Standard-Based Smart Card Access Control Architecture for Critical Infrastructures

ANTÓNIO FILIPE RAMIC SANTOS OLIVEIRA
(Licenciado)

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática e Multimédia

Orientadores: Doutor Tiago Miguel Braga da Silva Dias
Doutor Ricardo Jorge Fernandes Chaves

Júri:

Presidente:	Doutor Diogo Nuno Crespo Ribeiro Cabral
Vogais:	Doutor José Manuel De Campos Lages Garcia Simão
	Doutor Tiago Miguel Braga Da Silva Dias

Dezembro 2025

Standard-Based Smart Card Access Control Architecture for Critical Infrastructures

ANTÓNIO FILIPE RAMIC SANTOS OLIVEIRA

(Licenciado)

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática e Multimédia

Orientadores: Doutor Tiago Miguel Braga da Silva Dias, ISEL/Inst. Politécnico de Lisboa
Doutor Ricardo Jorge Fernandes Chaves, IST/Univ. de Lisboa

Júri:

Presidente: Doutor Diogo Nuno Crespo Ribeiro Cabral , ISEL/Inst. Politécnico de Lisboa
Vogais: Doutor José Manuel De Campos Lages Garcia Simão, ISEL/Inst. Politécnico de Lisboa
Doutor Tiago Miguel Braga Da Silva Dias, ISEL/Inst. Politécnico de Lisboa

Dezembro 2025

Acknowledgements

Above all, I thank God, who has never let me down.

This dissertation marks a significant milestone in both my academic journey and professional development. I learned a great deal. I experienced many very good moments, both socially and in terms of individual and collective growth. Together with my colleagues and friends, I overcame the natural challenges of academic life.

My undergraduate degree was in Electrical Engineering rather than Computer Science. I had only a handful of courses in programming and algorithms. Those few classes were enough to spark the curiosity that ultimately led me here. After graduating, I began my first full-time job at Lisbon Airport in the Electronics, Automation, and IT Maintenance Department. That experience quickly made me realize that I wanted to pursue an academic degree in computer science. I therefore enrolled in the Master's program in Informatics and Multimedia Engineering.

I began this journey during the COVID-19 pandemic. Teaching was remote, and I lacked some foundational knowledge, which demanded extra work and resilience. Nevertheless, I became genuinely fascinated by all the classes. Looking back, the Master's programme gave me enormous satisfaction to attend. I finally learned the topics that had long intrigued me—artificial intelligence, computer networks, software development, and distributed computing—with the support of a highly capable and professional faculty.

First and foremost, I am profoundly grateful to my advisers, Professors Tiago Dias, whose Cybersecurity class I attended, and Ricardo Chaves. Their availability, guidance, and commitment were decisive for the completion of this thesis.

I also thank my ISEL colleagues, with whom I worked on numerous group projects throughout my master's program. Together, we made this achievement possible.

To my family, I express my sincere gratitude. To my mother, who devoted her life to me and did everything within her power so that I could write these words and reach this milestone. She has been by my side at every step, and I will be forever grateful. Now it is my turn to give back. To Aunt Nerma, for her support and the crucial help that made my initial entry into university possible. To Aunt Ajša, for her consistent support throughout this journey. To my Father, a disciplined and disciplinarian man, whose technical knowledge, strong sense of duty, service, and determination have been a constant inspiration. To D. Marlene and Sr. João Rodrigues, for their support in difficult

moments. To my neighbours, José and Carminda, for being part of this journey as well.

I am grateful to my former and current managers for enabling me to balance my work with the demands of the Master's program. That flexibility was essential.

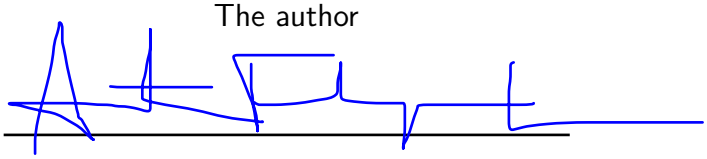
Finally, a word on friendship—a central part of my life. I thank my friends Zé, Crespo, João Carlos, Bruno, Hugo, Andreia, Ana Isabel, Nuno Almeida, Margaret, Luís, Mafalda, Neves, and all the others who have been and are part of my life. Each of you contributed, in your own unique way, to my personal and professional journey.

To all of you, my heartfelt thanks.

Statement of integrity

I declare that this dissertation is the result of my personal and independent research. Its content is original, and all sources listed in the bibliographic references were consulted and are duly mentioned in the text. I further declare that all scientific and technical references relevant to the development of the work are duly cited and included in the bibliographic references.

The author



Lisbon, December 17th, 2025

Standard-Based Smart Card Access Control Architecture for Critical Infrastructures

This document was created using the (pdf)LaTeX processor, based in the "iselthesis" template [70], developed at the DEETC of ISEL-IPL.

Abstract

This thesis proposes an enterprise-centered access control architecture that is technically compatible with widely adopted standards, enabling the convergence of physical and logical access on a single smart card credential while preserving interoperability and enterprise control over identity proofing, registration, and credential issuance. This need is particularly acute for operators of critical infrastructure, who must replace legacy badge-based systems, ensure cross-site and outage resilience, unify access control with support for rapid revocation, generate audit trails, and avoid vendor lock-in through the use of open standards. The proposed architecture was built to meet these needs.

The main outcomes of this work are a standards-aligned enterprise architecture for a multi-site access control system and a smart card credential with a well-defined, standardized data model and lifecycle. Additionally, this work showcases a proof-of-concept prototype consisting of a Card Management System that personalizes cards and issues Public Key Infrastructure (PKI) credentials. The prototype also features a Java Card applet that implements a standardized data model and command set, supporting asymmetric cryptography-based multi-factor authentication mechanisms. Furthermore, it includes an access control system emulator for testing purposes.

Using this stack, it is possible to issue cards and test them by performing strong authentication mechanisms, demonstrating the end-to-end feasibility of the devised solution for both physical and logical access scenarios.

Keywords: Critical Infrastructure, Access Control System, Authentication, Credential Management, Public Key Infrastructure, Smart Card, PIV, CIV.

Resumo

Esta tese propõe uma arquitetura de controlo de acessos para operadores de infraestruturas críticas, tecnicamente compatível com normas amplamente adotadas. A arquitetura proposta unifica o acesso físico e o acesso lógico numa única credencial de *smart card*, preservando a interoperabilidade e o controlo das organizações sobre os procedimentos de *enrollment* e emissão de credenciais. Existe uma necessidade premente para os operadores de infraestruturas críticas substituírem cartões de tecnologia antiga, manterem a resiliência entre instalações e durante falhas, unificarem o controlo de acessos com revogação célere, gerar registos de auditoria e evitar a dependência de fornecedores através do uso de normas abertas. A arquitetura proposta foi concebida para responder a estas necessidades.

As principais contribuições deste trabalho são a proposta de uma arquitetura de um sistema de controlo de acessos baseada em normas abertas e na utilização de *smart cards* como credenciais. A prova de conceito consistiu no desenvolvimento de um protótipo composto por um Sistema de Gestão de Cartões, que personaliza os cartões e emite credenciais PKI. Esse protótipo inclui ainda uma *applet* Java Card que implementa um modelo de dados e um conjunto de comandos padronizados, suportando mecanismos de autenticação multifator baseados em criptografia assimétrica, bem como um emulador de sistema de controlo de acessos para fins de teste.

Utilizando estes artefactos, é possível emitir cartões e testá-los através da execução de mecanismos de autenticação fortes, demonstrando a viabilidade, de ponta-a-ponta, da solução proposta para cenários de acesso físico e lógico.

Palavras-chave: Infraestruturas Críticas, Sistema de Controlo de Acessos, Autenticação, Gestão de Credenciais, Infraestrutura de Chave Pública (PKI), Smart Card, Verificação de Identidade Pessoal (PIV), Verificação de Identidade Comercial (CIV).

Contents

List of Figures

List of Tables

Listings

Acronyms

2K3DES	Two-Key Triple DES
3DES	Triple Data Encryption Standard
3K3DES	Three-Key Triple DES
AAL	Authenticator Assurance Level
ACL	Access Control List
ADF	Application Dedicated File
AES	Advanced Encryption Standard
AIA	Authority Information Access
AID	Application Identifier
AKI	Authority Key Identifier
APDU	Application Protocol Data Unit
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
CA	Certification Authority
CAD	Card Acceptance Device
CAK	Card Authentication Key
CAP	Converted Applet
CBC	Cipher Block Chaining
CC	Common Criteria
CCC	Card Capability Container
CCTV	Closed-Circuit Television
CHUID	Cardholder Unique Identifier
CIV	Commercial Identity Verification
CLA	Class (first) byte of a card command
CMAC	Cipher-Based Message Authentication Code
CMS	Card Management System

CMTC	Card Management System to Card
CP	Certificate Policy
CPS	Certification Practice Statement
CPU	Central Processing Unit
CRL	Certificate Revocation List
CRLDP	CRL Distribution Point
CSR	Certificate Signing Request
CSV	Comma-Separated Values
CTC	Cardholder to Card
CTE	Cardholder to External System
CVC	Card Verifiable Certificate
DER	Distinguished Encoding Rules
DES	Data Encryption Standard
DF	Dedicated File
DN	Distinguished Name
ECB	Electronic Codebook
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EEPROM	Electrically Erasable Programmable ROM
EF	Elementary File
EKU	Extended Key Usage
EMV	Europay, Mastercard, and Visa
FASC-N	Federal Agency Smart Credential Number
FID	File Identifier
FIPS	Federal Information Processing Standards
FPKI	Federal Public Key Infrastructure
GUID	Global Unique Identification number
HMAC	Hash-based Message Authentication Code
HR	Human Resources
HSM	Hardware Security Module
IC	Integrated circuit
IDMS	Identity Management System

IEC	International Electrotechnical Commission
INS	Instruction (second) byte of a card command
ISD	Issuer Security Domain
ISO	International Organization for Standardization
JCA	Java Cryptography Architecture
JCAPI	Java Card Application Programming Interface
JCE	Java Cryptography Extension
JCRE	Java Card Runtime Environment
JCVM	Java Cryptography Extension
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LACS	Logical Access Control System
MAC	Message Authentication Code
MF	Master File
MFA	Multi-Factor Authentication
NFC	Near-Field Communication
NIST	National Institute of Standards and Technology
NPU	Numeric Processing Unit
OCC	On-Card Biometric One-to-One Comparison
OCSP	Online Certificate Status Protocol
OID	Object Identifier
OS	Operating System
PACS	Physical Access Control System
PC/SC	Personal Computer/Smart Card
PIN	Personal Identification Number
PIV	Personal Identity Verification
PIX	Proprietary Identifier extension
PKCS	Public-Key Cryptography Standards
PKI	Public Key Infrastructure
PUK	PIN Unblocking Key
RAM	Random Access Memory

RFC	Request for Comments
RID	Registered Application Provider Identifier
ROM	Read-Only Memory
RSA	Rivest–Shamir–Adleman
SAN	Subject Alternative Name
SCP	Secure Channel Protocol
SFI	Short File Identifier
SHA	Secure Hash Algorithm
SKI	Subject Key Identifier
SM	Secure Messaging
SP	Special Publication
SW1	First byte of a 2-byte status word
SW2	Second byte of a 2-byte status word
TLS	Transport Layer Security
TLV	Tag-Length-Value
TPDU	Transmission Protocol Data Unit
UI	user interface
URN	Uniform Resource Name
USB	Universal Serial Bus
UUID	Universally Unique Identifier
VCI	Virtual Contact Interface

1 Introduction

Critical infrastructure refers to assets, systems, and networks, physical or logical, that are considered so vital to a nation that their incapacitation or destruction would have a debilitating effect on its security, economy, public health or safety, or any combination thereof [13]. Power plants, air traffic control systems, and government facilities are some common examples of such infrastructures. Unauthorized access poses a significant threat [14], as it can manifest in harm to personnel, incapacitation or destruction of facilities, networks, equipment, and systems, as well as service interruption. Therefore, robust and strict Physical Access Control Systems (PACSs) and Logical Access Control Systems (LACSs) are employed to prevent such threats.

Access control in large organizations spans both physical locations (e.g., facilities, equipment rooms) and logical resources (e.g., networks, applications). Multi-site operators — especially those responsible for critical infrastructure — must coordinate access across heterogeneous systems, vendors, and sites while maintaining enterprise control and resilience during outages. In this context, open, standards-based approaches that unify physical and logical access have become operationally and strategically important [15, 88].

A fundamental element of such systems is the authentication mechanism. This mechanism is used to verify the identity of individuals when they intend to access security-sensitive resources, so that a sound access control decision can be made. The three most common factors used to verify an individual's identity are something they know (such as a password), something they possess (such as a smart card), and a physical characteristic of them (such as a fingerprint) [53, 63].

Smart card-based authentication [9] is a commonly used type of authentication mechanism adopted in critical infrastructure [26]. It is based on the use of a plastic, credit card-sized card with an embedded integrated circuit (IC) chip that has memory capability, allowing it to store the personal information of individuals, such as their name and the organization they belong to, along with other credentials [75].

1.1 Motivation

In many organizations operating critical infrastructures, access control systems have evolved incrementally over time, resulting in heterogeneous technologies, inconsistent practices, and limited alignment with open standards. In other cases, the adoption of proprietary, vendor-specific platforms has created forms of lock-in that restrict flexibility and hinder the adoption of interoperable, standards-based solutions. These conditions have led to several open issues in the development and implementation of access control systems, summarized as follows:

- Lack of standardized approaches for access control systems credential lifecycle management in enterprise environments (e.g., issuance, activation, suspension, revocation, renewal).
- Absence of a common, vendor-neutral credential data model.
- Fragmentation caused by vendor-specific technologies and frameworks within the same sector or enterprise, hindering uniformity and cross-site interoperability.
- Replication of user account and access setup tasks across operational systems, PACS, and IT systems, and sometimes per individual system, creating siloed identity and access management processes.
- Operational consequences of duplication and fragmentation: slow onboarding for new hires, residual privileges after role changes or terminations, and difficulty conducting periodic access reviews [55].
- Limited enterprise control and outage resilience due to disjoint physical and logical access platforms that are not unified under a common architecture.
- Most studies emphasise PACS architecture, device integration and operations, and access control models, while paying comparatively little attention to credential data models or lifecycle management; when these latter topics are addressed, the literature skews toward government deployments—even though much critical infrastructure is operated by private enterprises [19].

These challenges demonstrate the need for a coherent, standards-based approach to access control in multi-site critical-infrastructure environments. This work is motivated by the aim of providing a structured basis for such an enterprise-controlled solution.

1.2 Objectives

The main goal of this thesis was to investigate, design, and prototype a standards-aligned, enterprise-centered access control architecture for critical infrastructures, with a unified

smart card credential for physical and logical access. The conducted work emphasizes the logical aspects of the credential (its data objects, keys, certificates, and protocols) rather than visual personalization. Beyond proposing the architecture, a proof-of-concept was successfully implemented based on three interoperating components: (i) a CMS with a desktop issuance/administration client; (ii) a Java Card applet that realizes the credential data model, key generation, and on-card cryptographic operations; and (iii) an access control system emulator that exercises one-factor (PKI-CAK) and two-factor (PKI-AUTH) authentication flows.

To achieve this goal, the work was guided by five concrete objectives:

- Analyze existing standards, technologies, and frameworks related to smart card lifecycle management in the context of critical infrastructures.
- Define functional and security requirements for an enterprise system that converges physical and logical access on a single credential, including policy flexibility, certificate validation, revocation handling, and auditing.
- Design a modular architecture that integrates the Identity Management System (IDMS), CMS, PKI services, physical and logical access control systems, and a smart card-based credential, that has a well-defined data model, lifecycle, and command set.
- Implement a working prototype of the key components, i.e., the CMS, the smart card application, and some form of access control system or equivalent that performs strong authentication mechanisms on issued credentials.
- Evaluate the prototype in terms of functional correctness, standards conformance, usability of the issuance workflow, and security properties within the scope of the prototype, identifying limitations and outlining a path to production.

By following these objectives, the thesis delivers not only an operational prototype but also an architectural blueprint and reference model that can guide future deployments of interoperable, standards-based access control solutions in critical-infrastructure environments.

1.3 Requirements

Given the objectives of this thesis, the proposed solution should satisfy the following set of high-level requirements, which are grouped into three categories:

Functional requirements

- Enable the issuance, update, and testing of smart cards for use in both physical and logical access control within critical infrastructures.
- Provide secure storage and management of cryptographic credentials.
- Offer a graphical administrative interface to support credential management by authorized personnel.

Security and interoperability requirements

- Adopt a standardized data model that meets the stringent security requirements of critical infrastructures.
- Remain decoupled from specific access control systems and external directory services, in order to maximize interoperability.
- Ensure compatibility with smart card compliant with ISO/IEC 7810 physical specifications [41].

Architectural and implementation requirements

- Follow a modular architecture capable of interfacing with external systems.
- Be released as open-source software, enabling transparent deployment and adoption by diverse organizations.
- Provide a design that supports extensibility and integration with future standards.

1.4 Document organization

This thesis presents a standards-aligned enterprise access control architecture for critical infrastructure, spanning the smart-card credential and data model, certificate profiles and PKI services, the Card Management System, and access control systems. Accordingly, it is organized as follows:

- Chapter 1 introduces the context, motivation, objectives, and requirements of this research work, as well as the overall document structure.
- Chapter 2 provides the theoretical background, covering cryptography, authentication mechanisms, access control, and smart card technology.
- Chapter 3 surveys the state of the art, including governmental frameworks, commercial solutions, and related academic work.

- Chapter 4 presents the proposed solution, detailing the architecture, credential model, lifecycle processes, and authentication mechanisms.
- Chapter 5 describes the implementation steps and evaluation of the prototype, including the smart card applet and its management application.
- Chapter 6 concludes the thesis by summarizing contributions, highlighting limitations, and outlining directions for future research.

2 Background

This chapter establishes the technical foundations for the thesis. Section 2.1 surveys the core cryptography that underpins certificate-based identity, such as symmetric and asymmetric primitives, X.509 certificates, message authentication codes, one-way hashes, digital signatures, and the role of public-key infrastructure (PKI). Section 2.2 introduces enterprise access control systems and their operational context. Section 2.3 reviews common credential types (passwords/PINs, access cards, and RFID tokens), motivating the move toward stronger and interoperable smart card credentials. Section 2.4 outlines various authentication methods, ranging from passwords and challenge–response to symmetric/public-key schemes, as well as multi-factor authentication. Section 2.5 focuses on smart cards: Application Protocol Data Unit (APDU)-based data exchange, file management concepts, relevant standards, security considerations, and the platforms used in this work (Java Card and GlobalPlatform). Together, these topics provide the conceptual and standards background needed to understand the architecture, implementation, and evaluation presented later.

2.1 Cryptography

Cryptography is the study of mathematical techniques related to aspects of information security such as data integrity, confidentiality, entity authentication, and data origin authentication [82].

Integrity corresponds to the assurance that messages are received without duplication, insertion, deletion, modification, reordering, or replay.

Confidentiality refers to the protection of transmitted data from passive attacks – unauthorized activities aimed at accessing or intercepting data without altering or compromising the integrity of the information. To assure confidentiality in a communication between two entities, the original message, known as the plaintext, is converted into an unintelligible coded message called ciphertext. The process of transforming plaintext

into ciphertext is known as encryption, and the reverse process, restoring plaintext from ciphertext, is called decryption. Both encryption and decryption consist of mathematical algorithms that use strings of bits called keys as inputs in order to code and decode data – algorithm outputs - as illustrated in Figure 2.1.

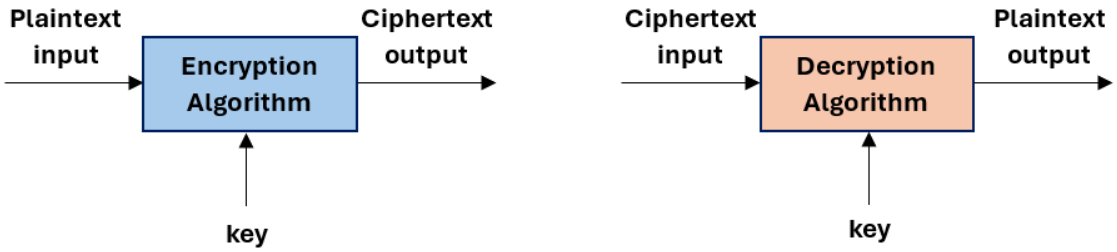


Figure 2.1: *Cryptography - Encryption and Decryption.*

Therefore, confidentiality is achieved if only the sender and receiver of the message are capable of accessing the plaintext. This implies that the encryption and decryption keys must be kept secret from all parties other than the sender and receiver. Only authorized people who have the key can decipher the code and access the original plaintext information. The set comprising encryption and decryption algorithms, plaintexts, and ciphers is called a cryptosystem.

Authentication is the assurance that the communicating entity is the one it claims to be. Two parties entering into a communication should identify each other.

Data origin authentication is the process of proving that a given message was sent by the specified party.

2.1.1 Symmetric Cryptography

Consider a cryptosystem consisting of an encryption algorithm E and a decryption algorithm D, each defined for keys drawn from the key space $\mathcal{K}$:

$$\{E_K\}_{K\in\mathcal{K}}, \qquad \{D_K\}_{K\in\mathcal{K}}$$

The cryptosystem is called symmetric when the encryption key and decryption key are identical or can be efficiently derived from each other. In practice, symmetric cryptosystems use the same key for both operations, so E_K and D_K rely on the same secret key K.

Consider the following example where two parties, A and B, intend to communicate with each other securely:

1. Initially, both parties agree on a cryptosystem and a secret key K.
2. Party A encrypts the message X using the algorithm E_K.

3. A sends the ciphertext message $Y = E_K(X)$ to B.

4. Party B decrypts the ciphertext message using D_K and recovers the original message.

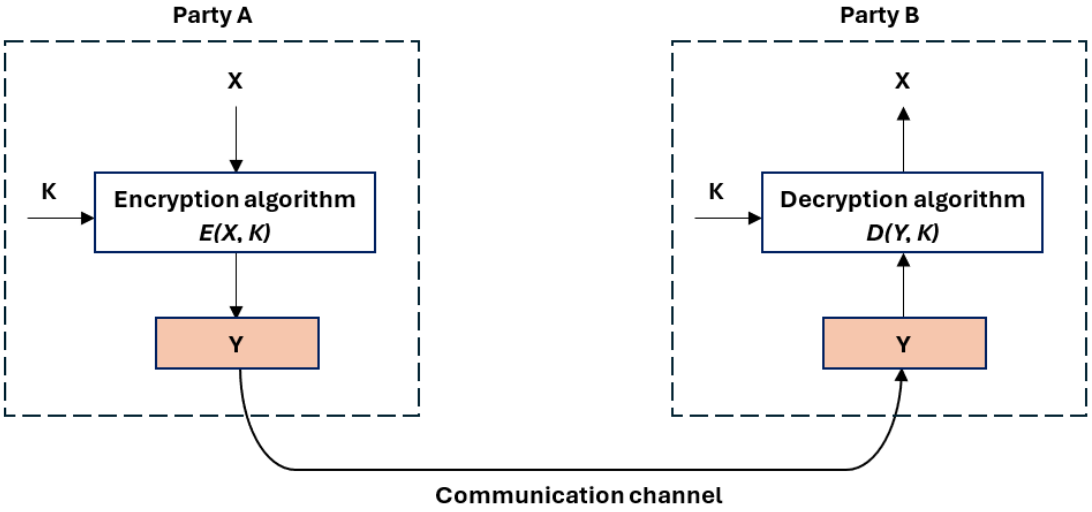


Figure 2.2: *Working principle of symmetric cryptography*

There are two requirements for secure use of symmetrical encryption. A strong encryption algorithm is needed, and the sender and receiver of the message must have obtained copies of the secret key in a secure manner and keep them safe. Additionally, it is essential to note that key length also plays a significant role in securing a cryptosystem. Knowledge of the keys compromises communication confidentiality since an unauthorized party may decrypt all messages encrypted with that key. Additionally, an unauthorized party may also impersonate the sender and send false messages.

The Advanced Encryption Standard (AES) is currently the symmetric cryptographic algorithm of choice for most applications that require electronic data protection. It was designed specifically for information systems used or operated by the U.S. government. The algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits. Each block consists of 16 bytes and is organized in a 4×4 column-major order matrix, referred to as the state array, as shown in Equation 2.1.

$$\begin{bmatrix} \text{byte}_0 & \text{byte}_4 & \text{byte}_8 & \text{byte}_{12} \\ \text{byte}_1 & \text{byte}_5 & \text{byte}_9 & \text{byte}_{13} \\ \text{byte}_2 & \text{byte}_6 & \text{byte}_{10} & \text{byte}_{14} \\ \text{byte}_3 & \text{byte}_7 & \text{byte}_{11} & \text{byte}_{15} \end{bmatrix} \tag{2.1}$$

AES consists of 10 to 14 transformation rounds, depending on the key length. For 128-bit keys, 10 rounds are used; for 192-bit keys, 12 rounds are used; and for 256-bit keys, 14 rounds are used. AES rounds are based on the following transformations: (i) SubBytes; (ii) ShiftRows; (iii) MixColumns; and (iv) AddRoundKey. Except for the last round, all

other rounds are identical. The order in which these transformations are executed is different for encryption and decryption. Figure 2.3 illustrates the overall structure of AES encryption and decryption schemes.

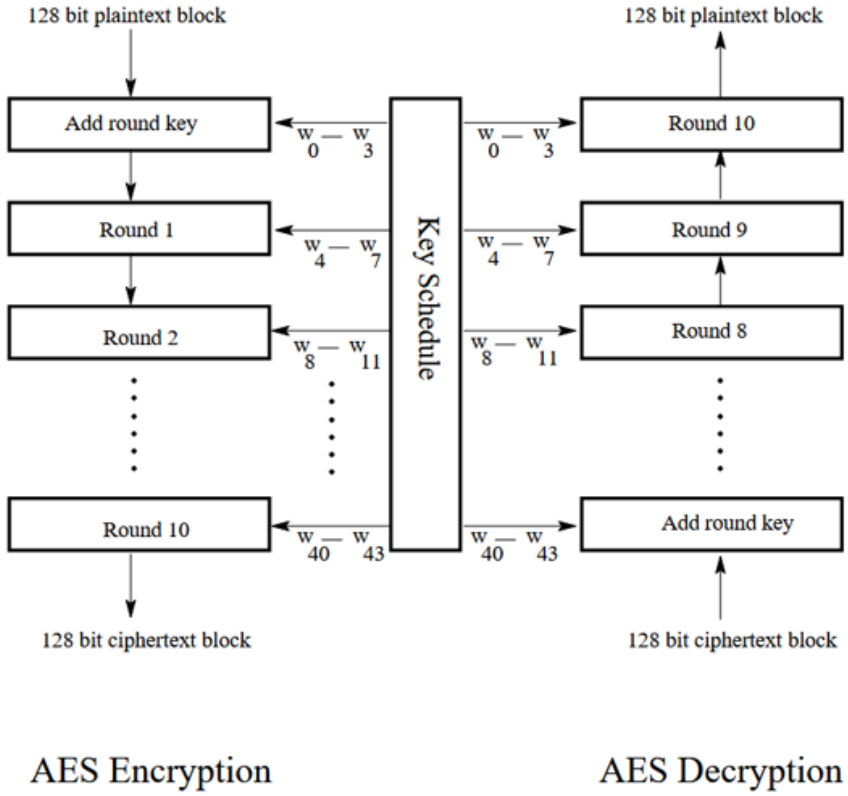


Figure 2.3: *AES algorithm – Encryption and Decryption schemes (obtained from [48]).*

The **SubBytes transformation** maps each block byte to another block using a mathematically precomputed lookup table, known as an S-Box, as shown in Figure 2.4.

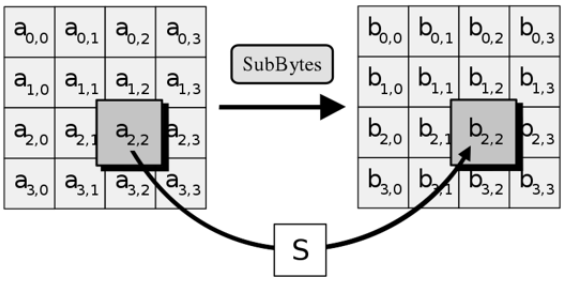


Figure 2.4: *Representation of the SubBytes transformation.*

The **ShiftRows transformation** is a simple byte transposition where the bytes in the last three rows of the state, the 4×4 matrix into which each block is organized, are cyclically shifted. For the second row, a 1-byte circular left shift is applied. For the third and fourth rows, 2-byte and 3-byte, respectively, left circular shifts are applied, as illustrated in Figure 2.5.

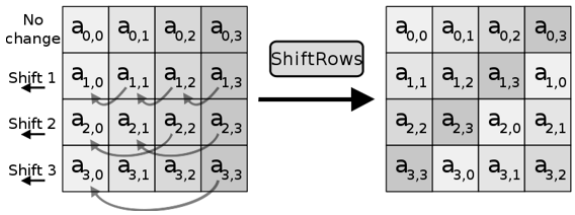


Figure 2.5: *Representation of the ShiftRows transformation.*

In the **MixColumns transformation** round, each column vector is multiplied by a fixed matrix that corresponds to a function of all the bytes in the same column vector, as depicted in Figure 2.6.

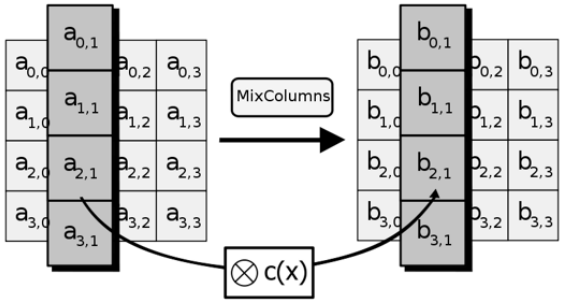


Figure 2.6: *Representation of the MixColumns transformation.*

Similarly to the state array, in the **AddRoundKey transformation**, the original key is also arranged in the form of an array of bytes whose dimension is dependent on the key size. From this key array, a key schedule is computed, providing round keys for each round of both the encryption and decryption schemes of the algorithm. This round key is XORed with the state array obtained from the previous round to obtain the current state array, as shown in Figure 2.7.

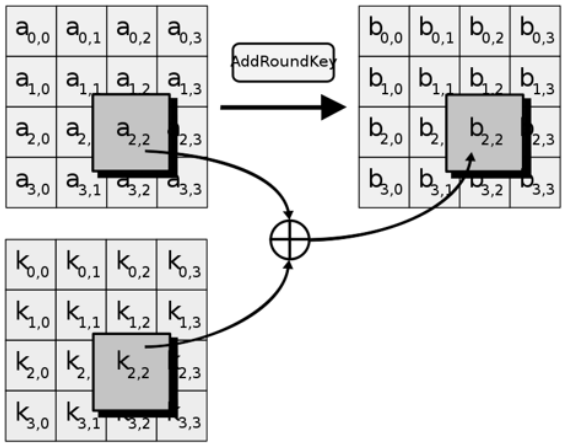


Figure 2.7: *Representation of the AddRoundKey step.*

At the end of the execution of the algorithm, the whole set of cipher blocks constitutes the cipher text.

2.1.2 Asymmetric Cryptography

Asymmetric cryptography utilizes two distinct keys for encryption and decryption, unlike symmetric cryptography, which relies on a single key for both purposes. In asymmetric cryptography, one of the keys is called the public key, in the sense that it is publicly available, while the other, called the private key, should be under the sole control of the entity that owns the key pair.

