Міністерство освіти і науки України
Український державний університет науки і технологій

Комп'ютерні технології і системи
(назва факультету)

Електронні Обчислювальні Машини
(повна назва кафедри)

Пояснювальна записка
до кваліфікаційної роботи
магістра
(ступінь вищої освіти)

Удосконалення механізмів інформаційного контролю та управління навчальним процесом університету.

на тему: Дослідження та розробка центру сертифікації ключів для університету

за освітньою програмою Комп'ютерна інженерія

зі спеціальності: 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

Виконав: студент Групи: КС2121

(підпис студента) Вікторія КУЛИК (Ім'я ПРІЗВИЩЕ)

Керівник: (підпис) професор, Ігор ЖУКОВИЦЬКИЙ (посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: (підпис) доцент Володимир ШАПОВАЛОВ (посада, Ім'я ПРІЗВИЩЕ)

Консультанти:

Використання власного сертифікатного центру (назва розділу)	(підпис)	старший викладач, Олексій ЗАЄЦЬ (посада, Ім'я ПРІЗВИЩЕ)
Розгортання центру сертифікату в університетському середовищі (назва розділу)	(підпис)	старший викладач, Олексій ЗАЄЦЬ (посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент (підпис)

Дніпро – 2022 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Computer technologies and systems
(faculty)

Electronic Computing Machines
(department)

Explanatory Note
to Master's Thesis
master

Improvement of mechanisms of information control and management of the educational process of the university.

on the topic: Research and development of a certificate authority for the university

according to educational curriculum Computer Engineering

in the Speciality: 123 Computer Engineering
(speciality and its code)

Done by the student of the group: KC2121 Viktoriia Kulyk
(name, surname)

Scientific Supervisor: professor, Ihor Zhukovytskyi
(position, name, surname)

Normative controller : associate professor Volodymyr Shapovalov
(position, name, surname)

Supervisors:

Implementing own certificate authority	senior teacher, Oleksiy Zaets
(Chapter title heading)	(position, name, surname)
Deployment of certificate authority into university environment	senior teacher, Oleksiy Zayets
(Chapter title heading)	(position, name, surname)

Dnipro – 2022

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Комп'ютерні технології і системи
Кафедра: Електронні Обчислювальні Машини
Рівень вищої освіти: магістр
Освітня програма: Комп'ютерна інженерія
Спеціальність: 123 Комп'ютерна інженерія
(шифр та назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри проф.
_____ Ігор ЖУКОВИЦЬКИЙ
(підпис) (Ім'я ПРІЗВИЩЕ)
Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу магістра
студенту Кулик Вікторії Антонівні
Удосконалення механізмів інформаційного контролю та управління навчальним процесом університету.
1. Тема роботи: Дослідження та розробка центру сертифікації ключів для університету
Керівник роботи: Жуковицький Ігор Володимирович, професор
(Прізвище, Ім'я, По батькові, науковий ступінь, вчене звання)
затверджені наказом від "___" ____ 202_ р. № ____

2. Строк подання студентом роботи: 21.12.2022 р.
3. Вихідні дані до роботи:
3.1 Протоколи та вимоги для реалізації центру сертифікації ключів
3.2 Технології реалізації центру сертифікації ключів
3.3 Організація дистанційного процесу навчання в університеті на сьогоднішній день
4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):
4.1 Аналітична частина:
4.1.1Огляд інфраструктури відкритого ключа (PKI)
4.1.2 Основні складові систем PKI: стандарти, протоколи та вимоги
4.1.3 Центр сертифікації як компонент PKI
4.2 Основна частина:
4.2.1 Використання власного сертифікатного центру
4.2.2 Розгортання центру сертифікату в університетському середовищі

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Завдання видав: (підпис консультанта, дата)	Завдання прийняв: (підпис студента, дата)
Використання власного сертифікатного центру	Заєць О. П., старший викладач	02.11.2022	02.11.2022
Розгортання центру сертифікату в університетському середовищі	Заєць О. П., старший викладач	02.11.2022	02.11.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вступ	10.10.2022	
2	Огляд інфраструктури відкритого ключа (PKI)	15.10.2022	
3	Основні складові систем PKI: стандарти, протоколи та вимоги	02.11.2022	
4	Центр сертифікації як компонент PKI	09.11.2022	
5	Використання власного сертифікатного центру	18.11.2022	
6	Розгортання центру сертифікату в університетському середовищі	04.12.2022	
7	Висновки	08.12.2022	
8	Оформлення роботи, підготовка демонстраційних матеріалів та доповіді	20.12.2022	
9	Подання кваліфікаційної роботи до кафедри	26.12.2022	
10	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	27.12.2022	

Студент _____ Вікторія КУЛИК
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ Ігор ЖУКОВИЦЬКИЙ
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра:

78с., 27 рис., 2 табл., 2 додатки, 26 джерел.

Об’єктом дослідження є розробка Центру сертифікації ключів як основного компоненту PKI системи, що може бути впроваджена в освітній процесс в університеті для управління електронним документообігом.

Метою дослідження є розробка прототипу основного компонента PKI – Центру сертифікації ключів для впровадження механізму сертифікованого цифрового підпису в навчальний процес в університеті.

Після детального дослідження різних архітектур, дизайнів, протоколів PKI та вимог, розроблено першу версію Центру сертифікації ключів.

На основі виконаних досліджень також було розроблено PKI клієнт, що контролює видання цифрових підписів для електронного документообігу. Створений проект також містить бібліотеки, які надають готові до використання функції для роботи з протоколами PKI та створення сертифікованих цифрових підписів.

Ключові слова: ІНФРАСТРУКТУРА ВІДКРИТИХ КЛЮЧІВ, PKI, ЦЕНТР СЕРТИФІКАЦІЇ КЛЮЧІВ, ЦСК, CMP, PKCS, СЕРТИФІКАТ, X.509, ВІДКРИТИЙ КЛЮЧ, ПРИВАТНИЙ КЛЮЧ, ЦИФРОВИЙ ПІДПИС, BOUNCY CASTLE, JAVA SECURITY, ПРОГРАМНИЙ ПРОДУКТ.

ABSTRACT

Explanatory note to the master's qualification project:

78 p., 27 fig., 2 tables., 26 sources, 2 appendixes.

The research subject is the development of Certificate Authority as a part of PKI system that is integrated to the university to provide electronic document management.

The goal of the project is to provide a first steps, particularly, implement the sample version of main PKI component – Certificate Authority to integrate certified digital signatures mechanism into an education process in university envoronment.

After detailed research of different architectures, designs, PKI protocols and requirements, the prototype of Certificate Authority has been developed. Based on performed investigations the sample of the CA client that services digital signatures for electronic documents has been also implemented. The project also contains several libraries that provide ready-to-use functions to work with PKI protocols and create verified digital signatures.

Keywords: PUBLIC KEY INFRACTRUCTURE, PKI, CERTIFICATE AUTHORITY, CA, CMP, PKCS, CERTIFICATE, X.509, PUBLIC KEY, PRIVATE KEY, DIGITAL SIGNATURE, BOUNCY CASTLE, JAVA SECURITY, SOFTWARE PRODUCT.

CONTENTS

LIST OF SYMBOLS

CA – Certificate Authority, main PKI component;

CAA – Certificate Authority Authorization;

CAA RR – Certification Authority Authorization resource record;

CMP – Certificate Management Protocol;

CP – Certificate policy;

CRL – Certificate revocation list, collection of revoked certificates;

CT – Certificate Transparency protocol;

EE – End Entity, client of issuing Certificate Authority;

FIPS – Federal Information Processing Standards;

OCSP – Online Certificate Status Protocol, that is used to get data about revoked certificates;

PEM – Privacy-Enhanced Mail;

PKI – Public Key Infractructure;

POP – Proof Of Possession, field in PKIMessage in Certificate Management Protocol;

RA – Registration Authority, optional PKI component;

INTRODUCTION

To mitigate the security risks of conducting business or managing an organization in an open environment while at the same time maintaining the cost advantages of doing so, enterprises are turning their attention to an emerging segment of the security market known as Public Key Infrastructure (PKI).

One of the most interesting security mechanisms based on public key cryptography is the generation of digital signatures. Nowadays in Ukraine the appropriately certified digital signature is equal to real one and legislatively accepted. That allows it to use it for electronic document management in company internally as well as between organizations on a legal basis. This can be an attractive feature for the education sector as a part of the remote education process.

In this project we represent the development of the key component in the PKI system - Certificate Authority. The implemented authority and related PKI elements can be used as a sample version and basis for further development in complex electronic document management systems in university. The prepared cryptographic and PKI-standard libraries can be used to simplify the realization of a custom PKI environment.

The project is organized into 6 sections that provide theoretical and practical basis in PKI technology and Certificate Authority particularly.

In section 1 we introduce the public key infrastructure, main components, required services and functions. Besides, the common security problems and weak parts are mentioned in this chapter.

In section 2 we give a detailed description of PKI protocols and standards. The main attention is given to the protocols used for communication between CA and other PKI components.

In section 3 the Certificate Authority is described in detail, including architecture types and required functionality. The key and certificate management definitions are also given.

In section 4 we move to the practical part: creating design and architecture of our system, choosing appropriate technology stack for the development.

In section 5 we show our model in work and describe the integration and usage of our project in experimental mode in the university.â€¦

1 OVERVIEW OF PUBLIC KEY INFRACTRUCTURE

1.1 PKI system components

Public Key Infrastructure is a complex system that consists of hardware, software, policies, and procedures that are required to manage, create, store, and distribute keys and certificates. To provide all this functionality, there are various components of PKI:

a) Certificate Authority

Usage of public keys as reliable security mechanism requires additional verification that public key is corresponds to certain private key and was not substituted. Without this protection, there is no guarantee that the sender of encrypted or signed data was not impersonated and the recipient of this data is definitely the targeted recipient.

Therefore, we need a third trusted entity that will provide such verification and all clients should rely on this entity as a verification center. This verification center gives a guarantee that a check on the key pair and its owner has been performed and the ‘document’ for this check is formatted as a certificate. Thus, the ‘verification center’ is called a Certificate Authority (CA).

To issue a certificate, CA signs it with his digital signature. CA is known to clients by his two attributes: the name and the public key. The CA includes his name in every certificate and revocation list (CRL) it issues and signs them with his private key.

By issuing a certificate, CA confirms that the owner of the issued certificate holds the private key that corresponds to the public key added to this certificate.

The CA functionality is protocolled and limited by Certificate Policy (CP) that determines the certificate content and extra security fields etc. By reviewing the CP and deciding that they trust a CA and its business operations, users can rely on certificates issued by that CA [6].

b) Registration Authority

Registration Authority (RA) is an optional component of PKI. RA acts between clients and CAs as verifier and registration point of clients' data that is supposed to be in the certificate.

Actually, the Registration Authority as a service can be a part of Certificate Authority itself and do not exist as separate PKI component, but in complex systems with significant amount of clients from different sides and big traffic of requests to CA, Registration Authority takes a responsibility from CA to process client information, prepare the required data for CA and, in general, to verify the client before request will be forwarded to CA.

The following tasks are provided by RA:

— receive and validate entity requests;

— forward checked requests to CA;

— receive an issued certificate from CA;

— send a certificate to the correct entity.

There can be multiple RAs used for one CA. In this case CA maintains a list of accredited Registration Authorities that are determined as reliable for this system.

The RA also must be verified by CA, so CA issues a certificate for RA. RA, in its turn, must provide sufficient security to protect its private key. When verifying the RA's signature on a message or document, the CA relies on the reliability of the information provided by the RA.

c) Certificate / CRL repository

Certificate / CRL repository is a special storage where issued and revoked certificates are stored.

The following requirements are applied to the repository:

— The certificate repository should provide trouble proof access to it’s services and to be hosted on high-capacity platform;

— The repository should be sufficiently protected;

— The repository must support the common protocols defined in PKI to access the data.

The repository is usually hosted in a directory system based on X.500 standard or Lightweight Directory Access Protocol (LDAP) server [6].

d) Certificate archive

Certificate archive is used for long-term storage of issued and revoked certificates with an ability for recovery.

Archived certificates can be used to provide evidence that any document, that has been signed by digital signature with verification of this certificate, was originally asserted with this digital signature in the past.

An archive can be either part of the local network of the system within the organization or implemented as an external service accessible via the Internet.

The certificate archive must follow the rules on how to store certificates according to Certificate Policy. If there is more than one archive in the system, the time, data type and processes may vary from one to another storage [2].

e) PKI clients

PKI clients (other name is End Entity – EE) are divided into two categories: certificate holders and relying parties. Certificate holder is an individual, legal entity or application service that possesses an issued certificate. Relying parties request and rely on information about the status of certificates [6].

1.2 PKI system: services and its functions

Having described the main components of Public Key Infrastructure, the next step is to give details about its services and required functionality. PKI system must provide the following services:

a) Cryptographic services. These services implement the next functions:

1) Generation of key pairs. This includes generating, secure storing, performing cryptographic computations, distribution and replication (if needed) of public/private key pairs for CA, RA or end entities.

2) Digital signature processing. The services are responsible for creation and / or verification of digital signature, thereby, assure the authenticity of the message and corresponding digital signature.

b) Certificate management services. These core PKI services provide:

1) Issuance of a certificate. CA can issue a certificate for users (either individual or legal entity or application server), for other CAs (certify authorities that stay at lower trust level or use cross-certification) and other components of the PKI such as Registration Authority.

2) Lifecycle management of certificates and keys. The PKI system must handle the issued certificates and generated keys and monitor / update its status.

   For example, when a new certificate is issued, it needs to be published in Certificate Repository. Before its expiration date, the certificate can be reissued (updated) in order to prolongate its valid status. In case of compromise or other unexpected failure, the certificate may be revoked and should be added to the CRL repository. After expiration, the certificate may also be archived.

3) Repository maintenance. As was already mentioned in the previous paragraph, when a certificate is issued or revoked, it must be saved in Certificate or CRL repository respectively. Typically, the repository is controlled by a CA, sometimes by a third party. Access to the registry may be restricted.

4) Certificate's history and archive. Issued certificates and certificate revocation lists usually need to be stored for a long time after its expiration or revocation. For this purpose, the archive storage service should be provided in the PKI system. The same is related to key pairs.

c) Auxiliary services. Except of the minimum set of required functions, PKI can manage the following aspects:

1) Registration and Authorization. The PKI system should verify clients before issuing a certificate. The general PKI design allows users to use various options from simple to complex to register and authorize the client as trusted.

2) Notarial authentication. As a trusted part, CA can be certified and have legal effect in e-commerce enterprise environments according to the laws of the country where this system is used.

