that the key is not stored entirely in one place, thus reducing the risk of key theft or exposure.

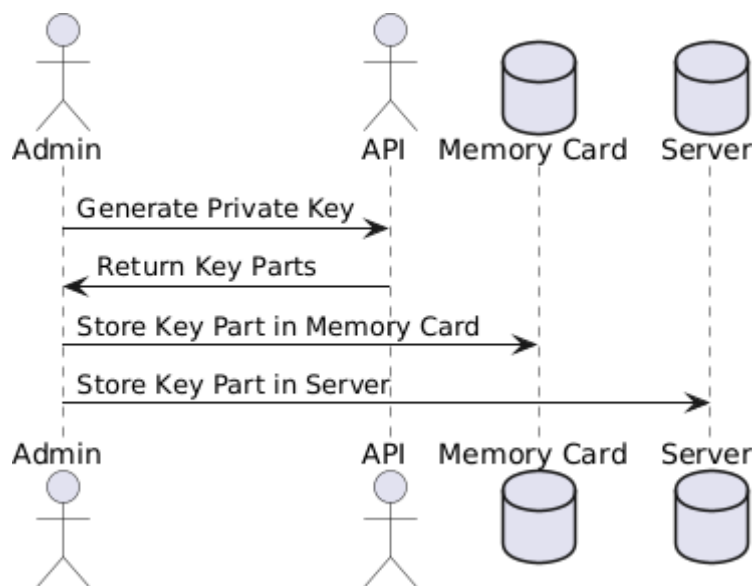


Figure 2: Key Generation Process

#### 3.2. Data Signing Process

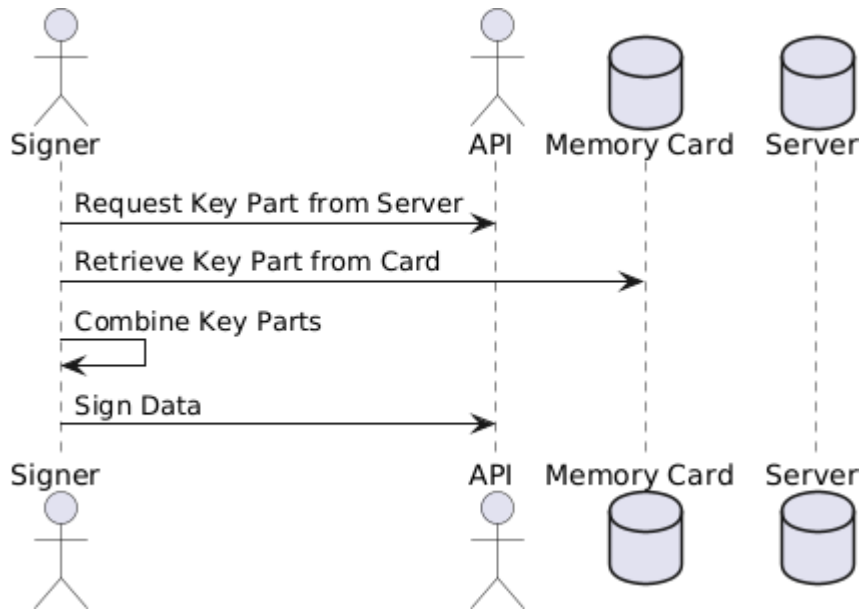
During the signing process, the SmartCardSigner retrieves the key parts from both the memory card and the server. The data is signed using the complete key, and a digital signature is generated.

##### Steps in the Data Signing Process:

1. The signer requests the key part stored on the server.
2. The signer retrieves the key part from the memory card.
3. The two key parts are combined.



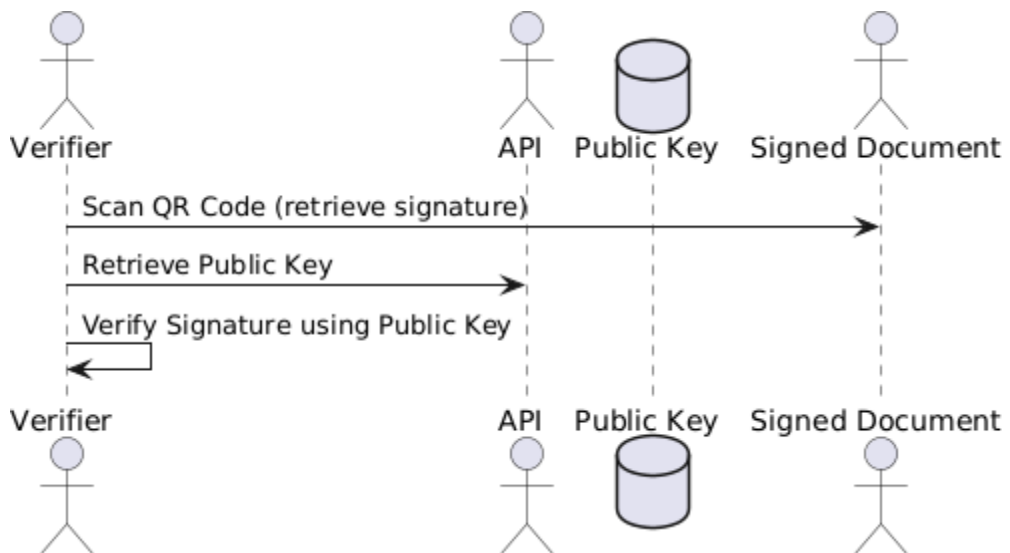
4. The signer signs the document.



**Figure 3: Signing Process**

### 3.3. Verification Process

Verification is a straightforward process. The verifier uses the public key and signature provided in the document or diploma to validate the signature. This is typically done without the need for the card or private key.