Consider a scenario where two parties, A and B, communicate using asymmetric cryptography. Each has its own public–private key pair. For confidentiality, if A wants to send a message to B, A encrypts it with B's public key. Only B can decrypt it with the matching private key. For authentication, if A wants to prove authorship of a message, A signs it with their private key. B can then verify the signature using A's public key, ensuring that the message was indeed created by A. This second case forms the basis of digital signatures, which is discussed in Section 2.1.6.

Asymmetric cryptography relies on specific algorithms for key pair generation and encryption and decryption. The most commonly used are Rivest–Shamir–Adleman (RSA) and Elliptic Curve Cryptography (ECC)-based algorithms.

Rivest–Shamir–Adleman (RSA) is an asymmetric cryptographic algorithm named after its inventors Rivest, Shamir, and Adleman [77], whose security relies on the computational difficulty of factoring the product of two large prime numbers. To generate the key pairs, two random primes p and q are chosen and multiplied to form the RSA modulus $n = pq$. These numbers must be large so that it is not computationally feasible for anyone to factor n. A public exponent e is then selected such that it is relatively prime to $(p-1)(q-1)$, and the private exponent d is computed as $d = e^{-1} \pmod{(p-1)(q-1)}$. The public key consists of the pair (e, n), while the private key consists of (d, n). The two primes p and q should also never be revealed. RSA key sizes are defined by the bit length of the modulus n, with contemporary standards prescribing a minimum of 2048 bits and recommending 3072 bits or more for long-term security [86]. Therefore, the computational overhead of RSA encryption and decryption increases as the size of the modulus integer increases. Encryption of a message M is performed through modular exponentiation: $C = M^e \pmod{n}$. The decryption of cipher text is similar, with the private exponent d being used: $M = C^d \pmod{n}$.

Elliptic Curve Cryptography (ECC) is a form of public-key cryptography based on the algebraic structure of elliptic curves over finite fields. Its security relies on the Elliptic Curve Discrete Logarithm Problem (ECDLP): given points P and $Q = kP$ on a curve, it is computationally infeasible to recover the scalar k. This problem is regarded as significantly harder than integer factorization, the foundation of RSA. An elliptic curve over a finite field $\mathbb{F}_p$ (with $p > 3$) is defined by Equation 2.2, where p is a prime

specifying the field size and a and b are curve parameters.

$$y^2 \equiv x^3 + ax + b \pmod{p}, \tag{2.2}$$

Cryptographic use of these curves requires domain parameters: a generator point G on the curve, its order n (the number of points generated by repeatedly adding G to itself), and the cofactor h (the ratio between the total number of curve points and n). For key generation, each user selects a random integer d as their private key and computes the corresponding public key $Q = dG$. Encryption, decryption, and key exchange protocols rely on scalar multiplication ($kP = P + P + \cdots + P$, k times), which is easy to compute in the forward direction but infeasible to invert without knowing k. ECC provides the same level of security as RSA but with much smaller keys. For example, a 256-bit ECC key offers security roughly equivalent to a 3072-bit RSA key (≈128-bit, comparable to a symmetric cipher with a 128-bit key). This efficiency translates into faster computations, lower memory requirements, and reduced bandwidth, making ECC particularly attractive for constrained environments such as smart cards, IoT devices, and mobile systems. To avoid insecure parameter choices, cryptographic standards define a fixed set of trusted curves. Among the most widely used is NIST P-256, adopted in U.S. federal systems [62] and communication protocols such as Transport Layer Security (TLS) 1.3 [76]. P-256 is defined by the equation $y^2 \equiv x^3 - 3x + b \pmod{p}$ with a fixed generator point G. However, because the seed used to generate the parameter b was never disclosed, concerns have been raised about potential hidden weaknesses [12]. Despite this, P-256 remains one of the most widely deployed curves. Other examples include Curve25519, known for its efficiency and robust security [3].

2.1.3 X.509 Certificates

Asymmetric cryptography also faces challenges related to key authenticity, as attackers may impersonate entities or perform man-in-the-middle attacks. To address this issue, public keys are validated or bound to identities through digital certificates, digitally signed documents issued by trusted Certification Authoritys (CAs) that bind a public key to an entity. The most widely used format is the X.509 standard [10], which defines the certificate structure and supporting infrastructure. An

X.509 certificate is divided into three parts: tbsCertificate, signatureAlgorithm, and signatureValue. The tbsCertificate contains the main data, including the X.509 version number of the certificate, a serial number, the signature algorithm identifier used by the CA, the issuer name, the validity period, the subject name, and the subject's public key information. The second part of the certificate, signatureAlgorithm, specifies the cryptographic algorithm used by the CA to sign the certificate, while the third part, signatureValue, contains the digital signature over the tbsCertificate, guaranteeing its integrity and authenticity.

The X.509 standard has defined three versions: Version 1 includes only basic fields, Version 2 adds issuer and subject unique identifiers, and Version 3 introduces extensions. Modern certificates are almost exclusively X.509 version 3, since the extension mechanism is essential for specifying constraints, usages, and additional identifiers. Typical v3 extensions include Basic Constraints to distinguish between CA and end-entity certificates, Key Usage and Extended Key Usage to define permitted cryptographic operations, Certificate Policies, Authority and Subject Key Identifiers for chain validation, and Subject Alternative Names for identifiers such as email addresses, IP addresses, or card-specific data.

The structure of X.509 certificates is defined using Abstract Syntax Notation One (ASN.1), a standardized notation for describing structured data independently of implementation. In practice, the definitions are encoded using Distinguished Encoding Rules (DER), a strict subset of ASN.1 Basic Encoding Rules (BER) that ensures a unique binary representation. ASN.1 also defines the Object Identifier (OID) type, a hierarchical numeric label that uniquely identifies algorithms, attributes, and extensions. OIDs are widely used in X.509 to specify signature algorithms, key usages, certificate policies, and extension fields, ensuring consistency and interoperability across systems.

2.1.4 Message Authentication Codes

Message authentication is the process of verifying that the received messages came from the alleged source and were not altered in transit.

A common way to provide message authentication is to append a MAC to messages. These codes consist of fixed-length values obtained by specific algorithms that are a function of the message and a secret key shared only between the two communicating parties. MACs can be thought of as a cryptographic checksum:

$$\text{MAC} = C(K, M) \tag{2.3}$$

where C is the MAC function, K is the shared secret key, and M is the input message.

When A intends to send a message to B, the message, along with its MAC, is sent. The recipient, B, calculates the MAC for the received message body, using the same key, and compares it with the appended MAC. If there's a match between both MACs, B gains assurance that the message has not been altered and came indeed from A since the key K is only known by these two parties.

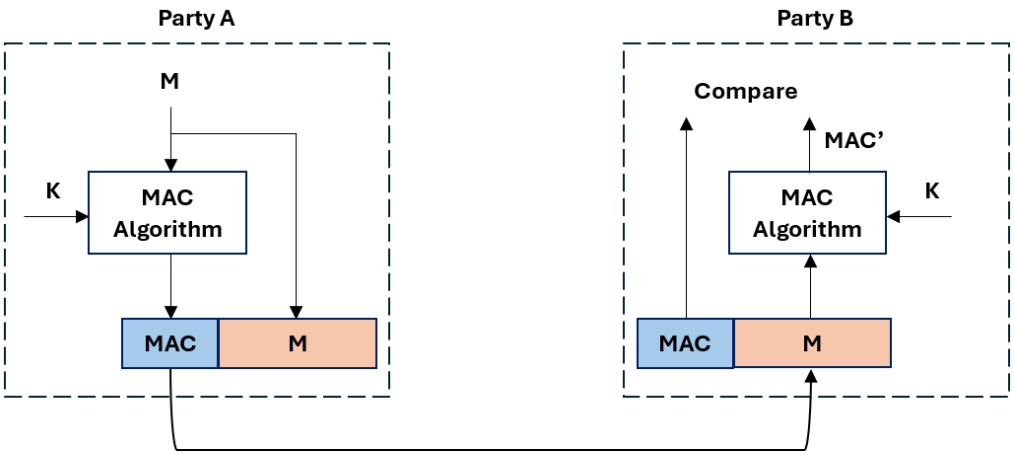


Figure 2.8: *Working principle of MACs in communication between two parties.*

In addition to message authentication, confidentiality can also be ensured if the entire block, including the message and MAC, is encrypted using either symmetric or asymmetric cryptography. In this scenario, communication security typically depends on the cryptographic strength of the algorithms and the key length, in bits. MACs can be based on existing hash functions or block ciphers. Hash-based algorithms are referenced as Hash-based Message Authentication Codes (HMACs), and cipher-based algorithms as Cipher-Based Message Authentication Codes (CMACs).

2.1.5 One-Way Hash Functions

A one-way hash function takes a message of arbitrary length as input and produces a fixed-size output called a message digest. The digest is generated through iterative compression functions, producing a condensed representation of the input. A key property of cryptographic hash functions is the avalanche effect: even the smallest change to the input will, with overwhelming probability, result in a completely different digest.

These properties make hash functions useful for verifying message integrity, since any tampering will alter the digest. They are also essential in the generation and verification of digital signatures and MAC, in password storage (where only the digest is stored), and in the generation of random numbers. For cryptographic use, a hash function h must satisfy two main security requirements: it must be computationally infeasible to find two different inputs x_1 and x_2 such that $h(x_1) = h(x_2)$ (collision resistance), and it must be infeasible, given a digest y, to find an input x such that $h(x) = y$ (preimage resistance).

The most widely used family of algorithms is the Secure Hash Algorithm (SHA) series, which includes SHA-1, SHA-2, and SHA-3. Although SHA-1 and SHA-2 follow similar design principles – padding the message, splitting it into fixed-size blocks, and applying a compression function to iteratively update the hash state – SHA-3 is based on a fundamentally different construction (sponge function). The variants differ in parameters such as block size, word size, and digest size.

A summary of these variants is presented in Table 2.1, which highlights the key properties of the SHA family. SHA-1, designed in the 1990s, is now considered insecure with well-known vulnerabilities [83, 91] and is being phased out by National Institute of Standards and Technology (NIST) [60]. SHA-2 and SHA-3 (standardized in 2015) are currently recommended, with SHA-2 still the most widely used in practice.

A more detailed description of the various SHA algorithms is given in [58].

Table 2.1: Secure Hash Algorithm properties.

Algorithm	**Message Size [bits]**	**Block Size [bits]**	**Word Size [bits]**	**Message Digest Size [bits]**
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

2.1.6 Digital Signatures

A digital signature is a data string produced by a signature generation algorithm that associates a message with a given entity. Suppose that A intends to send a message to another party, B. In addition, A wants to assure B that the message originated from A. To achieve that, A uses a secure hash function to generate a message digest. This digest, together with A's private key K_{PR}, serves as input to a signature generation algorithm G which produces a digital signature S for the message digest d using private key K_{PR}:

$$S = G(K_{\mathrm{PR}}, d) \tag{2.4}$$

A sends the message, along with its signature, to B. The recipient, B, calculates the digest for the message and provides it together with A's public key, K_{PU}, to a signature verification algorithm. If the algorithm returns the result that the signature is valid, B is assured that the message was signed by A. Since only A has the private key used to sign the message, no one else could have created a signature or altered the message in transit; therefore, authentication and integrity are assured.

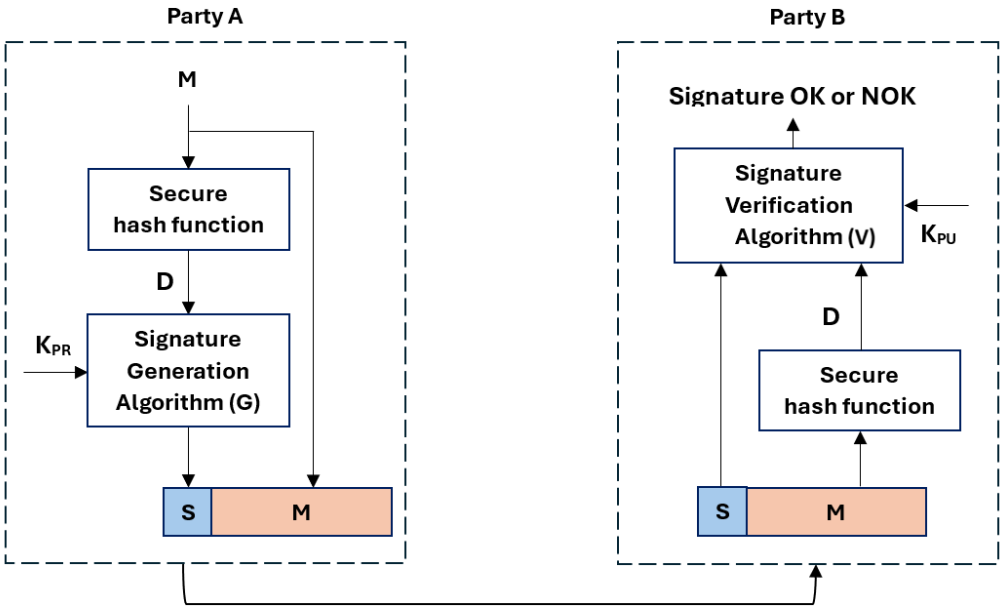


Figure 2.9: *Working principle of digital signatures when two parties A and B exchange messages.*

Given this, it is evident that for digital signatures to be of any cryptographic use, the private keys used should be securely stored, not only to prevent signature forgery but also to avoid confidentiality violations.

As discussed in Section 2.1.3, public-key certificates issued by certificate authorities are digitally signed. The example depicted in Figure 2.9 can also represent the process of obtaining this signature. Party A can be thought of as a certificate authority that signs a certificate to be issued to a specific entity. M represents that entity's' public key, ID information, ID_{A}, and a timestamp T for validity purposes. The private key K_{PR} corresponds to the CA private key. Therefore, the signature will consist of:

$$S_A = G\Big(K_{\mathrm{PR}},\, H\Big(T \,\|\, \mathrm{ID}_A \,\|\, K_{\mathrm{PU},A}\Big)\Big) \tag{2.5}$$

If another party B wishes to communicate with A, it obtains A's public-key certificate and verifies its authenticity using the CA's public key K_{PU} and the certificate's signature.

$$V(K_{\mathrm{PU}},\, S_A) \stackrel{?}{=} H\Big(T \,\|\, \mathrm{ID}_A \,\|\, K_{\mathrm{PU},A}\Big) \tag{2.6}$$

Since the signature verifies correctly under the CA's public key, B is assured the certificate came from the certificate authority and consequently that it is indeed communicating with A.

2.1.7 Public Key Infrastructure

Public Key Infrastructure (PKI) encompasses all the hardware, software, personnel, policies, and procedures that support the creation, distribution, use, storage, and revocation

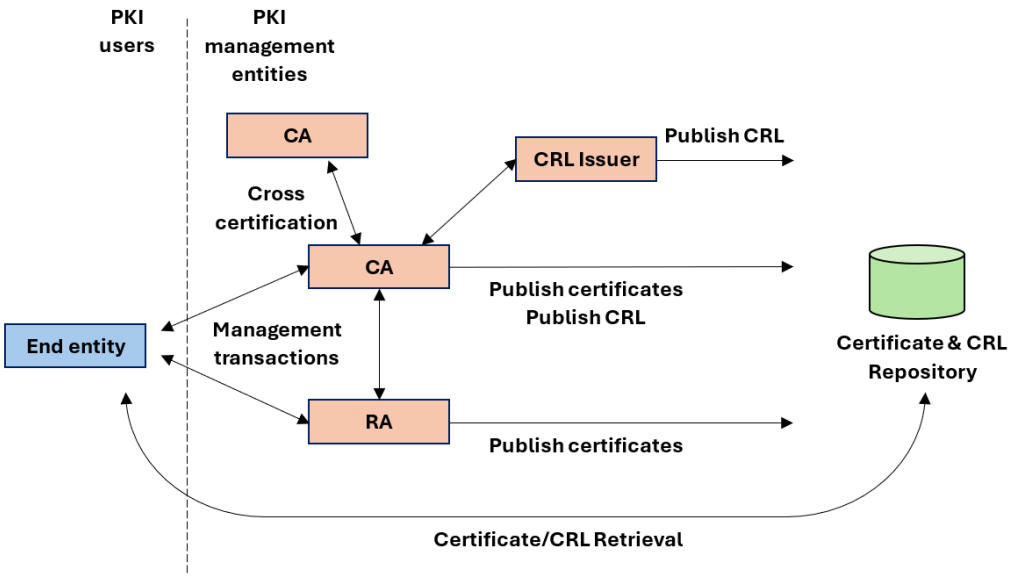


Figure 2.10: *PKI components and their interaction.*

of digital certificates. The widely adopted X.509 recommendation proposes an architectural model for PKI with the components depicted in Figure 2.10 [10]:

a) **End entities** are users of PKI certificates and/or end systems that are the subject of these certificates.

b) **Certification Authoritys (CAs)** are the entities responsible for issuing digital certificates and Certificate Revocation Lists (CRLs). CAs may also exchange information to support cross-certification between different PKIs.

c) **Registration Authorities (RAs)** are optional components, separate from CAs, that assume management functions delegated by the CA (e.g., identity authentication and name assignment).

d) **CRL issuer** is ann optional component that a CA can delegate to generate and sign lists of revoked certificates (CRLs).

e) **Repository** is a storage service that holds certificates and CRLs so they can be retrieved by end entities.

The whole concept of a PKI is based on trust. If there is no trust in the CA, then any certificate they have issued is not trusted.

2.2 Access Control Systems

Access control systems are electronic systems that implement automated approval for authorized personnel to access a physical location through a security portal and/or a logical resource, such as an information system. A physical location is a secured facility,

such as a building, parking garage, or server room, that personnel wish to access. A logical resource, on the other hand, is usually a network or a specific network location, such as a folder or a file. Such systems are composed of the following core components:

a) Credentials;

b) Credential readers;

c) Security portals and their respective locks;

d) Access control panels;

e) Servers;

f) Databases;

g) Client workstations.

Credentials consist of objects or data structures that authoritatively bind an identity to a card or a token possessed and controlled by a credential holder. Smart cards, badges, RFID transmitters, and biometric attributes such as fingerprints are examples of credentials used in access control systems (see Section 2.3 for details). Conversely, smart card readers, keypads, and biometric readers are examples of common credential readers.

Security portals are the entries into or out of a secured facility. Examples of security portals are automatic doors, turnstiles, and barrier gates.

Access control panels are responsible for implementing access control decisions (whether or not a user is allowed access to a physical resource) by releasing security portal locking mechanisms. Additionally, they connect to multiple readers, door position sensors, and other electronic components via physical cabling. These panels also communicate with the system servers to send event and log information as well as request user records in order to implement access control decisions.

The servers run all the application software that manages users, alarms, report generation, device configuration, and communication with external systems such as Closed-Circuit Television (CCTV) and fire detection systems. In addition, these servers also interact with the controller panels and the systems' databases that hold user data, access control events, and equipment and system configuration records.

Access control clients are workstations used for operational tasks. For instance, alarm and event monitoring, remote control of portal locking mechanisms, and credential management are examples of tasks performed in client workstations.

Figure 2.11 shows a typical architecture model for an access control system.

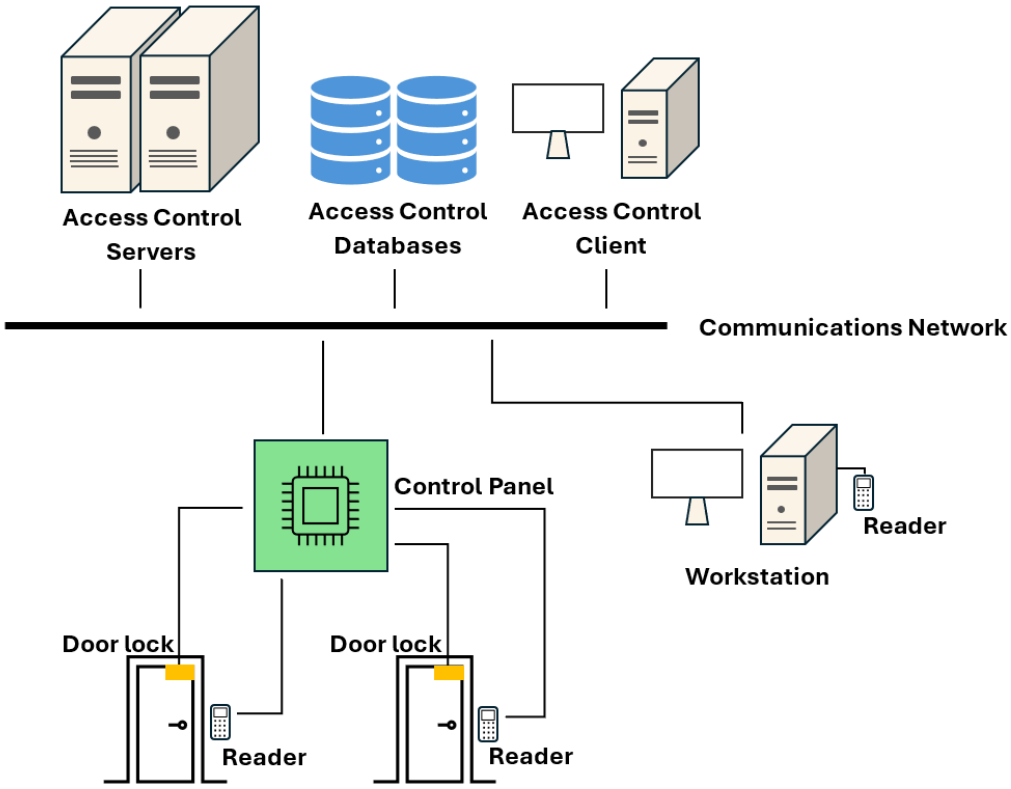


Figure 2.11: *Typical architecture model for an access control system.*

The interaction of these components in physical and logical access control is as follows.

For both scenarios, users present their credentials to a credential reader. The credential holder can be authenticated using one or some combination of authentication mechanisms. This is a common practice in critical infrastructures, such as government facilities.

In the case of physical access control, when applicable, after successful authentication of the credential holder, the reader sends the credential information to an access control panel. The control panel communicates with the access control server through a communication network, typically a TCP/IP network, to retrieve a database record associated with a user based on their respective credential information. This way, the control panel can determine whether there are access rights or permissions corresponding to that user and, if so, if it has assigned permissions to access the location in question. If authorization is granted, the control panel releases the security portal locking mechanism, allowing the portal to be opened. If authorization is unsuccessful, the locking mechanism remains locked.

With regard to logical access control, no control panels are involved in the process. Therefore, readers send credential information directly to authentication services that then enforce authorization to access the logical resource. These readers are directly connected to workstations or other types of terminals used to access logical resources. For instance, in the context of workstation access control, smart card-based authentication mechanisms are very common. These require users to present their smart cards to a

reader that is already embedded in the machine or externally connected to the workstation through a Universal Serial Bus (USB) cable, for example.

2.3 Credentials

Credentials consist of objects (tokens) or data structures that are authoritatively bound to an identity. Badges, RFID tags, access cards, passwords/PINs, and biometric attributes such as fingerprints are examples of credentials used in access control systems, with the latter four being the most common. In the next subsections, the following credentials are discussed: i) password/PIN; ii) access cards; iii) RFID token systems; and iv) biometric attributes.

2.3.1 Passwords and Personal Identification Numbers

Passwords typically consist of a string of characters, such as letters, numbers, and other symbols that are associated with an entity. Therefore, Traditional Passwords can be used to confirm an entity's identity to provide access to logical resources such as a website or a workstation. This type of credential is intended to be memorized by the user; however, it must also be of sufficient complexity that it would be impractical for an attacker to guess or otherwise discover it.

Personal Identification Numbers (PINs) consist of short numbers (4 to 8 digits) used as a memorized password to verify identity and gain access to a resource. In some cases, they can also be alphanumeric. In any case, they can be used for both logical and physical access. The user may have to provide a PIN to access a data center room or log in to a workstation. To prevent brute-force attacks, given its small key space, preventive measures are taken, such as blocking the system after a specified number of incorrect attempts have been made.

Common threats to the use of password credentials are wiretapping and spoofing. Passwords and PINs sent over insecure communication channels are prone to being captured by an attacker who is secretly wiretapping that channel. On the other hand, spoofing consists of a set of techniques used to steal a user's password or PIN. These involve the use of keyloggers, a form of malware or hardware that covertly tracks keys struck on a keyboard or PIN pad by a user, and social engineering tactics such as shoulder surfing. Another relevant threat is guessing techniques, such as brute-force or dictionary attacks, which involve attempting multiple combinations of passwords to find a match with the user's password. Therefore, organizations generally adopt password and PIN policies to require users to use strong passwords and PINs. In the case of passwords, it is generally required to set a password that consists of a mix of letters, numbers, and special characters, as well as a minimum length, to increase its uncertainty. As for PINs,

users are recommended to avoid using birthdates and sequential numbers. Additionally, these policies also require passwords and PINs to be changed periodically.

2.3.2 Access cards

Access control systems utilize various types of access cards that differ in technical features and physical characteristics. Regardless of the technology, it is very common for access cards to store a unique cardholder identifier code and other data objects that distinguish cardholders within an access control system, and other visual features such as a color photo of the cardholder, their name, expiration date, and other icons and logos representing entity and company affiliation.

Smart cards are a type of access card that feature one or more embedded integrated circuits. Although these cards may be of different formats, in access control applications, it is very common for the ID-1 (85.60 mm × 53.98 mm) format, specified in ISO/IEC 7810 (Identification Cards – Physical Characteristics), to be adopted. ISO/IEC 7810 is a widely used standard that specifies four different sizes of identification cards, as shown in Figure 2.12, each with a nominal thickness of 0.76 mm.

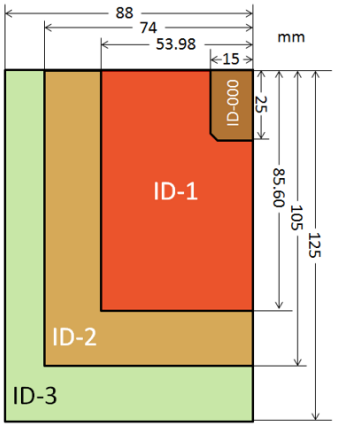


Figure 2.12: *Four different sizes of identification cards as specified in the ISO/IEC 7810 standard.*

The integrated circuits embedded in smart cards may be capable of performing the functions of a CPU, memory, and input/output interface, just like a embedded system. Section 2.5 discusses this in more detail. Smart cards connect with the outer world through a contact or contactless interface. Contact-based smart cards have a set of contact pads that establish a physical electrical connection with the pins of the outer device to communicate with it. By contrast, contactless smart cards exchange data with external devices without a physical connection. RFID is discussed in more detail in Section 2.3.3.

2.3.3 RFID token systems

Radio-frequency identification (RFID) is a technology that has been around since the 1940s. Due to its convenience, low maintenance requirements, and ease of use, it has been deployed in many different applications [54]. In addition to access control systems, automated toll collection, and asset tracking are common use cases of this technology.

An RFID system is composed of two elements: i) a transponder token and ii) a reader, used to read data from the transponder token and, in some cases, depending on the type of technology used, write data to the token [26]. Figure 2.13 illustrates these elements and their interaction.

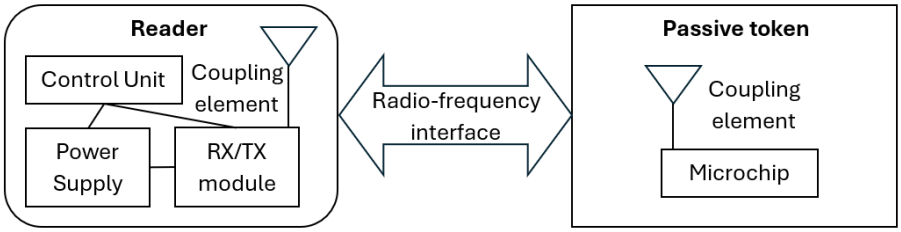


Figure 2.13: *RFID system.*

The transponder token consists of a coupling element, such as an antenna coil or a microwave antenna, and a microchip. It is used as a credential because it stores a Unique Identifier (UID) and other data on the microchip, which is associated to the token holder. Key fobs, illustrated in Figure 2.14, book tags, and some types of access cards are examples of RFID tokens.

Figure 2.14: *Example of an RFID token - a key fob.*

Depending on the source of power, RFID tokens are classified as either active or passive. Active tokens contain their own power supply, whereas passive tokens receive their energy from the reader, which is typically the case for access control applications.

The reader contains a radio frequency transmitter (TX) and receiver (RX) module (transceiver), a control unit, its own coupling element, and a power supply. In more complex RFID systems, the reader also has a communication interface to forward data received from the token to other systems.

The following focuses on the operating principle of passive RFID tokens, as these are

typically employed in access control systems. A comprehensive treatment of these concepts can be found in [54]. The working principle of these systems is based on inductive coupling. The readers' RX/TX module passes a large alternating current through its coupling element, which generates an alternating magnetic field in its vicinity. RFID systems can operate at different frequencies, ranging from 135 kHz (longwave) to 5.8 GHz in the microwave range. If the RFID token is positioned close to this field, an alternating voltage is induced in its coupling element, which is then used to power the microchip. Consequently, the token can send data, such as its unique identifier, to the reader by influencing the strength of this magnetic field. The reader recovers this information by monitoring changes to its magnetic field.

2.3.4 Biometric attributes

Biometrics are used as credentials – much like a card or PIN – by measuring physical or behavioral traits (e.g., fingerprints, facial features, the iris, or voice characteristics). To work in access control, a biometric system needs: a reader to capture raw data; a quality-assessment step followed by feature extraction to ensure a good sample and derive distinguishing features; a matcher/decision module to compare those features to stored templates; and a database where templates are stored, either in the central system or, sometimes, on a smart card.

Two of the most common traits are fingerprints and the iris. Fingerprints are ridge-and–valley skin patterns whose global shapes (loops, deltas, whorls) and local *minutiae* (ridge endings and bifurcations), plus ridge frequency and orientation, form a template matched against enrolled prints. The resolution and sensing area of the scanner strongly have a significant impact on matching performance.

Iris recognition uses a camera ("stop and stare") to image the colored ring around the pupil, isolates the iris boundaries, often maps it to a normalized rectangular representation using a polar transform, and extracts numerical features that remain stable despite pupil dilation. For a detailed treatment of biometric traits, capture methods, and matching algorithms, see *Handbook of Biometrics* [45].

2.4 Authentication

Authentication is the process of verifying an identity claimed by or for a system entity. In the context of access control systems, this process consists of two phases: i) configuration phase, and ii) operational phase.

The configuration phase consists of two steps: registration and provisioning. In the registration step, applicants request their credentials and then provide identity documents, such as identification cards, passports, and driver's licenses, to a registration authority

or security center for validation. Additionally, a photo is generally taken or provided, and biometrics may also be acquired. If all the provided documents and information are validated, the process advances to the provisioning step, where the credentials are personalized and issued to the intended applicant.

The operational phase takes place throughout the use of the access control system. It consists of an identification and verification step where users present or generate authentication information—*authenticators* – that verify the binding between the entity and the credential [80]. The system then checks if the authentication information is valid. Authenticators may consist of, or be derived from, one of the following factors:

- **Something the user knows (knowledge):** PINs, passwords, secret patterns, and answers to a prearranged set of questions.
- **Something the user possesses (possession):** physical or logical keys, access cards, RFID tokens, X.509 certificates.
- **Something the user is (biometrics):** fingerprints, iris images, facial images.

To increase the overall security of the identification step in the authentication's operational phase, Multi-Factor Authentication (MFA) is used. MFA combines two or more authenticator factors so that if one factor is compromised, the remaining factor(s) can still foil an attacker.