3) Non-repudiation. PKI also manages digital signatures that can be used in terms of real 'signature' on the document, providing the evidence that this document has been signed or this agreement has been dealt [6].

1.3 PKI risks and weak parts in scope of Certificate Authority

Speaking about PKI organization and services we must also mention the parts that can cause critical vulnerabilities in the system without sufficient security providing.

1.3.1 Attacks on Cetificate Authority

Certificate authority is the main component in PKI system and must be secured appropriately.

Certificate Authority (CA) systems may be the subject of attacks through external or internal ways (inside organization).

External attacks on the CA attempt to steal private keys using computers located outside the physical CA system environment.

Internal attacks are mainly based on illegal data usage by workers in the organization. CA private keys are an attractive target to those who work with them. Strict access control, electronic monitoring, and intruder-detection on each device should deter even the most tenacious would-be thieves.

The compromization of the root CA’s private key leads to multiple system problems and PKI services instability. Thus, the root private key must be protected by the best technologies and practices within the cryptographic community [1].

1.3.2 Root CA key protection. FIPS

To protect the root CA key, we should refer to FIPS 140-1 Level 3 Cryptographic Module.

FIPS are standards and guidelines for federal computer systems that are developed by National Institute of Standards and Technology (NIST) in accordance with the Federal Information Security Management Act (FISMA) and approved by the Secretary of Commerce [11].

The main security product manufacturers propose to manage root key-pair (generation, verification, storage and deletion) via hardware device rather than residing in software on a host computer. This approach prevents easy data compromisation and provides fast backup in case of disaster.

In order to provide general protocol for the security module in such cases, the FIPS 140-1 standards were introduced. FIPS is divided into four levels: from 1 to 4, from lowest security requirements to highest respectively.

Devices capable with FIPS 140-1 Level 3 standard or higher are required to provide cryptographic key generation and transport in hardware. This standard has multiple requirements, such as key backup mechanism, independent channel authentication methods etc.,that must be provided by device to meet the demands [1].

1.3.3 The risk of ‘trust’

As PKI is an integral part of e-commerce systems and deals with economic and legal aspects, there is a good point to determine the risks of the Public Key Infrastructure. And here we point to the main Certificate Authority risk: is the CA in this system really trusted and how it can be guaranteed? [7]

There are a large number of CAs that have the trust of our relying party out there on the Internet. And all of these CAs can issue a certificate for any valid domain name. Thus, it is possible to issue a certificate for some targeted domain without informing the owner of this domain. This certificate can be used for man-in-the-middle attacks because the relying party would trust this certificate since it was signed by a trusted CA.

The possible mitigations in this situation are the following:

a) Certificate Authority Authorization (CAA) DNS Resource Record

CAA Resource Record (CAA RR) allows a DNS domain name holder to set a restricted list of Certificate Authorities that can issue a certificate for this domain name. Here is one issue that must be handled: the certificate has it's expire date and if the certificate A had been issued at one time and then the CAA record list has been updated and domain is set in certificate A is no longer in this list, the certificate A will be no longer conformant with the current CAA records despite it was conformant with the previous version.

CAA records provide quite an efficient and simple way to solve the mis-issue problem. The cost of implementing CAA checks is very small, and the potential costs of a mis-issue event include the removal of an embedded truss anchor [10].

b) Certificate Transparency (CT)

Certificate transparency aims to resolve the problem of misissued certificates by providing publicly auditable logs of all issued certificates. This allows anyone to monitor certificate authority (CA) activity and notice the issuance of suspect certificates. Those clients who are concerned about misissue can monitor the logs, asking them regularly for all new entries, and can thus check whether domains they are responsible for have had certificates issued that they did not expect.

The log entry contains a certificate chain and can be audited by every CA client. While creating a new log entry with a certificate chain, the timestamp is also generated as a unique ID of this entry, which can be used later as evidence for clients that the chain has been logged. [7] [24]

1.4 Summary

In this chapter we have described the basics of Public Key Infrastructure: components, provided services and functionality.

The main component of PKI system is Certificate Authority that manages certificate lifecycle: issuance, reissuance, revocation and archival. CA is the core component of PKI and a trusted point that can be assigned as 'trusted' not only inside an organization or some local system, but also can be certified and have legal effect, thereby providing a legislative basis for digitally signed documents.

CA also manages and communicates with other PKI elements such as Certificate / CRL repository that provide storage for issued and revoked certificates, Archive Repository that saves certificates and generated keys for a long time period and handles an ability to recover it.

The PKI clients request the CA for new certificates and access repositories for certificate statuses. The responsibility of CA is to verify the client before issuing a certificate. Depending on the design, traffic and complexity of the PKI system, clients' requests can be routed to an additional security check component - Registration Authority that takes the task of client authentication, registration and verification from Certificate Authority and filters all invalid requests.

As for assignee of 'trust', the required security mechanisms must be provided for the Certificate Authority. The client entities rely on CA as on trust point. Besides, there are several protocols that provide protection from man-in-the-middle attacks.

2 PKI SYSTEM ORCHESTRATION:

STANDARDS, PROTOCOLS AND REQUIREMENTS

2.1 The X Series Standards. X.509

Any PKI system is required to follow the globally defined protocols and standards. These standards are applied to PKI components' implementation, certificates, key pairs, digital signatures and so on. The protocols, in their turn, are mandatory to follow in data exchange between PKI services, providing functions, security and other flows in the PKI environment.

The X series standards are the set of classified rules that define protocols, information, and authentication and have been proposed by International Telecommunication Union (ITU) and International Standards Organization (ISO). The X standards' list is wide but here we mention one that is the most used in PKI - X.509 standard. [4]

X.509 standard determines structure and data that can go into PKI certificates and various standards for certificates and CRLs. X.509 certificates authenticate the identity of a person by using cryptographic mechanisms and store information and privilege about entities in the form of attributes in the certificate.

The X.509 standard supports the following certificate versions:

a) X.509 version 1. The first version includes seven fields in certificate:

1) Certificate Version Number. The numeration starts from zero, so for version 1 this value equals 0, for 2nd — 1, for 3rd — 2.

2) Certificate Serial Number. This is a unique identifier among all certificates issued by a Trust Center.

3) OID of the signature method with which the certificate is signed.

4) Name (in X.500 format) of the CA that has signed the certificate — issuer name.

5) Name (in X.500 format) of the certificate holder — subject name.

6) Public key of the certificate holder. The public key is signed by a verified CA certificate that approves the owner and his key pair.

7) Validity period of the certificate.

b) X.509 version 2. The basic set of fields in version 1 had to be extended, therefore the second version of X.509 has been introduced. In this version two more fields have been added to the certificate. These two fields are optional and rarely used in certificates:

1) A unique identifier of the certificate holder (certificate owner).

2) A unique identifier of the CA. By this identifier one CA should be distinguished from other CAs.

c) X.509 version 3. The third version of the X.509 provides extension fields. The extension can be critical or non-critical, which is explisitly added to this field in certificate. If the value is set as critical but not supported by consuming service, the certificate must be resolved as invalid. From other hand, if the extension is non-critical it will be just ignored by the software.

The common extensions are: Authority key identifier, Key holder identifier, Purpose of the key, Private key usage period, Certificate policy, Additional holder name, Additional CA name, Name constraints, Policy constraints, CRL distribution points etc.

The key advantage of the third version is that by using field type as 'extension' we can wrap any field value inside, so there is no more necessity to define the separate field each time and extend the structure of the certificate and adapt the software to work with a new version of certificates. [17]

2.2 The basis: Certificate Management Protocol and PKCS#<X> standards

Management protocols assign the format and algorithm of data exchange between PKI services. For example, Certificate authority must interact with End Entities, Registration Authority or with other CA. The most commonly used PKI management protocols are PKCS#10, PKCS#7, Certificate Management Protocol (CMP), Certificate Management using CMS (CMC), and Simple Certificate Enrollment Protocol (SCEP).

Firstly, we may describe the Public Key Cryptography Standards (PKCS), namely PKCS#10 and PKCS#7.

2.2.1 Certificate request in PKCS#10 format

PKCS#10 — Certificate Request Syntax Standard, defines the syntax for certificate requests. In PKCS#10, certificate request has the following parts:

- certification request information, that, on its turn, consists of subject distinguished name, subject public key, and optional attributes;
- digital signature value. The certification request information data is signed by the entity's private key;
- signature algorithm identifier — the ASN.1 algorithm identifier of the signature mentioned previously.

The Figure 2.1 shows the general set of fields in PKCS: distinguished Name, public Keys amd other attributes contained in Certificate request info field, plus mandatory algorithm identifier name of digital signature and the digital signature value. The Figure 2.2 represents the structure of the full PKCS#10 request format.

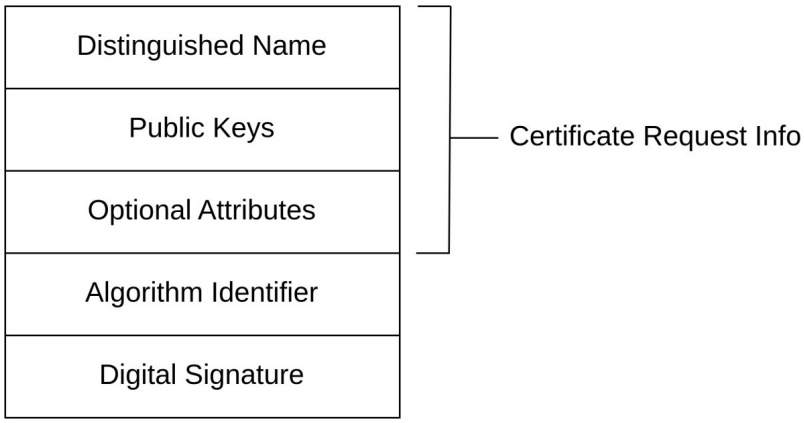

Figure 2.1 – Certificate request general scheme — PKCS#10

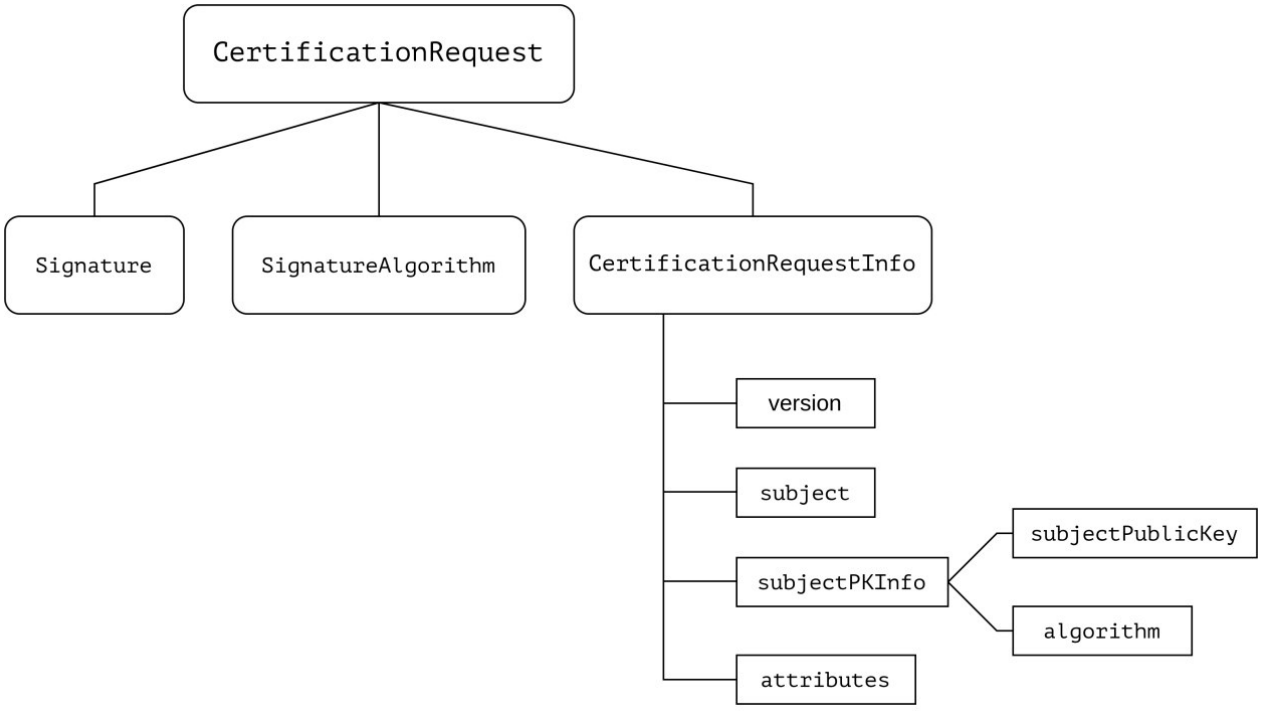

Figure 2.2 – Certificate request format — PKCS#10

The PKCS#10 request formats in the following steps:

- Construct CertificationRequestInfo with correcponding fields (subject name, subject public key, and attributes if needed).
- The fullfiled CertificationRequestInfo is signed with the subject's private key.
- The CertificationRequestInfo, signature and signature algorithm must be set and complete the top CertificationRequest.

When the CA receives the PKCS#10 CertificationRequest, CA (namely the RA service) authenticates the entity, verifies the entity's signature and, if the request is valid, constructs an X.509 certificate.

Usually, on PKCS#10 request, CA responds with PKCS#7 response. Other option is to publish new certificate into the repository that is accessible for PKI users. [16]

It is important to mention the limitations of the PKCS#10 due to which this protocol is less widespread nowadays and considered to be used to provide compatibility to old systems. A few limitations of PKCS#10 are:

- PKCS#10 is not algorithm-independent. The HTTP protocol is unsupported in PKCS#10 due to the lack of security (CA is unable to authenticate requester). Thus, the appropriate variant for PKCS#10 is, for example, SSL.
- In PKCS#10, the given data in request allows CA to authenticate the entity relaying only on digital signature that is ususally not enough for security requirements.
- There is also no confidence that a request has not been altered during its transit.

Finally, PKCS#10 defines the syntax for only for certificate request (to issue a new certificate), but not for other PKI functions, like request on certificate revocation or update and so on. [4]

2.2.2 Certificate response in PKCS#7 format

The PKCS#7 or Cryptographic Message Syntax Standard defines the syntax for cryptographic data, such as digital signatures. In the scope of PKI, this standard is used for CA response on certificate requests. The Figure 2.3 represents the general structure of PKCS#7. Hence, PKCS#7 format named ContentInfo consists of two fields: Content and its format identifier - ContentType. The ContentType is defined as one of six types that are shown in Figure 2.3.