**Figure 4: Verification Process**

## 4. Advantages and Limitations of Memory Card-Based PKI

### 4.1. Advantages

- **Cost-Effective:** Memory cards, like the SLE5542, are significantly cheaper than cryptographic smart cards, making them a viable option for large-scale deployments.
- **Enhanced Security:** By splitting the private key between the memory card and the server, the risk of key theft is reduced.
- **Simplicity:** The memory card system is easier to deploy and manage than systems that require complex cryptographic hardware.

### 4.2. Limitations

- **Security Limitations:** Memory cards like the SLE5542 do not offer built-in cryptographic functionality, making them more vulnerable to physical attacks.
- **External Security Dependencies:** Since memory cards do not perform cryptographic operations, additional layers of security (e.g., encryption) are needed on the server side to protect the key parts.

## 5. Applications and Use Cases

### 5.1. Diplomas and Certificates

One of the primary use cases for memory card-based PKI is the secure issuance and verification of diplomas. By signing diplomas digitally and including a QR code for verification, institutions can ensure that diplomas are tamper-proof and easily verifiable.

### 5.2. Identification and Access Control

Memory cards can be used for identification purposes, storing key information that allows individuals to authenticate themselves when accessing secure areas or systems.

### 5.3. Business and Financial Applications

In the financial sector, memory cards can be employed for secure document signing and transaction verification, particularly in environments where cost-effective security solutions are needed.

## 6. Comparison with Cryptographic Card-Based PKI

In this section, we compare the use of memory cards (non-cryptographic cards) with cryptographic smart cards (microprocessor-based) in the context of a Private Key Infrastructure (PKI). The comparison covers key aspects such as cost, security, key storage, performance, and scalability.

### 6.1. Cost

- **Memory Cards:** These cards are significantly cheaper as they lack embedded processors or advanced security features. This makes them cost-effective, especially when large numbers of users are involved.
- **Cryptographic Smart Cards:** The built-in processors and security features make these cards more expensive.
- *Justification:* Memory cards are a more affordable option for large-scale deployments, while cryptographic smart cards require a higher initial investment.

### 6.2. Security

- **Memory Cards:** They offer lower security and are vulnerable to physical attacks like cloning and tampering, as they do not perform cryptographic operations directly.
- **Cryptographic Smart Cards:** These cards are highly secure, storing keys and performing cryptographic operations on the card itself. This prevents key extraction and ensures a higher level of protection.

- *Justification:* While memory cards rely on external servers for security, cryptographic smart cards are self-contained and provide superior tamper resistance.

### 6.3. Private Key Storage

- **Memory Cards:** The private key is split, with part stored on the server and part on the card. This increases complexity and exposes the system to more attack surfaces.
- **Cryptographic Smart Cards:** The private key is securely stored entirely within the card and never leaves the chip, minimizing the risk of exposure.
- *Justification:* Memory cards add dependency on server security, whereas cryptographic cards offer a more secure, simplified approach to key storage.

### 6.4. Cryptographic Operations

- **Memory Cards:** Cryptographic operations like signing and decryption must be performed externally on a server.
- **Cryptographic Smart Cards:** These operations are carried out directly on the card, keeping sensitive data secure within the card's hardware.
- *Justification:* Offloading cryptographic operations to a server increases risk, while cryptographic smart cards ensure more secure and efficient operations.

### 6.5. Complexity

- **Memory Cards:** The system is more complex as it requires communication between the server and the card to retrieve part of the private key and perform operations.
- **Cryptographic Smart Cards:** Operations are handled on the card, making key management simpler and reducing overall system complexity.

- *Justification:* Memory cards require more infrastructure for secure key management, while cryptographic smart cards handle operations independently, reducing the complexity.

## 6.6. Performance

- **Memory Cards:** Performance depends on the server and network, and delays can occur during cryptographic operations due to server-side processing and network overhead.
- **Cryptographic Smart Cards:** These cards perform cryptographic operations on-chip, improving speed and reducing latency.
- *Justification:* Memory cards may introduce delays, while cryptographic cards offer faster, more efficient cryptographic operations.

## 6.7. Scalability

- **Memory Cards:** Their lower cost makes it easier to scale the system to accommodate a large number of users or devices.
- **Cryptographic Smart Cards:** The high cost of these cards makes scaling more expensive.
- *Justification:* For large deployments where cost is a concern, memory cards offer better scalability.

## 6.8. Flexibility

- **Memory Cards:** They are primarily used for storing data and rely on external servers for cryptographic operations.
- **Cryptographic Smart Cards:** These cards are more flexible, supporting multiple applications such as secure payments and user authentication.
- *Justification:* Cryptographic cards can be used for a wider range of applications, providing more flexibility in their deployment.

## 6.9. Maintenance

- **Memory Cards:** Maintenance is higher due to the need for server-side management of cryptographic operations and key storage.
- **Cryptographic Smart Cards:** Once issued, cryptographic cards require minimal maintenance and reduce server dependencies.
- *Justification:* Memory cards require more server maintenance, whereas cryptographic smart cards offer lower long-term maintenance requirements.

## 6.10. Tamper Resistance

- **Memory Cards:** These cards are more susceptible to tampering or duplication, as they do not contain on-card security mechanisms.
- **Cryptographic Smart Cards:** They are designed with tamper-resistant hardware, preventing key extraction and physical attacks.
- *Justification:* Cryptographic cards provide better protection against tampering and unauthorized access.

## 7. Conclusion

This chapter has presented a comprehensive view of memory card-based PKI systems, focusing on the role of memory cards as secure storage for private keys in environments where cost and ease of deployment are key concerns. While memory cards like the SLE5542 lack cryptographic functionality, splitting the key across the memory card and server provides enhanced security. The next chapter will explore secure communication and the challenges involved in real-world PKI implementations.