For the sake of simplicity, throughout this section, the operational phase of authentication is referred simply as *authentication*. An *authentication scheme* is a definition of the steps required for the authentication process. These schemes may be:

- **Single authentication:** Only one entity is authenticated by the scheme.
- **Mutual authentication:** Two communicating entities authenticate each other by means of the scheme.
- **Third-party authentication:** Two communicating entities are authenticated by an external server trusted by both.

In the following subsections, the most common authentication schemes are discussed in the context of physical and logical access.

2.4.1 Password and PIN-based authentication

Password-based authentication is the oldest authentication scheme for accessing logical resources. It falls under the category of knowledge-based authentication and is based on usernames and passwords. The user enters a username and a password, which provide a claim of identity and evidence supporting that claim, respectively. Then, the system

checks that the password matches the corresponding data it holds for that username, and that the stated identity is authorized to access the resource. Passwords can be stored in the system in the form of plaintext, ciphertext, or message digests obtained from hashing functions. Storing passwords in the form of ciphertexts or message digests implies additional processing in the checking process, but on the other hand, provides an additional layer of protection in case of information leakage of information or an intrusion, since attackers still won't be able to obtain the actual password. However, the security of this process is obviously dependent on the strength of the passwords, encryption keys, and algorithms used.

PIN-based authentication schemes are similar to password-based ones, with the exception of generally not requiring users to input a username. The user provides their PIN, and the system checks whether the PIN is valid or not. A common practice in the context of access control systems is to use PINs to enable access to other credentials resident on a cryptographic token that provides additional factors of authentication, such as a cryptographic key and digital certificates. In this case, the PIN is used to authenticate the user to the token, proving ownership of the token. Additionally, PINs are also used in conjunction with other authenticators, such as biometrics, in a multi-factor authentication environment, as discussed in Subsection 2.4.3.

2.4.2 Challenge-Response authentication

Challenge-Response authentication mechanisms are based on demonstrating knowledge of a secret without disclosing it to the verifier entity. To explain the working principle of this authentication scheme, consider an example where a given entity A intends to be authenticated by another entity B – unilateral authentication. Furthermore, K consists of a symmetric key that is known to both parties.

Entity B starts by sending a random value R_B to A. This random value (a *challenge*) is a cryptographic nonce – an unpredictable value used only once for a specific purpose. Using fresh randomness prevents replay of previously valid exchanges.

Consequently, entity A will return a response, which will be a message digest resulting from the hash of the combination of R_B and K, $H(R_B, K)$. This combination could be simply the concatenation of both items, for example. Entity B will then proceed to calculate its hash for the same combination of R_B and K, since it also knows K and the random value R_B was generated by entity B itself. If both digests match, entity B assumes it has successfully authenticated A.

Figure 2.15 illustrates this scheme taking place.

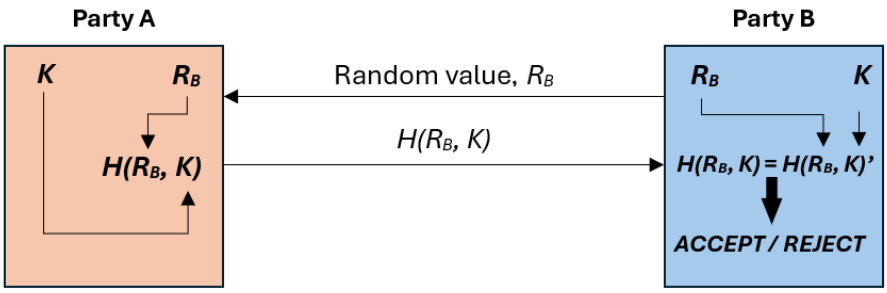


Figure 2.15: *Challenge-Response authentication scheme.*

This scheme could also be used to provide mutual authentication, if A sends its random value R_A along with its response to B. Afterwards, B will send its response to A, allowing it to also match digests. To increase the overall security of the scheme, the challenge and response should be encrypted to prevent interception and integrity violations.

It is also important to note that this authentication scheme allows different variations in terms of how the challenge and response values are produced. In terms of the challenge, they could be composed of identifying data, timestamps, random numbers, or even sequence numbers, provided appropriate replay protections are in place. In contrast, the responses may be computed using either symmetric-key or asymmetric-key cryptographic primitives.

In symmetric cryptography-based challenge-response, after R_B is sent from B to A, R_B is encrypted through a symmetric encryption algorithm with a key K shared by both A and B, E_K. A then sends this ciphertext to B.

$$A \leftarrow B : R_B \tag{2.7}$$

$$A \rightarrow B : E_K(R_B) \tag{2.8}$$

B decrypts the received cipher and checks that the random number R_B matches the one initially sent, which was generated by itself.

As for asymmetric cryptography-based challenge-response, B computes a challenge c

$$c = E_{K_{\mathrm{PU},A}}(r) \tag{2.9}$$

by encrypting R_B with A's public key $K_{\mathrm{PU},A}$. B then sends c to A. Upon receiving this information, A decrypts c using its private key to recover R'_B. Finally, A sends R'_B back to B, which concludes that authentication was successful if $R'_B = R_B$.

$$A \rightarrow B : K_{\mathrm{PU},A} \tag{2.10}$$

$$B \rightarrow A : h(R_B),\ E_{K_{\mathrm{PU},A}}(R_B) \tag{2.11}$$

$$A \rightarrow B : R'_B \tag{2.12}$$

Another way in which asymmetric cryptography could be used for challenge-response authentication schemes, is through the use of digital certificates, namely X.509. In this scenario, B starts by sending its generated random number R_B to A. Then, A responds with its public key digital certificate, cert_A, a random number generated by itself, R_A, and a digital signature, S_A, of R_A, and R_B. The signed R_A explicitly prevents chosen-text attacks, a form of cryptanalysis that can be used to obtain information about the encryption key or the plaintext.

$$A \leftarrow B : R_B \tag{2.13}$$

$$A \rightarrow B : \mathrm{cert}_A,\ R_A, S_A(R_A, R_B) \tag{2.14}$$

Upon receiving all this data, B uses the public key present in cert_A to verify that the signature provided by A, S_A, is valid.

The presented schemes provide unilateral authentication, but can be applied in both directions to a mutual authentication.

2.4.3 Multi-factor Authentication scheme

Multi-factor authentication was proposed to provide a higher level of security by combining two or more authentication factors for successful authentication. A common multi-factor authentication setup combines something the user knows – such as a username-password pair – with a biometric factor. Figure 2.16 illustrates such an authentication process where party A uses a multi-factor scheme based on a traditional password (username, password) pair, and biometrics to authenticate itself to party B.

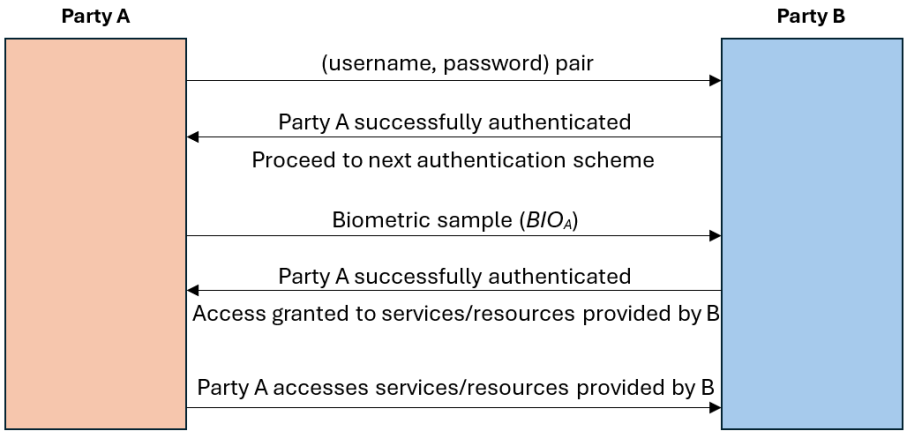


Figure 2.16: *Multi-factor Authentication scheme process where a given party A authenticates itself to a party B.*

Physical and logical resources are often subject to risk assessments to determine the appropriate authentication schemes and the number of authenticator factors required to

access them. For instance, the United States Federal Government facilities are categorized in three different security areas [61]: i) Controlled, which requires a single authentication factor; ii) Limited, which requires two authentication factors; and iii) Exclusion, which requires three authentication factors.

2.5 Smart Cards

The concept of smart cards emerged in the mid-1970s and has become increasingly popular over the years, being used in numerous applications. Smart cards are small plastic cards with embedded integrated circuits that perform functions similar to those of a regular computer. Their ease of use, ability to provide robust safety features (such as secret key storage with tamper-resistant security mechanisms), and support for on-card cryptographic operations are the most important reasons why these devices are used in a variety of applications, including mobile telecommunication, public transportation, identification, and payment systems.

The main components of smart cards are the same as of those of an embedded system — Central Processing Unit (CPU), an Numeric Processing Unit (NPU), memory, and input/output – and they may be implemented in a single or multiple smart card Integrated circuits (ICs). Smart cards include several types of memory for storing data and programs. The Random Access Memory (RAM) is the working memory of the card and is volatile, meaning it will lose its stored information immediately if power to the memory is removed. The Read-Only Memory (ROM) stores the card's operating system. The Electrically Erasable Programmable ROM (EEPROM), another non-volatile type of memory, is used to store data and applications. It can be electrically erased and reprogrammed via a reader/writer device. The CPU runs a dedicated operating system that makes the decisions concerning where data is stored and under which circumstances it will be transferred through its input/output interface. Additionally, the CPU is assisted by an NPU for cryptography-related numerical calculations. Figure 2.17 illustrates these components.

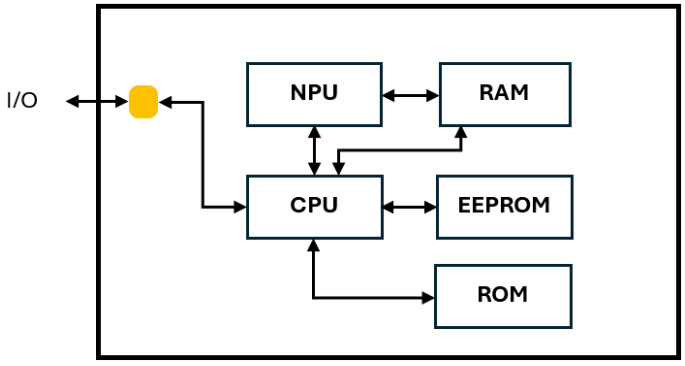


Figure 2.17: *Smart Card architecture.*

To communicate with the outside world, smart cards have a physical structure that allows them to interface with a reader/writer device for data exchange. Smart cards can be characterized by their physical interfaces – contact type, non-contact (or contactless), and hybrid (or dual-interface), which is a combination of both.

With a contact-type interface, the pins of the reader/writer must physically touch the smart card contact pad during data exchange. ISO/IEC 7816-3 specifies the physical characteristics, electronic signals, and transmission protocols of a contact smart card. The contact pad, shown in Figure 2.18, has eight pins with the following functions:

- **VCC:** used to supply the smart card with power; voltages can range from 5 V, 3 V, or 1.8 V (classes A, B, and C, respectively).
- **RST:** used to reset the smart card microcontroller (the card responds with an Answer-To-Reset – ATR).
- **CLK:** used to supply a clock signal that goes into the smart card.
- **GND:** the electrical ground.
- **I/O:** used to exchange data between the smart card and a terminal.
- **RFU (AUX):** "reserved for future use"; usually denoted as the AUX pin, used, for example, as an operation trigger or as a second I/O communication channel.

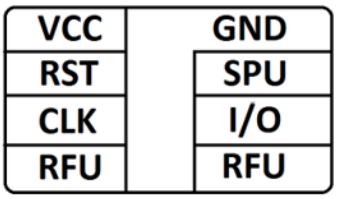


Figure 2.18: *Smart card contact pad according to ISO/IEC 7816-3.*

In contrast, a non-contact type interface works by transmitting and receiving information without a physical connection between the smart card and the reader/writer device. ISO 14443 outlines the specifications for contactless cards and terminals to ensure industry-wide compatibility.

This is achieved through radio frequency (RF) communication (13.56 MHz) or Near-Field Communication (NFC). In a non-contact interface, the smart card and the reader/writer communicate wirelessly through an RF interface, allowing for more convenience and durability, as there is no wear and tear associated with physical contact. Figure 2.19

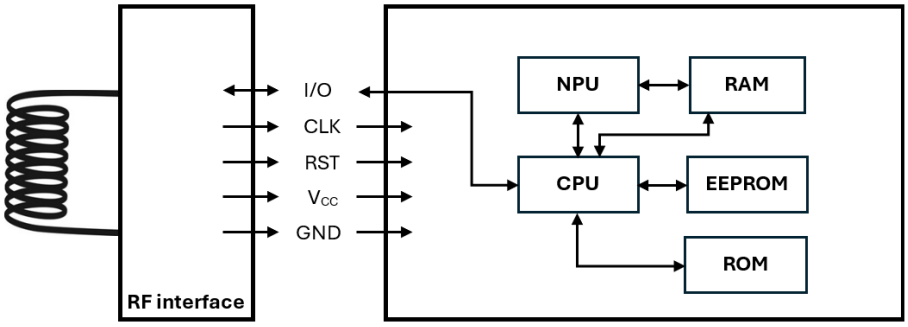


Figure 2.19: *Smart Card with contactless interface.*

The hybrid (or dual-interface) smart card combines both contact and non-contact interfaces, offering the flexibility to interact with different types of reader/writer devices.

2.5.1 Data Exchange

After a smart card has been inserted in an interface device, its contacts are physically connected to those of the interface device. Then, a set of procedures is conducted to operate and exchange information with the cards. These consist of: i) activation, where the card is powered by the interface device and information is exchanged between the two in order to establish the transmission protocol and parameters to be used during the exchange; ii) information exchange; and iii) deactivation, where the interface device stops the clock signal and removes power from the card.

For contactless smart cards, the overall communication process follows the same logical structure as for contact-based cards, with activation, information exchange, and deactivation phases. However, instead of relying on direct electrical contact, these steps are performed through the same inductive-coupling mechanism described in subsection 2.3.3.

Transmission protocols define the communication process and the mechanisms for handling transmission errors. There are three predominant protocols used worldwide: the $T = 0$ and $T = 1$ protocols, defined by ISO/IEC 7816-3 [39], and the USB protocol [90]. Both the $T = 0$ and $T = 1$ protocols are asynchronous and based on half-duplex transmission. This means that the same communication line, the I/O pin shown in Figure 2.18, is shared by the smart card reader and the smart card, so only one device can communicate at a time. In the case of contactless data exchange, the transmission protocol, defined by ISO/IEC 14443-4 [8], is also based on half-duplex transmission and its working principle is strongly based on the $T = 1$ protocol. The USB protocol only defines how smart card readers communicate with host systems over USB, not governing the communication between the reader and the card itself.

The T=0 transmission protocol The T=0 is a byte-oriented protocol historically used worldwide in GSM SIM cards. This means that the smallest unit transmitted is a

single byte. In the event of a transmission error being detected, retransmission of the incorrect byte must be requested immediately. The error detection mechanism of the T=0 protocol consists of the use of a parity check at the end of each byte. This mechanism provides reliable recognition of single-bit errors but does not detect two or more bit errors.

The T=1 transmission protocol The T=1 is a more recent transmission protocol used to exchange data sliced into several blocks. A block is a byte string conveyed in asynchronous characters. Both the smart card reader and the card may initiate the transmission of commands. In contrast to the T=0 protocol, in the event of a detected transmission error, an entire block, which can be composed of a sequence of bytes, must be retransmitted. The T=1 protocol also has the advantage of providing stronger error-handling and flow-control mechanisms than T=0.

The USB transmission protocol The USB protocol requires an additional hardware component, a USB interface, to be connected to the smart card CPU. The use of this protocol has an advantage, namely, a higher transmission rate than both the T=0 and T=1 protocols.

At the application level, communication between smart cards and terminals is based on the exchange of APDU whose structure is specified by ISO/IEC 7816-4. These data units are used to send commands and receive responses from the card. APDUs are encapsulated in Transmission Protocol Data Unit (TPDU) that are converted into electrical signals as specified by ISO/IEC 7816-3. Figure 2.20 illustrates this process.

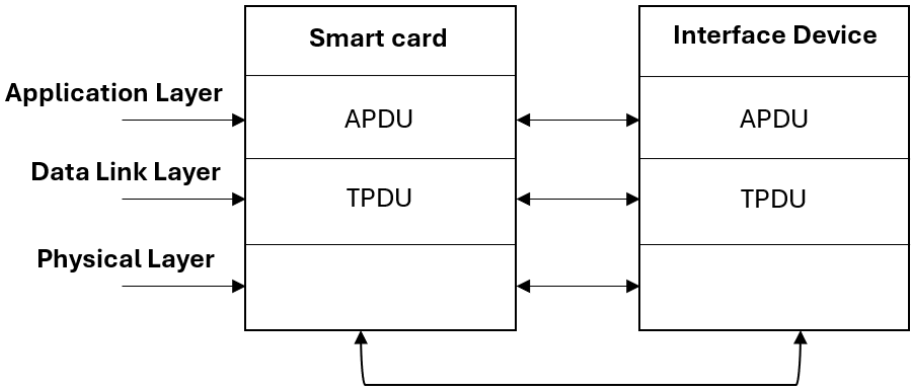


Figure 2.20: *Smart Card data exchange layers*

The T=0 transmission protocol provides little layer separation, as the TPDU and the APDU have quite a similar structure. In contrast, the T=1 protocol provides strict layer separation, as its TPDU has a different structure that resembles an Ethernet frame, including source and destination addresses for the block, as well as other protocol control bytes.

A command APDU is composed of a four-byte header: the class of command byte (e.g., interindustry or proprietary), CLA, the instruction code byte (e.g., 0xE4 means delete file), Instruction (second) byte of a card command (INS), and two parameter bytes P1 and P2 (e.g., provide the file ID when executing a create file instruction). Additionally, a command APDU can also include an optional body that varies in size. It can contain three components: the length of the data in the command, L_c, the data field, and the length of the data expected from the smart card response, L_e. Figure 2.21 depicts the different possible combinations of a command APDU.

Combination 4, command APDU

CLA	INS	P1	P2	L_C	Data	L_e

Combination 3, command APDU

CLA	INS	P1	P2	L_C	Data	(empty)

Combination 2, command APDU

CLA	INS	P1	P2	(empty)	(empty)	L_e

Combination 1, command APDU

CLA	INS	P1	P2	(empty)	(empty)	(empty)

Figure 2.21: *Smart Card data command APDU possible combinations.*

Due to its byte-oriented nature, the T=0 protocol does not support command APDU combination 4. As for the response APDU, it is composed of an optional data body and two trailer bytes, First byte of a 2-byte status word (SW1) and Second byte of a 2-byte status word (SW2). These bytes contain the response code indicating the command processing status. Figure 2.22 shows the different possible combinations of a response APDU.

Combination 2, response APDU

Data	SW1	SW2

Combination 1, response APDU

(empty)	SW1	SW2

Figure 2.22: *Smart Card data response APDU possible combinations.*

To better illustrate how APDUs are used to exchange data between smart cards and interface devices, consider an example scenario where a user inserts their access control smart card into a reader and is then prompted to enter their PIN, an 8-digit code. After inserting and submitting the PIN, a specific APDU, VERIFY, is sent to the smart card for PIN verification. The command structure is described in Figure 2.23.

CLA	INS	P1	P2	L_C	Data (the PIN)	L_e
00	20	XX	XX	08	38 30 31 32 38 30 31 32	(empty)

Figure 2.23: *Example APDU command.*

The class of command byte, CLA, assumes a value of "00," meaning it is an inter-industry class. The INS corresponds to "20", which means the command is of type VERIFY. For the sake of simplicity, Bytes P1 and P2 are considered irrelevant in this scenario. Lc indicates the length of the data field that contains the PIN. Since it's an 8-digit code, the length corresponds to 8 bytes; therefore, Lc assumes the value of "08". The data field contains the PIN itself. In this scenario, the typed PIN is correct. Therefore, the response APDU, represented in Figure 2.24, is sent back to the reader.

Data	SW1	SW2
(empty)	90	00

Figure 2.24: *APDU response to the previous example command.*

This particular combination of SW1 and SW2, according to ISO/IEC 7816-4, indicates that the command was executed successfully.

2.5.2 File Management

File management is particularly relevant in smart cards, as most applications are file-based and primarily involve providing read/write access, controlling file creation/deletion, and granting and monitoring access privileges. These tasks are performed by the smart card operating system.

The smart card file system is based on a tree structure specified by ISO/IEC 7816-4, a widely adopted standard for smart card file systems worldwide. This file system is composed of a single Master File (MF), which resembles the root directory of a computer operating system. The MF cannot store data directly, but it can contain other directories, called Dedicated Files, which act as folders. These Dedicated Files (DFs) can store application and operating system data in different available structures, called Elementary Files, which can also store data. DFs may contain lower-level DFs and/or Elementary Files (EFs) that logically belong together. More recently, a special type of file was introduced – the Application Dedicated File (ADF). This is a DF for an application that can be located in the file tree without any direct relationship to the root directory. Typically, an Application Dedicated File (ADF) holds all the files of a particular application.

As for EFs, they can be of two distinct types: i) working EFs, accessible to the exterior world; and ii) internal EFs, which are only accessible to the operating system. Figure 2.25 presents the taxonomy and file tree for these different types of files.

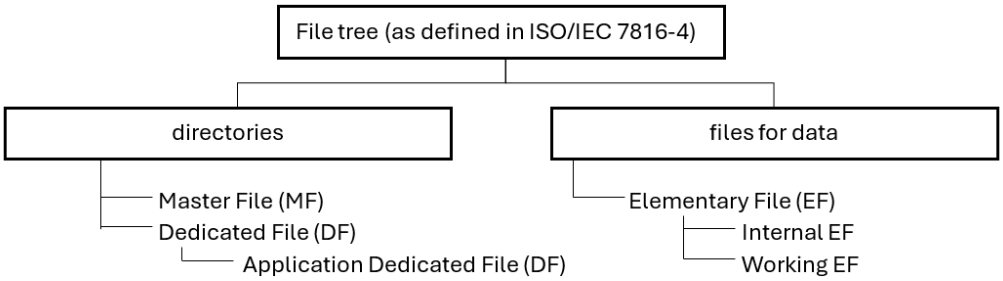


Figure 2.25: *Smart card file tree as defined in ISO/IEC 7816-4.*

Every file in a smart card has a header containing all the information relevant to itself, such as a two-byte File Identifier (FID) among other data elements [ISO/IEC 7816-4]. The MF has its own reserved File Identifier (FID) of '3F00'. All other FIDs can be freely chosen. As for Dedicated Files, a supplementary name termed DF Name is used in addition to their FID. The DF Name has a length of 1 to 16 bytes and includes an Application Identifier (AID) with a length between 5 and 16 bytes. The AID is composed of two fields – one mandatory and another optional. The mandatory field is the Registered Application Provider Identifier (RID), which has a fixed length of 5 bytes and is officially registered to identify an application worldwide. The optional field is the Proprietary Identifier extension (PIX) and acts as a proprietary identifier. The EFs are also assigned FIDs and, in addition, have a Short File Identifier (SFI), which can be provided as a parameter of a read or write command to select the EF directly without needing to specify its path.

2.5.3 Relevant Standards

This section examines the principal standards relevant to this work, grouped into three categories: (i) smart card physical and communication interfaces, (ii) application development frameworks and protocols, and (iii) security architectures and assurance mechanisms.

Smart Card Physical and Communication Interfaces International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 7810 describes the physical characteristics of identification cards, including card materials, construction, characteristics, and dimensions for four card sizes [41].

ISO/IEC 7816 is a series of standards that specify requirements applicable to identification cards intended for information exchange between the outside world and the integrated circuit in the card. Some of the most relevant standards are:

- ISO/IEC 7816-1 specifies physical characteristics for cards with contacts [37].
- ISO/IEC 7816-2 specifies dimensions and location of the contacts [38].

- ISO/IEC 7816-3 specifies electrical interface and transmission protocols for asynchronous cards [39].
- ISO/IEC 7816-4 specifies organization, security and commands for interchange [40].

ISO/IEC 14443 outlines the specifications for contactless cards and terminals to ensure industry-wide compatibility. This standard is divided into four parts outlining: i) physical characteristics; ii) radio frequency; iii) signal interface; iv) initialization and anti-collision and transmission protocol [5–8].

Security architectures and assurance mechanisms Smart cards used in high-security applications are often evaluated and certified under the Common Criteria (CC) (ISO/IEC 15408) [43], an international standard for assessing the security of IT products. For smart cards, dedicated Protection Profiles have been developed that address both hardware and software threats, including side-channel analysis and fault injection. Certification under Common Criteria provides assurance that the card platform meets recognized security requirements, which is why many government ID programs, banking applications, and other critical infrastructures mandate CC-certified smart cards.

Application Frameworks and Device Interfaces The Personal Computer/Smart Card (PC/SC) Specification [71] builds upon existing industry smart card standards – ISO 7816 and Europay, Mastercard, and Visa (EMV) [20, 36] – and complements them by defining low-level device interfaces and device-independent application APIs as well as resource management, to allow multiple applications to share smart card devices attached to general-purpose desktop computing systems.

2.5.4 Security

Although smart cards incorporate robust security features, they remain susceptible to various classes of attack. Therefore, it is important to understand both typical threats and the countermeasures used to mitigate them. The following paragraphs outline common attack classes and representative countermeasures. [75] provides a more in-depth discussion about these topics.

Social level attacks This type of attack is primarily targeted at IC designers, software developers, and cardholders. Some attacks are identical to those witnessed in computer systems, such as shoulder surfing to steal smart card PINs. Other examples include establishing relationships with developers and designers to leverage their psychology and emotions, thereby obtaining valuable architectural details or infiltrating the development supply chain to insert malware or backdoors. Social attacks can be mitigated by blocking the smart card upon exceeding a given number of bad PIN tries, using multi-factor

authentication, and, at the design and development level, enforcing secure software development methods such as tracking modifications to a source code repository, not allowing programmers to work alone on a project, mapping features to requirements, and prioritizing code reviews.

Logical level attacks These attacks exploit flaws in smart-card Operating Systems (OSs), applications, and protocols to gain unauthorized data access or perform operations outside the card/data lifecycle. Typical methods include impersonating readers/terminals, man-in-the-middle interception, command fuzzing (trying modified APDUs), and manipulating access rules.

To protect cryptographic execution from side-channel analysis, cards use countermeasures such as random delays, value masking/randomization (removed only at output), and random operation ordering. The OS enforces security by controlling all I/O access to memory and using atomic operations so that power glitches cannot yield partial, exploitable state changes.

Secure Messaging safeguards APDU exchange via per-message MACs or full encryption, often with send sequence counters to prevent replay attacks. At the application layer, many OSs isolate apps by encapsulating their dedicated files (DFs), preventing mutual interference.

Physical level attacks Physical attacks target the hardware of the smart card. Generally, the goal is to extract secrets or alter memory values. Such attacks require physical access and specialized tools. Examples include FPGAs [52], microscopes, lasers, and chemical etching. Microprobing [81] is one such attack that involves mapping a smart card's internal buses under a microscope to read data. Such attacks are rarer than social or logical ones due to the cost and expertise required.

Defenses start at the card body. Holograms and watermarks deter counterfeiting. Metallization shields block probing and are integrity-monitored. On-chip sensors monitor voltage, frequency, and temperature. Memory and bus scrambling hide addresses and signals. Memory and storage encryption protect data stored on the card.

2.5.5 Java Card Platform

Smart cards are used in a variety of applications, including access control, banking, and mobile communications. This is mainly due to the fact that the hardware and software present on them are sophisticated enough to address the requirements of all these different use cases, along with the security features they possess. Smart Card technologies available on the market have different goals, allowing system designers

to achieve varying levels of complexity and security. Despite the variety of smart card technologies, the Java Card Platform is a prominent platform for multi-application secure elements, supporting public-key cryptography and standardized application models.

The Java Card Platform enables programs written in the Java programming language to be run on smart cards and other resource-constrained devices. Due to the fact that these devices have significantly less processing power and resources than PC hardware, a more limited subset of Java has been carefully defined for this purpose. This subset is properly specified in the Java Card Virtual Machine Specification [47]. The Java Card Platform contemplates a runtime environment (Java Card Runtime Environment (JCRE)) which comprises the Java Cryptography Extension (JCVM), the Java Card Application Programming Interface (JCAPI) classes, and support services [46]. This runtime environment sits on top of the smart card's operating system and hardware platform. The general architecture of the Java Card Platform is depicted in Figure 2.26.

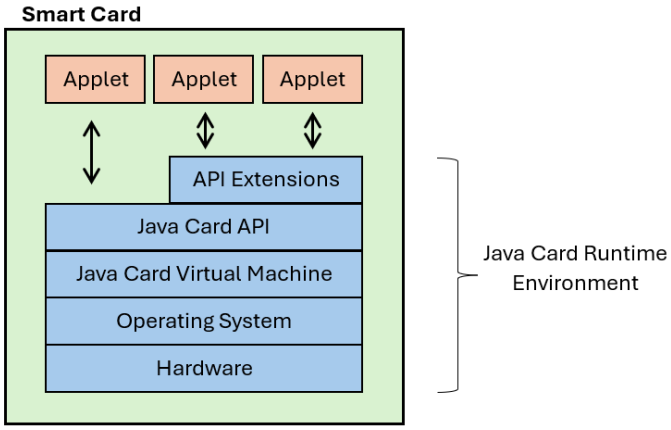


Figure 2.26: *Java Card Platform general architecture*

The Java Card Runtime Environment defines the application model and lifecycle, as well as the memory model. In addition, it enforces a security model based on the isolation of applets. Applets are the application software written for the Java Card Platform. The JCAPI defines a small subset of the traditional Java programming language API. It does not support Strings, multiple threads, wrapper classes like Boolean and Integer, or any Class or System classes. However, it has its own set of core classes specifically designed to support Java Card applications, defining important concepts such as the PIN and the APDU. Additionally, there are other packages that provide classes, interfaces, and functionality to support random number generation, message digest, symmetric and asymmetric cryptography, and biometry. The Java Card language maintains its object-oriented nature. Therefore, applet source code consists of one or more Java classes. The ?? is responsible for loading and executing code as well as controlling applet access to resources. Additionally, it is a more compact version of the traditional Java Virtual Machine (JVM). The ?? is based on a split architecture with an off-card component, the converter, and an on-card component, the interpreter. Java Card class files are compiled

off-card, using development tools and environments, by a Java compiler into one or more class files, which make up a Java package. When the applet is ready to be loaded onto the smart card, these class files are converted into Converted Applet (CAP) files using a Java Card Converter. CAP files are then loaded to the smart card through a card reader peripheral, the Card Acceptance Device (CAD), using an installation tool. The Interpreter is responsible for executing the CAP file in the smart cart environment.

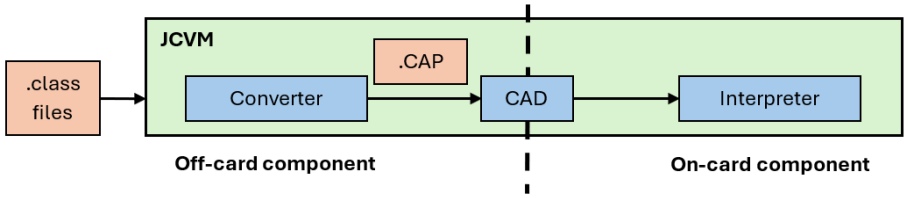


Figure 2.27: *Java Card Virtual Machine code loading process.*

The life of the applet starts after the installation process is completed and the applet becomes registered in the system registry. On the other hand, an applet's life ends when it is removed from the system registry. Java Card applets exchange data with off-card applications through the exchange of APDUs as defined in ISO/IEC 7816-4. The smart card is inserted into the CAD, initiating a session with a host for data exchange. The off-card application sends a SELECT FILE command to a given applet AID, making it the currently selected applet. Consequently, all APDU commands sent by the off-card application are forwarded by the JCRE to the currently selected applet for processing.