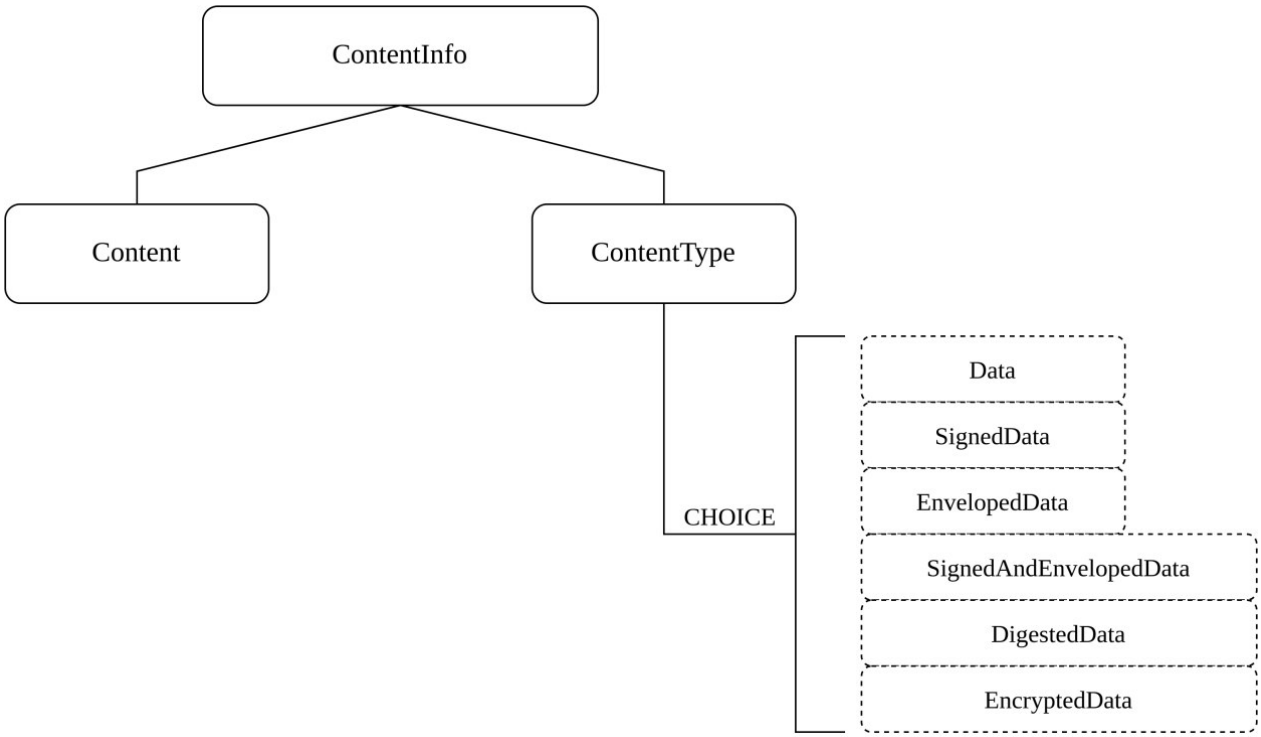

Figure 2.3 – Cryptographic Message Syntax format — PKCS#7

Here we describe more about SignedData as the most commonly used type. The content with SignedData type can contain any content format that is digested and signed by zero or more signers. Any type of content can be signed by any number of signers in parallel. There is one exceptional case: if there are no signers (zero), the ContentInfo value is not signed and considered to be irrelevant.

The process of SignedData construction involves the following steps:

a) The message digest is generated for each signer using a signer-specific message-digest algorithm. If there are more than one signer with the same algorithm, them any one of them is applied.

b) The message digest is encrypted by each one of signers by the providing to signer private key.

c) The encrypted message digests and other additional data are collected into a SignerInfo (the field in SignedData value)

d) The message-digest algorithms for all the signers and the SignerInfo values for all the signers are collected together with the content into a SignedData.

A recipient receives ContentInfo object and verifies the signatures by the signer's public key. After that the comparison is performed between recovered message digest and locally computed another one.

Combining PKCS#7 with PKCS#10 results in the formation of a complete and secure data exchange model in PKI system. However, both of them do not define any format for dealing with revoked certificates. [4] [15]

2.2.3 Certificate Management Protocol – CMP

2.2.3.1 CMP PKI operations

Certificate Management Protocol or CMP provides data exchange formats between different PKI entities. Unlike previosly described PKCS standards, CMP occupies a complete set of PKI management operations. The general scheme of these operations is shown in Figure 2.4.

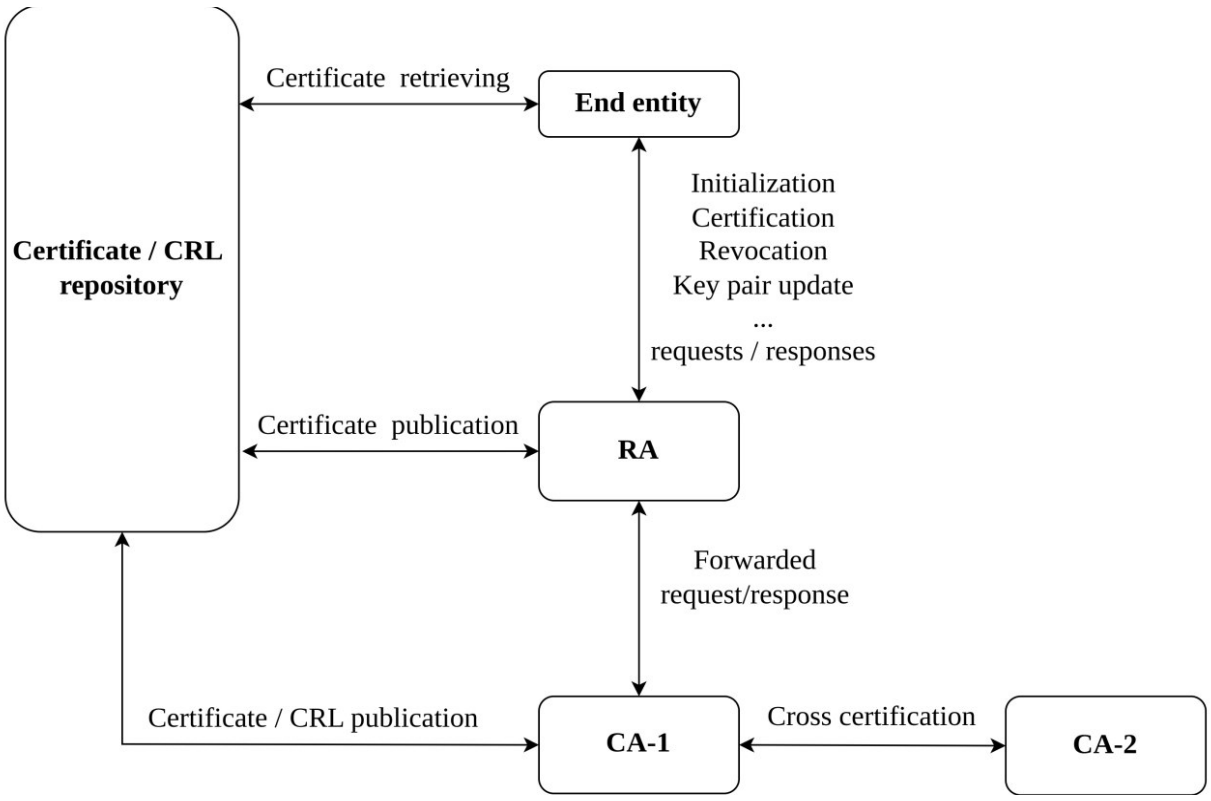

Figure 2.4 – CMP supported management operations

Certificate Management Protocol maintains the following PKI operations:

a) CA establishment. When establishing a new CA, certain steps are required to be done: generation of CA key pair, export of CA public key, creation of upper certificate for this CA, production of initial CRLs etc.

a) End-Entity Initialization. EE obtains root CA public key and retrives details about where PKI management entities are located and what services and options they provide.

b) Certification operations.

Initial registration and certification. In this process, the EE performs a request for a certificate and authenticates itself in RA or CA. The created certificate is returned to the entity and published to the repository. To this operation we can also regard a creation of EE's key pair.

Key pair update. End entity must update its key pair within defined periodicity. When the key pair has been generated, the new certificate must be also created.

Certificate update. The certificate must be recreated after its expiration period. The new certificate contains the same EE's info therefore this operation is defined as 'update'.

CA key pair update. The certificate of Certificate Authority itself also must be updated. In this case, CA must notify clients with a new public key and also provide a resolution for 'in-between process' when certificates signed with old or new versions of CA's key pair must be handled.

Cross-certification request. The operation when one CA requests another CA to certify its signing keys.

Cross-certificate update. CA updates its certificate using a cross-certification request, so the certificate is received in response to a cross-certification request.

c) Certificate and CRL Discovery operations. This group of processes includes publication of issued and revoked certificates to a certificate repository. The certificate publication is handled by using the CMP announcement messages CA Key Update Announcement or Certificate Announcement. The CRL publication is defined by CRL Announcement or Revocation Announcement.

d) Recovery Operations. The recovery is used when the end entity has "lost" its PSE (Personal Storage Environment). The example of such operation is key pair recovery that is used to recover or restore private keys or other materials that have been lost, where a backup version is available in a key archive.

e) Revocation Operations. This process deals with creation of new CRL or CRL entries and is assigned as a revocation request, when client requests CA to revoke the current certificate due to client’s key pair compromise and so on. [9][23]

2.2.3.2 CMP Message Format

An advantage of defined CMP data exchange format is that all PKI management operations are wrapped in general object — PKIMessage. Figure 2.5 represents the structure of PKIMessage.

As we can see, PKIMessage consists of required field PKIHeader that helds information about sender’s and transaction identity. The second mandatory field is PKIBody that can contain any of maintained by CMP PKI operations’ data in corresponding format (see Figure 2.5, PKIBody format and PKI operation relations). The next two fields are optional, they are PKIProtection and extraCerts. Protection of CMP messages may be achieved by the use of a MAC or signature to provide

integrity checking of the message contents and extraCerts item contains additional certificates to be used by CA to verify the requested certificate.

Due to such ‘wrapper’, there is no need to handle multiple standards and types for each PKI operation. Especially it simplifies a development because the lower level services just work with one type of object and do not mess with PKI level that is the responsibility of another layer of services.

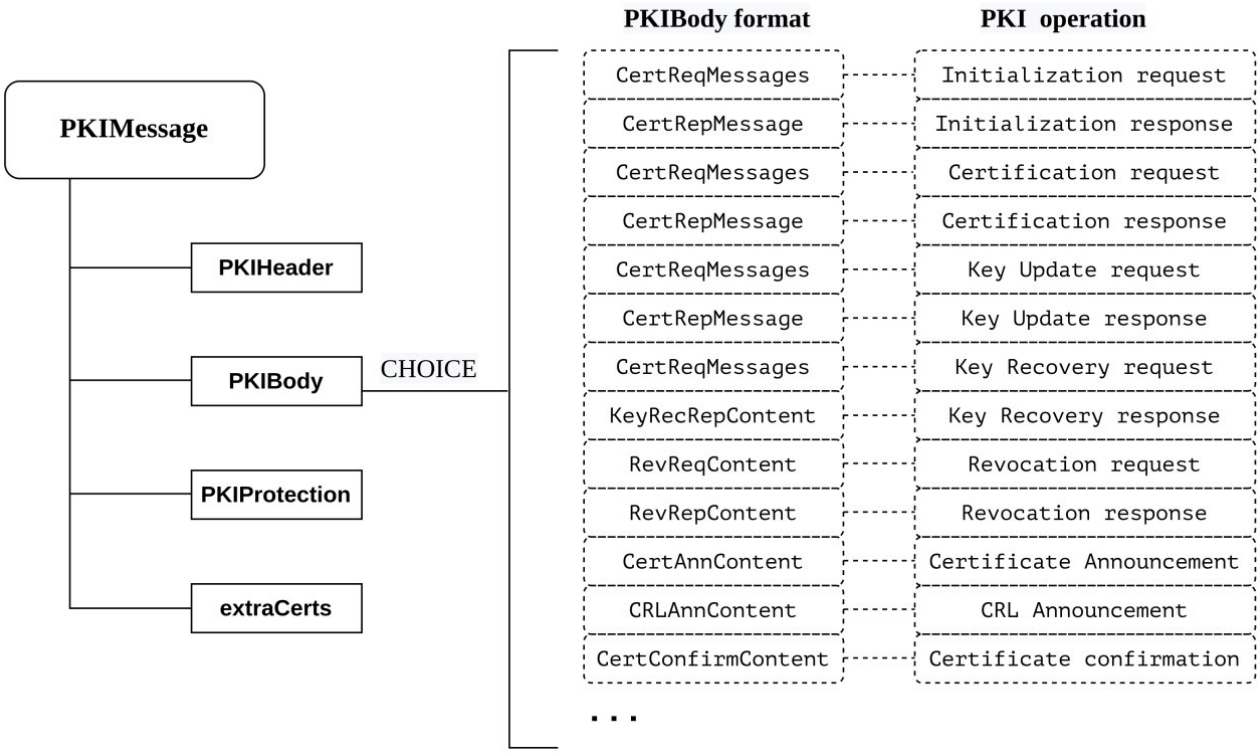

Figure 2.5 – PKIMessage format

2.2.3.3 Proof of Possession (POP)

Before to perform a certification procedure CA (or RA) verifies the EE’s data in incoming request. In order to ensure the end entity which data and public key is set in request, has the corresponding private key, the mechanism of the Proof of possession (POP) is used. Without proof of possession, it could be possible for the CA to bind the public key to the wrong entity.

To proove the ownership of the paired private key, the CA client needs to perform some operations to satisfy the indicated key use.

The proof of possession is achieved in different ways for different key types (if the same key can be used for different purposes — signing and encryption in the case of RSA — any one method can be used).

For example, for the key-pair has been generated for digital signature creation, the requester signs the request with a private key and CA can verify the signature by provided public key in this request.

Proof of Possession is an optional field in CMP message, but according to the PKI requirements, CA must support the function of verification of the POP. [9][13]

2.3 Summary

In second chapter we gave an information about PKI common standards and main protocols.

Firstly we described the X Series Standards, especially X.509 certificate standard as these standards are important part of PKI management protocols. For X.509 standard, we listed three versions and its specifics (certificate fields, addons etc.) as it is highly coupled with the protocols mentioned in the next paragraphs (PKCS and CMP).

Then, we described some details about PKCS#10 and PKCS#7 standards as it is directly related to Certificate Authority data exchange – certificate request and response respectively.