## Chapter III: EDiploma an electronic authentic diploma

### 1. Introduction

This chapter presents the detailed implementation of a **memory card-based Public Key Infrastructure (PKI)** that allows for the input and signature of a diplomas to be used by education systems. It explains how key actions such as user creation, key management, data input, signing, and verification are handled through four C# desktop applications interacting with the API and memory cards.

The core features of this system rely on a novel algorithm for private key splitting, where the private key is stored in two parts—one part on the memory card and the other part on the API server. The system is designed to manage digital signatures for the diplomas, ensuring their integrity and authenticity.

### 2. System Components and Architecture

The system we are building is based on a Public Key Infrastructure (PKI) that uses a private key split between a server-side API and a smart card for secure data signing. The system consists of four primary applications:

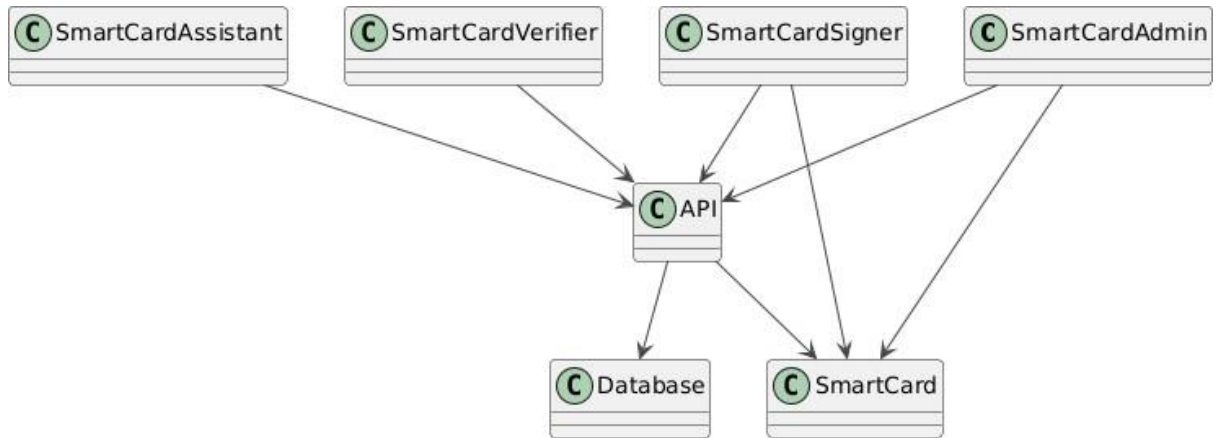
**SmartCardAdmin:** Used to create users and generate their keys, storing one part of the private key on a memory card and the other part in the API.

**SmartCardAssistant:** Allows assistants to input data (such as diploma records) and assign it to signers.

**SmartCardSign:** Enables users to sign documents by combining the two parts of their private key (from the memory card and the API).

**SmartCardVerify:** Verifies the authenticity of signed documents by checking their signatures.

These applications work along with the **API** that acts as a central server for managing user authentication, key storage, data storage and the signing process.



**Figure 5: System overview**

## 2.1. API Architecture

### 2.1.1. Overview

The API acts as the backbone of the entire system, handling secure communication between clients (the four applications) and the server. It provides endpoints for user management, key generation, data signing, and signature verification. The architecture follows RESTful principles, with stateless interactions, secure data transmission, and clearly defined roles for each component.

The central API interacts with the following:

- **Admin API** (/adminApi): Manages users, key generation, and administrative tasks.
- **Assistant API**: Submits data for signing.
- **Signer API**: Retrieves key parts to sign the data.
- **Verifier API**: Validates signatures against stored public keys.

### 2.1.2. Key Components and Endpoints

#### 2.1.2.1. Admin API endpoints

- **Purpose**: Provides endpoints for user management and key generation.



- **Endpoints:**
  - /createUser: Creates new users in the system.
  - /addUserKey: Adds cryptographic keys for users.
  - /updateUser: Updates user information.
  - /listUsers: Returns a list of all users in the system.
  - /listUserKeys: Returns a list of all keys in the system for a selected user.
  - /enableUserKey: allows the admin to set a key as enabled and can be used.
  - /disableUserKey: allows the admin to set a key as disabled and cannot be used.
  - /login: allows the admin to login and perform all other actions.

#### 2.1.2.2. Assistant API endpoint

- **Purpose:** Allows assistants to submit data that needs to be signed.
- **Endpoint:**
  - /listAssistantData: lists all data that is submitted by the current assistant.
  - /AddUserData: submit data to be signed by a signer.
  - /GetUserList: get list of users to assign data to the signer.
  - /login: allows the assistant to login and submit data to the api.

#### 2.1.2.3. Signer API endpoint

- **Purpose:** Handles the signing of data using split private keys.
- **Endpoints:**

- /GetUserKeys: Retrieves key parts from both the API and the memory card for signing.
- /AddSignature: Signs the submitted data by combining key parts from the server and the smart card.
- /listSignerData: Lists data assigned for the signer.
- /login: allows the signer to login and perform all actions allowed to him.

#### 2.1.2.4. Verifier API endpoint

- **Purpose:** Verifies the signatures of documents.
- **Endpoint:**
  - /VerifyData: Verifies the signature using the public key associated with the signed document.

#### 2.1.3. API Flow

The diagram below illustrates the flow of communication between the four applications and the backend API.

In this architecture:

- The **Admin** application interacts with the API to manage users and keys.
- The **Assistant** application submits data to the API to be signed.
- The **Signer** application retrieves key parts from the API and the smart card to sign the data.
- The **Verifier** application verifies the signature using the API.