2.5.6 GlobalPlatform Card Specification

GlobalPlatform is an international organization that defines standards for secure elements, including smart cards, embedded secure elements, and application processors. Its Card Specification [29] provides a vendor-neutral hardware-agnostic architecture that governs the full lifecycle of applets on smart cards. This lifecycle includes loading, installation, personalization, updating, and eventual deletion of applets. By standardizing these processes, GlobalPlatform enables multiple stakeholders—such as card issuers, application providers, and controlling authorities—to securely interact with the same card infrastructure.

A central component of the GlobalPlatform architecture is the Issuer Security Domain (ISD), a mandatory applet embedded during card manufacturing and commonly referred to as the Card Manager. The ISD acts as the root of trust for card management, enforcing security policies, and controlling operations such as applet loading and deletion. Access to the ISD is protected by establishing a Secure Channel Protocol (SCP) session between the card and an external management system. SCP sessions rely on mutual authentication and encryption to ensure the confidentiality and integrity of the exchanged messages. Several protocol variants exist, including SCP03, which is based on the AES

algorithm. Through these channels, applets, configuration data, and sensitive material, such as cryptographic keys or PIN values, can be securely provisioned to the card.

This architecture prevents direct host access to applet memory, thereby protecting the confidentiality and integrity of on-card data. By combining interoperability, security, and flexibility, GlobalPlatform has become the most widely adopted framework for secure chip management, supported by a large ecosystem and broad industry adoption.

2.6 Summary

This chapter introduced essential cryptographic primitives, authentication mechanisms, and provided an overview of access control systems and commonly used credentials. It also examined passive RFID tokens and smart cards, describing their working principles, data exchange procedures, and communication protocols. Furthermore, the chapter discussed the security aspects of smart cards, such as their tamper-resistant architecture, secure storage, and controlled execution environment. It also outlined the structure of APDU commands and responses, alongside command chaining and data combination mechanisms used during communication. Finally, the chapter reviewed the principal standards and platforms relevant to smart card technology, covering the ISO/IEC 7816 and ISO/IEC 14443 standards, the GlobalPlatform specifications, and the Java Card environment.

3 State of the Art

This chapter provides a comprehensive review of standards-based access control systems, emphasizing smart card–based architectures designed for critical infrastructure. The U.S. Federal Personal Identity Verification (PIV) program is presented as a mature, widely implemented reference model, with an analysis of its system architecture, credential lifecycle, card interfaces, data model, and authentication mechanisms. Building on this foundation, the chapter introduces Commercial Identity Verification (CIV), which adapts the PIV technical framework for enterprise environments with more flexible governance and policy requirements. The discussion then expands to proprietary commercial access control solutions, highlighting their typical design decisions and inherent limitations in open standards support, PKI credential support, and credential lifecycle management. Finally, related academic work is reviewed to identify gaps between current research and practical, standards-compliant credential management systems. These findings emphasize the necessity for a solution that integrates interoperable smart card credentials with robust support for secure issuance, lifecycle management, and deployment in critical infrastructure.

3.1 Personal Identity Verification

The Personal Identity Verification (PIV) program is the U.S. Federal Government's standardized system for issuing and managing secure identity credentials. It provides a uniform smart card–based credential used across agencies, supported by well defined data structures, interfaces, and assurance processes. Given its scale and operational maturity, PIV constitutes a relevant reference model when examining how strong and interoperable credential systems can be designed. Its formal definition and technical requirements are established in a set of standards, beginning with Federal Information Processing Standards (FIPS) 201-3.

FIPS 201-3 establishes the framework for the PIV system, ensuring compliance with

Homeland Security Presidential Directive-12 [35] control and security objectives [73]. It defines a standardized, government-wide identity credentials, including smart cards and derived credentials in various other form factors [21], issued by the U.S. Federal Government to employees and contractors. These secure credentials enable authentication for access to federally controlled facilities, information systems, and applications.

The standard specifies the policies, minimum requirements, and processes for binding identities to authenticators, covering initial identity proofing, credential issuance and revocation, lifecycle management, interoperability, and accreditation of issuing organizations. It also describes the physical characteristics, storage media, and data elements of PIV Cards, as well as the card interfaces and architecture for storing and retrieving identity credentials from these. FIPS 201-3 also references companion NIST publications that define the technical details of PIV systems. Of particular relevance are Special Publication (SP) 800-73-5 [44], which specifies the PIV data model, card interface, and client API, and SP 800-78-5 [24], which defines acceptable cryptographic algorithms and key sizes. NIST SP 800-76 [30] provides PIV Card biometric technical guidance.

3.1.1 System Architecture

An operational PIV system is organized into three cooperating subsystems: the PIV front-end, the PIV issuance and management subsystem, and the PIV relying subsystem. Figure 3.1 illustrates these subsystems.

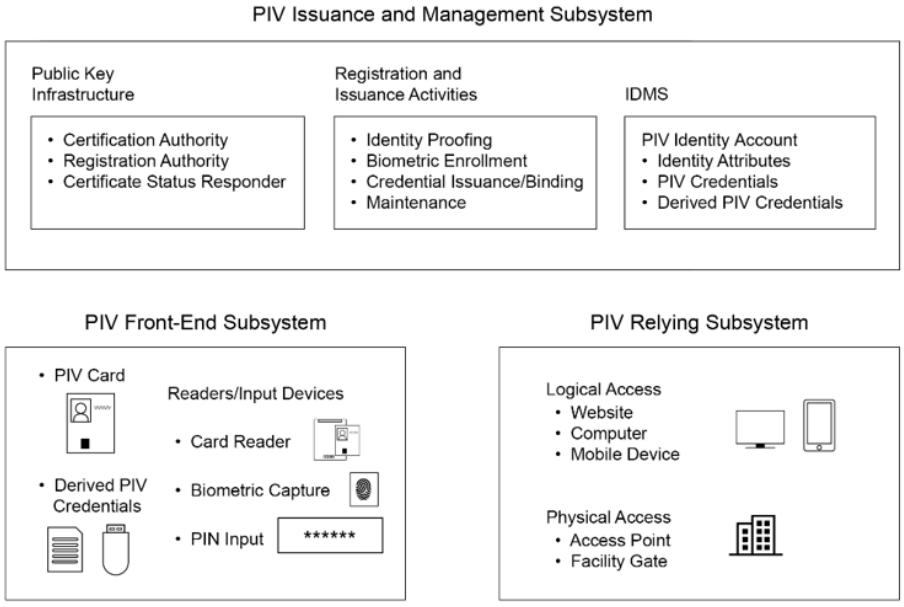


Figure 3.1: *PIV subsystems.*

The PIV front-end subsystem encompasses the PIV Card. It also includes derived PIV credentials [21], which may take alternative form factors for mobile devices, such as Secure Digital cards, USB tokens, or modern SIM cards. These tokens can also be embedded directly in the device as hardware or software cryptographic modules. Beyond the card

and its derived credentials, the PIV front-end also includes card readers, biometric capture devices, and PIN-entry pads used by cardholders to access federal physical and logical resources. Card readers, PIN-entry pads, and biometric capture devices communicate with a PIV Card to perform an authentication protocol and relay credential data to the access control systems for granting or denying access.

The PIV issuance and management subsystem comprises the components responsible for identity proofing and registration, card and key issuance (printing visual elements and loading applications, certificates, cryptographic keys, the PIN, and biometric templates), management, and the repositories and services that support verification, such as PKI directories and certificate-status servers. Complementing the PKI, an IDMS serves as the authoritative source of digital identities, maintaining the attributes that bind each card (and any derived credentials) to a PIV identity account and synchronizing those records with connected systems.

The PIV relying subsystem comprises the physical and logical access control systems, the protected resources, and the authorization data they consult. It includes the components responsible for determining a particular PIV cardholder's access to a resource, whether physical (e.g., a building, server room) or logical (e.g., a workstation, file, or database record). The subsystem relies on authorization mechanisms that define the privileges assigned to entities requesting access to a given resource. When a PIV Card or a derived PIV credential is presented, the relying subsystem identifies and authenticates cardholders by interacting directly with the PIV Card. Once authenticated, authorization mechanisms grant or deny access based on the cardholder's assigned privileges. Figure 3.2 depicts the various PIV system connections.

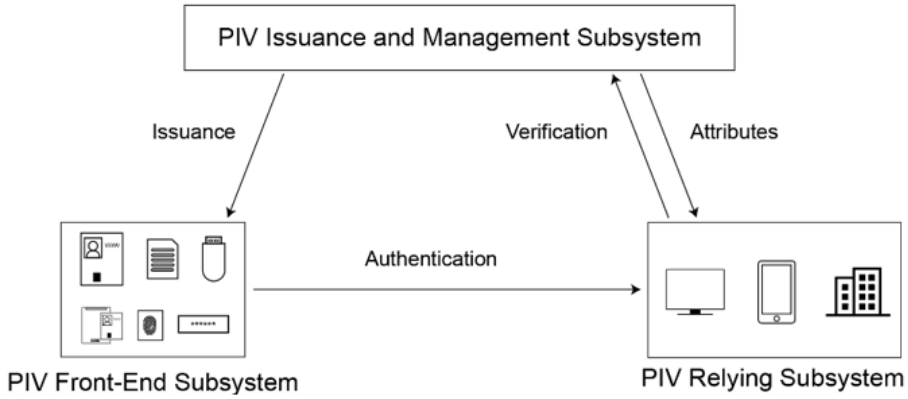


Figure 3.2: *PIV system connections.*

Because PIV credentials are used across agencies, the framework also defines interoperability mechanisms that enable authentication across facilities operated by different federal agencies. For facility access engineering, assurance mappings, and reader usage, see NIST SP 800-116 [22]. For practical interoperability guidance on card media, reader specifications, and PACS databases, see Technical Implementation Guidance: Smart Card

Enabled Physical Access Control Systems [87].

3.1.2 PIV Card Life Cycle Model

FIPS 201–3 organizes the operational life of a PIV Card into seven activities, as shown in Figure 3.3. It begins with PIV Card Request, which initiates and validates the issuance request. Identity Proofing and Registration follows, verifying the applicant's identity, validating source documents, capturing biometrics, and creating the enrolment record. Next, PIV Card Issuance personalizes the card both physically (printed visuals) and logically (applications, PINs, and biometrics), while PKI Credential Issuance generates keys and loads certificates onto the card. During PIV Card Usage, the credential is used to authenticate the cardholder for access; authorization follows successful authentication. PIV Card Maintenance covers updates to the physical card and some of its data, such as PINs, PKI credentials, applications, or biometric data. Finally, PIV Card Termination ensures that the card and the cryptographic material required for authentication are irreversibly destroyed or invalidated to prevent any future use.

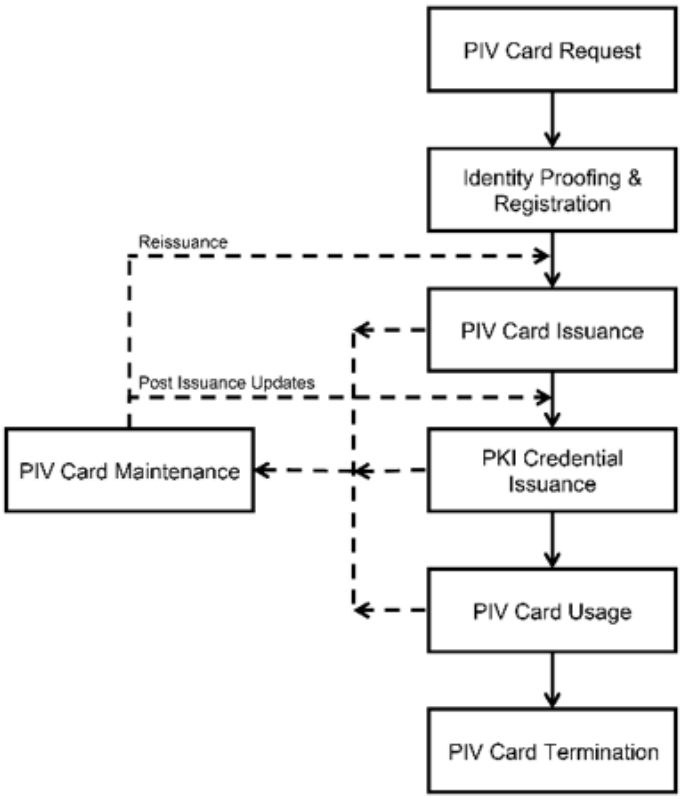


Figure 3.3: *PIV card lifecycle activities.*

The lifecycle activities of derived PIV credentials are described in [21], as these credentials follow issuance, maintenance, and termination processes that differ slightly from those of the physical card.

3.1.3 PIV Card Interfaces

PIV cards support both ISO 7816 contact and ISO 14443 contactless interfaces, each governed by different access rules. The Virtual Contact Interface (VCI) is a secure, contactless channel that emulates the contact interface using secure messaging. This enables commands defined for the contact interface, such as PIN-based activation and certain private-key operations, to be safely executed over contactless, while maintaining the same activation and policy controls. Without Virtual Contact Interface (VCI), such operations are not permitted on the raw contactless interface.

3.1.4 PIV Card Data Model

To support multiple authentication mechanisms, the standard defines the following credentials stored on the PIV Card [44, 73]:

- a PIN and its PIN Unblocking Key (PUK);
- PIV authentication data (one asymmetric private key and the corresponding X.509 certificate);
- card authentication data (one asymmetric private key and the corresponding X.509 certificate);
- two fingerprint templates;
- an electronic facial image.

The standard also allows for additional optional data credentials, such as a digital signature key, a key management key, and their corresponding certificates, depending on specific use cases or federal agency requirements. Of particular importance to card management is the symmetric PIV Card Application Administration Key, which is associated with the card management system and enables secure personalization and post-issuance administrative operations on the smart card.

Beyond the authenticators listed above, the card also carries other mandatory data objects required for discovery, interoperability, and integrity assurance, most notably the Cardholder Unique Identifier (CHUID), the Card Capability Container (CCC), and the Security Object.

The card supports activation with an **application PIN**. The PIN is 6–8 ASCII digits and is right-padded to eight bytes before being sent to the card. After a successful PIN verification, privileged operations using the card's PIV credentials are enabled. If the retry counter reaches zero, the card blocks PIN-gated operations until it is administratively recovered with a PIN Unblocking Key (PUK). For contactless use of PIN-gated keys,

activation must occur over the virtual contact interface with secure messaging. Otherwise, the card rejects the request.

Both the PIV Authentication data (AUTH) and the Card Authentication Key (CAK) data consist of an asymmetric key pair and an associated X.509 certificate.

For **PIV Authentication**, the private key is generated on the card and is never exportable. Private-key operations are available over the contact interface and may also be exposed via the Virtual Contact Interface (VCI), but are not permitted over the contactless interface. Once the card is activated, subsequent operations may proceed without re-entering the PIN. The corresponding X.509 certificate is stored on the card for public key validation.

For **Card Authentication**, the private key may be generated on the card or imported, and it is likewise non-exportable. Operations using the CAK are available over both the contact and contactless interfaces and are not available via the VCI. Unlike AUTH, CAK use does not require card activation. The corresponding X.509 certificate for public key validation is also stored on the card.

Section 5 of the Standard [73] specifies the certificate format and the key management infrastructure for both authentication data and key management.

The **PIV Card Application Administration Key** is an optional issuer-provisioned symmetric key that is unique to each card and used solely for administrative control. The CMSs use this key to activate the card for personalization and post-issuance updates, employing a challenge–response protocol. The **CHUID** is a data object that includes, as mandatory fields, the Federal Agency Smart Credential Number (FASC-N), the Global Unique Identification number (GUID), which is the card's Universally Unique Identifier (UUID) per Request for Comments (RFC) 4122, the card expiration date, and the issuer's digital signature. Optionally, the CHUID may include a Cardholder UUID (also per RFC 4122). The CHUID is used by PACS to locate and bind the presented card to an enrolled record. It is not an authenticator and shall not be used by itself to grant access. For formal definitions of these objects and additional details, see [25][87]. Table 3.1 illustrates the data elements defined by the standard, along with their maximum sizes.

Data Element	Type	Max. Bytes
FASC-N	Fixed	25
GUID	Fixed	16
Expiration Date	Date (YYYYMMDD)	8
Cardholder UUID (Optional)	Fixed	16
Issuer Asymmetric Signature	Variable	2816^{16}

Table 3.1: CHUID data elements and their maximum sizes.

The **Card Capability Container** (CCC) is a mandatory data object described by the Government Smart Card Interoperability Specification [79] that a PIV card exposes to

middleware so that it can discover the card's capabilities, supported applications, and command mappings, enabling interoperable communication without card-specific drivers.

The **Security Object** is a signed integrity record for selected data on the card. It enumerates which data items are covered and includes a cryptographic hash for each. The issuer digitally signs the Security Object, allowing relying systems to detect any alteration to the card's data objects. Its structure follows the ICAO e-passport model [17], enabling future interoperability with identity-document infrastructures.

The standard also defines associated access rules, which govern how applications may request and use the data objects stored on the card. These rules are discussed in Section 4.3.

3.1.5 Authentication Mechanisms

FIPS 201-3 defines several authentication mechanisms that a relying system can perform on a card [73]. Which mechanism is applied depends on the usage environment and the assurance level required. At a high level, PIV mechanisms fall into three categories:

- **Cardholder-To-Card (CTC)** authentication refers to credential elements used to prove the identity of the cardholder to the card, a process also known as card activation, to allow privileged operations using PIV credentials held by the card [73].
- **Cardholder-To-External (CTE)** refers to credential elements used by the card to prove the identity of the cardholder to an external entity, such as a host computer system [73].
- **Card-Management-To-Card (CMTC)** authentication refers to credential elements used by the card management system to authenticate to the card. The primary example is the PIV Card application administration key, which enables secure card management operations [73].

The PIV Card contains two mandatory asymmetric authentication private keys and corresponding certificates to support Cardholder to External System (CTE) authentication.

PKI-AUTH (PIV Authentication certificate). The relying system reads the PIV Authentication certificate and validates it in accordance with RFC 5280 [10]. The cardholder then activates the card (Cardholder to Card (CTC)) by either entering a PIN or completing an On-Card Biometric One-to-One Comparison (OCC). The relying system issues a random challenge and verifies the signature returned by the card's PIV Authentication private key. A unique identifier from the certificate is then used to determine whether the cardholder should be granted access. This mechanism is compatible with both contact readers and contactless readers that support the Virtual

Contact Interface (VCI), providing the "something you have" and "something you know" factors.

PKI-CAK (Card Authentication certificate). The relying system reads and validates the Card Authentication Certificate per RFC 5280 [10]. The relying system issues a challenge and verifies the signature produced by the CAKs private key. A unique identifier from the certificate is then used to determine whether the cardholder should be granted access. Unlike PKI-AUTH, PKI-CAK does not require PIN activation. This mechanism is usable with both contact readers and contactless readers (no VCI required) and provides the "something you have" factor.

Secure Messaging Authentication (SM-AUTH) is an optional CTE authentication mechanism. When this mechanism is adopted, the PIV Card includes a secure messaging key and a corresponding Card Verifiable Certificate (CVC) to establish symmetric keys for use with secure messaging. The same key, CVC, and key establishment protocol can also be used for authentication, since the PIV Card is authenticated as part of establishing the secure messaging channel. Details of the SM-AUTH authentication mechanism are specified in [44] and [24].

FIPS 201–3 also defines biometric authentication using either off-card one-to-one comparison (BIO/BIO-A) or on-card comparison (OCC-AUTH). BIO/BIO-A performs CTE authentication while OCC-AUTH is used as a means to activate the card (CTC).

BIO / BIO-A (off-card biometric comparison). The relying system first reads the CHUID (or, alternatively, the PIV Authentication or Card Authentication certificate) and verifies its digital signature to confirm the card originates from a trusted source and has not expired. The cardholder then enters a PIN to activate the card (CTC). Next, the relying system reads the signed biometric record(s) from the card and verifies their signature(s), prompts the cardholder to capture a live sample, and performs a one-to-one comparison off-card (CTE). The FASC-N or card UUID from the CHUID (or, alternatively, the PIV Authentication or Card Authentication certificate) is then checked against the corresponding identifier in the signed attributes of the biometric record to ensure consistency, and that unique identifier is used as input to the authorization decision. BIO-A is the higher-assurance variant of BIO, performed under the supervision of an operator such as a security guard. Both BIO and BIO-A are usable with contact readers and with contactless readers that support VCI. BIO provides the "something you are" factor, while BIO-A provides "something you are" and, through attended verification of card possession, also covers "something you have."

OCC-AUTH (on-card biometric comparison). The card hosts an OCC algorithm and stores OCC data that cannot be exported. Initially, a fingerprint biometric template is supplied to the card to perform CTC authentication. The card responds with a positive or negative biometric verification decision. No PIN is required, and biometric retry

blocking is mandated. OCC-AUTH requires that Secure Messaging Authentication (SM-AUTH) be successfully performed beforehand, so both the fingerprint request and the response are protected with MACs, ensuring integrity and binding to the authenticated card. As a result, this mechanism provides two factors: something you have (the card) and something you are (the fingerprint). Finally, for Card Management System to Card (CMTC) authentication, the standard defines a simple challenge-response protocol that activates the card for personalization or update. This mechanism is based on the Card Application Administration key, a symmetric cryptographic key unique to each PIV card. This protocol is discussed in detail in Section 4.6.

Physical Access Control Considerations In a physical access control environment, each PIV authentication mechanism provides one or two factors of authentication [61]. The mechanisms can be combined to achieve up to three authentication factors. Table 3.2 and Table 3.3 summarize PIV authentication mechanisms and their authentication factors when used on the contact and contactless interfaces, respectively, for these environments.

PIV Authentication Mechanism	Have	Know	Are	Authentication Factors
BIO			x	1
BIO-A			x	1
PKI-CAK	x			1
PKI-AUTH (with PIN activation)	x	x		2
PKI-AUTH (with OCC activation)	x		x	2
OCC-AUTH	x		x	2
PKI-CAK + BIO(-A)	x	x	x	3

Table 3.2: PIV Authentication Mechanisms on the Contact Interface.

PIV Authentication Mechanism	Have	Know	Are	Authentication Factors
PKI-CAK	x			1
OCC-AUTH	x	x		2

Table 3.3: PIV Authentication Mechanisms on the Contactless Interface.

Logical Access Control Considerations Logical access control is divided into two scenarios: Remote/Network Access and Local Workstation Access. This is due to the fact that the risks and protections differ depending on whether access occurs across a network or directly at a workstation. Each scenario relies on its own set of authentication mechanisms and corresponding assurance measures.

When a user needs to authenticate to systems or applications across a network, the assurance requirements are higher due to the risk of eavesdropping, credential replay, or impersonation. For this use case, FIPS 201-3 aligns with NIST SP 800-63's Authenticator Assurance Levels (AALs), which represent the degree of confidence that an authenticator, or combination of authenticators, can be relied upon to correctly verify a claimant's

identity during authentication. Table 3.4 lists the authentication mechanisms defined in FIPS 201-3 that support access control for remote/networked access.

PIV Authentication Mechanism	Supported Authenticator Assurance Level
PKI-CAK	AAL1 (some confidence)
PKI-AUTH	AAL3 (very high confidence)

Table 3.4: PIV Authentication Mechanisms for Remote/Network Access.

When the user authenticates to a workstation or resource that is physically co-located with them, the risk model differs, and the session does not fall under the SP 800-63 AAL framework. Instead, FIPS 201-3 provides its own graduated assurance descriptions for local authentication. The assurance provided by each PIV authentication mechanism for local use cases is summarized in Table 3.5.

PIV Authentication Mechanism	Assurance Provided
PKI-CAK	Some confidence in the asserted identity
BIO	Medium confidence in the asserted identity
BIO-A	High confidence in the asserted identity
OCC-AUTH	High confidence in the asserted identity
PKI-AUTH	High confidence in the asserted identity

Table 3.5: PIV Authentication Mechanisms for Local Workstation Access.

3.2 Commercial Identity Verification

The PIV program has demonstrated that a single, standards-based credential can be scaled across multiple organizations and systems. Agencies align on common goals, issuance policies, and technical specifications, allowing vendors to build to a single baseline of hardware and software products. Cross-certification through the Federal PKI Bridge, the federal CA that links participating PKIs in the PIV ecosystem, enables inter-agency trust. The scale of adoption of the PIV program is significant. U.S. government agencies have issued well over five million PIV cards to federal employees and contractors over the years [72].

The Commercial Identity Verification (CIV) [89], developed by the Smart Card Alliance Physical Access Council, provides the same technical stack to enterprises without the burden of federal policy. It is built on the PIV technical specifications and the data model defined in NIST SP 800-73. It is intended for commercial deployments and works with the growing ecosystem of readers, middleware, and other products developed for PIV [31]. However, a CIV issuer defines its own identity proofing and registration policies, as well as card issuance policies.

Compared with proprietary, vendor-specific access control and identity management solutions, CIV's main advantage is convergence on open, widely deployed standards. A

single credential serves both physical access control and logical authentication, covering facilities, endpoint, and network logon, remote access, and application single sign-on. Centralized identity and credential management enables unified monitoring, auditing, and automated deprovisioning when an employee departs or a credential expires. Because CIV aligns with PIV, organizations can procure interoperable components from multiple vendors and benefit from standards that evolve with backward compatibility, which reduces vendor lock-in and long-term lifecycle costs. Figure 3.4 illustrates a multi-site enterprise network in which a central CIV credentialing infrastructure supports local access control and policy enforcement.

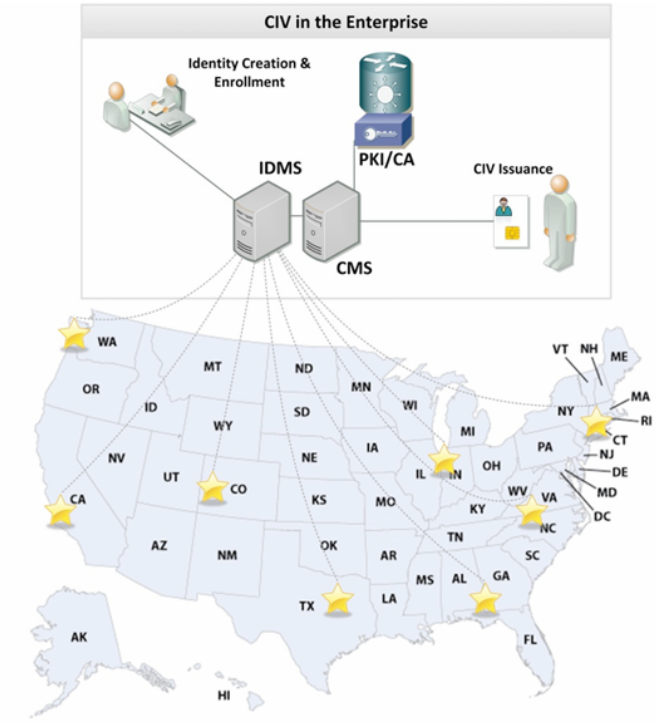


Figure 3.4: *Multi-site enterprise network with CIV credentialing infrastructure supporting local access control.*

The key differences between PIV and CIV lie in governance and identifiers rather than in on-card technology. Both use the SP 800-73 card application and data model; however, PIV follows FIPS 201-3 processes and relies on the Federal PKI Bridge for cross-organizational trust, whereas CIV follows issuer-defined policies and is trusted within the enterprise or a chosen federation. A CIV credential will never be accepted as "trusted" by the U.S. Federal Government. As the credential identifier, PIV uses the FASC-N, while CIV replaces it with an RFC 4122 UUID [51]. In CIV, the only mandatory certificate on the card is the Card Authentication certificate; all other certificates are optional.

Biometrics in CIV are optional, and enterprises choose whether to include them according to their requirements and use cases. When implemented, the supported interoperable modality is fingerprint, stored on-card as two standardized fingerprint templates defined in the SP 800-73 data model. These templates are securely stored on the card and can

only be accessed through the contact interface after successful PIN verification. If an organization requires contactless matching without PIN, the biometric must be stored off-card and referenced by the card's UUID. In this case, other biometric modalities can be adopted, although they may not guarantee interoperability. Biometrics are not a replacement for the card's cryptographic authenticators. Instead, they act as an additional "who you are"factor that complements certificate-based mechanisms such as Card Authentication and private key-based user authentication.

See Table 3.6 for a concise comparison of PIV and CIV credentials.

	PIV	**CIV**
Identity Proofing and Registration	Follows FIPS 201-3	Follows the corporation's policies
Credential Issuance	Follows FIPS 201-3	Follows the corporation's policies
Card data model	Follows SP 800-73	Follows SP 800-73
Credential identifier	FASC-N	UUID
PIV authentication data	Mandatory	Optional. Follows the corporation's requirements
Biometrics (Two Fingerprint templates)	Mandatory	Optional. Follows the corporation's requirements

Table 3.6: Comparison of PIV and CIV credentials.

3.3 Commercial Solutions

Commercial access control and credential-management platforms are typically delivered as proprietary, closed-source solutions that integrate badge lifecycle administration but not full cryptographic credential management. They provide robust capabilities for credential issuance, device provisioning, and system administration, but they generally do not include native support for high-assurance, PKI-based credentials, such as those defined by FIPS 201—3 unless purchased as optional modules or add-ons. In many cases, these systems do not natively support smart cards and instead rely on legacy contactless RFID badges. Some vendors offer higher-end proprietary credential technologies, such as HID Seos [34] and MIFARE DESFire EV3 [56], which use symmetric cryptographic keys and improve on legacy proximity formats, yet still operate outside an open, or standards-based smart-card model and do not implement asymmetric private key operations or the PIV/CIV application model.

Commercial systems typically operate in one of two modes. In the first, the system and the card share a symmetric secret and perform mutual authentication using that shared

key. In the second step, the system reads a cardholder identifier stored on the credential and matches it to a corresponding database record.

Representative vendors in this sector include Genetec (Synergis access control system) [85], LenelS2 (OnGuard) [64], and HID Global. Genetec and LenelS2 integrate proprietary CMSs components coupled with their access control and video platforms. These vendors do offer PKI-based credentials through PIV, but they are delivered via add-on enablement components –most commonly HID Global's pivCLASS hardware [32] and software [33] – rather than as native, out-of-the-box support. pivCLASS software is proprietary and licensed. Finally, these offerings are primarily oriented toward PACS deployments and, in general, do not incorporate standards-based support for logical access (e.g., workstation logon, VPN authentication, or federated identity services).

3.4 Related Work

Academic proposals on smart card–based access control typically emphasize architectural design, system integration, and operational workflows. By contrast, far fewer studies discuss credential data models and their lifecycle management. The three works reviewed [18, 49, 57] highlight this imbalance and motivate a solution that explicitly addresses these topics, with particular attention to the security requirements of critical-infrastructure environments.

Mohandes [57] proposes a two-part platform – CMSs and Access Management System (AMS) – built on contactless MIFARE cards, a legacy technology with well-documented cryptographic weaknesses [11] [50] [27]. The CMSs supports card and cardholder management, as well as card-request processing. Although the implementation includes a PKI component, its use is restricted to securing reader–server communications rather than on-card asymmetric operations. Their paper details the CMSs database E-R model and workflows, but does not define the card's data model or its supported authentication mechanisms.

Du and Tang [18] present a web-based campus PACS with modules for cardholder/card management, field-device management and monitoring, and other administrative features. Their work is integration-focused, as it demonstrates how access control operations and data synchronization can be orchestrated at the system level. However, it does not specify credential data models and lifecycle management.