After that, we gave details about Certificate Management Protocol (CMP) and mentioned its advantages comparing with PKCS. The main features of CMP are the following:

- CMP protocols all communication between clients and CA, including certificate and revocation requests, update key requests and cross-certification;
- CMP message – PKIMessage acts as a universal wrapper that contains PKIBody in appropriate format and metadata for transaction, thereby CMP can

also be used at higher level for already implemented PKCS in the existing PKI system;

— CMP’s PKIMessage provides extra security metadata inside – ProofOfPosession. This mechanism assures the CA that the client that sends request to CA with a public key in mesasge is real owner of this key pair and has the corresponding private key.

3 CERTIFICATE AUTHORITY AS A PKI COMPONENT

3.1 Comparing CA architectures: advantages and weaknesses

The PKI system is supposed to have a particular architecture of Certificate Authorities. From one side we can classify architecture depending on CA layers. Here is two types:

a) Two-Tier Architecture

Two-Tier Architecture contains two layers of CA: root and issuing. On the top layer is root CA that is the end point of the certification chain (see Figure 3.1). Root CA issues certificates that are self-signed and can be used for either end entities or other CAs. The second layer is placing an issuing CA (or end entity CA) that issues certificates for PKI users.

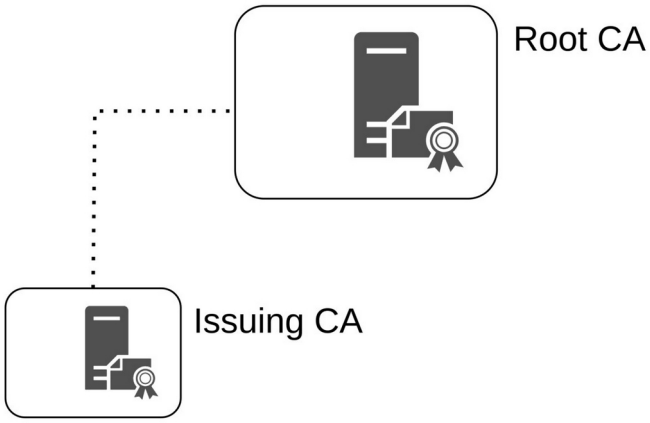

Figure 3.1 – Two-Tier Certficate Authority architecture

b) Three-Tier architecture

Three-Tier architecture includes 3 types of certificate authorities: root CA, intermediate CA (one and more) and issuing CA, as it is shown on Figure 3.2. The additional kind of CA here is ‘intermediate’. The Intermediate CA is placed in the middle of the certificate chain – it can be certified either by root or other intermediate authority on the upper tier and is issuer for intermediate authorities at the level below. The system can contain single or multiple intermediate CAs.

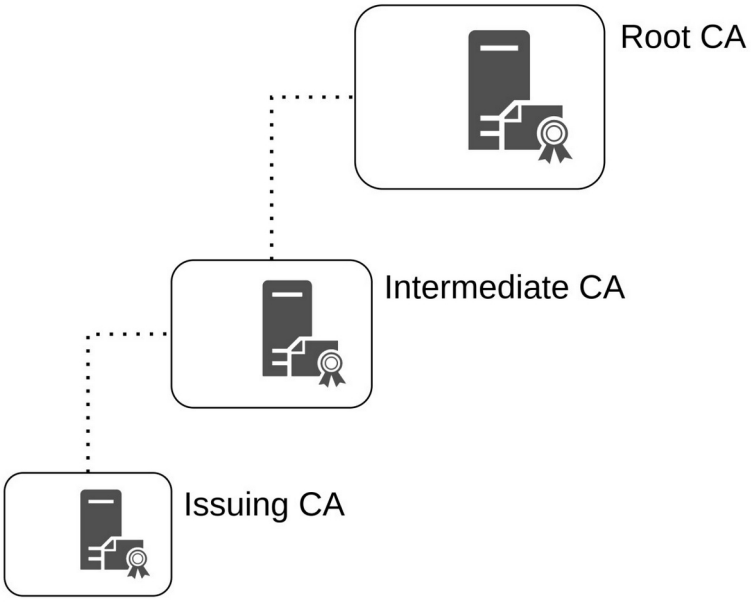

Figure 3.2 – Three-Tier architecture of Certificate Authorities

The choice of architecture depends on the design of the system. For small, locally-bounded systems with one or two issuing CAs the two-tier type is applicable. For complex, loaded system with multiple issuing CAs and users, with external root CA, the preference should be given to three-tier architecture as more secure option where are more links in the chain that would need to be broken by attackers. [22]

Another division is based on arrangement and relationship between certificate authorities. Let’s take a closer look on there types:

a) Single CA architecture

In this basic type of architecture, there is just one CA who issues and distributes certificates and Certificate Revocation Lists (CRLs) to the end entities, as shown on Figure 3.3. EE1 and EE2 obtain certificates from CA1 (only one CA persisting in the system).

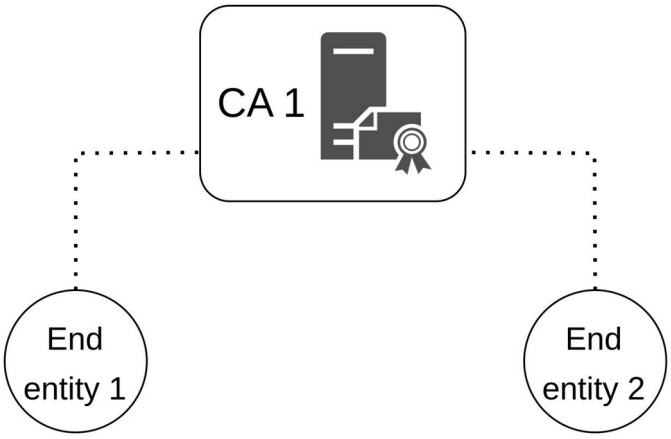

Figure 3.3 – Single CA architecture

The main advantage of Single CA architecture is simplicity of configuration and deploying. Only one CA must be managed, which also causes several problems:

- this type of architecture is not scalable — it is not allowed to add new CAs to the system;
- single CA is actually a single point of failure. If the private key is compromised, all certificates issued by this CA will become invalid, which can lead to a complete failure of the PKI system. Becides, this CA must provide a notificating mechanism to inform all entities about the problems. CA must also provide a re-establishing function in case of failure, with reissure of all issued certificates. [4]

b) Hierarchical CA architecture

The architecture contains multiple CAs in hierarchical structure, so here is a root Certificate authority, which issues a self-signed certificate and acts as issuer for subordinate CAs. Subordinate CAs can issue certificates for CAs below them in the hierarchy or for users.

The certificate chain is illustrated in Figure 3.4. For example, we want to verify the certificate of EE3. Then we need to verify the EE3's certificate, issued by CA-5, then certificate of CA-5, issued by CA-2, after that CA-2's certificate, issued by root

CA. In a hierarchical PKI, each relying party knows the signing public key of the root CA. To rely on a certificate, we must ensure that:

— each certificate in the chain is signed with the private key of the next certificate in the chain;
— the certificate has not expired and not canceled;

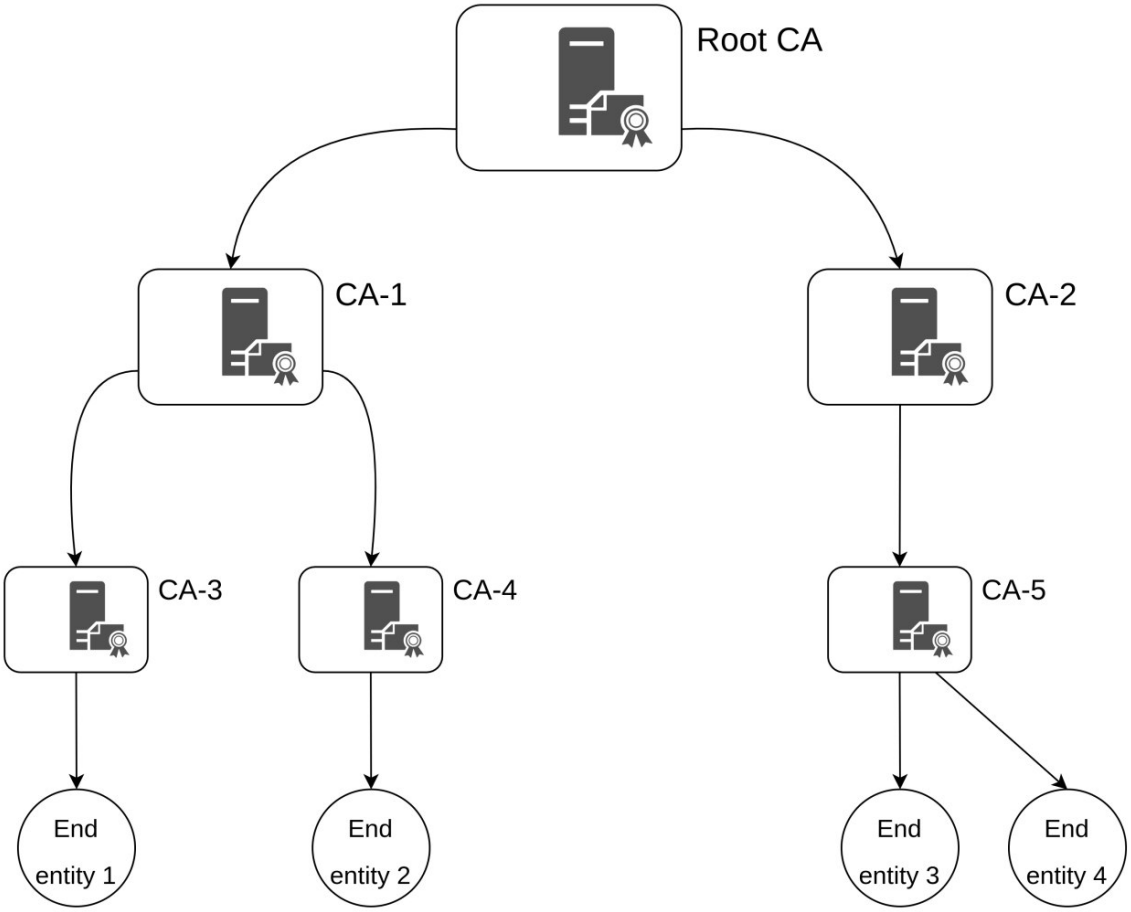

Figure 3.4 – Hierarchical CA architecture

— each certificate satisfies a number of criteria, set by certificates located higher in the chain. [6]

The advantages of hierarchical architecture include:

— Scalability. We can easily add new subordinate CA and build a new variant of certificate chain.
— Ease of management and deployment. Having only superior-subordinate relations, certificate chains are quite short and the majority of functions are

concentrated in root CA. Establishing root CA we can add/remove subordinate CAs and add/remove chains respectively.

However, with a compromisation of private key in key component - root CA, the complete PKI architecture will break down. [4]

c) Mesh CA architecture

In this type, CAs have a peer-to-peer relationship. All CAs in a mesh PKI can be trust points and all CAs need to cross-certify each other.

Cross-certification is the process of bidirectional establishing of trust between two CAs. Cross-certification helps to avoid an ‘Achilles’ heel’ in the system. If we have multiple trust points, then compromise of a single CA cannot result in a breakdown of the complete system. If one CA is compromised, there are still other CAs that use not compromised trust points and can provide certificates.

Using Figure 3.5 we can simulate a CA-1's compromisation and actions in this error scenario. We need to verify EE2's certificate. Usually, we do this in the shortest way: EE2’s → CA-4’s → CA-1’s → Root CA.

We check the EE2’s certificate, issued by CA-4 (see sub-path 1 in Figure 3.5), then CA-4’s certificate, issued by CA-1 (sub-path 2) and CA-1’s certificate, issued by Root CA (sub-path 3). After CA-1 has been compromised, we still can build a trust path, it will be longer although: EE2’s → CA-4’s → CA-3’s → CA-5’s → Root CA. Now we should check the EE2’s certificate, trusted by CA-4 (the same sub-path 1, Figure 3.5), then CA-4’s certificate, trusted by CA-2 (sub-path 4), then CA-2’s certificate, trusted by CA-3 (sub-path 5) and CA-3’s certificate, issued by Root CA (sub-path 6). [4]

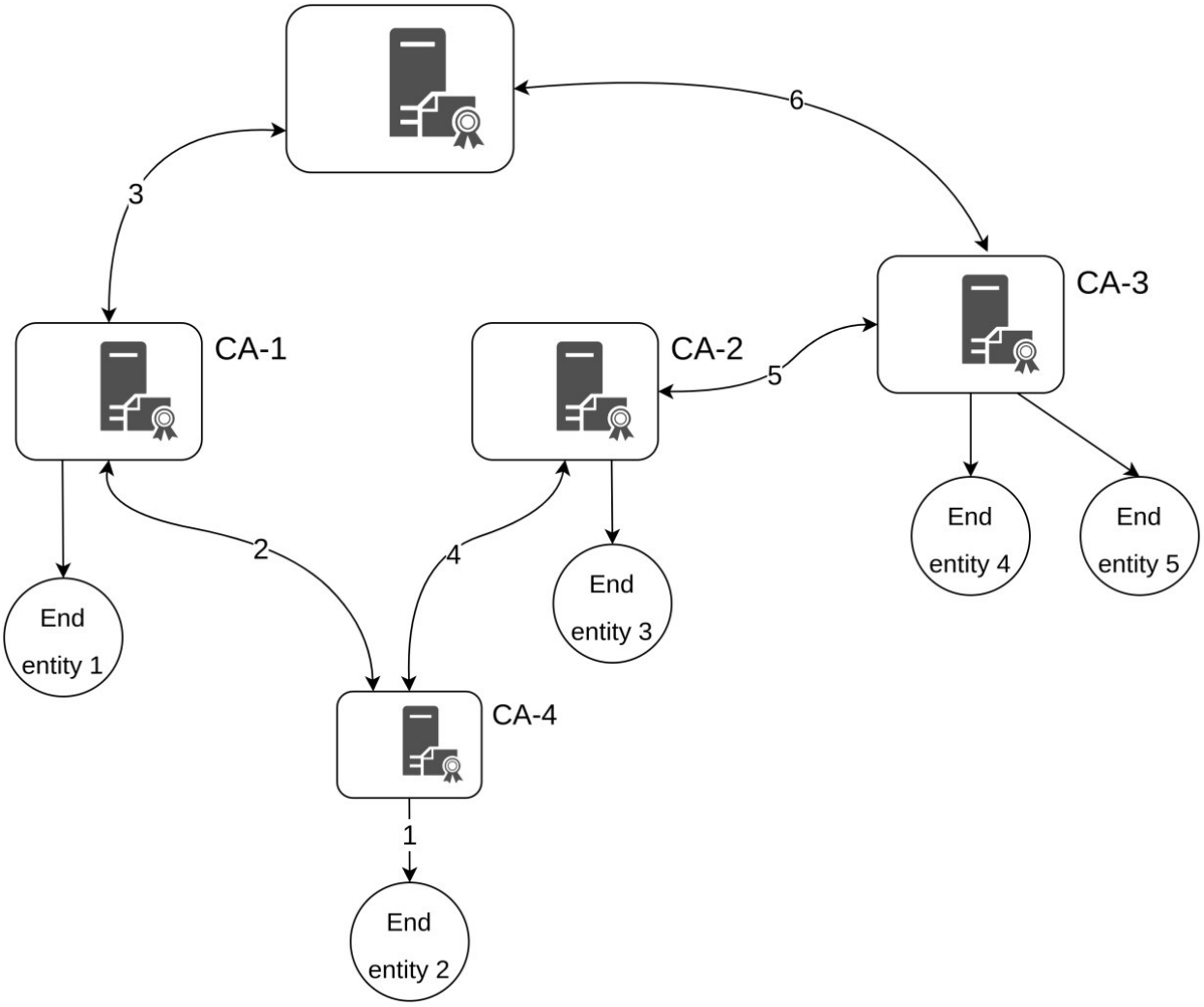

Figure 3.5 – Mesh CA architecture

d) Hybrid architecture

Every CA architecture has strong and weak parts, following the demands of the systems, we can suggest one type for one system and different types for another. As a result, we have a complete PKI system consisting of parts with various architectures. This kind is called ‘hybrid architecture’.