## 2.2. API Security

Security is a critical part of our system. the key security measures implemented in our Api are mentioned bellow:

### 2.2.1. Authentication:

JSON Web Tokens (JWT) are used to authenticate API requests, ensuring only authorized users can access sensitive endpoints.

### 2.2.2. Encryption:

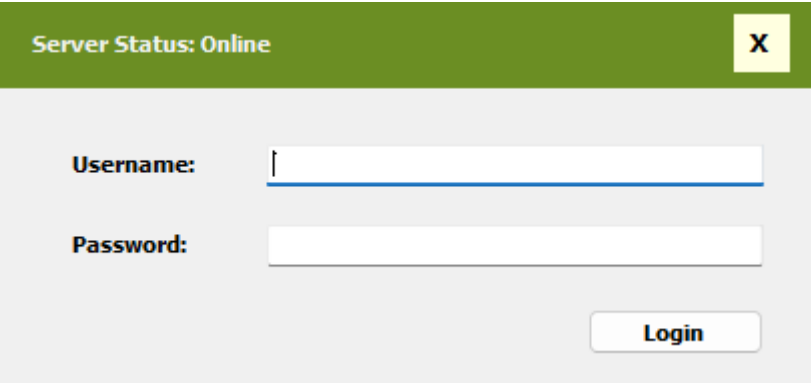
All data exchanged between clients and the API is encrypted using SSL/TLS, ensuring that no sensitive information is intercepted during transmission.

### 2.2.3. Key Splitting:

The private key used for signing is split between the server and the smart card, providing an additional layer of security by preventing exposure of the full key in any single location.

## 2.3. Application user interface

All the applications must go through the login interface before getting access to functionalities except the verifier app which is considered to be public.



The image shows a standard login user interface. At the top, there is a green header bar with the text "Server Status: Online" on the left and a yellow square with a black "X" on the right. Below the header, the login form is displayed on a light gray background. It features two input fields: "Username:" with a white input box and a blue underline, and "Password:" with a white input box. A "Login" button is located at the bottom right of the form.

**Figure 6: Standard login UI**

### 2.3.1. SmartCardAdmin:

- This application is responsible for managing users and keys.

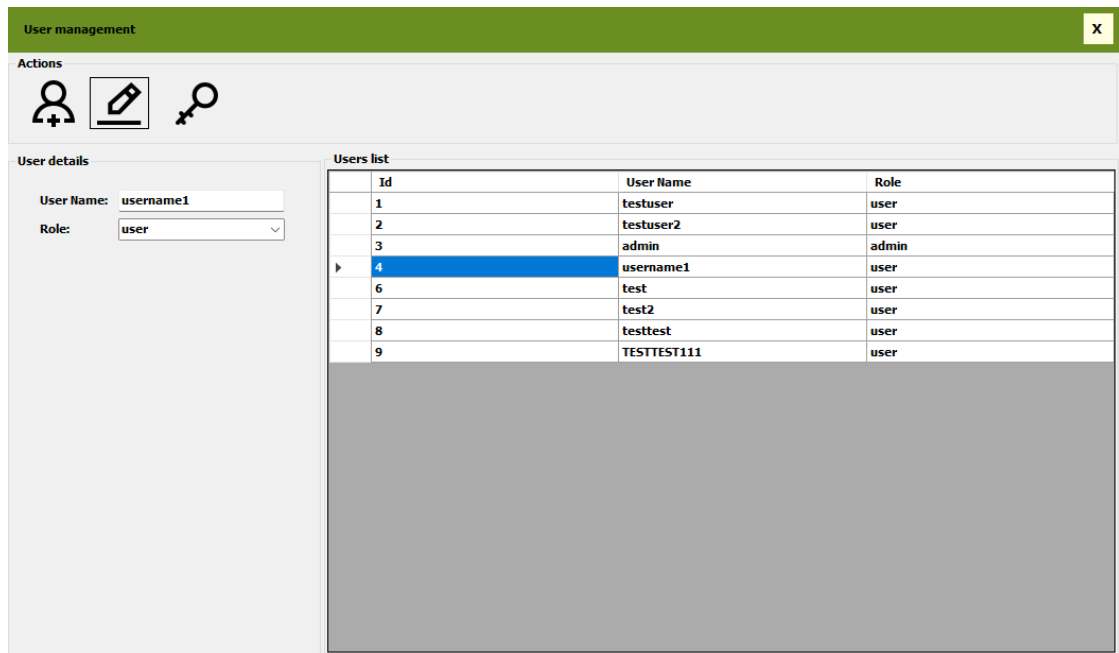


Figure 7: Main view of the admin

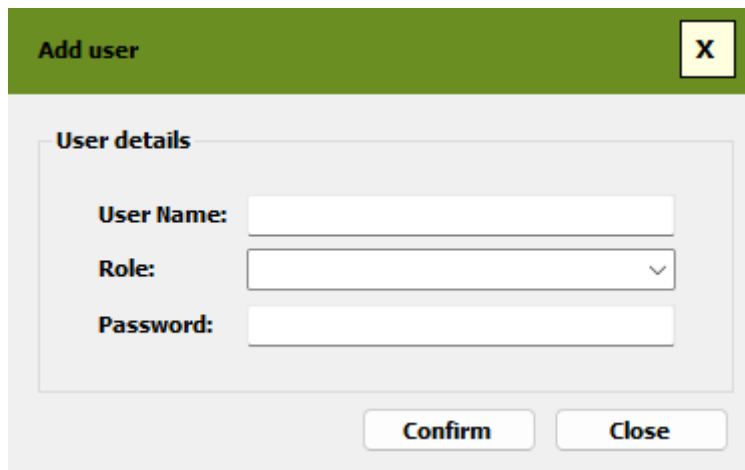


Figure 8: Add new user UI

Manage keys For user : username1 X

**Smartcard Reader details**

Reader list: ↻ Load readers ↻ Connect reader 🔧 Prepare the card

**Actions**

🔑 Generate Key 📡 Send keys to server 📄 Write keys to card ✅ Verify Key on card 🔒 Enable Key 🔓 Disable Key