Kapusta and Lindström [49] analyse and evaluate authentication standards and protocols for high-security PACS such as FIPS 201-3 (PIV) and CIV. They also develop a prototype in which an Android smartphone acts as a CIV-derived credential, and they implement and test an asymmetric-key authentication mechanism using that credential. The result is a useful authentication prototype. However, by design, it does not address credential

lifecycle management, particularly credential issuance.

3.5 Summary

FIPS 201-3 defines processes for binding verified identities to authenticators, such as the PIV Card and derived PIV credentials used in a PIV system. These credentials support mechanisms that authenticate individuals who require access to U.S. government-controlled facilities, information systems, and applications. PIV relies on well-established Public Key Infrastructure (PKI) and NIST technical specifications for strong authentication, making it suitable for environments requiring high security. FIPS 201-3 also references companion NIST publications, such as SP 800-73-5, which contain the technical specifications for interfacing with the PIV Card to retrieve and utilize the identity credentials.

PIV was designed for U.S. federal physical and logical access control across agencies. However, many enterprises – often operators of critical infrastructures – lack a comparable, standards-based approach outside the federal policy framework. Most deployments of physical access control systems are based on proprietary commercial PACS solutions that frequently default to legacy RFID badges. When these commercial solutions are operated in federally controlled facilities, PIV functionality is often delivered via proprietary add-ons rather than as a native, first-class capability.

Commercial Identity Verification (CIV) fills this gap by reusing the PIV technical stack, especially the SP 800-73 card application and data model, while allowing organisations to define their own identity-proofing, registration, and issuance policies. This makes PIV-like credentials practical in enterprise settings.

In the research community, most studies tend to emphasise PACS architecture, device integration, and operations, with comparatively little attention to the credential data model or lifecycle management. Representative examples include Mohandes [57], which delivers a MIFARE-based CMS with rich operational features but no PIV/CIV-aligned on-card model or on-card asymmetric operation supporting strong authentication mechanisms; Du and Tang [18], who focus on administration, device monitoring, and data exchange rather than credential data modelling or lifecycle; and Kapusta and Lindström [49], who prototype a CIV-derived smartphone credential and validate an asymmetric authentication flow, but do not address issuer-side flows.

This analysis emphasizes a clear need for an access control architecture that extends beyond access control integration and addresses both the full lifecycle and the data model of credentials within a standards-compliant framework. Such an architecture must natively support smart card–based credentials aligned with the PIV/CIV data model rather than relying on legacy badge technologies or proprietary extensions. Furthermore,

the architecture must treat the credential data model and its lifecycle as first-class design concerns, ensuring interoperability, auditability, and long-term maintainability. Addressing these requirements is essential to enabling critical infrastructure operators to deploy high-assurance identity credentials with the same level of rigor, security, and scalability demonstrated by the PIV program.

4 Proposed Solution

Building on the gaps discussed in Section 3.5, this chapter presents a complete access control architecture for critical infrastructures that is technically compatible with PIV and policy-aligned with CIV, delegating enrolment and identity proofing to enterprises. This ensures interoperability with existing standards while providing the flexibility required by operators of critical infrastructure.

Rather than proposing only a CMS, this work presents a cohesive system that spans: (i) the enterprise access control system and its components; (ii) the smart-card application (card applet) implementing the SP 800-73 data model and authentication mechanisms; and (iii) the CMS that issues, maintains, and terminates credentials, supporting both physical and logical access. The result is a system that accommodates the diverse realities of enterprises, allowing each organization to define its own identity proofing, registration, and issuance policies, while preserving credential interoperability across vendors and sites and ensuring strong, standards-based authentication.

The remainder of this chapter is organized as follows. Section 4.1 presents the functional and non-functional requirements that guided the design of the proposed solution. Section 4.2 provides a system overview, introducing the main components and explaining their interactions. Section 4.3 defines the smart card credential profile, the data model adopted, and the adaptations required in a CIV setting. Section 4.4 presents the credential lifecycle model from enrollment through termination. Section 4.5 details the CMSs, its responsibilities across lifecycle activities, and its external interfaces with the IDMS, PKI, and PACS/LACS. Section 4.6 describes the authentication mechanisms considered in this work and outlines the message flows.

4.1 Requirements

The proposed access control architecture is guided by a set of functional and non-functional requirements derived from the analysis of existing standards, commercial

solutions, and related academic work. These requirements capture the technical, security, and operational constraints that shape the design of the proposed solution. They are summarized as follows:

1. **Standards-aligned foundation:** The solution shall be compatible with broadly accepted industry practices and standards so that products from multiple suppliers can interoperate, be tested independently, and be procured without vendor lock-in. Interfaces, data formats, and protocols shall be sufficiently open and well specified to enable multi-vendor deployments and cross-site interoperability.

2. **Enterprise policy flexibility:** Organizations shall retain control over identity proofing, registration, and issuance policies. The architecture shall accommodate different operational realities without sacrificing interoperability.

3. **Integrated architecture:** The solution shall integrate with PKI services (certificate issuance and status), card printers/laminators (visual personalization), and biometric acquisition devices. It shall also support data exchange with external systems via well-defined APIs.

4. **Smart card as a unified credential:** Smart cards shall be the primary identity credential. Each cardholder shall have one identity account and a smart card that provides multi-site access to physical and logical resources, ensuring credential convergence.

5. **Well-defined smart card lifecycle and data model:** The smart card should adopt a well-defined data model and lifecycle activities covering card request, identity proofing as registration, card issuance, PKI credential issuance, card usage, card maintenance, and card termination. The data model shall specify access rules and conditions for each data element (e.g., PIN-gated use of particular private keys) and define which operations are permitted over the contact and contactless interfaces. Cryptographic algorithms and key sizes shall meet contemporary security requirements.

6. **Secure card-management operations:** Loading, installation, updating, and deletion of applets and card data shall be performed only over authenticated communication channels.

7. **Certificate-based authentication:** The card shall support X.509 certificates for high-assurance authentication and use of digital signatures, via on-card private key operations, to resist credential cloning and tampering. Asymmetric key pairs shall be generated on-card, and all on-card private keys shall be non-exportable.

8. **Optional biometric support:** Where required, the system shall support biometric verification using either off-card comparison or on-card comparison, with protected template storage, retry/lockout policy, and binding of biometric data to the cardholder.

9. **Certificate policy enforcement and status checking:** System components that validate credentials shall enforce certificate content and policy, perform full path validation in accordance with RFC 5280 [10], and check certificate revocation status via Online Certificate Status Protocol (OCSP) or CRLs.

10. **Protection against revoked or compromised credentials:** Physical and logical access control systems shall prevent the use of revoked credentials and employ challenge–response mechanisms using on-card private keys to protect against credential forgery.

11. **Support for multiple authentication factors and zone-based protection:** Authentication mechanisms shall support different factor counts and combinations to meet protection requirements for areas and resources with differing assurance levels in the asserted identities.

12. **Public-key infrastructure (PKI) integration and hosting flexibility:** The system shall integrate with PKI for certificate issuance and status checking, and support internally hosted PKI components where required.

13. **Service continuity under network issues:** PACS and logical components (where applicable) shall continue to operate through network outages using cached certificate status and policy, reconciling events once connectivity is restored.

14. **Auditability, logging, and compliance evidence:** Card management operations, access control decisions, successful and unsuccessful authentication attempts, and revocation/status checks shall be logged. Operators shall be able to audit and monitor all related events centrally to support compliance and investigations.

4.2 System Overview

The proposed solution adopts a CIV-like credential that leverages PIV technical specifications, notably the NIST SP 800-73 [44] card application, data model, and strong, certificate-based, authentication mechanisms, while delegating identity proofing and registration to the enterprise. Its main components are: the CMS with its issuance client application, the Identity Management System (IDMS), the Public Key Infrastructure (PKI), containing certificate issuance and validation services, the smart-card credential, Physical Access Control Systems (PACSs), and Logical Access Control Systems (LACSs). Figure 4.1 illustrates the devised architecture.

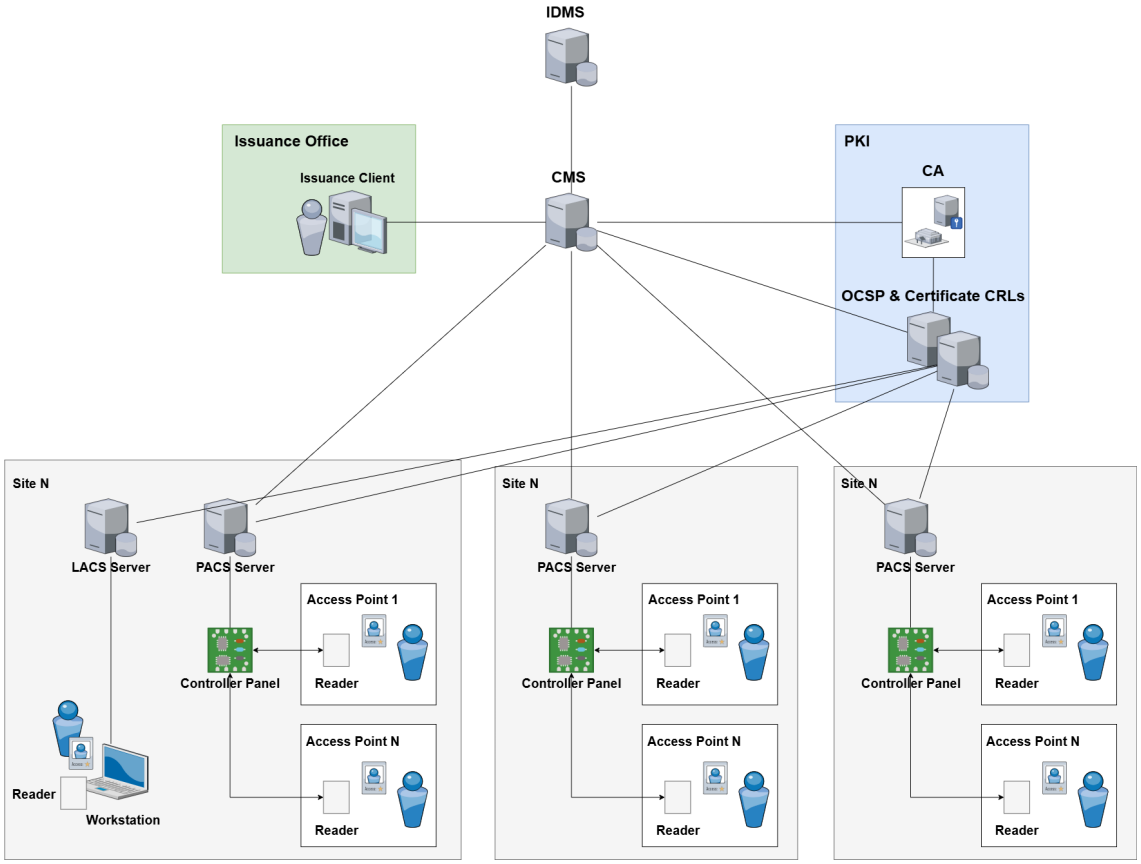


Figure 4.1: *Architecture for the proposed solution.*

The **Identity Management System (IDMS)** is the enterprise's central repository for identities and related enrolment data (e.g., attributes, photos, biometrics, background checks). It is owned and operated by the enterprise, and its internal workflows are outside the scope of this work. The IDMS collects events from Human Resources (HR) (e.g., hire, role changes, termination) and forwards them to the CMS, which, in turn, requests card-associated certificate issuance or revocation from the PKI.

The **Card Management System (CMS)** manages the full lifecycle of the smart card credentials, including processing of enrolment information, card personalization, key management, certificate provisioning, post-issuance updates, and termination. It is composed of a backend software service and a desktop issuance client application. The CMS service exposes an API for the issuance client application and integrates with the enterprise PKI and IDMS. The CMS receives identity data from the IDMS for card issuance and to update cardholder status. The issuance client runs on operator workstations in enrolment/issuance offices. It provides an administrative UI for lifecycle operations, and interfaces with card printers, cameras, and smart cards via a PC/SC-compliant reader (typically USB). The CMS is deployed on-premises within the enterprise environment.

The **Public Key Infrastructure (PKI)** is enterprise-operated and responsible for issuing and validating certificates bound to smart card credentials. A private root CA anchors

trust, and one or more intermediate CAs issue end-entity certificates from Certificate Signing Requests (CSRs) submitted via the CMS. Certificate status is provided through CRLs and OCSP, which are consumed by systems that verify certificates, namely PACS controllers, workstation/logon services (LACS), and the CMS. These systems perform certificate path discovery and validation against configured trust anchors and recheck revocation status at authorization time. While certificate path information may be cached, revocation status must be revalidated at each access attempt when connectivity is available. Revocation is triggered by CMS and IDMS events (e.g., card loss, contract termination) and propagated promptly via updated CRLs or OCSP to prevent the use of invalid credentials.

The system uses a **CIV smart card credential**, implemented as a dual-interface token compliant with ISO/IEC 7810, 7816, and 14443. Logically, it follows the PIV card application and data model as specified in NIST SP 800-73-5 (Parts 1 and 2) [23, 25], where applicable, while omitting U.S. Federal Government–specific elements and adopting enterprise identifiers consistent with CIV. The card securely stores private keys, corresponding public-key certificates, and biometrics, as required by enterprise policy. Cardholder-to-card (CTC) verification is enforced through a PIN with retry counters and lockout mechanisms. For higher-assurance use cases, match-on-card biometrics are available. As a design choice of the proposed solution, only on-card biometric comparison is supported.

CIV is adopted to address the gaps identified in Section 3.5, namely, the absence of a standards-based enterprise credential outside the U.S. Federal Government framework, the prevalence of proprietary or legacy PACS that rely on insecure RFID badges, and limited lifecycle standardization. CIV retains the well-defined data model, authentication mechanisms, and FIPS-conformant technical controls used by PIV, while giving critical-infrastructure operators the flexibility to define their own identity proofing, registration, and issuance policies. The credential supports both contact and contactless authentication for PACS and LACS, utilizing certificate-validated authentication with online status checks and digital signature-based challenge-response protocols. Section 4.3 details the card application structure and data objects.

At each enterprise site, the **Physical Access Control System (PACS)** enforces physical access decisions at controlled access points. Contact and contactless credential readers – optionally with biometric acquisition sensors – communicate with controller panels over secure, bidirectional links. The PACS server maintains Access Control Lists (ACLs) that map each cardholder to areas, time schedules, and privileges. Any commercial PACS that is PIV/CIV-compatible can be used. Cardholder records are provisioned by the CMS. For resilience during network outages, controller panels cache a hotlist and a minimal authorization set, then resynchronize revocations, expirations, and privilege changes when connectivity returns. The authentication mechanisms executed by the PACS depend on

the required assurance levels and are discussed in Section 3.1.5.

The **Logical Access Control Systems (LACSs)** govern access to enterprise IT resources (e.g., workstations, VPNs, applications) using the proposed CIV smart-card credential. Client middleware (via PC/SC) performs PIN-based cardholder-to-card (CTC) authentication to unlock on-card private-key operations used for certificate-based authentication. The LACS server validates the certificate chain against enterprise trust anchors, checks revocation via CRL or OCSP, and maps the certificate to an IDMS record. Authorization is then enforced per policy, and deployments may require multi-factor authentication.

Table 4.1 summarizes the interactions between the architecture components.

Source Component	Target Component	Purpose of Communication
HR System	IDMS	Provides lifecycle events (hire, role change, termination) that update identity records.
IDMS	CMS	Supplies identity data required for card issuance and status updates.
CMS (issuance client)	Smart Card	Personalizes card, generates on-card keys, and loads certificates and data objects.
CMS	PKI	Submits certificate signing requests and triggers revocation.
PKI	CMS	Returns issued certificates for card issuance and maintenance operations.
PKI	PACS/LACS	Provides revocation and validation services (CRLs/OCSP) for access decisions.
CMS	PACS	Provisions cardholder records and access control list updates.
Smart Card	PACS/LACS	Cryptographic authentication (digital signatures, certificate presentation).

Table 4.1: Interactions between the architecture components.

4.3 Smart Card Credential

The proposed solution utilizes a CIV credential profile based on the PIV card application and data objects defined in NIST SP 800-73, with a limited number of enterprise-oriented adaptations. The credential comprises the following data elements and credentials:

- an application PIN and its respective PUK;
- Card Authentication data (one asymmetric private key and the corresponding X.509 certificate);

- PIV-like authentication data (one asymmetric private key and the corresponding X.509 certificate);
- two fingerprint templates (optional);
- an electronic facial image;
- the Card Application Administration Key;
- the Cardholder Unique Identifier (CHUID);
- the Card Capability Container (CCC); and
- the Security Object.

As described in Section 3.1.4, a RFC 4122-compliant UUID is adopted in place of the FASC-N. This is the recommended approach for non-federal issuers, preserving interoperability with PIV-aware readers.

In CIV, biometrics are optional. Yet, when implemented, the interoperable modality is a fingerprint. In such cases, up to two standardized templates are stored on the card and are accessible only via the contact interface after successful PIN activation.

By adopting a CIV credential built on the PIV technical specifications and data model, enterprises can select standardized products and processes that have been widely tested and deployed, and that interoperate with the existing ecosystem of readers and middleware.

Although CIV mandates only the storage of a Card Authentication certificate, the proposed solution includes a PIV-like Authentication certificate so that, with PIN activation, the credential natively supports two factors ("something you have" and "something you know"). If the enterprise decides to use biometrics, the credential is capable of providing the three factors ("have", "know", "are"). Support for all three authentication factors makes the proposed solution well-suited to the critical-infrastructure context.

These data elements shall conform to the read-access rules defined in NIST SP 800-73, Part 1, for both contact and contactless interfaces. Table 4.2 summarizes these rules.

Data object	**Access Rule for Read**	
	Contact	**Contactless**
X.509 Certificate for Card Authentication	Always	Always
X.509 Certificate for PIV-like Authentication	Always	VCI
Cardholder Fingerprint templates	PIN	VCI and PIN
Cardholder Facial Image	PIN	VCI and PIN
Card Capability Container	Always	VCI
Cardholder Unique Identifier	Always	Always
Security Object	Always	VCI

Table 4.2: Access rules for reading PIV data objects.

X.509 Certificates To support two-factor authentication, the devised data model considers not only the CIV-mandated Card Authentication certificate but also a PIV-AUTH–like certificate. PIV certificate profiles contain U.S.-federal-only elements (notably Federal Public Key Infrastructure (FPKI) policy OIDs and the FASC-N in subjectAltName) that must be replaced in an enterprise context. Taking this into account, the extension and attribute settings common to both certificates are as follows (the highlighting shows where they differ from PIV):

- Assert enterprise certificate-policy OIDs instead of the FPKI OIDs;
- Include only an RFC 4122 UUID in the Subject Alternative Name (SAN), encoded as a Uniform Resource Name (URN);
- Key Usage set to digitalSignature only and marked as critical, consistent with PIV;
- Set validity periods that do not exceed the card's lifetime;
- Include standard X.509 extensions (basicConstraints with the CA attribute set to false, Subject Key Identifier (SKI), Authority Key Identifier (AKI), and revocation pointers (Authority Information Access (AIA)/CRL Distribution Point (CRLDP)) as specified in the enterprise Certification Practice Statement (CPS)).

Per-certificate specifics are minimal. The PIV-like Authentication certificate includes Extended Key Usage (EKU) id-kp-clientAuth (1.3.6.1.5.5.7.3.2). Deployments that require Windows interactive logon may also include Microsoft Smartcard Logon (1.3.6.1.4.1.311.20.2.2). The Card Authentication certificate includes the Extended Key Usage (EKU) id-PIV-cardAuth (2.16.840.1.101.3.6.8). This EKU is defined for card authentication and is appropriate for non-federal issuers.

Table 4.3 provides a side-by-side comparison of the PKI-AUTH and PKI-CAK certificate profiles under PIV versus the proposed CIV profile, highlighting common elements and calling out federal-specific items that must be replaced in enterprise deployments.

Attribute	PKI-AUTH (PIV)	PKI-AUTH (CIV)	PKI-CAK (PIV)	PKI-CAK (CIV)
Certificate Policy	Federal	Enterprise policy OID (non-federal)	Federal	Enterprise policy OID (non-federal)
SubjectAltName	FASC-N and UUID (encoded as a URN)	UUID only (encoded as a URN)	FASC-N and UUID (encoded as a URN)	UUID only (encoded as a URN)
Key Usage (critical)	digitalSignature			
Extended Key Usage	Optional MS Smartcard Logon as needed		id-PIV-cardAuth	
Validity	3 years and must not exceed card lifetime.	Enterprise policy. Must not exceed card lifetime	3 years and must not exceed card lifetime.	Enterprise policy. Must not exceed card lifetime
Revocation/ AIA/ CRLDP	Populated. Per federal CPS/profile	Populated. Per enterprise CPS	Populated. Per federal CPS/profile	Populated. Per enterprise CPS

Table 4.3: Differences between PIV and CIV certificate profiles.

Cryptographic Algorithms and Key Sizes Because CIV reuses the PIV card application and aligns with the cryptographic profiles in NIST SP 800-78-5, the same algorithm and key-size principles apply. The permitted options for card keys and digital signatures are detailed below.

The PIV Authentication and Card Authentication key pairs may be RSA 2048 or 3072 or Elliptic Curve Digital Signature Algorithm (ECDSA) Curve P-256 or P-384 through December 31, 2030. Beginning in 2031, new keys shall be RSA 3072 or ECDSA curves P-256 or P-384. For physical access, elliptic-curve keys are preferable because private-key operations are significantly faster than those of RSA, and there is no specified phaseout date for either P-256 or P-384 curves. Accordingly, while the ultimate choice is an enterprise policy decision, we recommend ECDSA P-256 for its performance and long-term compliance. The Card Management System authenticates to the card using the Card Application Administration Key, a symmetric key used in a challenge–response protocol for personalization and post-issuance updates. Acceptable algorithms for this key are 3TDEA (deprecated) and AES-128/192/256 through December 31, 2030. From January 1, 2031, only AES-128/192/256 is permitted. We recommend AES-128/192/256, with the specific size selected according to enterprise risk, performance, and platform constraints.

Table 4.4 summarizes the permitted algorithms and key sizes for these keys, from which the enterprise may select.

PIV Key Type	Algorithms and Key Sizes Through 2030	Algorithm and Key Sizes for 2031 and beyond
PIV Authentication Key	RSA (2048 or 3072 bits) ECDSA (Curve P-256 or P-384)	RSA 3072 bits ECDSA (Curve P-256 or P-384)
Asymmetric Card Authentication Key		
Card Application Administration Keys	AES-128/192/256	

Table 4.4: Permitted algorithms and key sizes for PKI-AUTH, CAK, and Card Application Administration keys.

Approved digital signature algorithms are defined in FIPS 186 [16], and NIST SP 800-78-5 [24] specifies the permitted combinations for digitally signed PIV information. Importantly, RSA 2048 is permitted only through December 31, 2030, and is not applicable to new signatures starting January 1, 2031. We therefore recommend selecting from the remaining options, with the specific choice determined by the enterprise policy, performance needs, and platform support. Table 4.5 summarizes the recommended algorithms, key sizes, hash functions, and padding schemes for digital signatures.

Public Key Algorithms and Key Sizes	Hash Algorithms	Padding Scheme
RSA (3072 or 4096)	SHA-256 or SHA-384	PKCS #1 v1.5
	SHA-256 or SHA-384	PSS
ECDSA (Curve P-256)	SHA-256	N/A
ECDSA (Curve P-384)	SHA-384	N/A

Table 4.5: Recommended digital signature algorithm and key size requirements.

Card Activation The PIV Card Application requires cardholder activation for privileged operations. By default, activation is accomplished by verifying a numeric PIN of 6–8 digits. Activation is blocked after excessive failures, with a maximum of 10 consecutive attempts permitted per activation method. If the enterprise enables biometrics on the card, OCC may also be used to activate the card. A successful OCC sets the same security state as a successful PIN verification and has its own retry counter within the same 10-attempt limit. PIN and OCC-gated private-key operations are available over the contact interface and may be exposed over the Virtual Contact Interface (VCI), but not over the raw contactless interface. Within these standard bounds, the enterprise defines its activation policy: it may set a longer minimum PIN, choose whether to enable OCC, adopt a retry cap below 10 for either method, specify change and reset procedures (for example, in person, kiosk, remote, or supervised remote), and define VCI requirements, provided these choices remain within the standard's limits to preserve interoperability.

The card may also be activated by the CMS to support personalization and post-issuance updates. This activation takes place by performing a challenge-response-based authentication protocol using the card's unique Card Application Administration Key. This

authentication protocol is discussed in detail in Section 4.6.

4.4 Credential Lifecycle

The proposed solution adopts a lifecycle model analogous to that used for PIV Cards. Accordingly, the 'PIV' prefix is omitted, and the following activities are considered: Enrolment (card request and identity proofing & registration), Card Issuance, PKI Credential Issuance, Card Usage, Card Maintenance, and Card Termination. Figure 4.2 illustrates the adopted lifecycle model. The lifecycle for Derived Credentials is outside the scope of this work.

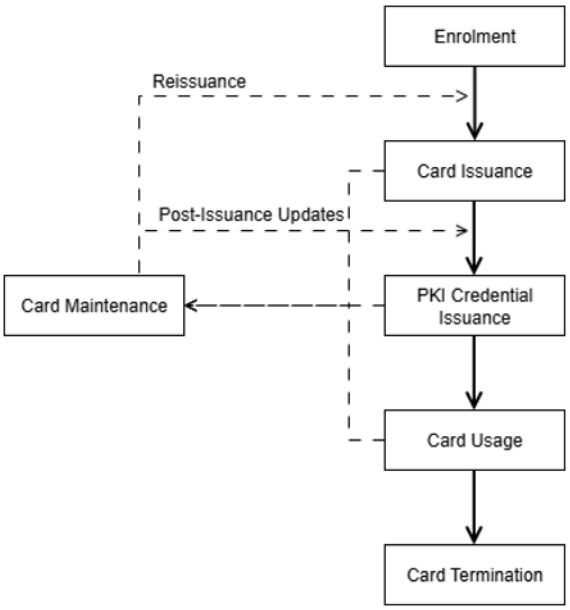


Figure 4.2: *Credential Lifecycle model for the proposed solution.*

Enrolment is entirely enterprise-managed and therefore not addressed in this work. Its output is an approved card applicant record with the attributes required for issuance. Those attributes are discussed later in this section.

The Card Issuance activity loads and installs the card application, provisions the Card Application Administration Key to protect management sessions, and sets the PIN and PUK in accordance with enterprise policy. Then, it writes the CHUID, the Card Capability Container (CCC), and the electronic facial image. If biometrics are enabled, it also loads up to two cardholder fingerprint templates. The Security Object is generated at the end of this activity, after all covered signed data objects have been written, and is then stored on the card.

The PKI Credential Issuance activity establishes the card's public-key credentials. The Card Authentication (CAK) and PIV-like Authentication (PKI-AUTH) key pairs are generated on the card, and the corresponding public-key certificates are stored in the

appropriate data containers on the card. This process is discussed in greater detail in Subsection 5.4.1.

During the Card Usage activity, the card is presented to PACS and LACS to obtain access to protected physical and logical resources. These access control systems perform certificate path validation and revocation checking on the presented certificate and enforce authorization decisions.

The Card Maintenance activity comprises routine post-issuance operations. These include certificate renewal, re-keying, and replacement. It also covers PIN change, reset, and unblock. When biometrics are enabled, it supports the addition or removal of fingerprint templates. The activity also applies updates to the remaining data objects defined by the credential data model. When any covered signed object is modified, the Security Object is regenerated. All operations follow enterprise policy and are executed through predefined CMS workflows.

The enterprise defines the conditions under which post-issuance updates or full reissuance occur. The CMS enforces these rules through controlled workflows and audit mechanisms. Finally, during the Card Termination activity, the card and all associated authentication data and keys are permanently destroyed or invalidated, preventing any future use of the credential for authentication purposes.

4.5 Card Management System

The Card Management System (CMS)) is a central component of the proposed solution and is responsible for coordinating credential lifecycle operations and interactions with external enterprise systems.

4.5.1 Role and Responsibilities

The CMS serves as the policy enforcement mechanism for the enterprise credential lifecycle. Enterprise governance sets the requirements. The CMS implements them as auditable and conformant workflows. These workflows manage the card application, keys, certificates, biometric data, and on-card data objects. The CMS is active in the following lifecycle activities: Card Issuance, PKI Credential Issuance, Card Maintenance, and Card Termination. Card Usage is enforced by PACS and LACS. Moreover, it underpins it by ensuring the correct provisioning of credentials and the timely publication of status updates.

The CMS interfaces with cards to load and install the card application, set their PIN and PUK, and the respective Card Application Administration Key. It writes data-model elements to the card, such as CHUID, CCC, mandatory facial image, and, if enabled,

fingerprint templates. Additionally, it triggers on-card generation of new CAK and PKI-AUTH key pairs. The corresponding public keys are used to construct certificate signing requests in accordance with the enterprise Certificate Policy (CP). The CMS obtains the issued certificates from the PKI and stores them in the appropriate containers on the card. Table 4.6 summarizes the CMS's responsibilities by credential lifecycle activity, the primary outputs and emitted events, and the principal external dependencies.

Lifecycle activity	CMS Context information
Card Issuance	**Responsibilities** Load and install card application; Generate and load Card Application Administration Key; Generate and load PIN and PUK; Store data model elements.
	Output Card is personalized.
	Event Output N/A
	External Dependencies IDMS for card applicant data.
PKI Credential Issuance	**Responsibilities** Trigger on-card generation of asymmetric key pairs; Apply for corresponding public key certificates per enterprise certificate policy; Store the issued public-key certificates on the card.
	Outputs Fully issued smart card credential; Cardholder data provisioning message to PACS.
	Event Output `CREDENTIAL_ISSUED`
	External Dependencies Enterprise PKI for certificate issuance; PACS connector for cardholder enrolment.
Card Maintenance	**Responsibilities** Certificate renewal/rekey/replacement; PIN change/reset/unblock; Add/remove fingerprint templates; Update card data objects.
	Outputs Updated smart card credential.
	Event Output `CREDENTIAL_UPDATED`
	External Dependencies Enterprise PKI for new certificate generation; IDMS for cardholder data changes; PACS for cardholder status update.
Card Termination	**Responsibilities** Revoke smart card credential certificates; Mark smart card credential unusable.
	Outputs Terminated smart card credential.
	Event Output `CREDENTIAL_TERMINATED`
	External Dependencies Enterprise PKI for certificate revocation; PACS for cardholder status update.

Table 4.6: Lifecycle activities and CMS context information.

4.5.2 External interfaces

As depicted in Figure 4.1, the CMS must interoperate with several enterprise systems, namely the IDMS, the enterprise PKI, and PACS/LACS. Given the diversity of products and custom deployments, prescribing a single interface for each system is unrealistic. Instead, this work defines logical interfaces: canonical data contracts and event types that specify the messages the CMS must send or receive, independent of technology. In each deployment, adapters bind the logical interfaces to local systems, for instance by publishing to the enterprise event bus, invoking REST APIs, or integrating via vendor SDKs. Given the critical-infrastructure context of this work, we recommend that all CMS interfaces must enforce mutual authentication and confidentiality.

Messages exchanged with the CMS consist of JSON records with a fixed envelope and a per-message detail object as shown in Listing 4.1.

```
{
  "version": "1.0",
  "id": "...",
  "detail_type": "...",
  "datetime": "2025-09-23T14:05:00Z",
  "source": "...",
  "account": "...",
  "detail": {
    ...
  }
}
```

Listing 4.1: Proposed message schema for exchanging messages.