To link heterogeneous public key infrastructures in hybrid architecture, the ‘bridge’ component is used. The bridge component is a CA that links one CA sub-system with another. The bridge authority does not issue any certificates directly to end entities and can not act as a root CA, so it is an intermediate CA. A bridged CA establishes peer-to-peer relationships between different PKI systems.

In Figure 3.6 we can see a hybrid PKI with two subsystems: hierarchical (CA-1, CA-2, CA-3 and CA-4) and mesh (CA-5, CA-6, CA-7). The bridge CA establishes a connection between these systems. Actually, it acts as an adapter. From the hierarchical side, bridge CA is linked with CA-1 that is a root CA in the system. From the mesh side, bridge CA can be connected with one of multiple trustpoints, e. g. With CA-5. [6]

The hybrid architecture makes our system less exposed to break downs. In the hierarchical part the root CA is not anymore a single trust and will not cause a complete crash in case of compromisation as cross-certified architecture is provided. In this architecture, either the root CA or the subordinate CA of a particular PKI

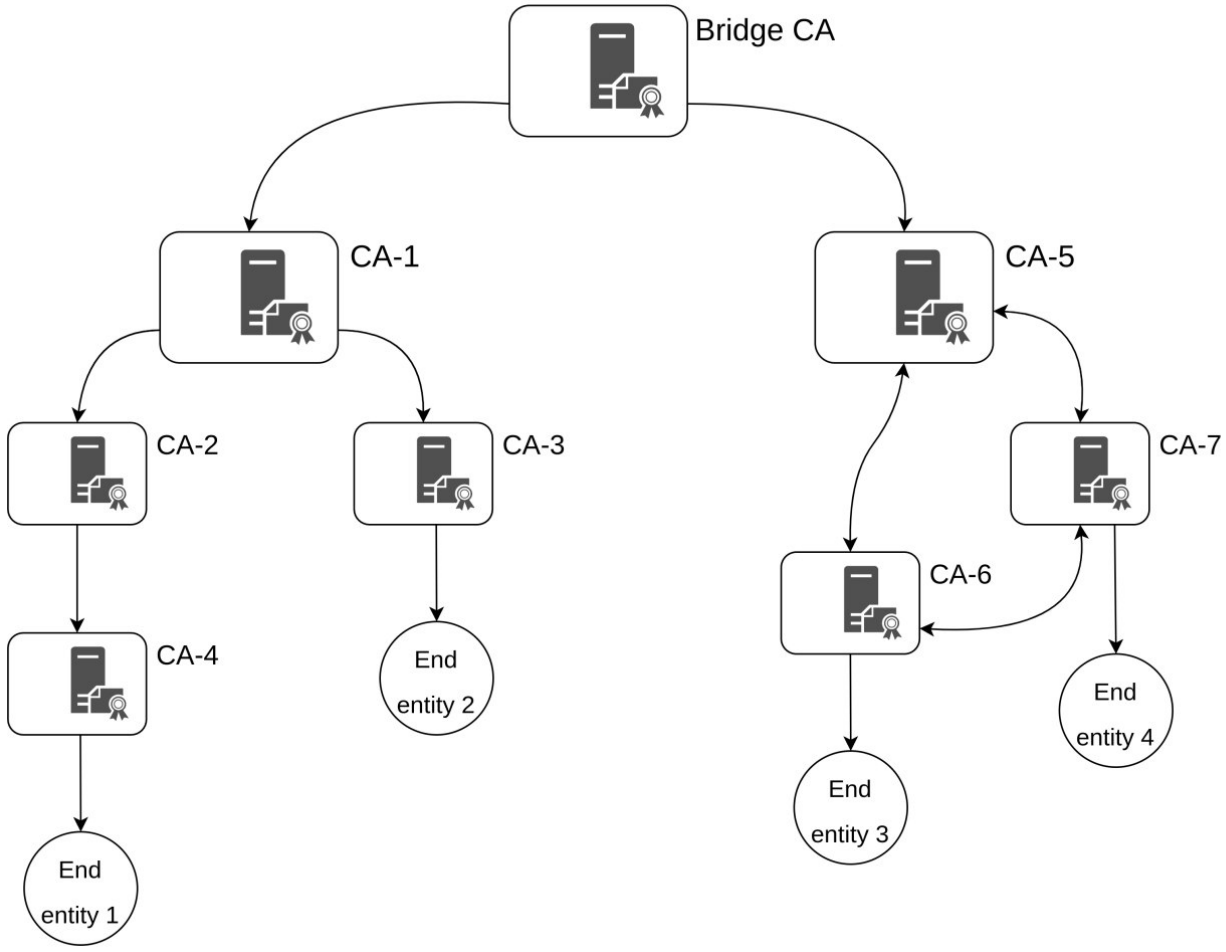

Figure 3.6 – Hybrid CA architecture with bridged authority

establishes a peer-to-peer trust relationship with the root CAs or subordinate CAs of another PKI. [4]

3.2 CA functions

The main certificate authority function is issuing certificates for the clients. Based on this function, CA must also provide maintenance of certificate lifecycle, more precisely, to take care of issued (valid) and revoked or expired (invalid) certificates, and host a repository for them. In addition, CA must have a mechanism of client verification to ensure that 'client' (in the previous chapter we saw that it can be either EE or other CA) is trusted. Let's provide more details about each function. [21]

Issuing Certificates. Certificate authority must provide a service to issue a certificate on client request. The list of protocols are used in PKI during certificate processing (not only issuing, but also update, revocation, etc.) contains Certificate Management Protocol (CMP) and Public Key Cryptography Standards — Certificate signing request or briefly PKCS#10.

Firstly, the client should have a key pair and generate a template of the certificate that he expects to obtain. The public key with certificate template and additional data are sent via request to the Registration Authority (RA) (see step 1 in Figure 3.7).

RA identifies and authenticates client (step 2) and after this forwards the request to the CA (step 3). To mention briefly, RA is not required to be a separate service and can be implemented in Certificate authority. CA receives request and validates data in certificate template (step 4). If all data is valid, CA signs the requested certificate (step 5) and sends a response with certificate to RA (step 6), otherwise CA sends a response with failure status (according to the PKI protocols). Finally, RA forwards a response (either successful or failed) back to the client (step 7). [4] [9]

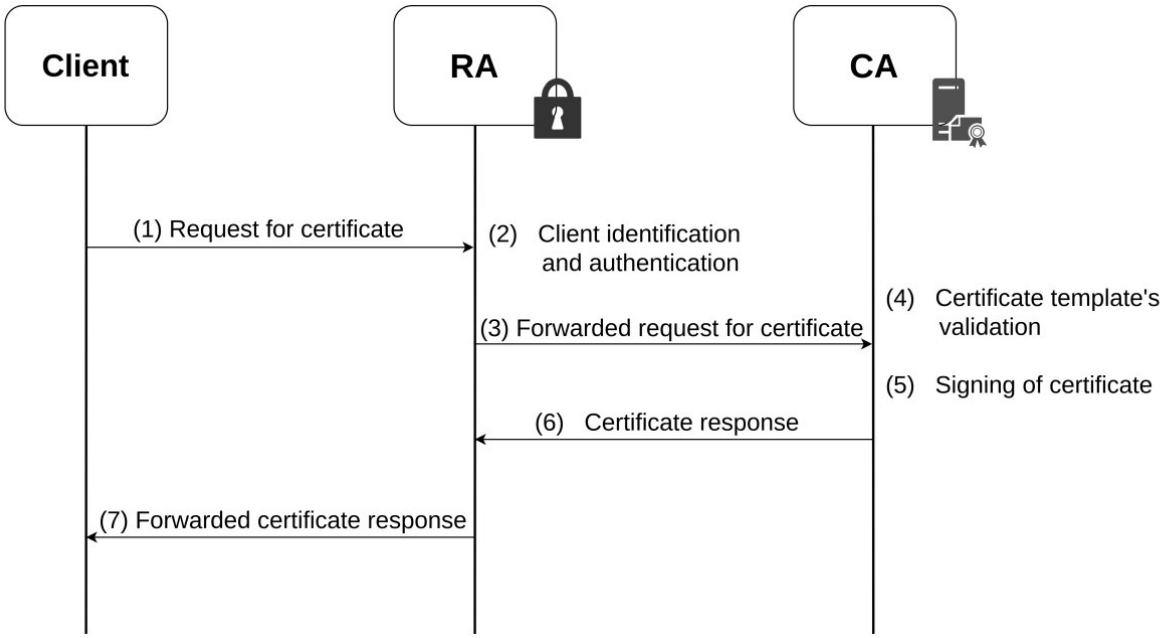

Figure 3.7 – Certificate issuance flow

Revoking Certificates. The issued certificate is intended to be valid only for its validity period and after this time it becomes expired and non-useful. However, various circumstances may cause a certificate to become invalid prior to the expiration of the validity period. The list includes:

— update in name, subject and other EE's data in certificate;
— certificate holder is not a part of organization and certificate should not be used anymore;
— compromise or suspected compromise of the CA’s private key.

Under such circumstances, the CA needs to revoke the certificate. [25]

Certificate authority provides the following methods to manage revoked certificates:

a) Periodic Publication Mechanism.

This mechanism includes Certificate Revocation List (CRL). CRL is a list of revoked certificates that is digitally signed to prove authenticity. CRLs are regularly updated. During every update of the CRL, newly revoked certificates are inserted into the list. Once a client wants to obtain revocation information, he downloads the CRL

and verifies its digital signature. After that, the client searches and checks whether the certificate that he is interested in is contained in the CRL.

There is one performance issue with CRLs is that after long system exploitation, the list becomes too large and not efficient. Thus, delta CRLs have been introduced. These CRLs contain only the certificates that have been revoked after the publication of the last full CRL.

b) Online Query Mechanisms

As an alternative to CRL list the online certificate status protocol (OCSP) has been invented. It allows clients to query an OCSP server about the revocation status of individual certificates. Upon advantages of this method are:

1) the client receives fresh information on each request. Instead of downloading large CRL that can take a lot of time and process already out-of-date information, OCSP server takes a responsibility to give the up-to-date data to the clients.

2) OCSP is that it does not require much storage. Only the revocation information about the certificate under examination must be stored. The clients must provide an online services to receive a revocation data although. [3]

c) Other Revocation Mechanisms

One of the other methods to manage certificate revocation is the Novomodo method. Novomodo uses a hash chain together with a single digital signature. The plus of this method is that the cost of one signature is amortized over many proofs of validity.Another method is usage of short-lived certificates. If the certificate's validity period is very short, there is less chance the such certificate will be revoked. [3] [20]

*Formulating a Certificate Policy***.** As Certificate authority receives a request to issue a certificate, it validates the given 'certificate template' that must contain all

required data and valid formats to be signed by CA. After issuing, this certificate can be used and verified according to applied rules in the PKI system. These rules are defined and called Certificate Policy (CP). Different certificates are issued following different practices and procedures, and may be suitable for different applications and purposes. A certificate policy needs to be followed by both the issuer and user of a certificate.

Certificate policy also constructs an authority accreditation. Every CA is accredited against one or more certificate policies that it provides. When one authority issues a certificate for another CA, the issuing CA must assess the set of certificate policies for which it trusts the subject CA. The assessed set of certificate policies is then indicated by the issuing CA in the certificate.

Certificate policy serves to determine the security policy for our PKI system and references rules for other organizations that need to establish a domain-trust relation with us.

Certificate policies are documented and practiced by using a Certification Practice Statement (CPS). A CPS specifies the policies and procedures that a company follows for issuing and managing certificates.

How users can receive data related to certificate policy and CPS? This is accomplished by the use of policy object identifiers (OID). Each OID points to a certificate policy, which is, in turn, related to one or more CPS.

To define a ceritificate policy CA should perform the following tasks for formulating a CPS:

- issue a CPS;
- ensure that the RA and the subscribers are following the policies and CPS;
- Ensure that any other certificate authority with whom it cross-certifies complies with certificate policies and CPS that are mutually agreed. [4] [19]

3.3 Key and Certificate management

To provide functions and services in our PKI environment we must follow the requirements defined for PKI components and suggested protocols for protected data exchange between these components. On the basis of this high-level flow stays the complex management of cryptographic keys and certificates. There is no sense to implement sophisticated algorithms for PKI elements and its communication if no sufficient security is provided for private keys.

3.3.1 Keys management and security requirements

Public/private key management can be divided into the following services:

a) Generation of the key pair

Firstly, the key pair object must be generated. Besides of bits value of the public and private keys, the key object contains metadata about the algorithm that is used to create keys, time of creation and some other fields related to parsing process. The recommended key sizes for key pairs are given in Table 3.1

The next aspect of key generation is the place where the key should be created. Here is several options:

— generate by owner of the key pair (i.e., the entity that uses the private key in the cryptographic operations);
— on the side of service that distributes the key pair;
— the owner and service in a cooperative process.

For key pairs used for digital signatures it is preferable to use the first method. The owner is responsible for his key generation that also simplifies the evidence of non-repudiation.