**Key parts**

Public modulus: qr76050aRCmdhtBLD6w1i2Rinet04GmJFUL+GrzKXbZZo2Bo  
+OqoRbyZ/xT3v52MdMdh4Tm822mA7J8c3KCeV9tFRW0zgnSFx  
+U92VfVG5v8vnPck3FvnWlOzsvfOnvXs4Tscnc:1010R8+URGF5MX3RdWambIX5m/cAiiRxsF7K7aDN55hrr0uIAiKPRRR/ckl

Public exponent: AQAB

Private part: GUSRRBaQkmv/4AI8rR8Qrd/14IOha0xQGxhRGz5MLhV/z1rXTysL4yekGZCG15ZDNfwvag8LVMoCCnCFRjw4gl/N8FeH9EoR  
OYXGQkeVIIF2OCM1Chksq54pBftl5v2LkbK9hyjx+gd1qhwTLPMqz5EQKfu+bd5thlJlpIbhE=

**User key list**

	Id	Enabled	Created At	Updated At
▶	6	<input checked="" type="checkbox"/>	23-09-2024	23-09-2024

Status : Status

**Figure 9: Admin key manager for users**

2.3.2. *SmartCardAssistant:*

- This application interface allows assistants to input data to be signed.

Assistant X

**Actions**

➕ Add New Record

**Details:**

Diploma Number: test number  
Name: student name  
Place Of Birth: student place  
Date Of Birth: 29-09-2024  
Domain: test domain  
Field: test field  
Specialty: test specialty  
Date Issued: 29-09-2024

**Diplomas list:**

	Id	Number	Name	Domain	Issued
▶	1	test number	student name	test domain	29-09-2024 1...
	3	test	test	test	01-10-2024 1...

**Figure 10: Assistant main view**

**Add new record**
X

**Record details:**

**Diploma Number:**

**Name:**

**Place Of Birth:**

**Date Of Birth:**

**Domain:**

**Field:**

**Specialty:**

**Date Issued:**

**Signer:**

**Figure 11: Add new diploma record view**

### 2.3.3. SmartCardSigner:

- This application enables signers to sign documents using the split private key.

**Smart Card Signer**
X

**Card Reader**

Reader list :      Verify Key on card

**Actions**

**Details:**

**Diploma Number:** test number

**Name:** student name

**Place Of Birth:** student place

**Date Of Birth:** 29-09-2024

**Domain:** test domain

**Field:** test field

**Specialty:** test specialty

**Date Issued:** 29-09-2024

**Diplomas list:**

	Id	Number	Name	Domain	Issued
▶	1	test number	student name	test domain	29-09-2024 1...
	3	test	test	test	01-10-2024 1...

Status : Status

**Figure 12: Signer main view**

### 2.3.4. SmartCardVerifier:

- This application verifies the signatures of documents using a public key.

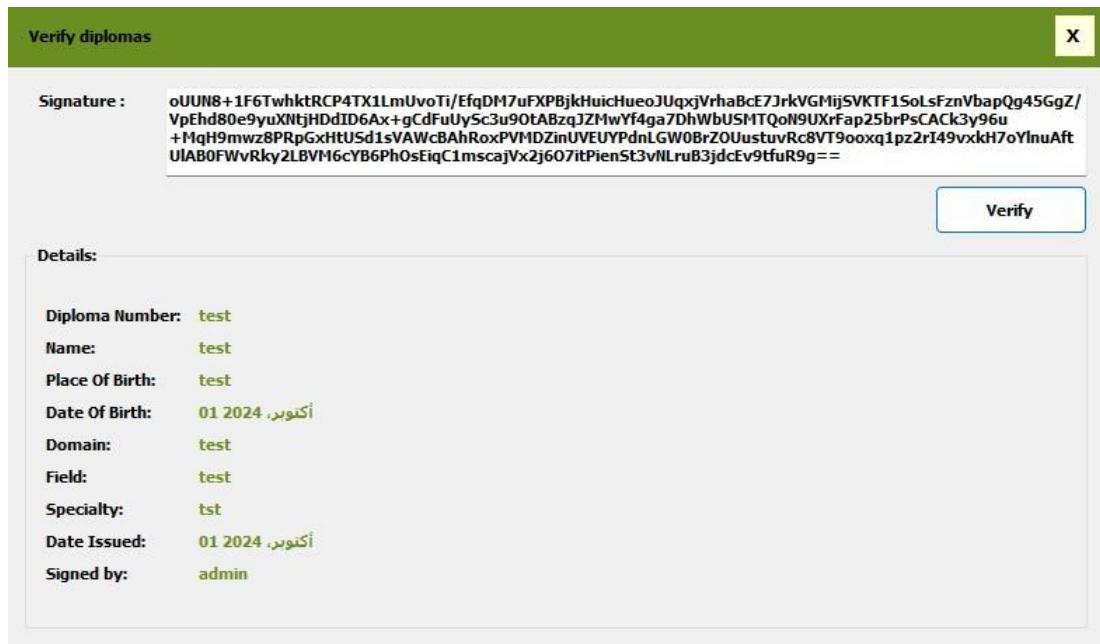


Figure 13: Verifier app view

## 3. Key Management Processes

Key management in this system revolves around the process of **user creation**, **key generation**, and **key storage**.