The envelope contains the fields version, id, detail_type, datetime, source, account, and detail. The version field enables message schema evolution and backward compatibility. The id field uniquely identifies the message. We recommend using UUID v7, which embeds a creation timestamp for natural time ordering and facilitates simpler debugging and indexing. The detail_type declares the semantic type, allowing receivers to route and validate without inspecting the payload. It follows the convention COMPONENT_NAME.EVENT_NAME, for example, CMS.CREDENTIAL_ISSUED. The datetime field records the date and time the message was created by the sender. The use of the ISO 8601 format is recommended. The source and account identify the emitting system or service. The detail object carries the message-specific fields defined for that detail_type.

CMS – IDMS interaction. The detail object in messages originating from the IDMS and destined for the CMS comprises a single field, pending_applicants,

which contains a list of applicant records. Each record includes a stable id, to be used by both systems, name, given_name, surname, employee_affiliation, organization_affiliation_1, and organization_affiliation_2. The message detail_type is IDMS.ENROLLMENT_RECORD. The source and account envelope fields are populated by the IDMS. If the exchange is initiated by the CMS as a periodic poll, the request uses detail_type CMS.ENROLLMENT_QUERY and an empty detail object. The IDMS response then uses IDMS.ENROLLMENT_RECORD as above.

Listing 4.2 illustrates an example message for the detail_type IDMS.ENROLLMENT_RECORD.

```json
{
  "version": "1.0",
  "id": "01912f3a-8b7e-7e2a-b3d4-9a1c2f3e4b56",
  "detail_type": "IDMS.ENROLLMENT_RECORD",
  "datetime": "2025-09-23T14:05:00Z",
  "source": "idms://hr-core/lisbon",
  "account": "svc.idms.enrollment@enterprise.example",
  "detail": {
    "pending_applicants": [
      {
        "id": "EMP-2025-001234",
        "name": "Ana Ferreira",
        "given_name": "Ana",
        "surname": "Ferreira",
        "employee_affiliation": "Employee",
        "organization_affiliation_1": "Operations",
        "organization_affiliation_2": "Airport Security"
      },
      {
        "id": "CTR-2025-000987",
        "name": "Michael Smith",
        "given_name": "Michael",
        "surname": "Smith",
        "employee_affiliation": "Contractor",
        "organization_affiliation_1": "IT Department",
        "organization_affiliation_2": "Identity Services"
      }
    ]
  }
}
```

Listing 4.2: Example message for the detail_type `IDMS.ENROLLMENT_RECORD`.

CMS – PKI interaction. The CMS interacts with the enterprise PKI to request, renew, and revoke public-key certificates for on-card key pairs. This integration does not use the previously defined JSON message schema. Instead, the CMS includes a PKI adapter capable of communicating with the enterprise CA using a standard X.509 certificate enrolment and management protocol, such as CMP [1] or CMC [78], or a vendor API. The adapter submits certificate signing requests over a mutually authenticated channel and stores the returned X.509 certificate in the appropriate container on the card.

CMS – PACS interaction. The CMS sends provisioning messages that enable PACS to correlate the cardholder record with the presented card. Under the defined

JSON schema, the following detail_type values are used: CMS.CREDENTIAL_ISSUED, CMS.CREDENTIAL_UPDATED, and CMS.CREDENTIAL_TERMINATED. For each, the detail object shall contain card_uuid (the authoritative identifier assigned by the CMS), name, given_name, surname, employee_affiliation, organization_affiliation_1, and organization_affiliation_2. Access permissions (for instance, areas, roles, and schedules) are not managed by the CMS and are instead administered within PACS. The message source shall identify the emitting CMS component/instance (for example, the enrolment or issuance office), and the account shall identify the responsible officer or service principal. For automated flows, a dedicated service account shall be used.

4.6 Authentication Mechanisms

While FIPS 201-3 defines a broad set of authentication mechanisms, the proposed solution considers only four of those mechanisms: PKI-CAK, PKI-AUTH, OCC-AUTH, and a simple challenge–response protocol used for CMTC authentication. The choice of adopting these mechanisms was motivated by several factors, starting with the fact that they are supported by the proposed data model.

By default, CIV supports PKI-CAK, which supports a minimum of one authentication factor in physical access control scenarios and also supports logical access (remote/network access and local workstation access) with the minimum assurance level.

PKI-AUTH utilizes PIV-like authentication data, as defined in the proposed credential data model, to enable two-factor authentication for PACS and achieve higher assurance levels for LACS.

OCC-AUTH was choosen as a means to provide the third authentication factor ("something you are"), so that the proposed solution supports all three factors. However, this comes at the cost of ensuring that the selected smart card supports OCC. BIO(-A) was not considered since it requires off-card biometric comparison, which could raise privacy concerns since the biometric templates would be stored off-card, risking exposure.

Finally, the challenge–response protocol used for CMTC authentication is of high relevance, because it is how the CMS establishes a secure administrative session with a card during issuance and maintenance operations.

As described in Section 3.1.5, these mechanisms – PKI-CAK, PKI-AUTH, and CMTC – are based on challenge–response protocols. The request containing the nonce, the private key identifier (key reference), and the algorithm identifier derived from the card's X.509 certificate are forwarded by the reader to the smart card applet. The key and algorithm identifiers follow the FIPS PIV assignments. Table 4.7 summarizes the identifiers for PIV cryptographic keys and credentials, as well as their security conditions for use.

Key Identifier (HEX)	Key Type	Security Conditions for Use	
		Contact	Contactless
'80'	Card Application PIN	Always	VCI
'81'	PIN Unblocking Key	Always	Never
'96'	Primary Fingerprint (OCC)	Always	Secure Messaging (SM)
'97'	Secondary Fingerprint (OCC)	Always	Secure Messaging (SM)
'9A'	PIV Authentication Key	PIN or OCC	VCI and (PIN or OCC)
'9B'	PIV Card Application Administration Key	Always	Never
'9E'	Card Authentication Key	Always	Always

Table 4.7: PIV key and algorithm identifiers and their security conditions for use.

In contrast, Table 4.8 summarizes the algorithm identifiers supported by PIV that the proposed solution adopts.

Algorithm Identifier (HEX)	Algorithm – Mode
'05'	RSA 3072-bit modulus, $65537 \leq$ exponent $\leq 2^{256} - 1$
'07'	RSA 2048-bit modulus, $65537 \leq$ exponent $\leq 2^{256} - 1$
'08'	AES-128 – ECB
'0A'	AES-192 – ECB
'0C'	AES-256 – ECB
'11'	ECC: Curve P-256
'2E'	ECC: Curve P-384

Table 4.8: PIV Algoritm identifiers

The following describes the interactions between the cardholder, the smart card, the card reader, and the access control system as the parties involved in the authentication flow for all authentication mechanisms. For the sake of simplicity, the access control system is considered to be a "black box" that represents the generalization of all the access control system servers, controller panels, databases, and public key infrastructure.

PKI-CAK The PKI-CAK authentication mechanism relies on an asymmetric challenge-response protocol that utilizes the card authentication X.509 certificate and its associated private key, known as the CAK. The main interactions are described as follows:

1. The cardholder presents the smart card to a reader, and a connection is established;
2. The card application is selected;

3. The reader requests the CAK certificate from the card application;
4. The card retrieves the CAK certificate;
5. The card reader forwards the CAK certificate to the access control system;
6. The Access Control System performs full path validation of the CAK certificate in accordance with RFC 5280. It also parses the certificate to extract the public key and algorithm, and the Card UUID;
7. The access control system generates a challenge string;
8. The challenge string is sent to the reader along with the algorithm and Card Authentication Key identifiers;
9. The reader sends a request to the card to sign the challenge, including the same algorithm and private-key identifiers;
10. The card responds to the challenge by signing it using the CAK private key;
11. The signature is sent back to the reader;
12. The reader requests the access control system to validate the signature;
13. The access control system verifies the signature using the public key in the CAK certificate;
14. The Card UUID is used to perform an authorization check to determine whether the cardholder should be granted access;
15. An access decision is made for the cardholder.

Any verification failure aborts the attempt. The main interactions of the PKI-CAK authentication mechanism are summarized in Figure 4.3 as a UML simplified sequence diagram.

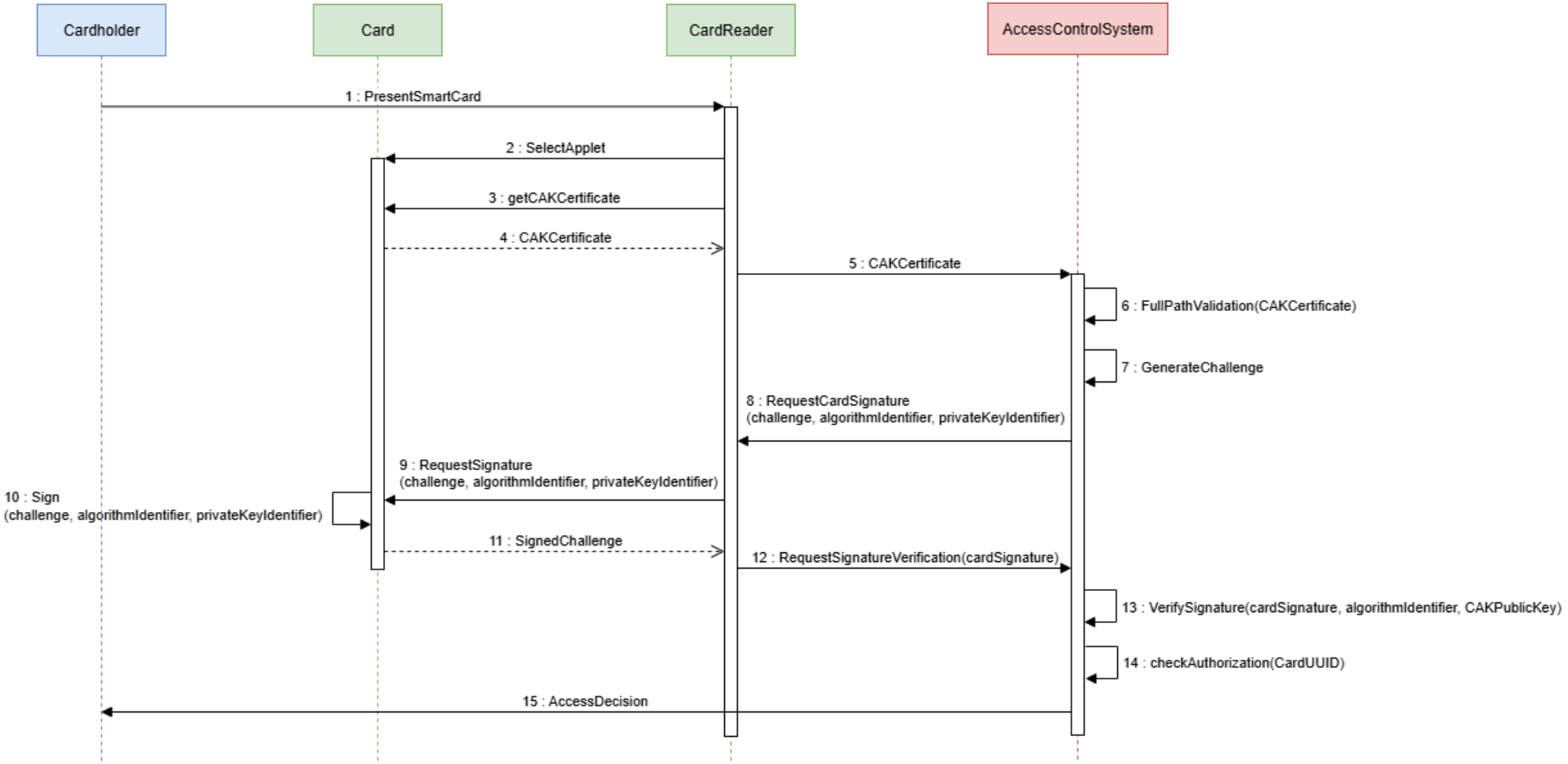


Figure 4.3: *PKI-CAK authentication mechanism sequence diagram.*

PKI-AUTH The PKI-AUTH authentication mechanism relies on an asymmetric challenge-response protocol that uses the PIV-like authentication X.509 certificate and its associated private key, the associated Authentication private key. The main interactions are as follows:

1. The cardholder presents the smart card to a reader, and a connection is established;
2. The card application is selected;
3. The reader requests the Authentication certificate from the card application;
4. The card retrieves the Authentication certificate;
5. The card reader forwards the Authentication certificate to the access control system;
6. The Access Control System performs full path validation of the Authentication certificate per RFC 5280. It also parses the certificate to extract the public key and algorithm, and the Card UUID;
7. The access control system requests the card activation (CTC authentication);
8. The reader prompts the cardholder to enter a PIN;
9. The cardholder enters and submits a PIN;
10. The card verifies the submitted PIN;
11. The card is now activated;
12. The access control system generates a challenge string;
13. The challenge string is sent to the reader along with the algorithm and authentication key identifiers;
14. The reader sends a request to the card to sign the challenge, including the same algorithm and private-key identifiers;
15. The card signs the challenge using the Authentication private key;
16. The signature is sent back to the reader;
17. The reader requests the access control system to validate the signature;
18. The Access Control System verifies the signature using the public key in the Authentication certificate;
19. The Card UUID is used to perform an authorization check to determine whether the cardholder should be granted access;

20. An access decision is made for the cardholder.

Any verification failure aborts the attempt. The main interactions of the PKI-AUTH authentication mechanism are summarized in Figure 4.4 as a UML simplified sequence diagram.

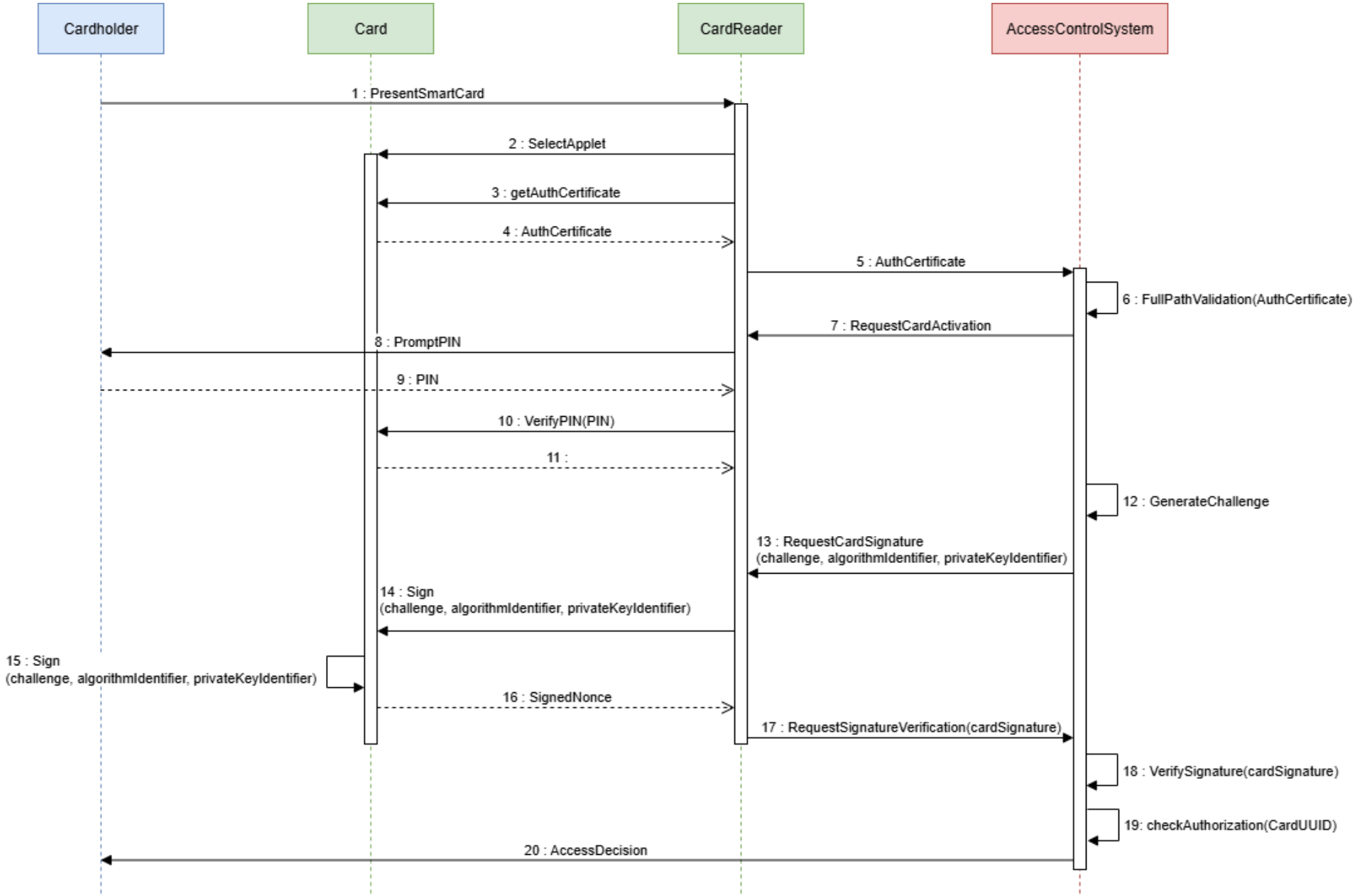


Figure 4.4: *PKI-AUTH authentication mechanism sequence diagram.*

OCC-AUTH OCC-AUTH requires that Secure Messaging Authentication (SM-AUTH) be successfully established prior to use. The card hosts the OCC algorithm and stores non-exportable reference biometric data. The main interactions are described as follows:

1. The cardholder presents the smart card to a reader, and a connection is established;
2. The card application is selected;
3. SM-AUTH is performed. The access control system authenticates the card, establishing secure messaging;
4. The cardholder provides a live-scan fingerprint and the reader derives a template compatible with the card's OCC format;
5. The reader sends an OCC verification request under secure messaging. The request is protected by a MAC;
6. The card compares the supplied template with the on-card reference using an OCC algorithm;
7. The card returns a verification decision within the MAC-protected secure-messaging session;
8. The relying system treats a positive verification decision as the biometric factor and proceeds.

Challenge-Response Protocol for Card Management Procedures To activate the card for personalization or data updates, the CMS performs a challenge-response protocol using the Card Application Administration Key. The protocol supports either unilateral authentication of the CMS to the card or mutual authentication between the CMS and the card. It is assumed that the card has already been presented to the reader, a connection has been established, and the card application has been selected. For simplicity, it is considered that the exchange occurs directly between the CMS and the card application. The main interactions are as follows:

1. The CMS requests a cipher from the card (a "witness"), indicating the algorithm and Card Application Administration Key, K, identifiers;
2. The card application generates 16 bytes of challenge data, C_{CARD}.
3. The card application encrypts C_{CARD} with K to produce the witness, W;
4. The card application returns W along with the algorithm and key identifiers;

5. The CMS decrypts W with its copy of K for that specific card to recover the challenge data, C'_{CARD};

6. The CMS generates its own 16 bytes of challenge data, C_{CMS}.

7. The CMS requests the card to encrypt C_{CMS} under the same key and algorithm, sending along the recovered challenge data, C'_{CARD} (proof it decrypted W).

8. The card verifies that the supplied value equals its original C_{CARD}. If correct, the CMS is authenticated to the card (unilateral authentication complete at this point).

9. The card encrypts C_{CMS} to produce $E_K(C_{\mathrm{CMS}})$;

10. The card returns $E_K(C_{\mathrm{CMS}})$ and signals that the CMS has been authenticated.

11. The CMS verifies $E_K(C_{\mathrm{CMS}})$ using K. If valid, the card is authenticated to the CMS, and mutual authentication is complete.

On success, the card enters an authenticated management state, and CMS-authorized personalization or update commands may proceed. Any verification failure aborts the session. Figure 4.5 summarizes these interactions as a UML simplified sequence diagram.

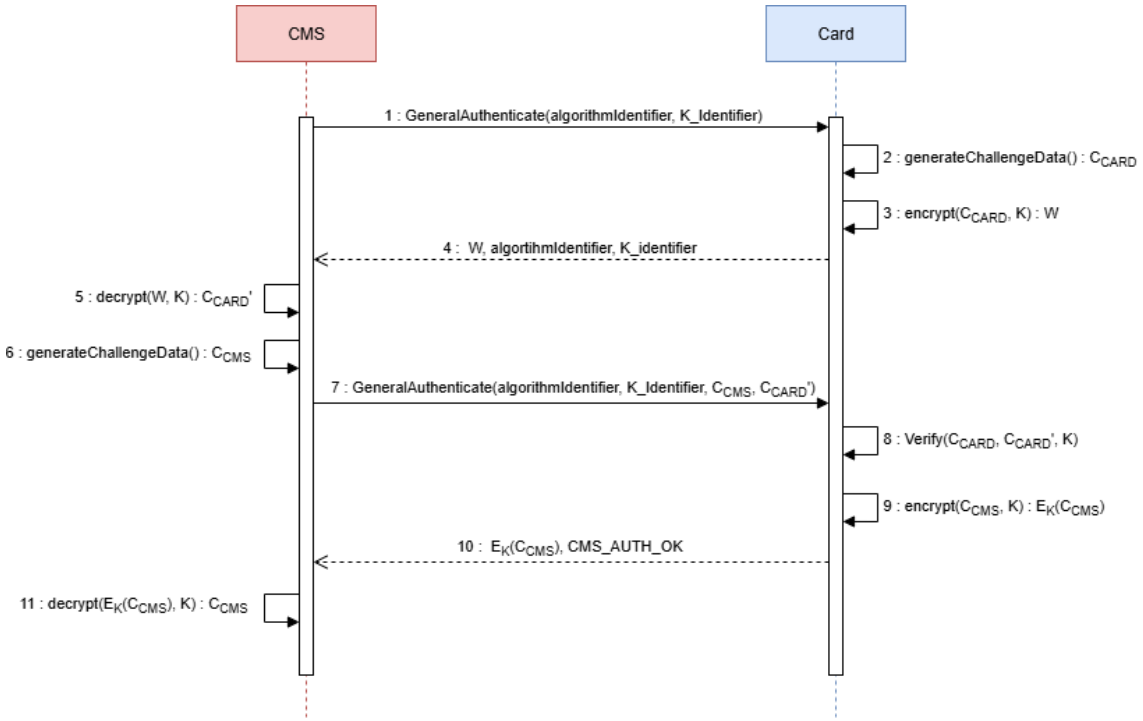


Figure 4.5: *CMS to Card authentication mechanism sequence diagram.*

4.7 Summary

This chapter presents a standard-based access control architecture designed for critical-infrastructure environments that addresses the gaps identified in Chapter 3.

The architecture is technically compatible with PIV and policy-aligned with CIV, enabling enterprises to maintain control over identity proofing and registration, and credential issuance workflows while supporting interoperability with PIV vendors. A CIV-based smart-card credential, compatible with the PIV data model and authentication mechanisms, acts as a unified credential for both physical and logical access. Enterprise-defined identifiers and certificate policy elements replace federal-specific PIV components, while full RFC 5280–compliant certificate validation and revocation checking are preserved. Authentication relies on a subset of PIV mechanisms using on-card, non-exportable private keys, and biometric verification.

A well-defined credential lifecycle, combined with a central role for the CMS, ensures policy enforcement, auditability, and seamless propagation of credential status updates across systems.

The next chapter details the implementation and validation of the proposed solution.

5 Implementation and Results

This chapter presents the proof-of-concept prototype developed to showcase the proposed solution described in Chapter 4. This prototype comprises an operational CMS with a user interface, a smart card applet that is loaded onto the card during Card Issuance, and an emulator that plays the role of an access control system. Together, these components enable the issuance of credentials and the simulation of their use at access points that require different authentication factors, by invoking distinct authentication mechanisms, such as PKI-CAK for one-factor authentication and PKI-AUTH for two-factor authentication.

Accordingly, Section 5.1 presents the prototype architecture and the test use cases. Section 5.2 describes the smart-card credential used in the prototype and the implemented data model, while Section 5.3 covers the card applet. Then, Section 5.4 details the CMS features and their interactions with other prototype components and Section 5.5 describes the access control system emulator. Finally, Section 5.6 presents the tests and results.

5.1 Architecture and Test Use Cases

The development of the prototype is based on the architecture proposed in Chapter 4 and instantiated as shown in Figure 5.1. Because no production enterprise environment was available, the entire prototype was implemented on a single workstation. Specific simplifications are detailed in subsequent sections.

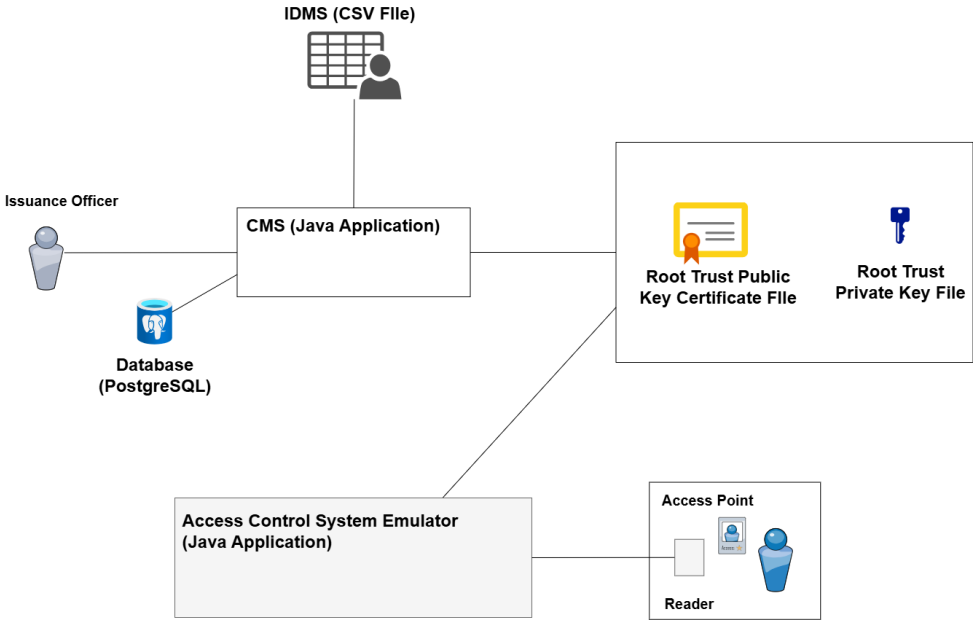


Figure 5.1: *Prototype architecture for the proof-of-concept implementation.*

The **Card Management System (CMS)** is a Java application that combines backend service logic with a JavaFX [65] administration user interface (UI) for the issuance officer. Smart card I/O uses APDU4J [68] for APDU exchange and GlobalPlatformPro [69] for loading and installing applets. The CMS persists cardholder and credential records in a locally hosted PostgreSQL [74] database. The workstation's embedded PC/SC reader connects the CMS to the smart cards. For certificate issuance and other handling operations, Java cryptography APIs (Java Cryptography Architecture (JCA)/Java Cryptography Extension (JCE)) were used in combination with the Bouncy Castle library [4]. Instead of integrating with a full enterprise PKI, the prototype employs an OpenSSL-generated [66] self-signed root (public key certificate and private key) imported by the CMS, which then issues end-entity card certificates. In production, certificate issuance would be performed by the enterprise CA using a standards-based enrolment protocol via a PKI adapter, as described in Section 4.5. An **Identity Management System (IDMS)** was not emulated. Instead, pending applicants are provided to the CMS by importing a Comma-Separated Values (CSV) file through the CMS user interface. In production, this data would arrive over a mutually authenticated channel using the JavaScript Object Notation (JSON) message schema defined in Section 4.5.

The **Physical and Logical Access Control System (PACS/LACS)** behaviour is represented by a Java application with a JavaFX UI and service logic. The emulator implements the two certificate-based mechanisms adopted in the solution – PKI-CAK (one-factor) and PKI-AUTH (two-factor) – and can read and decode the CHUID for debugging purposes. It displays authentication outcomes and logs the execution of the mechanism step-by-step. The emulator validates the card certificates by building a chain to the locally trusted self-signed root CA and checking the certificate's validity.

Revocation status (CRLs/OCSP) is not implemented in this prototype and would be enforced in a production deployment. Automated enrolment of cardholders from the CMS via the JSON message schema defined in Section 4.5 was not implemented in this prototype and is left as future work. Instead, records were supplied manually during the testing process.

The prototype uses a Java Card–based smart card (see Section 5.2) to store the required data elements and credentials and to execute cryptographic operations. The card is accessed via the workstation's embedded PC/SC reader. Contactless operation is not implemented. OCC is also not implemented, as no biometric reader was available and the selected card platform does not support OCC.

Given these architectural constraints and simplifications, and the requirements discussed in Section 4.1, a limited set of representative test scenarios was defined to exercise and validate the most relevant aspects of the solution, from issuance to usage of a credential (authentication and verification). These scenarios include both one-factor and two-factor authentication, and validate error handling and access denial. The goal is to demonstrate functional correctness, alignment with PIV and CIV standards, and observable behavior during usage under realistic constraints. The defined test use cases are as follows:

1. **Credential issuance:** (both personalization and PKI credential issuance);
2. **PKI-CAK (success):** successful one-factor authentication;
3. **PKI-CAK (failure):** authentication denied due to an expired certificate;
4. **PKI-AUTH (success)**;
5. **PKI-AUTH (failure):** authentication denied due to invalid PIN;
6. **CHUID retrieval/parsing:** for traceability and debugging.

5.2 Smart Card Credentials

The credential used in the prototype is a Java Card–based smart card, specifically the ACOSJ-G 95K from Advanced Card Systems [2]. Java Card was selected because it is a mature, standards-aligned platform for secure elements that supports multiple isolated applications on tamper-resistant hardware and is widely deployed in identity and security contexts [42, 84]. The chosen card conforms to GlobalPlatform 2.2.1 [28] for secure applet management and implements Java Card 3.0.4 Classic [67]. It is a dual-interface token with approximately 95 KB of EEPROM and both ISO/IEC 7816 (contact; T=0/T=1) and ISO/IEC 14443 (contactless) interfaces. Its on-card cryptographic toolkit includes DES/3DES, AES-128/192/256, RSA (up to 2048 bits), ECC (up to 384 bits), and

SHA-1/224/256/384/512. Table 5.1 summarizes the supported algorithms and key sizes. This model does not support OCC.

Symmetric Ciphers	Asymmetric Ciphers	Elliptic-Curve Cryptography (ECC)	Hashing Algorithms
DES, 2K3DES, 3K3DES (ECB and CBC)	RSA (768 to 2048 bits)	12/128/160/192/224/256/384-bitcurves	SHA-1/224/256/384/512
AES-128/192/256 (ECB/CBC)			

Table 5.1: Supported algorithms and key sizes for the smart card used in the prototype.

Adopted data model The data model implemented consists of a subset of the one defined for the proposed solution. Because the card does not support OCC and no biometric reader was available, fingerprint templates are not considered. The electronic facial image is also omitted, as it is not essential to the prototype's objectives. To keep the implementation focused on validating PKI-AUTH and PKI-CAK, the model also does not consider the Card Capability Container (CCC) and the Security Object. The resulting subset contains solely the elements necessary to exercise the two certificate-based authentication mechanisms and the challenge–response protocol used for card-management procedures (CMS–card application). Specifically, the prototype includes:

- an application PIN and its PUK;
- Card Authentication data (one asymmetric private key and corresponding X.509 certificate);
- PIV-like Authentication data (one asymmetric private key and corresponding X.509 certificate);
- the Card Application Administration Key; and
- the Cardholder Unique Identifier (CHUID).