Table 3.1 Recommended Algorithms and Key Sizes

Key Type	Algorithms and Key Sizes
Digital Signature keys used for authentication	RSA – 2048 bits ECDSA – Curve P-256
Digital Signature keys used for non-repudiation	RSA – 2048 bits ECDSA – Curves P-256 or P-384
CA and OCSP Responder Signing Keys	RSA – 2048 or 3072 bits ECDSA – Curves P-256 or P-384
Key Establishment keys	RSA – 2048 bits Diffie-Hellman – 2048 bits ECDH – Curves P-256 or P-384

Second generation method is called central key generation. This method simplifies management of key archival and recovery, but also raises multiple security issues because the service that takes a lot of responsibility on working with keys, becomes a very sensitive security component. Many PKI implementations prefer to use a central generation for encryption keys that need to be archived or recovered, but client side generation for digital signature keys that should not be archived. [5]

b) Distribution

When we want to share the public key to some other PKI entity (receiver), the following requirements must be fulfilled:

1) The receiver of the public key should be provided with assurance that the owner of public key of consumed message is the actual owner of the key;
2) When service distributes the key to client, the purpose (usage) of the key must be defined (e.g., for digital signatures or authentication);
3) All attributes related to the key are known (e.g., domain parameters);
4) The public key value is valid

c) Key Archival

After the active state (usage in cryptographic operations) keys can be archived in storage – repository. The keys are usually archived for the recovery purpose, so they can be retrieved from the archive repository back.

The key must be stored with all necessary metadata to parse the value, know the creation algorithm, the cryptoperiod etc, but not with some other operational data that can be used in pair with key in malicious way. It is also suggested to store the key in several physically separated locations in case of problems with one archive storage. Nevertheless, not all keys need to be archived. The recommendations for PKI public/private keys archiving are given in Table 3.2.

Table 3.2 Key Archival

Type of Key	Archive?	Retention period (minimum)
Private signature key	No	
Public signature-verification key	Yes	Until no longer required to verify data signed with the associated private key
Private authentication key	No	
Public authentication key	Yes	
Private key-transport key	Yes	Until no longer needed to decrypt keys encrypted by this key
Public key-transport key	Yes	No real use after its usage period

d) Key Recovery

As was previously mentioned, key recovery is the opration of of retrieving or reconstructing a key from key archives. The purpose of key recovery is to process cryptographically protected information, for instance, verify signature on signed information or decrypt encrypted data.

There are several requirements for the process of recovering and recovered key usage that must be followed:

— recovered keys should not be used for cryptographic computations if the key lifetime has ended;

— After performing all necessary cryptographic operations with recovered key, the key object must be destructed and removed from the cryptographic process for which it was recovered. The key can be returned to archive though and stored there as long as it is needed. [8]

3.3.2 Certificate lifecycle

The processes performed with PKI certificates can be divided into main stages:

a) Certificate generation phase. The starting point of a certificate is the formation of a request for certificate creation. This request typically contains a certificate template with information related to the requester (and future owner of the certificate), public key, entity registration data and other metadata.

After processing of the request by RA and CA, successful registration and data validation the certificate is issued.

Once a certificate is issued, it is delivered to and accepted by its owner.

After issuing the certificate, it needs to be published to a certificate repository to be available for other PKI entities that want to establish trust with the current certificate owner.

b) Certificate Validity Phase. The certificate is supposed to assure client that the public key contained in the certificate belongs to the certificate owner. The client can verify certificate by its signature, check the validity period and its revocation status. Usually there is more than one CA is involved in PKI system, which means that client must check the chain of certificate until root.

If certificate’s lifetime is close to expire date but client wants to use this certificate longer, the cerificate can be reissured before its expiration period.

c) Certificate Invalidity Phase. Certificate can become invalid due to expiration (the lifetime period has ended) or revocation (the lifetime has not ended yet, but the certificate has been revoked because of certain circumstances).

The key pair that has been certified by this certificate can not be used anymore if the main facilities of the corresponding certificate are in invalid state.

Usually, invalid certificates are not deleted totally after the invalidation. Either due to expiration or revocation, the certificate may have to be archived. [3]

3.4 The common standards and legislative framework for CA in Ukraine

The PKI system is widely used in e-commerce organizations to manage and secure processing with electronic documents. In Ukraine digital signature has a legal binding, thereby it is important to mention some aspects from legislative framework for CA.

According to the official government resources, the Certificate Authority can manage the following data:

— maintain digital signatures and certificates;
— obtain necessary data for registration and certificate issuance from the client (legal entity or individual or from its authorized representative).

On the other hand, the Certificate Authority is obliged to provide:

— protection of information in automated systems in accordance with legislation;
— protection of personal data received from the client in accordance with the law;
— ensure the possession of public key, which is set in request for certificate) to the entity that has corresponding private key;
— reissuance, revocation and annulment certificates according to the protocol;

- verify the legality of requests for revocation, annulment and reissuance of certificates and keep documents on the basis of which key certificates were revoked, annulled or reissued;
- trouble proof service for the clients to issue, reissue, revoke and annul certificates;
- maintenance for issued, revoked and annulled certificates;
- trouble proof service for clients that want to access the certificate repositories through public telecommunication channels;
- storing of issued certificates during the period provided by the law for the storage of relevant documents on paper;
- custom support on issues related to the digital signature.

Access and storing of private keys of clients is prohibited for Certificate Authority. [18]

In this moment, the developed universitie’s Certificate Authority has more experimental target and is not registered officially to have a legal binding in Ukraine and ability to certify digital signatures not only for education inside the university, but also provide verified proof in inter-university processes and financial flows, but surely this system can be extended, improved and integrated into the real e-commerce environment.

3.5 Summary

In this section we described the various CA architectures and its facilities. In general CA architectures can be divided into two-tier and three-tier with required root CA and issuing CA in both cases and optional intermediate CA in second type. By Ca arrangement we can define the single, hierarchical, mesh and hybrid types. The first two types are good for smaller organizations with internal usage, very restricted access to the root CA and small number of clients. The second two are sophisticated

in configuration and management, but less influenced by any CA compromisation or failure and more effective with a large count of certificate holders and other users.

Then we define CA functions, that include: issuance, reissuance, revocation, publication of certificates and CRLs and also formulating the Certificate Policy that defines any services or actions provided by CA.

After that we determine the aspects of public/private key pair management and the stages of certificate lifecycle. The PKI system should provide appropriate key pair generation, storing, updating, archiving and recovery (if it is required in specific cases). The Certificate Authority, on its turn, manages the certificates. Firstly, to be an actor in trust flow, the certificate must be generated, issued and published. After that it can be obtained by clients and verified. Then, depending on the situation, the certificate can be reissued (updated), just expired or revoked. Finally, certificate can be archived in a long-term storage and recovered in extra cases to prove the actions performed with its usage.

In the final part, we mention the main requirements to the Certificate Authority to be certified at a legislative level in our country.

4 IMPLEMENTING OWN CERTIFICATE AUTHORITY

4.1 Ready-to-use solutions

Before creation of the design and implementation of an own Certificate Authority PKI component, it is useful to take a look into some ready solutions from verified organizations.

The first product we can investigate is HashiCorp Vault Certificate Authority. According to the official documentation, HashiCorp Vault provides a PKI environment built in Vault server that allows users to obtain X.509 certificates on request.

All functionality is encapsulated inside of the Vault's PKI secrets engine - the part of Vault server. To prepare the PKI system, the user needs to generate a root CA certificate. We can also configure an Intermediate CA and then establish an Issuing CA. After that, the user can easily request for new certificates, revoke existing, cleanup certificate archives, check repositories etc. All these operations are available through Vault API or command-line interface.

HashiCorp Vault's PKI solution is great for small organizations as it provides easy maintenance and administration, straightforward configuration and fast deployment. The Vault actually manages the Certificate Authority, Certificate/CRL repository, Certificate archive and its communication by itself. Thereby, if a user makes some flaws during the configuration process, Vault engine will take default properties to avoid future system faults. [12]

However, when we performed an analysis for the application of Vault's PKI product into our university system, we faced several problems.

Firstly, the PKI component, such as Certificate Archive, are already built in HashiCorp Vault PKI engine and managed internally, which means that we do not

have the ability to provide the custom configuration for these components and provide the custom algorithms for communication between components.

Explaining the previously mentioned, the main purpose of the PKI system in university is to provide security basis for electronic document management, more precisely, provide verified digital signatures for students and university staff. In the current flow, when a student wants to pass a laboratory or course work or diploma etc., he puts his signature on a paper variant and this paper variant must be stored in university for a long-term period, and, what is important, much longer than this person will be a student of the university. This aspect is required to be handled also via digital signatures, therefore, all certificates must be achieved for a long time (10+ years) with an ability to be restored. In order to fulfill these requirements that are also dependent on Ukrainian laws, we need a reliable own implementation of Certificate Archive.

The second issue is raised about Key Repository. As we want to maintain and store users' key-pairs on our servers, we need to create a Key Repository service for storage of certified key-pairs. The Key Repository is a very sensitive component with confidential data. At the same time, every time when a user wants to sign any document, his data must be retrieved from Key Repository, hence this service must be sustained.

PKI Vault does not provide a Key Repository, so if we want to use PKI Vault, we need to provide a key storage from 'outside' (from our side to Vault's engine). This can cause the problem with access and network restrictions. We want to keep the Key repository in the local network and with very limited access. In this case we need to set an additional layer in our system that is shown in Figure 4.1.

This structure of layers also causes the third issue. As Vault has every PKI component in-built, the API provided by Vault has been created for PKI clients (actually, PKI users in the Figure 4.1).

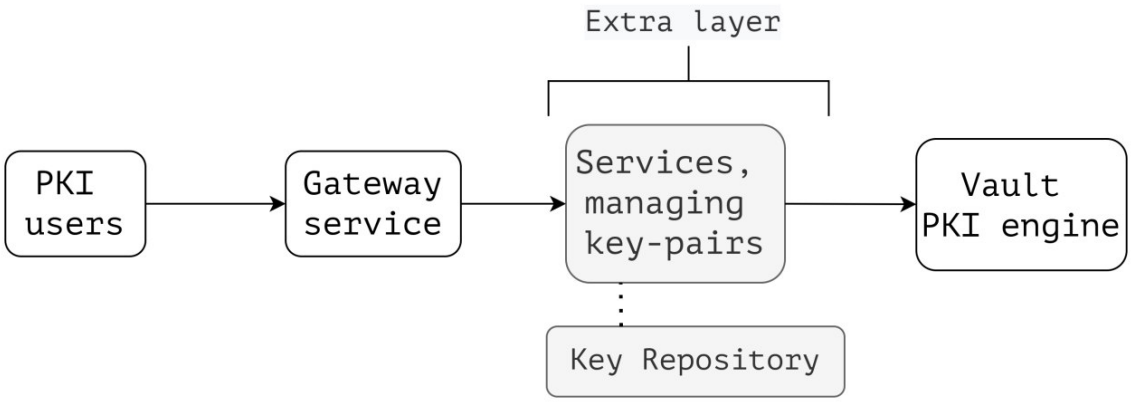

Figure 4.1 – Certificate issuance flow

All authentication and other security checks Vault encapsulates inside the of engine. Hence, the middle layers in Figure 4.1 will act as an extra proxy, forwarding the user's request to Vault's engine, that will perform all mapping, constructing of Certificate requests etc. inside. This seems to be an overhead and non-efficient design. For example, we can already provide a creation of a Certificate request in gateway service. The same is related to the authentication.

Among other problems we can mention:

— choice of CA architecture type (Vault offers only a hierarchical type);
— choice of management protocol (e.g. to use PKCS or CMP);
— choice of transport (TCP, PEM or HTTP) as it is encapsulated in Vault engine;
— currently, Vault provides only CRL mechanism and no OCSP;
— user is limited in usage of some extra security features, such as Certificate Transparency protocol and others;
— the user is limited in usage of authentication options. There is no special configuration for Registration Authority and so on;
— storage choice: Vault uses its internal storage by default. There is an option to add integrated storage, but the list of supported databases is also limited.

Switching to another solution, the second product we have checked is EJBCA. EJBCA provides a whole system required to issue certificates. Unlike a HashiCorp Vault product, EJBCA provides more freedom in custom configuration, for example it

offers usage of CA or CA with RA, but the archive and key repositories problem remains similar to HashiCorp Vault.

EJBCA also has much more detailed documentation about the internal structure of the system. Thus, it is useful to look at specification, used technologies and design, to follow 'best and common practices'. Besides, the EJBCA products can be obtained in enterprise or community versions. The community version code sources are open and available on Github, so it is also an important source of good programming approaches on how to deploy an own CA environment. [14]

Having considered all issues mentioned above, we decided to decline the PKI Vault solution and build our own PKI system taking the best practices and common designs into account. The created system is called PKI Clavis and its detailed description is given in next chapters.

4.2 The project architecture and design issues

After performing analysis of PKI solutions, we designed our own PKI system - PKI Clavis. The general structure of the created system is given below in Figure 4.2.

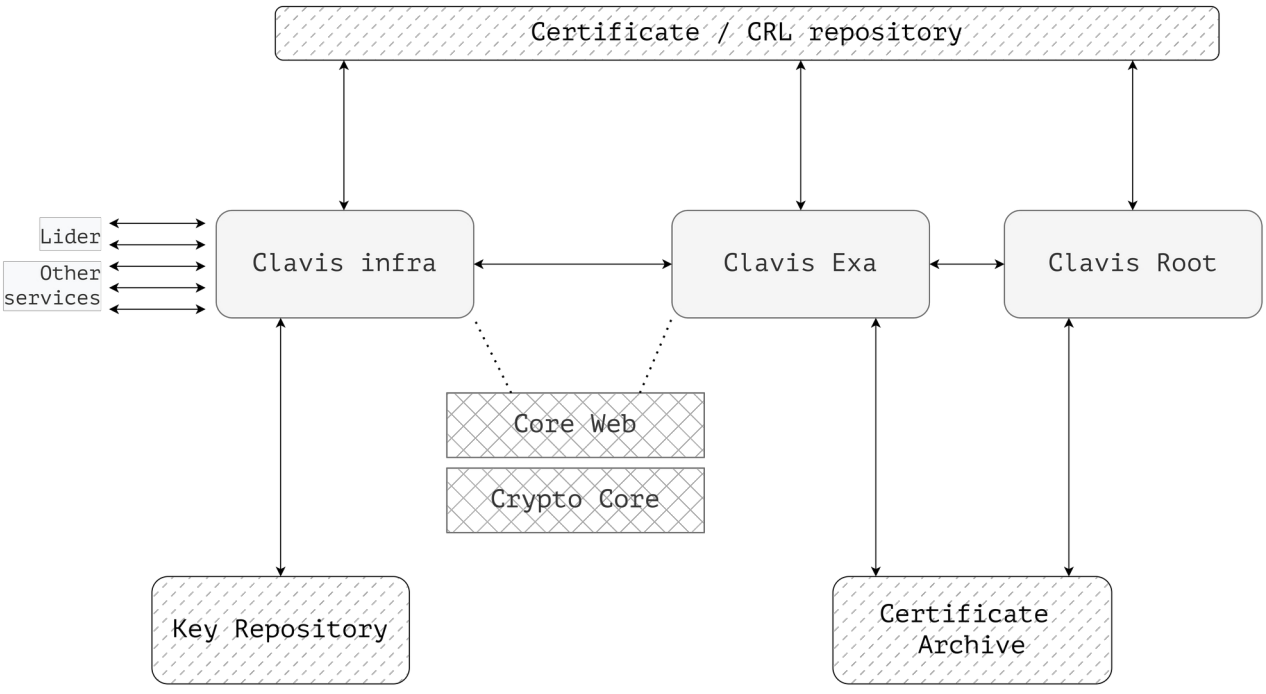

Figure 4.2 – PKI system general structure

The scheme shown in Figure 4.2 provides basic structure of PKI system (Root and issuing CAs, Certificate / CRL Repository, Certificate Archive and PKI clients are mandatory components) with additional services including Key Repository and Clavis Infra. We give more details about each component.