### 3.1. User Creation and Authentication

When a new user is created using **SmartCardAdmin**, a unique **public-private key pair** is generated. The private key is split into two parts, one stored on the memory card and the other in the API.

#### API Endpoint for User Creation:

- **Endpoint:** POST /adminApi/createUser
- **Description:** Creates a new user and generates a key pair for them.

#### 3.1.1. Request Example:

```
{
  "Username": "test",
  "Password": "test1",
  "Role": "user"
}
```

### 3.1.2. Response Examples:

```
{
  "error": "Username already exists"
}
```

```
{
  "message": "User created successfully"
}
```

## 3.2. Key Generation

After a user is created, the admin uses the **SmartCardAdmin** app to generate and store the keys in both the Api server and on the smart card.

### 3.2.1. Key splitting algorithm

The apps use the following algorithm for both the generation the private key parts that are to be stored on the server and on the smart card.

<b>Algorithm: Key Splitting Function</b>
<ol style="list-style-type: none"> <li>1. function SplitKey()</li> <li>2. // Step 1: Retrieve the private exponent (d)</li> <li>3. dBytes = _rsaParams.D</li> <li>4. // Step 2: Check if the private exponent is large enough</li> <li>5. if length of dBytes &lt; 128 then</li> <li>6.     throw "Private key exponent (d) is too small to split."</li> <li>7. end if</li> <li>8. // Step 3: Initialize cardPrivatePart to store the split key</li> <li>9. cardPrivatePart = new byte[255]</li> <li>10. // Step 4: Copy the first 128 bytes of d into cardPrivatePart</li> <li>11. copy first 128 bytes of dBytes to cardPrivatePart</li> </ol>



```

12. // Step 5: Retrieve P and Q values
13. pBytes = _rsaParams.P
14. qBytes = _rsaParams.Q
15. // Step 6: Copy portions of P and Q into cardPrivatePart
16. pSize = minimum(63, length of pBytes) // Limit size of P to 63 bytes
17. qSize = minimum(64, length of qBytes) // Limit size of Q to 64 bytes
18. copy first pSize bytes of pBytes to cardPrivatePart at index 128
19. copy first qSize bytes of qBytes to cardPrivatePart at index 128 + pSize
20. // Step 7: Prepare remaining part of the private exponent (d) for the server
21. remainingPrivatePart = new byte[length of dBytes - 128]
22. copy bytes of dBytes from index 128 to remainingPrivatePart
23. // Step 8: Prepare remaining parts of P and Q for the server
24. remainingP = new byte[length of pBytes - pSize]
25. copy bytes of pBytes from index pSize to remainingP
26. remainingQ = new byte[length of qBytes - qSize]
27. copy bytes of qBytes from index qSize to remainingQ
28. // Step 9: Prepare the server public part with modulus (n) and exponent (e)
29. serverPublicPart = new ServerPublicPart
30. serverPublicPart.pm = Base64Encode(_rsaParams.Modulus)
31. serverPublicPart.pe = Base64Encode(_rsaParams.Exponent)
32. serverPublicPart.rp = Base64Encode(remainingPrivatePart)
33. serverPublicPart.remainingp = Base64Encode(remainingP)
34. serverPublicPart.remainingq = Base64Encode(remainingQ)
35. // Step 10: Return both parts
36. return (cardPrivatePart, serverPublicPart)
37. end function

```

The following is the explanation of Each Step of the previous algorithm.

1. **Retrieve Private Exponent:** The function begins by fetching the private exponent  $d$  from the RSA parameters.
2. **Check Size:** It checks if the length of  $d$  is less than 128 bytes. If so, it raises an error since a key that small cannot be properly split.

3. **Initialize cardPrivatePart:** A byte array, cardPrivatePart, is initialized to hold the first part of the split key.
4. **Copy Exponent to Card:** The first 128 bytes of the private exponent d are copied to cardPrivatePart.
5. **Retrieve P and Q:** The function retrieves the prime factors P and Q from the RSA parameters.
6. **Copy Portions of P and Q:** It determines how many bytes of P and Q can be safely copied to cardPrivatePart without exceeding set limits (63 for P and 64 for Q). The relevant portions are then copied to cardPrivatePart.
7. **Prepare Remaining Private Exponent:** The remaining part of the private exponent d (beyond the first 128 bytes) is prepared for sending to the server.
8. **Prepare Remaining Parts of P and Q:** The leftover parts of P and Q (after the copied portions) are extracted and prepared for the server.
9. **Prepare Server Public Part:** A new instance of ServerPublicPart is created. It includes:
  - The modulus n (encoded in Base64).
  - The public exponent e (encoded in Base64).
  - The remaining part of d.
  - The remaining parts of P and Q (all encoded in Base64).
10. **Return the Results:** Finally, the function returns both the cardPrivatePart (to be used on the smart card) and the serverPublicPart (to be sent to the server).

This pseudo-code captures the logic of the function while providing a clear step-by-step breakdown of the operations performed.

### 3.2.2. *Key Reconstruction Algorithm*

This algorithm will reconstruct the private key using parts stored on the server and parts stored on the smart card.