According to the solution proposal, the CHUID uses a 16-byte Universally Unique Identifier (UUID) as specified in RFC 4122 in place of the FASC-N. For the PIV-like Authentication and Card Authentication key pairs, the ECDSA P-256 was selected (a recommended option), because on-card ECC private-key operations are typically faster than RSA on the same platform. In initial tests, CHUID signatures and authentication challenge–response signatures were noticeably smaller with ECDSA-P256 than with RSA-2048, reducing communication and storage overhead. Certificates issued with ECDSA-P256 were likewise substantially smaller. For the Card Application Administration Key, the baseline AES-128 in ECB mode was used for the challenge–response protocol implemented by the applet for card management procedures. This choice was merely for simplicity.

The PIN/PUK policy matches the constraints discussed in Section 4.3. The PIN is a 6–8-digit value (bytes are American Standard Code for Information Interchange (ASCII) '0'– '9' characters, padded with 0xFF), and the PUK is an 8-byte value (0x00–0xFF). A default retry limit of 3 was established for both the PIN and the PUK. Table 5.2 contrasts the full credential data model proposed for the solution with the subset realized in the prototype, listing the algorithm/key parameters where applicable.

Data Element	**Proposed model**	**Prototype**	**Algorithm / key parameters (if applicable)**
Application PIN and PUK	Yes	Yes	N/A
Card Authentication data	Yes	Yes	ECDSA P-256
PIV-like authentication data	Yes	Yes	ECDSA P-256
Two fingerprint templates	Optional	No	N/A
Electronic facial image	Yes	No	N/A
Card Application Administration Key	Yes	Yes	AES-128 ECB
Cardholder Unique Identifier (CHUID)	Yes	Yes	N/A
Card Capability Container (CCC)	Yes	No	N/A
Security Object	Yes	No	N/A

Table 5.2: Comparison between the data model adopted in the prototype and in the proposed solution.

X.509 Certificates The card certificates used in the developed prototype are issued by a locally generated root CA using Bouncy Castle [4]. Common elements across both certificates are:

- basicConstraints extension with the CA attribute set to false (and marked critical);
- keyUsage extension set to digitalSignature (and marked critical);
- Authority Key Identifier (AKI) and Subject Key Identifier (SKI) extensions present;
- subjectAltName extension containing the card's UUID encoded as URN.

Per-certificate specifics are as follows:

- For the **PIV-like authentication certificate**, the Extended Key Usage (EKU) includes id-kp-clientAuth (1.3.6.1.5.5.7.3.2) and Microsoft Smartcard Logon (1.3.6.1.4.1.311.20.2.2) to support Local Workstation Access;
- For the **Card Authentication certificate**, the EKU includes id-PIV-cardAuth (2.16.840.1.101.3.6.8).

For expedience during testing, the certificatePolicies extension asserts the PIV policy OIDs, namely id-fpki-common-authentication for PKI-AUTH and id-fpki-common-cardAuth for PKI-CAK. In a production environment, these policy OIDs would be replaced in accordance with enterprise CPS. The emulator does not enforce revocation checking, therefore AIA and CRLDP are omitted from certificates. Validity intervals were set to 3 years – and sometimes intentionally expired – to exercise error handling. In a real deployment, validity must align with the enterprise CPS and must not exceed the card's lifetime.

All card keys and signature operations used ECDSA P-256, which reduces signature and certificate sizes and speeds private-key operations on the card. Subject Distinguished Names (DNs) and the root CA naming are placeholders for test purposes only.

Tables 5.3 and 5.4 compare the proposed CIV certificate profiles for PKI-AUTH and PKI-CAK, respectively, with their prototype implementations.

Attribute	Proposed profile	Prototype
Version	v3	v3
Issuer	Enterprise CA hierarchy	Locally generated root CA
Validity		
Not Before	Per enterprise policy	`<issuing date>`
Not After	Per enterprise policy	`<issuing date>` + 3 years
Subject Public Key Info		
Algorithms supported	RSA 2048/3072; ECDSA P-256/P-384	ECDSA P-256
X.509v3 Extensions		
Authority Key Identifier	Present	Present
Subject Key Identifier	Present	Present
Key Usage	digitalSignature only	digitalSignature only
Certificate Policy Identifier	Per enterprise policy	id-fpki-common-authentication
Subject Alternative Name	UUID only (encoded as URN)	UUID only (encoded as URN)
Basic Constraint (CA)	false (critical)	false (critical)
Extended Key Usage	id-kp-clientAuth; MS Smartcard Logon (if needed)	id-kp-clientAuth; MS Smartcard Logon
CRLDistributionPoints	Populated per enterprise policy	Not populated
Authority Information Access	Populated per enterprise policy	Not populated
Signature Algorithm	RSA 2048/3072; ECDSA P-256/P-384	ECDSA P-256

Table 5.3: Comparison between the PKI-AUTH certificate policies of the proposed solution and prototype.

Attribute	**Proposed profile**	**Prototype**
Version	v3	v3
Issuer	Enterprise CA hierarchy	Locally generated root CA
Validity		
Not Before	Per enterprise policy	`<issuing date>`
Not After	Per enterprise policy	`<issuing date>` + 3 years
Subject Public Key Info		
Algorithm	RSA 2048/3072; ECDSA P-256/P-384	ECDSA P-256
X.509v3 Extensions		
Authority Key Identifier	Present	Present
Subject Key Identifier	Present	Present
Key Usage	digitalSignature only	digitalSignature only
Certificate Policy Identifier	Per enterprise policy	id-fpki-common-cardAuth
Subject Alternative Name	Card UUID only	Card UUID only
Basic Constraint (CA)	false (critical)	false (critical)
Extended Key Usage	id-PIV-cardAuth	id-PIV-cardAuth
CRLDistributionPoints	Populated	Not populated
Authority Information Access	Populated	Not populated
Signature Algorithm	RSA 2048/3072; ECDSA P-256/P-384	ECDSA P-256

Table 5.4: Comparison between the PKI-CAK certificate policies of the proposed solution and prototype.

5.3 Smart Card Applet

The prototype includes a Java Card applet that has been loaded onto the test cards during the Card Issuance lifecycle activity. This section describes the features of the applet, its internal architecture, APDU-level command interface, and security model.

The devised applet implements secure storage of the cardholder data and credentials, namely the CHUID, PIN/PUK, Card Application Administration Key, and the PKI-CAK and PIV-like Authentication key pairs and certificates, as well as on-card generation of asymmetric keys, and execution of the PKI-CAK and PKI-AUTH authentication mechanisms. The applet does not support contactless operation.

The SP 800-73-5 Part 1 [25] was followed for namespace, data model, and representation, in order to maximize interoperability with PIV-aware middleware and readers. The application was configured with the PIV AID ('A0 00 00 03 08 00 00 10 00 01 00'). Yet, the developed prototype omitted the two-byte version component.

The applet exposes the following command set: SELECT, GET DATA, VERIFY, PUT DATA, GENERATE ASYMMETRIC KEY PAIR, and GENERAL AUTHENTICATE. Moreover, it enforces read/write and activation rules consistent with NIST SP 800-73 [23, 25]. SELECT (INS 'A4') selects the application (using the PIV AID; see above). GET DATA (INS 'CB') returns the contents of a single data object identified by the tag in the command. VERIFY (INS '20') compares the submitted reference data (e.g., PIN) against the stored data and is also used to probe the remaining retries. GENERAL AUTHENTICATE (INS '87') performs a cryptographic operation (challenge–response/signature) with the private key referenced in the command. PUT DATA (INS 'DB') replaces the contents of a single data object. GENERATE ASYMMETRIC KEY PAIR (INS '47') generates a key pair and returns the public key parameters. Command security conditions and interface availability follow SP 800-73-5 Part 2 [23]. Table 5.5 summarizes these conditions for the commands implemented in the prototype card applet. The CMS and the access control emulator interact with this card applet by sending APDU that invoke this command set.

Type	Name	Contact Interface	Security Condition for Use	Command Chaining
Data Access	SELECT	Yes	Always	No
	GET DATA	Yes	Data Dependent.	No
Authentication	VERIFY	Yes	Always	Yes
	GENERAL AUTHENTICATE	Yes	Key Dependent.	Yes
Credential Initialization and Administration	PUT DATA	Yes	PIV Card Application Administrator (Activation by CMS)	Yes
	GENERATE ASYMMETRIC KEY PAIR	Yes	PIV Card Application Administrator (Activation by CMS)	Yes

Table 5.5: Summary of the commands exposed by the prototype card applet.

As discussed in Section 5.2, only the subset of data elements required by the proposed solution was implemented:

- the application PIN and PUK;
- Card Authentication key pair and certificate;
- PIV-like Authentication key pair and certificate;
- the Card Application Administration Key;
- and the CHUID.

During installation, the CMS passes a Tag-Length-Value (TLV) payload containing the PIN (tag 0x01), PUK (0x02), and Card Application Administration Key (tag 0x03). The applet constructor parses this TLV and initializes the corresponding objects. The PIN and PUK are stored in OwnerPIN instances with retry limits set to 3. The Card Application Administration Key is stored in an AESKey instance. An AES cipher instance (Cipher.ALG_AES_BLOCK_128_ECB_NOPAD) is provisioned to support the CMS–card mutual authentication, which activates the card and enables administrative operations. A Signature instance (Signature.ALG_ECDSA_SHA_256) is used for ECDSA signatures, and a RandomData instance (RandomData.ALG_SECURE_RANDOM) generates challenges. ECC P-256 domain parameters are defined as static byte arrays. To support objects larger than a single APDU (e.g., CHUID and X.509 certificates), the

applet employs utility classes for APDU chaining and transient auxiliary buffers. These transient buffers are cleared when the card is removed. All data model elements and credentials are maintained as byte arrays in persistent memory. Key references (e.g., 0x9A for PIV-AUTH and 0x9E for CAK) are defined as constants.

The activation state of the card is tracked with Boolean flags. One flag tracks PIN activation, and another tracks activation by the CMS via the Card Application Administration Key. These flags are used to control access to commands and data. For instance, PUT DATA and GENERATE ASYMMETRIC KEY PAIR require successful CMS administrative activation. GENERAL AUTHENTICATE using the PIV-AUTH private key requires prior PIN activation. Read-access rules for data elements follow SP 800-73-5 Part 1 [25] and apply to the contact interface used in the prototype. Contactless was not considered.

The applet supports RSA-2048 and ECDSA-P256. Initial experiments with RSA-2048 resulted in larger signatures and certificate payloads, which increased I/O and risked storage pressure during chained APDU. The prototype, therefore, defaults to ECDSA-P256 for key generation and signatures to reduce payload sizes and improve performance.

5.4 Card Management System

The Card Management System (CMS) is implemented in Java and comprises two layers: a JavaFX desktop client for issuance and administration, and a backend service layer that executes the core lifecycle logic. Smart card I/O utilizes apdu4j for APDU transport and GlobalPlatformPro for loading and installing CAP file applets, as well as opening Secure Channels. Data persistence is provided by PostgreSQL, which the CMS accesses via the PostgreSQL JDBC driver. X.509 certificate generation and parsing rely on the JCA/JCE cryptography APIs and the Bouncy Castle libraries.

The prototype CMS implements two lifecycle activities from the proposed data model: Card Issuance and PKI Credential Issuance. Card Termination was not implemented because its procedures are quite regular and add limited insight. Card Maintenance largely reuses the same operations as issuance (e.g., generating keys and writing objects with the PUT DATA command), and PIN management is a routine process. These two implemented activities, therefore, cover the core aspects of the proposed solution. Given time constraints, we prioritized the path from enrollment to usage. In the user interface, these choices are reflected on the initial screen. As shown in Figure 5.2, the CMS opens to a default Main Menu where the card-lifecycle activities are displayed in a layout that mirrors the lifecycle diagram presented in Figure 4.2. The buttons for the implemented activities – Card Issuance and PKI Credential Issuance – are enabled, while the remaining activities are intentionally disabled.

All lifecycle workflows execute off the JavaFX Application Thread. Each lifecycle activity

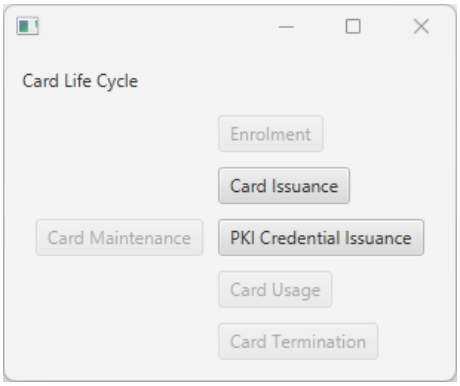


Figure 5.2: *Main menu of the CMS showing lifecycle activities and the implemented activities enabled.*

is implemented as its own javafx.concurrent.Task extended class. The call() method performs the background work, while progress and status updates are synchronized with the UI thread via the task's observable properties, keeping the interface responsive.

The database schema comprises two tables. The first stores cardholder records. In the prototype, applicant data supplied by the IDMS, which in this case is represented by a simple CSV file, is converted into cardholder entries at issuance and stored in this table. The second stores credential metadata and status.

5.4.1 Card Issuance workflow

In the prototype, the card issuance activity begins by importing a CSV file containing card applicant records, as shown in Figure 5.3.

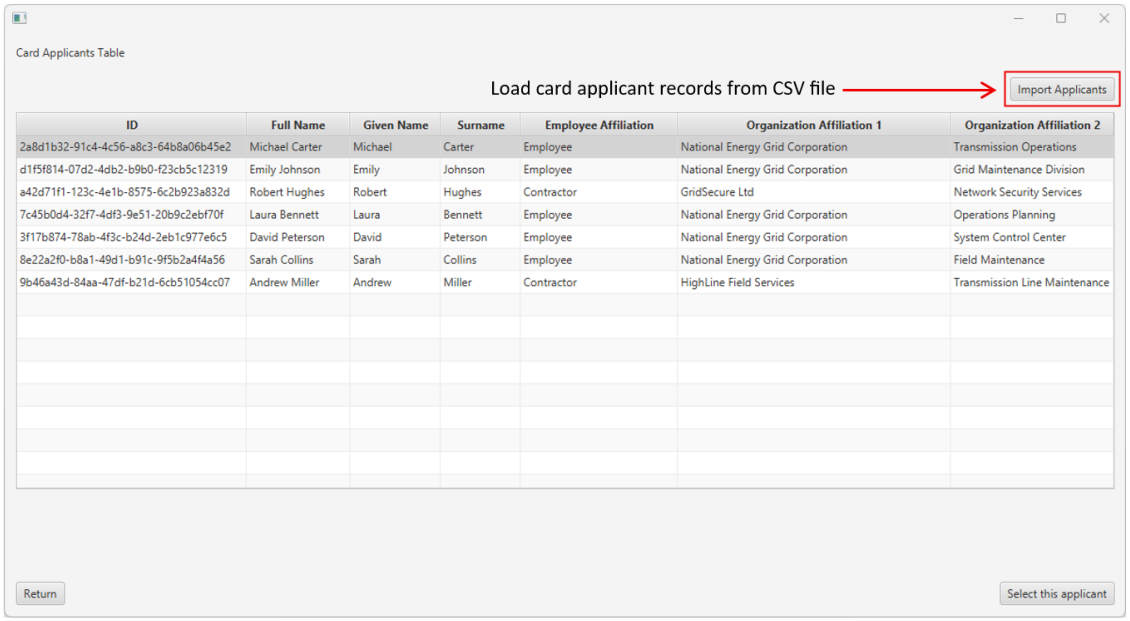


Figure 5.3: *Card applicants table of the CMS after loading the respective records from a CSV file.*

Its columns mirror the fields that, in a production environment, would arrive from the

IDMS as an IDMS.ENROLLMENT_RECORD message as defined in Subsection 4.5.2. Once an applicant is selected, a screen showing only the selected applicant's information is presented. When the "Issue Card" button is clicked, a confirmation prompt is displayed. Upon confirmation, CardIssuanceTask is launched to execute the issuance workflow, as illustrated in Figure 5.4.

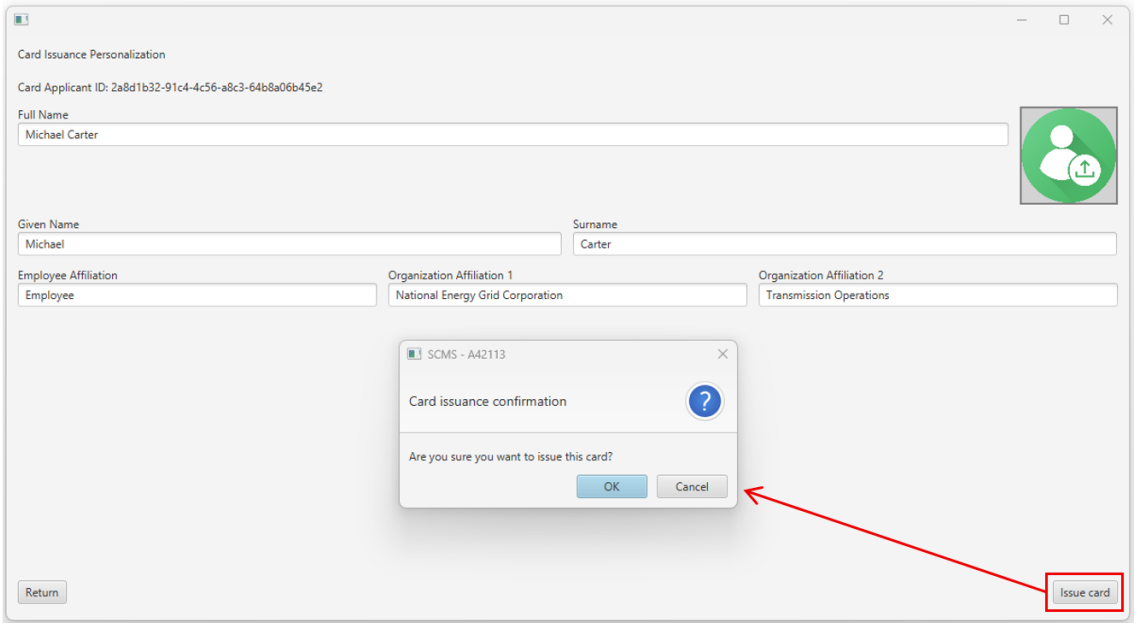


Figure 5.4: *Card Issuance workflow trigger steps in the CMS.*

In this workflow, the CMS generates a CHUID, creates a cardholder record, and initializes the following card's keys:

- a PIN of 6-8 numeric digits (length randomly varied between 6 and 8 in tests to exercise padding);
- an 8-byte PUK; and
- a Card Application Administration Key (AES-128, ECB) used to activate the card for administrative operations.

These values are persisted in the smart card table. In the prototype, the Administration Key is stored in Base64 encoding, and the PIN and PUK are stored in plaintext for ease of testing. Only the initially assigned PIN is retained to support issuance hand-off (subsequent PIN changes are not stored). In a production system, keys should be protected with strong encryption, strict access controls, and audit.

The workflow continues with the CMS establishing a GlobalPlatform secure channel to the card using the default development keys. In a production deployment, unique GlobalPlatform keys are set per card and stored only with strong encryption, strict access controls, and audit. Secure-channel establishment should use SCP03 (AES) whenever

supported by the card. With the channel established, the CMS loads the CAP file and installs the applet, passing a TLV payload as an installation parameter that contains the PIN (tag 0x01), PUK (0x02), and the Card Application Administration Key (tag 0x03). The applet's constructor parses the TLV structure and sets the keys accordingly.

After installation, the CMS selects the applet and performs mutual authentication using the Card Application Administration Key, which is shared by both parties. Successful authentication activates the card and enables administrative operations. The CMS then writes the CHUID using a PUT DATA command. The GP secure channel is closed, and the card is recorded as PERSONALIZED in the database. A confirmation dialog is displayed by the CMS indicating whether the card issuance succeeded or failed, as shown in Figure 5.5.

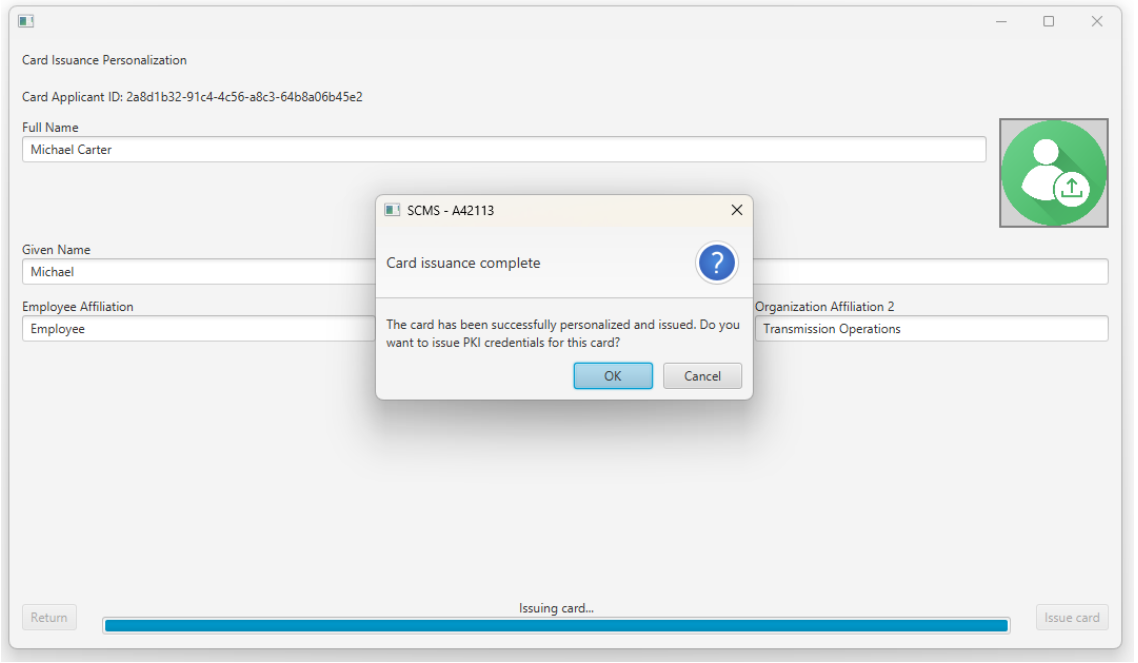


Figure 5.5: *Card issuance completion dialog.*

5.4.2 PKI Credential Issuance workflow

If the card issuance workflow completed successfully, the issuance officer is brought to a new screen to initiate PKI credential issuance, as illustrated in Figure 5.6. When PKI credential issuance is triggered, an instance of IssuePKICredentialsTask is launched. The task opens a GlobalPlatform secure channel, reads the card's CHUID via GET DATA, and uses that value to look up the corresponding Card Application Administration Key in the local database. Using that key, the CMS conducts mutual authentication with the applet, which activates the card and enables administrative operations. The CMS then issues GENERATE ASYMMETRIC KEY PAIR twice – first with key reference 0x9A for the PIV-like Authentication key, and then with 0x9E for the Card Authentication key – selecting the ECC curve P-256 algorithm (see Tables 4.7 and 4.8). The card

generates both key pairs and stores each in its designated data objects. Private keys remain non-exportable, and public key parameters, X and Y coordinates of point P, are returned to the CMS.

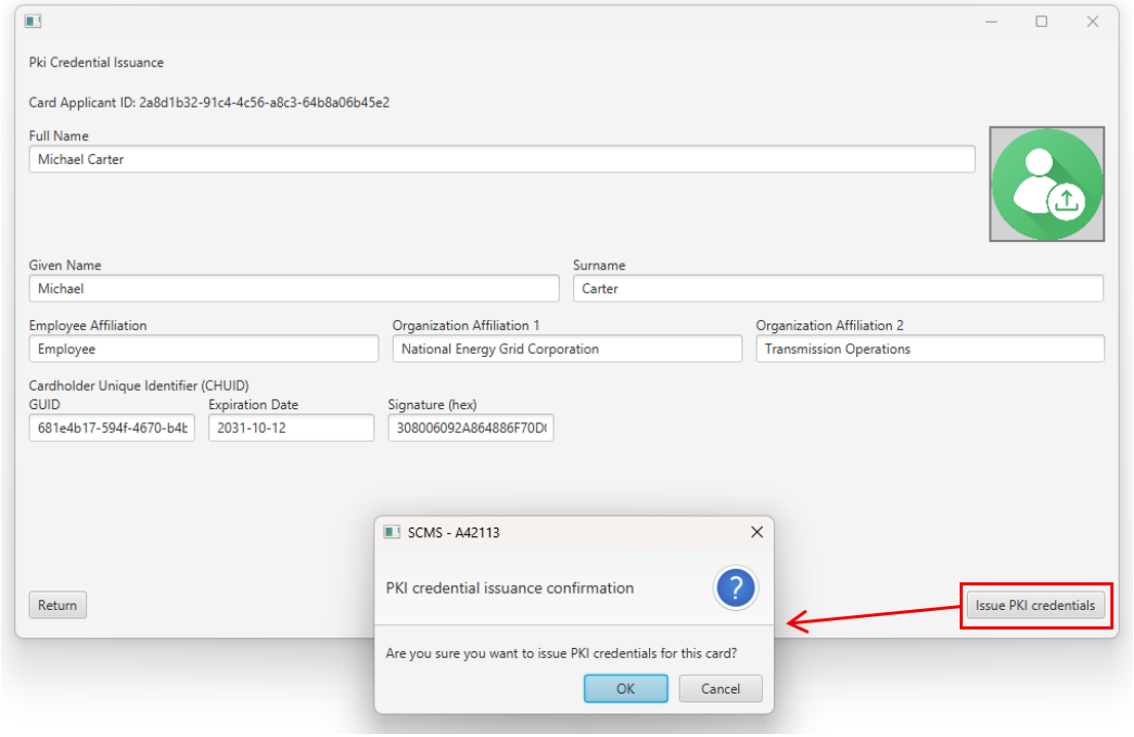


Figure 5.6: *PKI Credential Issuance workflow trigger steps in the CMS.*

The CMS uses Bouncy Castle and the Java Cryptography API (JCA/JCE) to instantiate a PublicKey object from the returned parameters and to construct the card X.509 certificate, following the prototype profiles described in Section 5.2. In this prototype, the CMS acts as the issuing CA, applies the selected subject and extension sets, and signs an end-entity X.509 certificate using the locally generated root CA private key. For simplicity, AIA and CRLDP are omitted, and revocation checking is disabled. In a production environment, AIA/CRLDP would be published, and revocation mechanisms such as CRL and OCSP would be specified in the certificate.

The resulting X.509 certificates are encoded as ASN.1.1 DER byte arrays and written to their designated data objects on the card using the PUT DATA command, with APDU chaining where required. The CMS marks the card as fully issued in the database. The credential can subsequently be tested with the access control emulator using the PKI-CAK and PKI-AUTH mechanisms. A confirmation dialog is displayed by the CMS indicating whether the PKI credential issuance succeeded or failed, as shown in Figure 5.7.

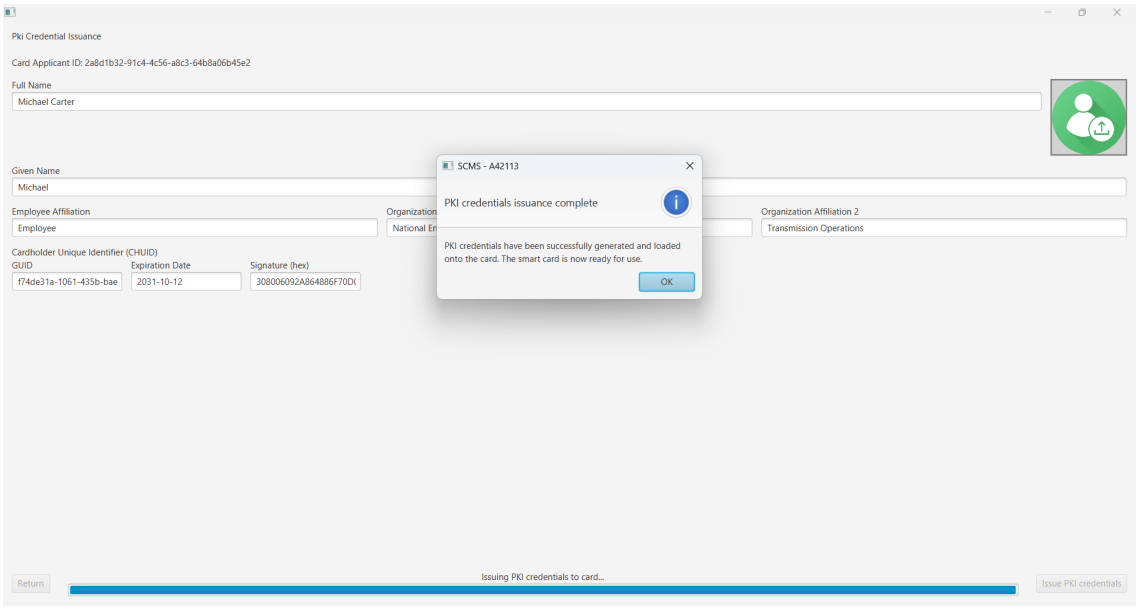


Figure 5.7: *PKI credential issuance completion dialog.*

5.5 Access Control System Emulator

The access control system emulator implements the behavior of an enterprise PACS/LACS. It is implemented in Java with a JavaFX user interface and a minimal service layer. The UI provides controls for each mechanism, displays step-by-step logs of APDU and validation actions, and shows the outcome (success or failure). Long operations run off the JavaFX Application Thread using javafx.concurrent.Task, with progress and status synchronized via observable properties to maintain UI responsiveness.

The emulator supports PKI-CAK (one-factor) and PKI-AUTH (two-factor) authentication, and can read and decode the CHUID for debugging. OCC-AUTH (on-card biometric comparison) is not supported because biometrics are beyond the scope of this work and the card platform lacks OCC support. Figure 5.8 shows the main menu of the emulator with the controls to access these features.

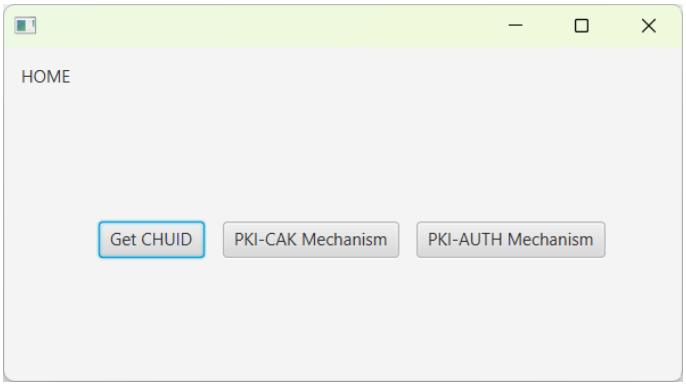


Figure 5.8: *Main menu of the emulator.*

The implemented authentication mechanisms follow a common flow:

1. the emulator reads the relevant certificate via the DATA command;

2. the emulator validates it by building a certification path to the locally trusted self-signed root and checking the validity interval;

3. and then performs a challenge-response using the GENERAL AUTHENTICATE command with the appropriate key reference.

PKI-AUTH adds a PIN step. The emulator probes the remaining PIN retries and, if the counter is zero, the attempt fails. Otherwise, it prompts for the PIN, as illustrated in Figure 5.9, then sends it to the card for verification and subsequent activation, enabling the challenge–response protocol.