Clavis Infra

Clavis Infra component is a complex system of multiple services providing various functions for PKI clients. Clavis Infra acts as a gateway to the PKI environment. External services, such as Lider and other existing ones or future services should use Clavis Infra as a proxy to access the PKI services (to issue or verify certificates, for example). Clavis Infra modules also maintain keys and, thereafter, digital signatures of the PKI clients.

The feature list of Clavis Infra can be updated and modified with a development process, but the main purpose should be static - it is an entry point from the Internet to the internal system.

Clavis Exa

Clavis Exa is an Issuing Certificate Authority component in our system. The current CA architecture type is hierarhical that contains two layers of CA: root and issuing. Thereafter, we have two CAs: Clavis Exa and Clavis Root as issuing and root respectively. The details about root CA will be given in the next paragraph, issuing CA is specified with:

a) Service architecture

Clavis Exa is an online CA that is considered to work in a local network and to be proxied by Clavis Infra. The optional components, such as Registration Authority as a separate service can also be added to the Clavis Infra system. The ability to extend the Clavis Infra allows, at the same time, to keep CA’s architecture clear and plain, to avoid frequent updates and increase stability.

b) Management protocol.

As we have already described in section 2, to communicate with CA we can use the following protocols: CMP or PKCS. Based on have already been mentioned advantages of CMP, this protocol is desided to be implemented in our system. Clavis Infra acts as an end entity, therefore CMP protocol actually must be handled only by CAs and some of Clavis Infra services (CA's clients).

c) Transport protocol

One of the requirements to PKI management protocols is to be independent from methods of data exchange. Common practices include: TCP/IP, FTP, PEM or HTTP protocols. In our case, the HTTP has been provided and there are several reasons for this. Firstly, usage of HTTP gives us convenient implementation and maintenance. There are a large number of different libraries providing easy ways to use HTTP transport that do not need a high barrier to entry. Secondly, as we should use metrics, tracing, health check and backup tools that use extra or existing HTTP endpoints to monitor the system, we avoid mixing of protocols in PKI services. Thirdly, we can also add another level of security via TLS in spite of the CMP protocol providing sufficient protection of sensitive data.

d) Issuing Certificates

The PKI Clavis system supports X.509v3 certificates. In Figure 4.3 we represent a simplified flow of issuance of certificates in our PKI.

Firstly, Clavis Infra receives request from external services to create a certified key-pair for user, e.g., student. (see step 1). At this moment, Clavis Infra authenticates user, verifies his data and generates a new key-pair for this user.

Then Clavis Infra sends certification request to Clavis Exa (step 2). Clavis Exa, on its turn, verifies the request payload and in case of successful verification, issues

certificate for the user and responds to Clavis Infra with status ‘accepted’ or ‘grantedWithMods’ (step 2, case a – 2a).

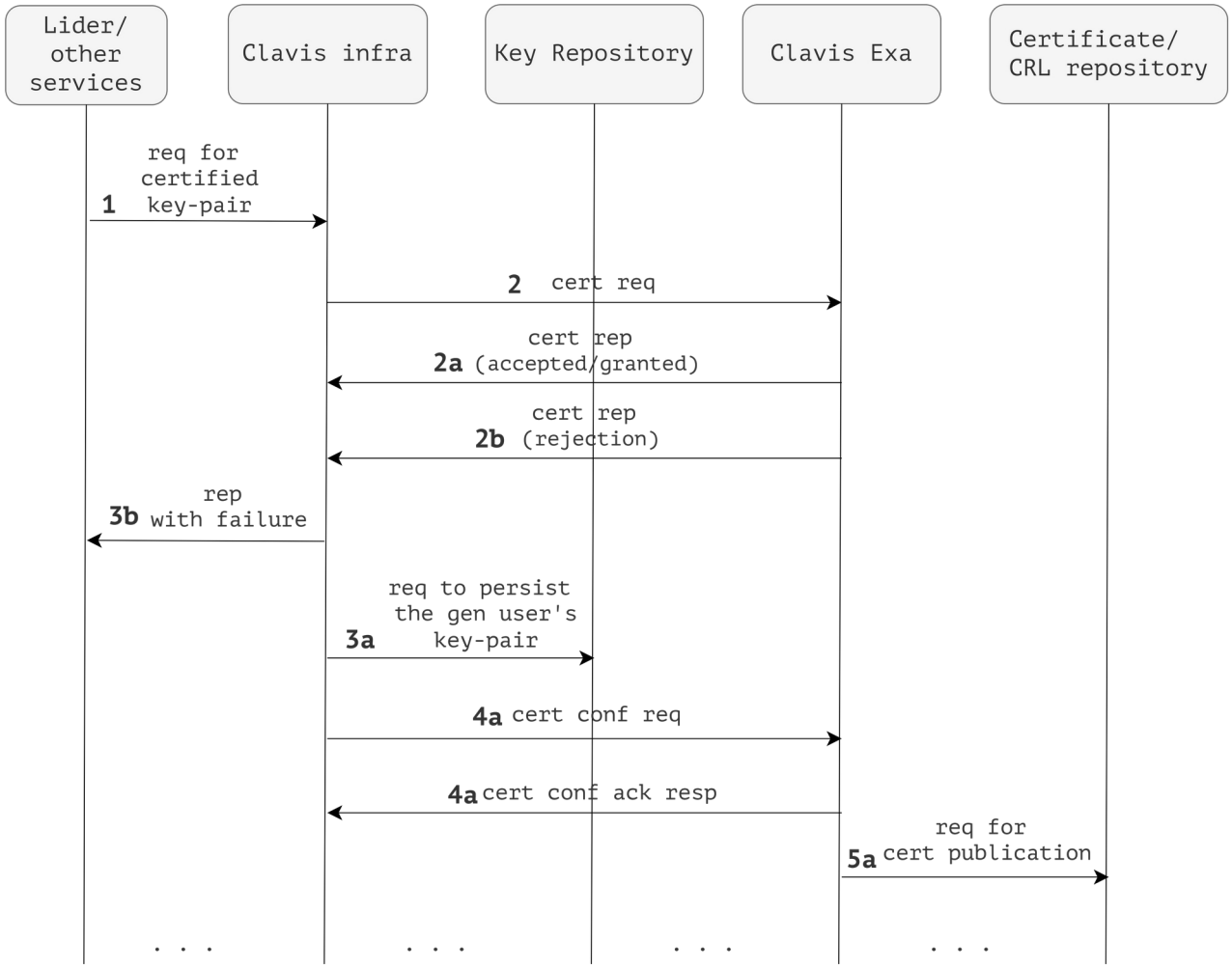

Figure 4.3 – General flow of certificate issuance

In error case of, such as CA performed verification of and it has failed, Clavis Exa responds with status ‘rejection’ (step 2b) and Clavis Infra should destruct the key pair and respond to external service with failure (step 3b). With this, the ‘*a’* flow is ended.

Returning to the case *a*, after receiving the response from CA, Clavis Infra requests Ket Repository for key-pair persistence (step 3a). After this, Clavis Infra sends certificate confirmation request of receiving an issued certificate to Clavis Exa (step 4a). The last one may respond with certificate confirmation acknowledge response.

Finally, after receiving the confirmation from client, CA publishes the issued certificate into Certificate repository (step 5a).

According to PKI specification, the particular order *'cert conf req → cert conf ack rep → cert pub into repo'* is not defined, so actually, certificate can be published even before confirmation from client. The one thing must be handled here is if certificate has already been published and CA do not receive any confirmation within defined period of time, CA must revoke the issued certificate. In our case, we have few services in Clavis Infra that are CA's clients and processing requests for certificates from other components. Thereat, we can manage these services effectively, set short period for confirmation answer and publish certificate for the repository only after receiving the confirmation, thereby avoid revocation process on this stage.

Clavis Root

Clavis Root is root CA in PKI Clavis. According to the best practices [26], Clavis Root tends to be an offline service that generates a self-signed long-term certificate and also certifies the Clavis Exa. Current version of the PKI system has two-tier CA architecture that means we do not provide Intermediate CAs. However, if we plan to extend our system to support certificates for different kinds of users and different purposes, we may need to provide a three-tier type. For example, we can issue a certificate for an individual such as student, professor, laboratory assistant, accounting staff, secretary etc. or service, such as Cursor or Lider in order to have a possibility to sign various types of documents or have trusted HTTPS connection. For this we can add with online Intermediate CAs that will be responsible for main sectors, at least one Intermediate CA can certify Issuing CAs for HTTPS and other one can support certified digital signatures.

Key Repository

This component provides secure storage for certified generated key-pairs for digital signatures. The component can have several services, including also Key Archive and Key Recovery services.

Certificate / CRL repository

This repository provides persistence for issued and revoked certificates. It also includes service (distribution point) that populates API for PKI clients to retrieve the issued Certificate and CRL.

Certificate Archive

This component serves a long-term storage of expired certificates. It is extremely important for certificates that are used for digital signatures. The documents were signed by digital signatures may be stored for a long time in university databases (as provided with paper variants now), therefore we must provide a reliable archive for certificates with possibility to restore certificates and verify the document signatures.

Core Web and Crypto Core

Core Web and Crypto Core have been added to the scheme as main libraries are implemented and used in PKI components. Crypto Core provides ready-to-use methods to easily generate key-pair, create a digital signature, create certificate template etc. Core Web contains multiple CMP request/response builders with simple configuration that allows a developer to set up communication between PKI components fastly, without dealing with low-level ASN.1 structures.

4.3 Certificate revocation problem

The issued certificate can be revoked at any time before its expiration in cause of the following situations:

— Certificate Authority that issued the certificate has been compromised;

— CA on any upper level (intermediate or even root) of authority that issued the certificate has been compromised;

— The private key of certificate holder has been compromised;
— Certificate holder is not anymore a client of organization, for which the certificate was issued;

The listed cases above are considered to be ‘extra’ and must not appear in normal flow, but must be handled in any PKI system.

Thus, what can be a problem with certificate revocation in PKI Clavis? We may look more precisely at university processes. The person becomes a student in university and generates own key-pair for a digital signature that is certified by our PKI Clavis system. The certificate is valid for the education period, for example 4 years for Bachelor’s degree, 1.5 years for Master’s degree etc. Here the last item in listed reasons for certificate revocation comes into action. A student can be expelled for bad marks or go from university by his own reasons, take an academic leave, switch the university via student exchange and so on. The list of reasons to leave the university is various.

To provide a clear and secure management of user rights, when a person becomes not a student of our university anymore, the certificate must be revoked immediately. In addition, the endpoint for users for document signing will be some university service (such as new module Lider or new part in Cursor) the management of user access to document signing is firstly handled by these services and only after by our PKI Clavis. The access-check mechanisms of these services are out of PKI Clavis scope, that is why we can not blindly trust these services. The problematic situation here can be: the person has been expelled for incomplete multiple course works and module works, he still can access some of the university services and, without additional check, he can sign some of his works that can raise a conflict and issue if the reason for expulsion was adequate.

Having described the problem in detail we need to perform analysis and estimate the amount of revoked certificates to provide an efficient protocol for revoked

certificates management. Here are three main options: to return the whole CRL, to use delta CRLs or to provide OCSP protocol. With two first options, the time of retrieval of the CRL list is important.

Based on our department we gather some statistics about students who leave the university during their Bachelor's degree period. Each year we have 6-7% of leave, so during the whole 4 years this value is considered to be 24-28%. For our calculations we will round this value till 30%, including the university staff that is also changing during years. Let the amount of students + stuff be equal around 2000 persons or 2000 certificate holders. Therefore we have 600 certificate owners and 600 certificates to be revoked.

With these estimated data we can choose the right protocol for revocation management. Let analyze the 'whole CRL' option first (as the simplest and most reliable in implementation). The main issue for this choice is how big will the whole CRL be? Too big list requires more time for downloading and processing on the client side.

To estimate the average time of CRL processing, we need to define the average CRL size. We have investigated the most common CAs, such as VerySign, LetsEncrypt, GoDaddy, Entrust and DigiCert; we found that CRL with 736 entries takes 27KiB (DigiCert). With average internet speed in 50 Mbit/sec this size can be downloaded within less than 5ms, so actually with our 600 certificates (hence, 600 list entries) the most time will be spent on request / response processing.

Let's consider the worst case: compromise of root CA when all certificates must be revoked at one time. In this case the CRL will contain 2000 entries. The Entrust's CRLw contains 4607 entries and takes around 208,4KiB. This size can be downloaded within less than 30ms, so for our system it will be twice less (around 15ms).

The results show us that there is no problem with too big size of CRL and we can use the ‘whole CRL’ method in an efficient way. Moreover, in PKI Clavis the PKI clients are Clavis Infra services (we have no external PKI users). Thus, we can easily manage the process of revocation in our system internally, maintain the network traffic and also apply the best configuration for CRL processing in these services.

Based on this simple analysis, we can sum up that processing of whole CRL can be used in PKI Clavis efficiently. To avoid overengineering, we will not provide extra delta CRLs or OCSP.

4.3 Technology stack

As was previously mentioned, two libraries: Core Web and Crypto Core have been implemented as high-level tool for PKI Clavis development. These libraries are mainly based on common open-source cryptography API provided by Legion of the Bouncy Castle Inc.