**Algorithm: Key Reconstruction Function**

```
1. function ReconstructKey(cardPrivatePart, serverPublicPart)
    // Step 1: Decode Base64 values from serverPublicPart
2. pm = Base64Decode(serverPublicPart.pm) // Modulus (n)
3. pe = Base64Decode(serverPublicPart.pe) // Public Exponent (e)
4. rp = Base64Decode(serverPublicPart.rp) // Remaining private exponent part
5. remainingP = Base64Decode(serverPublicPart.remainingp) // Remaining part of
   P
6. remainingQ = Base64Decode(serverPublicPart.remainingq) // Remaining part of
   Q
    //Step 2: Reconstruct the private exponent (d)
7. fullPrivateExponent = new byte[255]
8. copy first 128 bytes from cardPrivatePart to fullPrivateExponent
9. copy rp to fullPrivateExponent starting from index 128
    // Step 3: Reconstruct the P and Q values
10. fullP = new byte[length of cardPrivatePart - 128 + length of remainingP]
11. fullQ = new byte[length of cardPrivatePart - 192 + length of remainingQ]
12. copy bytes from cardPrivatePart (starting at index 128 up to pSize) to fullP
13. copy bytes from cardPrivatePart (starting at index 128 + pSize) to fullQ
14. append remainingP to fullP
15. append remainingQ to fullQ
    // Step 4: Construct the RSAParameters object for the private key
16. rsaParams = new RSAParameters
17. rsaParams.Modulus = pm
18. rsaParams.Exponent = pe
19. rsaParams.D = fullPrivateExponent
20. rsaParams.P = fullP
21. rsaParams.Q = fullQ
    // Step 5: Return the reconstructed RSA parameters
22. return rsaParams
```

23. end function

The following is the explanation of Each Step of the previous algorithm.

1. **Decode Base64 Values:** The function begins by decoding the Base64-encoded values from serverPublicPart for the modulus (pm), public exponent (pe), remaining private exponent part (rp), and remaining parts of P and Q (remainingP and remainingQ).
2. **Reconstruct the Private Exponent:** A new byte array, fullPrivateExponent, is initialized. The first 128 bytes from cardPrivatePart are copied into fullPrivateExponent, followed by the remaining private exponent bytes from rp.
3. **Reconstruct P and Q:** The fullP and fullQ arrays are created by combining the portions from cardPrivatePart and remainingP / remainingQ.
  - o The portions of P and Q stored in cardPrivatePart are appended with remainingP and remainingQ from the server to complete P and Q.
4. **Construct RSA Parameters Object:** With all components reconstructed, an RSAParameters object is created, assigning Modulus, Exponent, D, P, and Q to complete the private key structure.
5. **Return the Reconstructed RSA Parameters:** The rsaParams object containing the private key components is returned.

### 3.3. Key Distribution

After key generation, the **cardPrivateKeyPart** must be securely transferred to the memory card.

#### API Endpoint for Key Distribution:

- **Endpoint:** POST / adminApi/addUserKey
- **Description:** Transfers the private key part to the memory card.

#### Request Example:

```
{  
  
  "userId": 1,  
  
  "pm": "uDWS+T6iTMnMaN4T+xYs9IKgJ3NLkYcw...",  
  
  "pe": "AQAB",  
  
  "rp": "4XQ3tkg1pOxijjHqoZZn4Nu5TCwmXYcGAEHFIy3eJ...",  
  
  "remainingp": "pN63JuvK13URyePayUZY9fYtOeXPo5BNCRjqj...",  
  
  "remainingq": "w2TD+4dm/W7nsxvNPzyeJndNJ0Dw3TdrFCpm8EFc..."  
  
}
```

**Response Example:**

```
{  
  
  "error": "Invalid input"  
  
}  
  
{  
  
  "message": "Key registered with success"  
  
}
```

## 4. Data Signing and Verification

### 4.1. Data Signing Process

Signing data involves using the private key (both parts from the server and the memory card) to create a digital signature. The following steps outline how to sign data in the system.

**API Endpoint for Data Signing:**

- **Endpoint:** POST / api/AddSignature

- **Description:** Signs the data using both parts of the private key.

**Request Example:**

```
{  
  
  "dataId":1,  
  
  "signature":"vBXXJ3F1NCs83wFt3tj9iNJQdETsmRkVIEqfR856bhNb...  
  
}
```

**Response Example:**

The response is empty if no problem arises.

## 4.2. Verification Process

Verification ensures the data's integrity by confirming the signature using the public key. The verification can happen online or offline, in our case the verification works on the verifying app. The app requests the public key for the user who did login then the apps reconstruct the public key only, that public key is used to verify the authenticity of the data.

**API Endpoint for Data Verification:**

- **Endpoint:** POST /api/VerifyData
- **Description:** Verifies the signed data using the public key.

**Request Example:**

```
{  
  
  "signature": "MEUCIQCg27Dvu0jyt..."  
  
}
```

**Response Example:**

```
{
```

```

"error": "Invalid Signature"
}

{
  "id": 1,
  "data": "{\\"Name\\":\\"student name \\",\\"DateOfBirth\\":\\"2024-09-29T13:42:00.4538218+01:00\\",\\"PlaceOfBirth\\":\\" student place of birth\\",\\"Domain\\":\\"test domain\\",\\"Field\\":\\" test field\\",\\"Specialty\\":\\"test specialty\\",\\"DateIssued\\":\\"2024-09-29T13:42:00.4528233+01:00\\",\\"DiplomaNumber\\":\\"test number\\"}\n",
  "updated_at": "2024-10-27 23:53:09",
  "signature": "BYDUHXXqIaqMW71WDVeLedqp3....",
  "pm": "o9m6aXZ1vr6rUt0fTrd9ZjiKa3+Hzy3Vinrgb/.....",
  "pe": "AQAB",
  "username": "admin",
  "role": "admin"
}

```

## 5. Security Considerations

### 5.1. Key Security

In the memory card-based PKI system, private key security is critical. By splitting the private key between the memory card and the server, we ensure that compromising one part does not lead to a full key compromise. Additional measures include:

- **Encrypted Key Transfers:** All communication involving key distribution can be encrypted using SSL/TLS.