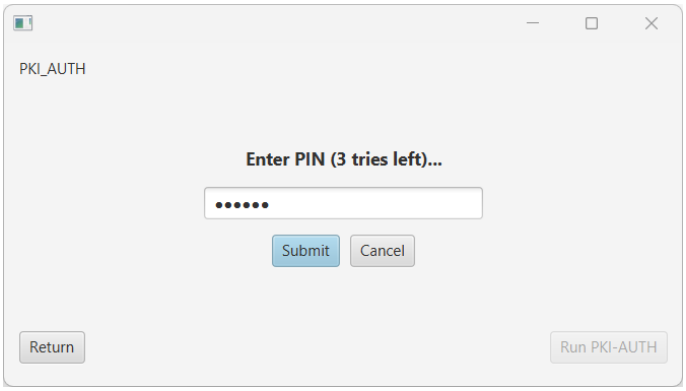


Figure 5.9: *PIN prompt of the emulator for the PKI-AUTH mechanism.*

After a successful challenge-response protocol, the emulator derives an authorization decision by checking the card UUID (extracted from the certificate Subject Alternative Name (SAN)) against an in-memory allow/deny list that emulates the ACLs typically used in production PACS/LACS. These ACLs generally map each cardholder to areas, time schedules, and privileges. Revocation status (CRLs/OCSP) is not implemented in this prototype and would be enforced in production. Automated enrolment from the CMS via the JSON schema in Section 4.5 is left as future work. For testing, cardholder and credential records were supplied manually.

5.6 Tests and Results

This section evaluates the developed prototype against the design objectives defined in Section 4.1. The goal is not to produce production-grade benchmarks, but to demonstrate that the core lifecycle – issuance, certificate provisioning, and use at an access point – functions correctly, adheres to the intended profiles, and fails safely under common error conditions.

5.6.1 Test environment

All the tests were conducted on a single laptop-class workstation with an embedded USB PC/SC reader. The prototype components were executed locally on this host:

- **Credential:** ACOSJ-G 95K Java Card (dual-interface; tests used the contact interface only).
- **CMS:** Java 23 application with a JavaFX UI; apdu4j for APDU transport; GlobalPlatformPro for secure CMS-card channel establishment, CAP file (applets) load, and install; data persistence by PostgreSQL via the PostgreSQL JDBC driver; certificate operations handled with JCA/JCE and Bouncy Castle.
- **Access control emulator:** Java/JavaFX application using the same card I/O stack and crypto libraries as the CMS.
- **PKI for tests:** locally generated self-signed root CA (OpenSSL); the CMS issued end-entity certificates from this root for the card public keys.

The considered environmental constraints are as follows: single-workstation deployment, contact interface only, no enterprise IDMS/PKI, no network OCSP/CRLs distribution, and no biometrics. The prototype data model did not include the Security Object or Card Capability Container, so integrity binding of signed data objects and capability metadata was not exercised.

5.6.2 Functional Evaluation

This section presents the six use cases used to evaluate the prototype and the corresponding obtained results.

Use case 1 – Credential issuance

Objective: Create a usable credential from an applicant record.

Procedure:

1. Import applicants (CSV) into the CMS;
2. Select an applicant;
3. Run Card Issuance (PIN/PUK/Administration Key generation; CAP load/install; mutual authentication; CHUID write);
4. Run PKI Credential Issuance (on-card generation of PKI-AUTH and CAK key pairs; local CA issuance of card certificates; execute PUT DATA command to load certificates onto the card).

Result: Cards were consistently marked PERSONALIZED after CHUID write and ISSUED after certificates were successfully stored in the card. The certificate contents matched the prototype profiles.

Use case 2 – PKI-CAK (success)

Objective: One-factor authentication using the Card Authentication certificate.

Procedure:

1. Present the card to the reader;
2. Trigger the PKI-CAK mechanism in the emulator UI;
3. Emulator reads CAK certificate (GET DATA), validates chain to local root and validity interval, then issues a challenge via GENERAL AUTHENTICATE with the CAK key reference;
4. Emulator provides feedback on the outcome of the authentication decision.

Result: The returned signature was verified under the certificate's public key; the emulator recorded `Authentication: SUCCESS` and issued `Access: ALLOW` when the UUID was present in the allow list.

Use case 3 – PKI-CAK (failure: expired certificate)

Objective: Deny authentication when the CAK certificate is expired.

Procedure:

1. Present a card that was deliberately issued with a past `NotAfter` (negative test) to the reader;
2. Trigger the PKI-CAK mechanism in the emulator UI.

Result: The emulator rejected the certificate at validation time and recorded `Authentication: FAIL (expired)`; no challenge was attempted.

Use case 4 – PKI-AUTH (success)

Objective: Two-factor authentication using the PIV-like Authentication certificate plus PIN.

Procedure:

1. Present the card to the reader;
2. Trigger the PKI-AUTH mechanism in the emulator UI;
3. Emulator reads and validates the PKI-AUTH certificate;
4. Emulator probes remaining PIN retries;
5. There are remaining tries. Emulator prompts for PIN;
6. Cardholder introduces PIN;
7. PIN is correct. Emulator performs challenge–response (GENERAL AUTHENTICATE with PKI-AUTH key reference and ECC P-256 identifier).

Result: The returned signature was verified under the certificate's public key; `Authentication: SUCCESS` and `Access: ALLOW` (UUID in allow list).

Use case 5 – PKI-AUTH (failure: invalid PIN)

Objective: Deny authentication when the PIN is incorrect.

Procedure:

1. Present the card to the reader;
2. Trigger the PKI-AUTH mechanism in the emulator UI;
3. Emulator reads and validates the PKI-AUTH certificate;
4. Emulator probes remaining PIN retries;
5. There are remaining tries. Emulator prompts for PIN;
6. Submit an invalid PIN to validate retry counter decrement; do not unblock.

Result: Emulator reported `VERIFY PIN: FAIL`, decremented the retry counter accordingly, and did not issue the cryptographic challenge.

Use case 6 – CHUID retrieval/parsing

Objective: Aid traceability and debugging.

Procedure:

1. Present the card to the reader;
2. Trigger the GET CHUID feature in the emulator UI;

3. Emulator reads CHUID and displays its fields in the UI.

Result: Parsed fields matched values written by the CMS during Card Issuance.

5.6.3 Security Evaluation

Within the scope of the prototype, cryptographic controls behaved as expected, and failure modes were handled appropriately. Positive test cases succeeded, while negative cases were rejected before any cryptographic challenge–response was attempted. Both PKI-CAK and PKI-AUTH used nonce-based challenges transported via the applet's GENERAL AUTHENTICATE command. Authentication keys (PKI-AUTH and CAK) were generated on the card and remained non-exportable. For PKI-AUTH, private-key use required correct card PIN input. Retry counters are decremented on failure, and a zero-retry state blocks further attempts. Revocation status (CRLs/OCSP) is not implemented in this prototype but would be enforced in a production environment. To address revoked credentials, the prototype utilizes an in-memory allow/deny list. Following a successful challenge-response protocol, the emulator determines an authorization decision by checking the card UUID against that list.

5.7 Summary

The developed prototype demonstrates that the proposed architecture is technically feasible and can be realized using open-source frameworks and the Java Card platform. The CMS supports card personalization and on-card generation of cryptographic keys, issues X.509 credentials bound to those keys, and enables certificate-based authentication using PKI-CAK and PKI-AUTH mechanisms.

The experimental results confirm correct behavior across both success and failure scenarios, including expired certificates and invalid or blocked PINs. The use of ECDSA-P256 proved effective in reducing certificate and signature sizes, limiting APDU chaining during personalization and authentication.

The main gaps relative to a production deployment are environmental rather than architectural. These include the absence of an enterprise CA/RA with published AIA/CRLDP, lack of revocation checking, contact-only operation, omission of biometric mechanisms, and prototype-grade key management. Addressing these aspects, together with systematic performance measurements and interoperability testing against a commercial PACS/LACS, constitutes a clear and well-defined direction for future work.

6 Conclusion

This thesis aimed to close a practical gap that enabled enterprises – especially operators of critical infrastructure – to converge physical and logical access on a single, standards-aligned smart card credential, while maintaining control over identity proofing, registration, and credential issuance. This work proposed a CIV-style credential model that is technically compatible with the FIPS 201-3 Standard, integrates with enterprise systems (IDMS, PKI, PACS/LACS), and defines open, testable interfaces to avoid vendor lock-in.

6.1 Main Conclusions

The proposed architecture was validated through a proof-of-concept prototype, which comprised a CMS, a Java Card applet implementing a PIV-compatible subset of the data model and commands, and an access control system emulator. The prototype demonstrates end-to-end feasibility, as cards can be issued and used for one-factor (PKI-CAK) and two-factor (PKI-AUTH) authentication, utilizing standards-conformant APDUs and certificate processing. In doing so, it articulated enterprise-appropriate certificate profiles, exercised on-card key generation and non-exportable private key operations.

Prototype limitations included single-workstation deployment, a local self-signed root (public key certificate and private key) rather than an enterprise public-key infrastructure, no revocation checking, no contactless interface, no on-card biometric comparison, and a reduced data model omitting the Card Capability Container and the Security Object. These constraints do not undermine the central claim. The core issuance and authentication flows work as designed, and the system's interfaces and data structures align with widely deployed standards, positioning the solution for multi-vendor, multi-site interoperability.

6.2 Future Work

A clear path from prototype to production emerges from the implementation, tests, and results. Card applicant ingestion should transition from manual CSV import to real-time IDMS integration. Specifically, the CMS can implement the message schema introduced in Section 4.5 over a secure communication channel.

On the CMS side, a production build should introduce a PKI adapter that supports a standards-based enrollment protocol adopted by the deployed CA. Key management should be strengthened by assigning unique GlobalPlatform keys to each card, enforcing SCP03 as the default protocol, and storing administration keys in secure storage mechanisms such as a Hardware Security Module (HSM). Adding role-based administration with audit is another suggested feature. The CMS client application should integrate with printers/laminators for visual personalization, support photo capture, and even write the facial image to the card, thereby completely conforming to the proposed data model.

Support for temporary/visitor credentials is a natural extension.

For the card applet, completing the SP 800-73 data model is recommended. This includes adding optional biometrics to enable three-factor authentication and implementing support for the contactless interface, as discussed in Section 4.3. The applet should also expose the Card Capability Container (CCC) and maintain a Security Object to provide explicit capability metadata and integrity protection across covered data objects.

Access and audit logging should be prioritized for enhanced security. This could be achieved by emitting structured, tamper-evident logs for lifecycle actions and authentication outcomes, and providing dashboards for compliance and investigations. This feature is particularly important for operators of critical infrastructure.

Finally, broaden interoperability and deployment testing: exercise workstation logon (e.g., Windows Smart Card Logon) and remote access (e.g., VPN access), running tests with PIV-aware commercial readers and controllers, and following formal PIV/CIV conformance/interoperability test suites [59]. This work would help validate assumptions and reveal gaps in relation to the CIV and PIV specifications, while further strengthening the practical applicability of the proposed solution.

Bibliography

[1] C. Adams, S. Farrell, T. Kause, and T. Mononen. *Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)*. RFC 4210. Defines CMPv2; see also RFC 9480 for updates. Internet Engineering Task Force (IETF), 2005. URL: https://www.rfc-editor.org/rfc/rfc4210 (cit. on p. 73).

[2] Advanced Card Systems Ltd. *ACOSJ-G (Contact/Contactless/Combi) Functional Specification*. Version 2.09. Advanced Card Systems Ltd. 2023. URL: https://www.acs.com.hk/download-manual/7796/FSP-ACOSJ-G-2.09.pdf (cit. on p. 87).

[3] D. J. Bernstein. "Curve25519: New Diffie–Hellman Speed Records". In: *Public Key Cryptography – PKC 2006*. Ed. by M. Yung, Y. Dodis, A. Kiayias, and T. Malkin. Vol. 3958. Lecture Notes in Computer Science. Springer, 2006, pp. 207–228. DOI: 10.1007/11745853_14 (cit. on p. 13).

[4] Bouncy Castle Project. *Bouncy Castle Java Documentation*. n.d. URL: https://www.bouncycastle.org/documentation/documentation-java/ (cit. on pp. 86, 89).

[5] *Cards and security devices for personal identification — Contactless proximity objects — Part 1: Physical characteristics*. International Organization for Standardization (ISO/IEC) (cit. on p. 36).

[6] *Cards and security devices for personal identification — Contactless proximity objects — Part 2: Radio frequency power and signal interface*. International Organization for Standardization (ISO/IEC) (cit. on p. 36).

[7] *Cards and security devices for personal identification — Contactless proximity objects — Part 3: Initialization and anticollision*. International Organization for Standardization (ISO/IEC) (cit. on p. 36).

[8] *Cards and security devices for personal identification — Contactless proximity objects — Part 4: Transmission protocol*. International Organization for Standardization (ISO/IEC), 2018 (cit. on pp. 31, 36).

[9] H. Y. Chien. "An Efficient and Practical Solution to Remote Authentication". In: *Computers & Security* 21.4 (2002), pp. 372–375. ISSN: 0167-4048. DOI: 10.1016/S0167-4048(02)00415-7. URL: https://www.sciencedirect.com/science/article/abs/pii/S0167404802004157 (cit. on p. 1).

[10] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and T. Polk. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. Internet Engineering Task Force (IETF), 2008. URL: https://www.rfc-editor.org/rfc/rfc5280 (cit. on pp. 13, 18, 47, 48, 59).

[11] N. T. Courtois. "The Dark Side of Security by Obscurity — and Cloning MiFare Classic Rail and Building Passes, Anywhere, Anytime". In: *Proceedings of the International Conference on Security and Cryptography (SECRYPT 2009), ICETE*. 2009 (cit. on p. 53).

[12] Credelius Group. *Why I don't trust NIST P-256*. 2015. URL: https://credelius.com/credelius/?p=97 (cit. on p. 13).

[13] Cybersecurity and I. S. A. (CISA). *Critical Infrastructure Sectors*. URL: https://www.cisa.gov/topics/critical-infrastructure-security-and-resilience/critical-infrastructure-sectors (cit. on p. 1).

[14] Cybersecurity and I. S. A. (CISA). *Defining Insider Threats*. URL: https://www.cisa.gov/topics/physical-security/insider-threat-mitigation/defining-insider-threats (cit. on p. 1).

[15] Cybersecurity and Infrastructure Security Agency. *Cybersecurity and physical security convergence*. Publication No. 508-01.05.2021 [PDF]. 2021. URL: https://www.cisa.gov/sites/default/files/publications/Cybersecurity%20and%20Physical%20Security%20Convergence_508_01.05.2021.pdf (cit. on p. 1).

[16] *Digital Signature Standard (DSS)*. National Institute of Standards and Technology, 2023. DOI: 10.6028/NIST.FIPS.186-5. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf (cit. on p. 66).

[17] *Doc 9303: Machine Readable Travel Documents (MRTD), Eighth Edition — All Parts*. Series reference; e.g., Part 1 (Introduction), Part 4 (MRPs), Part 12 (PKI). International Civil Aviation Organization (ICAO), 2021. URL: https://www.icao.int/publications/doc-series/doc-9303 (cit. on p. 47).

[18] Z. Du and Y. Tang. *Web-based Multi-level Smart Card Access Control System on University Campus*. 2014. URL: https://ieeexplore.ieee.org/document/6933737 (cit. on pp. 53, 54).

[19] S. E. Eckert. "Protecting Critical Infrastructure: The Role of the Private Sector". In: (2006). URL: https://www.files.ethz.ch/isn/21268/Eckert.pdf (cit. on p. 2).

[20] *EMV Integrated Circuit Card Specifications for Payment Systems (Books 1–4)*. EMVCo (cit. on p. 36).

[21] H. Ferraiolo, D. Cooper, S. Francomacaro, A. Regenscheid, J. Mohler, S. Gupta, and W. Burr. *Guidelines for Derived Personal Identity Verification (PIV) Credentials*. NIST Special Publication 800-157. National Institute of Standards and Technology, 2014. URL: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-157.pdf (cit. on pp. 42, 44).

[22] H. Ferraiolo, N. Ghadiali, J. Mohler, V. Johnson, and S. Brady. *Guidelines for the Use of PIV Credentials in Facility Access*. NIST Special Publication 800-116 Rev. 1. National Institute of Standards and Technology, 2018. DOI: 10.6028/NIST.SP.800-116r1. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-116r1.pdf (cit. on p. 43).

[23] H. Ferraiolo, K. Mehta, S. Francomacaro, R. Chandramouli, and S. Gupta. *Interfaces for Personal Identity Verification (PIV): Part 2 — PIV Card Application Card Command Interface (Revision 5)*. NIST Special Publication 800-73-5, Part 2. National Institute of Standards and Technology, 2024. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-73pt2-5.pdf (cit. on pp. 61, 93).

[24] H. Ferraiolo and A. Regenscheid. *Cryptographic Algorithms and Key Sizes for Personal Identity Verification*. NIST Special Publication 800-78-5. National Institute of Standards and Technology, 2024. DOI: 10.6028/NIST.SP.800-78-5. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-78-5.pdf (cit. on pp. 42, 48, 66).

[25] H. Ferraiolo et al. *Interfaces for Personal Identity Verification (PIV): Part 1 — PIV Data Model (Revision 5)*. NIST Special Publication 800-73-5, Part 1. National Institute of Standards and Technology, 2024. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-73pt1-5.pdf (cit. on pp. 46, 61, 93, 95).

[26] K. Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*. 3rd ed. Chichester, UK: John Wiley & Sons, 2010. ISBN: 978-0-470-69506-7 (cit. on pp. 1, 23).

[27] F. D. Garcia, P. van Rossum, R. Verdult, and R. Wichers Schreur. "Wirelessly Pickpocketing a MIFARE Classic Card". In: *2009 30th IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2009, pp. 3–15. DOI: 10.1109/SP.2009.6 (cit. on p. 53).

[28] GlobalPlatform. *GlobalPlatform Card Specification*. Version 2.2.1. GlobalPlatform. 2018. URL: https://globalplatform.org/wp-content/uploads/2018/06/GPC_Specification-2.2.1.pdf (cit. on p. 87).

[29] *GlobalPlatform Card Specification*. Specification GPC_SPE_034. Version 2.3.1. Public Release. 2018. URL: https://globalplatform.org/wp-content/uploads/2018/05/GPC_CardSpecification_v2.3.1_PublicRelease_CC.pdf (cit. on p. 39).

[30] P. Grother, E. Tabassi, et al. *Biometric Specifications for Personal Identity Verification*. NIST Special Publication 800-76-2. National Institute of Standards and Technology, 2013. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-76-2.pdf (cit. on p. 42).

[31] I. (GSA). *FIPS 201 — Approved Product List (APL)*. 2025. URL: https://www.idmanagement.gov/fips201/ (cit. on p. 50).

[32] *HID® pivCLASS Readers*. HID Global. 2025. URL: https://www.hidglobal.com/product-mix/pivclass (cit. on p. 53).

[33] *HID® pivCLASS® Software*. HID Global. 2025. URL: https://www.hidglobal.com/product-mix/pivclass-software (cit. on p. 53).

[34] *HID® Seos®*. HID Global. 2025. URL: https://www.hidglobal.com/product-mix/seos (cit. on p. 52).

[35] T. W. House. *Homeland Security Presidential Directive 12: Policy for a Common Identification Standard for Federal Employees and Contractors*. 2004. URL: https://www.cac.mil/Portals/53/Documents/HSPD-12.pdf (cit. on p. 42).

[36] *Identification cards — Integrated circuit cards (ISO/IEC 7816 series, all parts)*. International Organization for Standardization (ISO/IEC) (cit. on p. 36).

[37] *Identification cards — Integrated circuit cards — Part 1: Cards with contacts — Physical characteristics*. International Organization for Standardization (ISO/IEC) (cit. on p. 35).

[38] *Identification cards — Integrated circuit cards — Part 2: Cards with contacts — Dimensions and location of the contacts*. International Organization for Standardization (ISO/IEC) (cit. on p. 35).

[39] *Identification cards — Integrated circuit cards — Part 3: Cards with contacts — Electrical interface and transmission protocols*. International Organization for Standardization (ISO/IEC), 2016 (cit. on pp. 31, 36).

[40] *Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange*. International Organization for Standardization (ISO/IEC) (cit. on p. 36).

[41] *Identification cards — Physical characteristics*. International Organization for Standardization (ISO/IEC) (cit. on pp. 4, 35).

[42] Infineon Technologies AG. *Card Applet for Electronic ID and Travel Documents*. Product brief. Infineon Technologies AG. 2017. URL: https://www.infineon.com/assets/row/public/documents/30/45/infineon-card-applet-for-electronic-id-and-travel-documents-pb-en.pdf (cit. on p. 87).

[43] *Information technology — Security techniques — Evaluation criteria for IT security (Common Criteria)*. International Organization for Standardization (ISO/IEC) (cit. on p. 36).

[44] *Interfaces for Personal Identity Verification (PIV), Revision 5 (Parts 1–3)*. NIST Special Publication 800-73-5. Suite includes Part 1 (Data Model), Part 2 (Card Command Interface), Part 3 (Client API). National Institute of Standards and Technology, 2024. URL: https://www.nist.gov/news-events/news/2024/07/personal-identity-verification-piv-interfaces-cryptographic-algorithms-and (cit. on pp. 42, 45, 48, 59).

[45] A. K. Jain, P. Flynn, and A. A. Ross, eds. *Handbook of Biometrics*. Springer, 2008 (cit. on p. 24).

[46] *Java Card Platform, Runtime Environment Specification, Classic Edition, Version 3.2*. Specifies the JCRE (VM lifecycle, applet model, memory, firewall, RMI, install/delete). Oracle. 2023. URL: https://docs.oracle.com/en/java/javacard/3.2/jc-re-spec/F74157_02.pdf (cit. on p. 38).

[47] *Java Card Platform, Virtual Machine Specification, Classic Edition, Version 3.2*. Defines the Java Card VM subset and CAP file format. Oracle. 2023. URL: https://docs.oracle.com/en/java/javacard/3.2/jc-vm-spec/F74158_02.pdf (cit. on p. 38).

[48] A. Kak. "Lecture 8: AES: The Advanced Encryption Standard". Lecture Notes on "Computer and Network Security- Purdue University. 2025. URL: https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf (cit. on p. 10).

[49] M. Kapusta and N. Lindstrom. *Access Control With High Security Credentials*. 2016. URL: https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=8820688&fileOId=8820693 (cit. on pp. 53, 54).

[50] G. de Koning Gans, J.-H. Hoepman, and F. D. Garcia. "A Practical Attack on the MIFARE Classic". In: *CARDIS 2008*. 2008 (cit. on p. 53).

[51] P. J. Leach, M. Mealling, and R. Salz. *A Universally Unique Identifier (UUID) URN Namespace*. RFC 4122. Obsoleted by RFC 9562 (2024). Internet Engineering Task Force (IETF), 2005. URL: https://datatracker.ietf.org/doc/html/rfc4122 (cit. on p. 51).

[52] J. F. C. Lopes. "Smart Card Power Analysis: From Theory To Practice". Dissertação de Mestrado. Lisboa, Portugal: Instituto Superior Técnico, Universidade de Lisboa, May 2017. URL: https://fenix.tecnico.ulisboa.pt/downloadFile/1689244997257556/Dissertacao.pdf (cit. on p. 37).

[53] B. Madhuravani, P. B. Reddy, and P. L. Reddy. "A Comprehensive Study on Different Authentication Factors". In: *International Journal of Engineering Research & Technology (IJERT)* 2.10 (2013). URL: https://www.ijert.org/research/a-comprehensive-study-on-different-authentication-factors-IJERTV2IS100472.pdf (cit. on p. 1).

[54] K. E. Mayes and K. Markantonakis. *Smart Cards, Tokens, Security and Applications*. London, UK: Springer, 2008. ISBN: 978-1-84882-015-4 (cit. on pp. 23, 24).

[55] J. McCarthy. *Identity and Access Management for Electric Utilities*. en. NIST Special Publication NIST SP 1800-2. Gaithersburg, MD: National Institute of Standards and Technology, July 2018. DOI: 10.6028/NIST.SP.1800-2. URL: https://doi.org/10.6028/NIST.SP.1800-2 (cit. on p. 2).

[56] *MIFARE DESFire EV3 | Secure Contactless IC*. NXP Semiconductors. 2025. URL: https://www.nxp.com/products/MF3DHx3 (cit. on p. 52).

[57] M. A. Mohandes. "A Smart Card Management and Application System". In: *2010 IEEE International Conference on Progress in Informatics and Computing (PIC)*. 2010 (cit. on pp. 53, 54).

[58] National Institute of Standards and Technology. *FIPS PUB 180-4: Secure Hash Standard (SHS)*. FIPS Publication 180-4. NIST, 2015 (cit. on p. 16).

[59] National Institute of Standards and Technology. *PIV Card Application and Middleware Interface Test Guidelines (SP 800-73-4 Compliance)*. Tech. rep. NIST Special Publication 800-85A-4. Supersedes NIST SP 800-85A-2 (2010). National Institute of Standards and Technology, 2016. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-85A-4.pdf (cit. on p. 108).

[60] National Institute of Standards and Technology. *NIST Retires SHA-1 Cryptographic Algorithm*. https://www.nist.gov/news-events/news/2022/12/nist-retires-sha-1-cryptographic-algorithm. 2022 (cit. on p. 16).

[61] National Institute of Standards and Technology (NIST). *Guidelines for the Use of PIV Credentials in Facility Access*. NIST Special Publication 800-116 Rev. 1. National Institute of Standards and Technology, 2018. URL: https://csrc.nist.gov/pubs/sp/800/116/r1/final (cit. on pp. 29, 49).

[62] National Institute of Standards and Technology (NIST). *Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters*. Tech. rep. NIST Special Publication 800-186. U.S. Department of Commerce, 2023. URL: https://doi.org/10.6028/NIST.SP.800-186 (cit. on p. 13).

[63] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, and Y. Koucheryavy. "Multi-Factor Authentication: A Survey". In: *Cryptography* 2.1 (2018), pp. 1–31. DOI: 10.3390/cryptography2010001. URL: https://www.mdpi.com/2410-387X/2/1/1 (cit. on p. 1).

[64] *OnGuard: Physical Access Control System*. LenelS2 (Honeywell Building Technologies). 2025. URL: https://buildings.honeywell.com/us/en/brands/our-brands/lenels2/security-products/onguard (cit. on p. 53).

[65] OpenJFX Project. *OpenJFX*. n.d. URL: https://openjfx.io/ (visited on 12/2025) (cit. on p. 86).

[66] OpenSSL Software Foundation. *OpenSSL 3.2 Documentation: Introduction*. n.d. URL: https://docs.openssl.org/3.2/man7/ossl-guide-introduction/ (visited on 12/2025) (cit. on p. 86).

[67] Oracle Corporation. *Java Card Specifications*. n.d. URL: https://www.oracle.com/java/technologies/javacard-specs-downloads.html (visited on 12/2025) (cit. on p. 87).

[68] M. Paljak. *apdu4j*. GitHub repository. n.d. URL: https://github.com/martinpaljak/apdu4j (cit. on p. 86).

[69] M. Paljak. *GlobalPlatformPro*. GitHub repository. n.d. URL: https://github.com/martinpaljak/GlobalPlatformPro (cit. on p. 86).

[70] M. Pato. *The ISELthesis LaTeX Template's Manual*. Instituto Superior de Engenharia de Lisboa (ISEL-IPL). 2024. URL: https://github.com/matpato/iselthesis (cit. on p. viii).

[71] PC/SC Workgroup. *PC/SC Specifications: Interoperability Specification for ICCs and Personal Computer Systems*. URL: https://pcscworkgroup.com/specifications/ (cit. on p. 36).

[72] *Personal Identity Verification (PIV) in Enterprise Physical Access Control Systems (E-PACS)*. Guidance. Version 3.0. See Sec. 1.1 for definitions. Federal Identity, Credential, and Access Management (FICAM) Program, GSA, Mar. 26, 2014. URL: https://www.idmanagement.gov/docs/pacs-piv-epacs.pdf (cit. on p. 50).

[73] *Personal Identity Verification (PIV) of Federal Employees and Contractors*. National Institute of Standards and Technology, 2022. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.201-3.pdf (cit. on pp. 42, 45–47).

[74] PostgreSQL Global Development Group. *PostgreSQL*. n.d. URL: https://www.postgresql.org/ (visited on 12/2025) (cit. on p. 86).

[75] W. Rankl and W. Effing. *Smart Card Handbook*. 4th ed. Wiley, 2010 (cit. on pp. 1, 36).

[76] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. Internet Engineering Task Force (IETF), 2018. URL: https://datatracker.ietf.org/doc/rfc8446/ (cit. on p. 13).

[77] R. L. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Communications of the ACM* 21.2 (1978), pp. 120–126. DOI: 10.1145/359340.359342 (cit. on p. 12).

[78] J. Schaad and M. Myers. *Certificate Management over CMS (CMC)*. RFC 5272. Internet Engineering Task Force (IETF), 2008. URL: https://www.rfc-editor.org/rfc/rfc5272 (cit. on p. 73).

[79] T. T. Schwarzhoff, J. Wack, et al. *Government Smart Card Interoperability Specification (GSC-IS), Version 2.1 (E2003)*. NIST Interagency Report NISTIR 6887. National Institute of Standards and Technology, 2003. URL: https://nvlpubs.nist.gov/nistpubs/legacy/ir/nistir6887e2003.pdf (cit. on p. 46).

[80] R. R. Shirey. *Internet Security Glossary*. RFC 2828. Internet Engineering Task Force (IETF), 2000. URL: https://www.rfc-editor.org/rfc/rfc2828 (cit. on p. 25).

[81] S. Skorobogatov. "How Microprobing Can Attack Encrypted Memory". In: *2017 Euromicro Conference on Digital System Design (DSD)*. 2017, pp. 244–251. DOI: 10.1109/DSD.2017.69 (cit. on p. 37).

[82] W. Stallings. "Introduction to cryptography". In: *Cryptography and Network Security: Principles and Practice*. 8th ed. Global Edition: Pearson, 2017, p. 33 (cit. on p. 7).

[83] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov. "The First Collision for Full SHA-1". In: *Lecture Notes in Computer Science*. Springer, 2017 (cit. on p. 16).

[84] STMicroelectronics. *STeID-Java: Secure Java Card Solutions for e-Identity and e-Government*. Data brief DB5287. STMicroelectronics. 2024. URL: https://www.st.com/resource/en/data_brief/steid-java.pdf (cit. on p. 87).

[85] *Synergis Access Control System*. Genetec Inc. 2025. URL: https://www.genetec.com/products/unified-security/synergis (cit. on p. 53).

[86] S. S. Team. *New minimum RSA key size for code signing certificates*. Accessed: December 2025. 2021. URL: https://www.ssl.com/blogs/new-minimum-rsa-key-size-for-code-signing-certificates/ (cit. on p. 12).

[87] *Technical Implementation Guidance: Smart Card Enabled Physical Access Control Systems (TIG SCEPACS)*. Technical Guidance. Practical guidance for PACS media, readers, and databases. Federal Identity, Credential, and Access Management (FICAM) / IDManagement.gov, 2005. URL: https://www.idmanagement.gov/docs/pacs-tig-scepacs.pdf (cit. on pp. 44, 46).

[88] R. Thayer. *Unified physical and logical access using industry standards and protocols*. Accessed: 17 de dezembro de 2025. SecurityInfoWatch. 2019. URL: https://www.securityinfowatch.com/access-identity/article/21069112/unified-physical-and-logical-access-using-industry-standards-and-protocols (cit. on p. 1).

[89] *The Commercial Identity Verification (CIV) Credential — Leveraging FIPS 201 and the PIV Specifications: Is the CIV Credential Right for You?* Smart Card Alliance Physical Access Council, Oct. 2011. URL: https://www.securetechalliance.org/resources/pdf/CIV_WP_101611.pdf (cit. on p. 50).

[90] *USB Device Class Specification for Smart Card CCID*. USB Implementers Forum (USB-IF), 2005 (cit. on p. 31).

[91] X. Wang, Y. L. Yin, and H. Yu. "Finding Collisions in the Full SHA-1". In: *Advances in Cryptology – CRYPTO 2005*. Vol. 3621. Lecture Notes in Computer Science. Springer, 2005, pp. 17–36 (cit. on p. 16).

@is@a@figure