- **Tamper Detection:** If a memory card is tampered with, the system will not allow the associated key to be used for signing.

## 5.2. Revocation and Recovery

In case a user loses their memory card, the system provides key revocation and new key generation.

### API Endpoint for Key Revocation:

- **Endpoint:** POST /adminApi/disableUserKey
- **Description:** Revokes a key pair, rendering it unusable.

### Request Example:

```
{  
  
  "keyId":"1"  
  
}
```

### Response Example:

```
{  
  
  "message":"Key disabled successfully"  
  
}
```

## 6. Use Cases and Applications

The memory card-based PKI system is versatile and applicable in various fields:

- **Diplomas and Certificates:** Securely sign and verify academic qualifications.
- **Identification and Access Control:** Used in secure ID cards for controlling access to facilities.
- **Business and Financial Applications:** Digital signatures for financial transactions.



## **7. Conclusion**

This chapter detailed the implementation of a memory card-based PKI system, exploring key management processes, cryptographic operations, and tests using Postman. The implementation offers a cost-effective alternative to cryptographic smart cards with secure key splitting, ensuring robust data security for various applications. Through comprehensive Postman tests, we ensured the correct functioning of key components, such as key generation, data signing, and verification.

## **Conclusion**

In conclusion, this thesis underscores the critical role of cryptography and authentication in safeguarding digital communications, particularly in the context of electronically signed diplomas. By examining the principles of cryptography and the importance of authentication, we have established a framework for understanding how these technologies can be applied effectively. The ediploma system exemplifies this application, demonstrating a practical solution for ensuring the integrity and authenticity of digital diplomas. As educational institutions continue to adopt digital credentials, the insights gained from this work contribute to the broader efforts to enhance security in the digital landscape.

## References

- [1] R. Oppliger, *Cryptography 101: From Theory to Practice*, Artech House, 2021.
- [2] V. Geetha, P. Singh, N. S. Patil and S. S. S. Reddy, *Introduction To Cryptography And Network Security*, AG PUBLISHING HOUSE (AGPH Books), 2023.
- [3] M. Abd Zaid and S. Hassan, "Survey on modern cryptography," *Journal of Kufa for Mathematics and Computer*, vol. 7, p. 1–8, 2020.
- [4] M. Bellare, R. Canetti and H. Krawczyk, "Keying hash functions for message authentication," in *Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16*, 1996.
- [5] W. Diffie and M. E. Hellman, "New directions in cryptography," in *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, 2022, p. 365–390.
- [6] A. M. Abdullah and others, "Advanced encryption standard (AES) algorithm to encrypt and decrypt data," *Cryptography and Network Security*, vol. 16, p. 11, 2017.
- [7] F. I. P. S. Pub, "Data encryption standard (des)," *FIPS PUB*, p. 46–3, 1999.
- [8] W. Stallings, "Digital signature algorithms," *Cryptologia*, vol. 37, p. 311–327, 2013.
- [9] T. Dierks and C. Allen, "The TLS protocol version 1.0," 1999.
- [10] A. Roy and S. Karforma, "A survey on digital signatures and its applications," *Journal of Computer and Information Technology*, vol. 3, p. 45–69, 2012.
- [11] W. Fang, W. Chen, W. Zhang, J. Pei, W. Gao and G. Wang, "Digital signature scheme for information non-repudiation in blockchain: a state of the art review," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, p. 1–15, 2020.
- [12] S. Delgado-Segura, C. Pérez-Sola, G. Navarro-Arribas and J. Herrera-Joancomartí, "Analysis of the bitcoin utxo set," in *Financial Cryptography and Data Security: FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers 22*, 2019.
- [13] N. Komninos, "PKI systems," *Network Security: Current Status and Future Directions*, p. 409–418, 2007.
- [14] C. Adams and S. Lloyd, *Understanding PKI: concepts, standards, and deployment considerations*, Addison-Wesley Professional, 2003.
- [15] Z. A. Alizai, N. F. Tareen and I. Jadoon, "Improved IoT device authentication scheme using device capability and digital signatures," in *2018 International conference on applied and engineering mathematics (ICAEM)*, 2018.

- [16] S. F. Al-Janabi and A. K. Obaid, "Development of certificate authority services for web applications," in *2012 International Conference on Future Communication Networks*, 2012.
- [17] A. Loconsolo, "Securing digital identities: from the deployment to the analysis of a PKI ecosystem with virtual HSMs leveraging open-source tools," 2024.
- [18] A. Yang and G. P. Hancke, "RFID and contactless technology," *Smart Cards, Tokens, Security and Applications*, p. 351–385, 2017.
- [19] E. Milanov, "The RSA algorithm," *RSA laboratories*, p. 1–11, 2009.
- [20] D. E. Standard and others, "Federal information processing standards publication 46," *National Bureau of Standards, US Department of Commerce*, vol. 23, p. 1–18, 1977.
- [21] C. Paar and J. Pelzl, *Understanding cryptography*, vol. 1, Springer, 2010.
- [22] M. J. Dworkin, E. B. Barker, J. R. Nechvatal, J. Foti, L. E. Bassham, E. Roback and F. J. Dray Jr, "Advanced encryption standard (aes)(fips pub 197)," *Nat. Inst. Standards Technol.(NIST), Gaithersburg, MD, USA, Tech Rep*, vol. 197, 2001.
- [23] G. N. Brijwani, P. E. Ajmire and P. V. Thawani, "Future of quantum computing in cyber security," in *Handbook of Research on Quantum Computing for Smart Environments*, IGI Global, 2023, p. 267–298.