The reason for creating of high-level libraries over the Bouncy Castle API is that last one is basically realized on ASN.1 structures to allow user be free in any possible configuration of PKI protocols and requirements. During implementation of own CA, we define the common techniques that we will use for development and these functions we prepared in Crypto Core and Core Web libraries.

The programming language of current version of PKI Clavis is Java, however there are no rectrictions in programming languages or frameworks for PKI services because the communication between them is handled via HTTP.

The web framework used in Clavis Infra and Clavis Exa is Spring Framework that provides a high-level management of incoming and outcoming data and Spring Boot module autoconfigures the required web container – Apache Tomcat.

For key and certificate storage PostgreSQL database is used. PostgreSQL provides reliable security and fail safe mechanisms and also effective processing of byte arrays data.

4.4 Summary

In this section we overview our PKI Clavis system architecture and its components. We specify the functions of Clavis Infra as CA client and gateway to the PKI environment, describe the general flow of certificate issuance in our system and define techniques are efficient to use with our Certificate Authority, such as CA architecture type, PKI management protocol, transport protocol, method to retrieve the CRL.

We have also described some of ready-to-use PKI solutions: HashiCorp Vault and EJBCA, pointed its advantages and disadvantages (that are actually the reason why we have chosen to implement our own libraries and build the system ourselves).

We performed analysis and provide a resolution for certificate revocation issue and explained why in our system this issue is important to solve.

We have also listed used technologies, libraries and frameworks in the development. Java is a good choice for server side development and has a extensive set of tools providing crypto- API for keys, digital signatures, encryption and X.509 certificates.

5 DEPLOYMENT OF CERTIFICATE AUTHORITY INTO UNIVERSITY ENVIRONMENT

5.1 Initial configuration and testing of the system

In this section we present the system in work with testing the main flow of issuing certificate that has been already shown in previous section (Figure 4.2).

To perform a request for creation of certified key-pair, firstly we need to perform some initial configuration before to start the services.

Firstly, the configuration of Clavis Root must be completed, the key-pair must be generated and self-signed certificate must be issued. As out root CA is offline, we can use command line tool – openssl to create keys and certificate.

```
$ openssl genrsa -out root_ca_keys.pem 4096
```

```
$ openssl req -new -key root_ca_keys.pem -out root_ca_req.csr -config
cert_req_root.cnf
```

Figure 5.1 – Listing of root CA's key-pair generation and creation of certificate signing request

In Figure 5.1 we show the listing of commands how to generate the root key-pair (and any key-pair) and create a certificate signing request (for Clavis Root this request will be sent to the Clavis Root itself) with usage of configuration files. The sample configuration file with data for signing request is given in Appendix A.

After formatting a signing request, we can issue a self-signed certificate for root CA (see Figure 5.2). After this, the basic configuration of root CA is finished.

```
$ openssl x509 -req -days 1095 -in root_ca_req.csr -signkey root_ca_keys.pem
-out clavis_root.crt -extfile cert_req_root.cnf
```

Figure 5.2 – Listing of root CA's certificate issuance

The next step will be to certify the Issuing CA – Clavis Exa. First two steps are similar to Clavis Root, but with different signing request data in configuration file (see Figure 5.3). Similar to Clavis Root, we set all necessary data to file cert_req_exa.conf, that is shown in Appendix B.

```
$ openssl genrsa -out exa_ca_keys.pem 4096

$ openssl req -new -key exa_ca_keys.pem -out exa_ca_req.scr -config
cert_req_exa.cnf
```

Figure 5.3 – Listing of issuing CA's key-pair generation and creation of certificate signing request

Then, we issue a certificate for Clavis Exa using Clavis root keys:

```
$ openssl req -config cert_req_exa.cnf -new -key exa_ca_keys.pem -out
ca_exa.csr
```

Figure 5.4 – Listing of issung CA's certificate issuance

After performing all mentioned steps, we are ready to start Clavis Infra and Clavis Exa web services and test the issuance of student's certificate.

To send request for creation of certified key-pair, we will use Postman API platform, which design is very human. To track the process, we will use logging output of Clavis Infra and Clavis Exa services. The resulting key-pair and certificate should be stored in database, for work with Dbs we will use the DataGrip JetBrains IDE.

Firstly, we start Clavis Infra and Clavis Exa services and send a request to acquire certified key-pair for certain student via Postman tool, as it is shown in Figure 5.5.

In payload we need to send the unique student identifier in the whole university system, in this case this is ID of student's record-book. 'authToken' and 'encryptionToken' are used for authentication and private key encryption parts respectively.

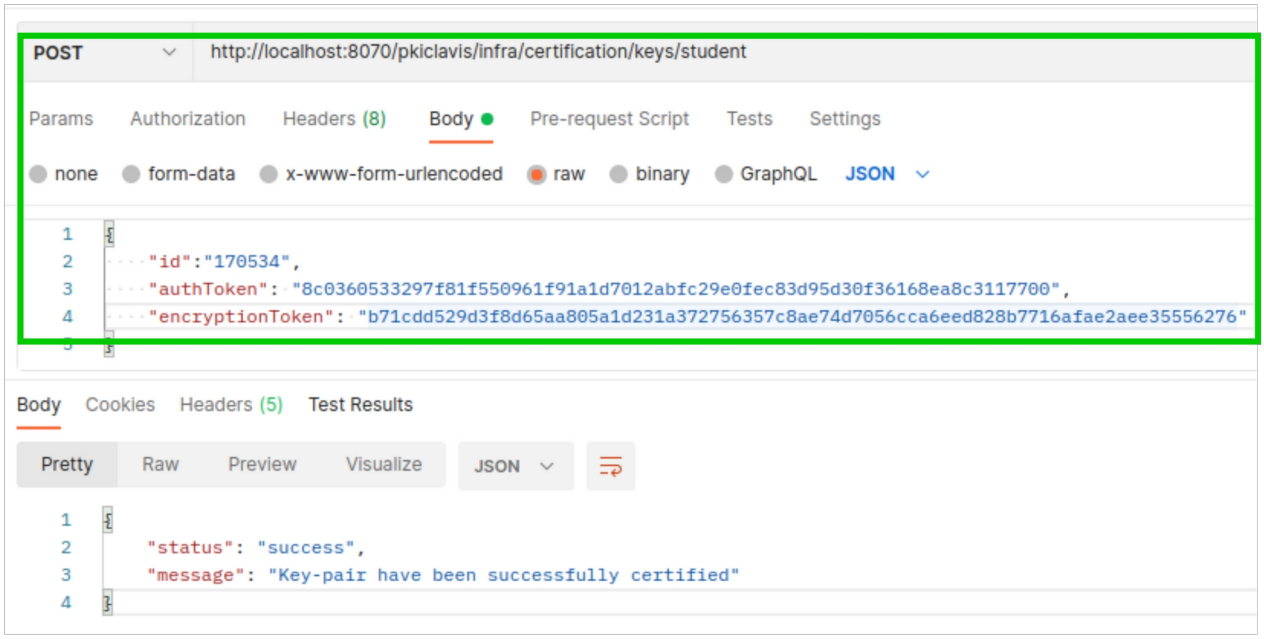

Figure 5.5 – Request for certified key-pair

On the second step, we switch to logs of Clavis Infra service that are represented in Figure 5.6. Clavis Infra generates new key pair for the client and then constructs certification request to the CA.

```
: Request for key-pair creation: StudentCertData(id=170534, authToken=8c0360533297f81f5
: Generating new key-pair for user: 170534
: [170534] Key-pair has been generated
: [170534] Generating CertificationRequest
: [170534] CertificationRequest has been generated
: [170534] Received CertificationResponse with status: 1 ( GRANTED_WITH_MODS )
: [170534] Generating ConfirmationRequest
: [170534] ConfirmationRequest has been generated
: [170534] Key-pair has been certified
: [170534] Persisting key-pair in storage...
: [170534] Key-pair has been persisted
```

Figure 5.6 – Log output of Clavis Infra (preparing the Certification request)

Then, on the side of CA, we receive and process the request. The log output of Clavis Exa is shown in Figure 5.7. CA verifies the incoming request data, issues the certificate and build a certification response for the client.

Besides, according to the defined flow, we will publish certificate only after receiving confirmation from client. Therefore, we persist certificate in separate storage, marked as ‘unconfirmed’.

```
: Certification request for user: 170534
: [170534] Generating CertificationResponse
: [170534] CertificationResponse has been generated

: [170534] Persisting issued certificate as unconfirmed...
: [170534] Certificate has been saved as unconfirmed...
: Confirmation request from user: 170534
: [170534] Searching for unconfirmed certificate...
: [170534] Unconfirmed certificate has been found
: [170534] Publishing issued certificate
: [170534] Certificate ahs been published
```

Figure 5.7 – Log output of Clavis Exa (processing the Certification request)

After this, we check Clavis Infra to receive the CA’s response and format confirmation request (see Figure 5.8) During this part, Clavis Infra also persists the certified key-pair into repository (see Figure 5.9).

```
: Request for key-pair creation: StudentCertData(id=170534, authToken=8c0360533297f81f55
: Generating new key-pair for user: 170534
: [170534] Key-pair has been generated
: [170534] Generating CertificationRequest
: [170534] CertificationRequest has been generated
: [170534] Received CertificationResponse with status: 1 ( GRANTED_WITH_MODS )
: [170534] Generating ConfirmationRequest
: [170534] ConfirmationRequest has been generated
: [170534] Key-pair has been certified
: [170534] Persisting key-pair in storage...
: [170534] Key-pair has been persisted
```

Figure 5.8 – Log output of Clavis Infra (preparing the Confirmation request)

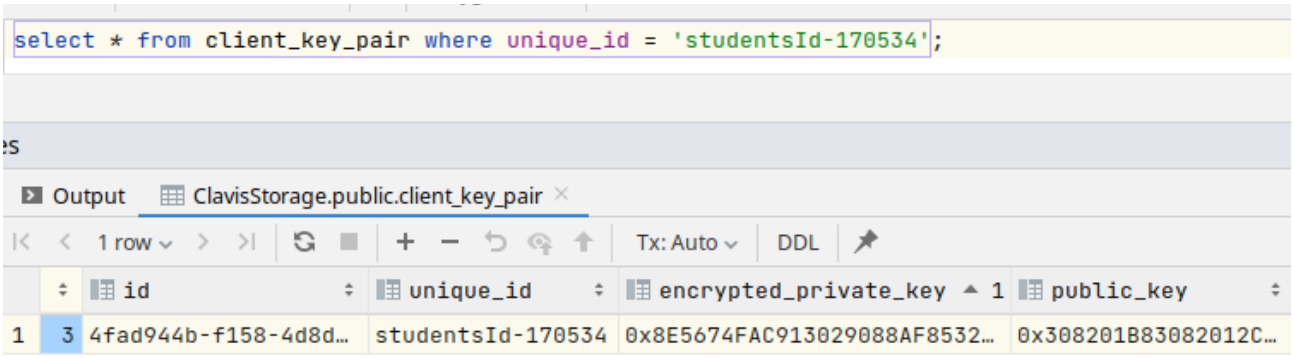

Figure 5.9 – Persisted certified key-pair

Finally, we switch back to the Clavis Exa and check the received confirmation request, publish confirmed certificate to repository (see Figure 5.10 and 5.11 respectively).

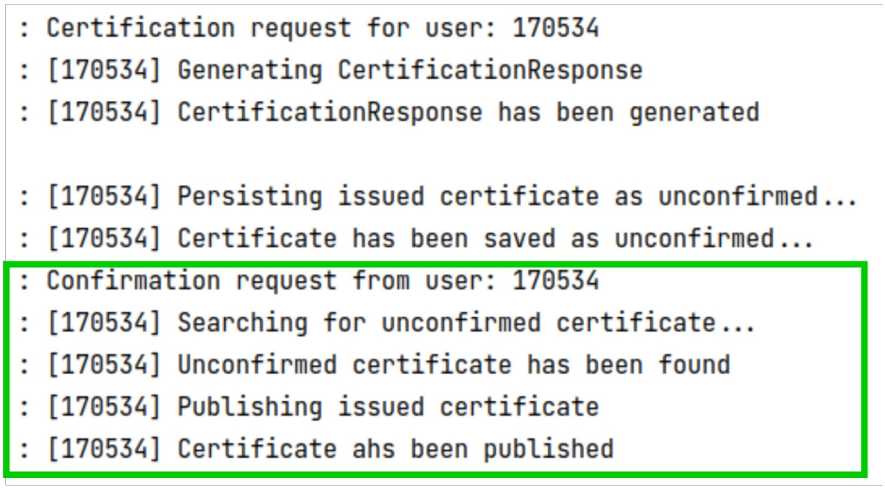

Figure 5.10 – Log output of Clavis Exa (processing the Confirmation request)

At the end we can switch back to the Postman and see the successful response from CA Infra that is shown in Figure 5.12.

```
            Version: 3
       SerialNumber: 1
           IssuerDN: CN=CA Clavis,O=org.diit.evm.ca-clavis,UniqueIdentifier=CAId:12345
         Start Date: Mon Dec 12 00:11:59 CET 2022
         Final Date: Fri Dec 16 04:11:59 CET 2022
          SubjectDN: CN=Clavis Infra,O=org.diit.evm.clavis-infra,UniqueIdentifier=170534
         Public Key: DSA Public Key [6f:d1:f9:34:5c:be:51:b9:b7:14:63:db:3c:7b:4f:14:62:e4:7b:b0]
          Y: 5c6be2e39bd9c4fe7104c20a745d7a2d714aabd95fd0f847c2e5b80e15413ab53949d204a9d00e3b32c2f

Signature Algorithm: SHA512WITHRSA
          Signature: e16766b182dadd01baabefa0d343ec46e0634248
                     911ba6ef65279f87439529b5c70486b61981e10c
                     88fa027d36e719196038cdb9b211029cbc7b1941
                     61f743052d3cce18856dd8623c4a8c9a3c781b92
                     a1313db3dcfa5460c8b2793be5b8c59b3dc270af
                     33f6cb371d4f3a6c14044646159dce9b2a61f94b
                     02758cacca184464e7cbc0a525e3b184a2b24b06
                     4ceb708a9231be25a0742f5e54a006b64c05f0bb
                     0d112416edea797b8e917b695e045322e99d5637
                     e0f1bbc1c8db8307d1f29ad948b4059ac1693e38
                     3e002bd47418a0cd9bc8b7276db1fe004e74867b
                     81e330d81b7cf999a3c8e8c9210dc40fbf30c3f7
                     3a6ac9332a4398e959ced750871c2fd4327887c9
                     a27f91d43774b5735a16546af4224d386f34a914
```

Figure 5.11 – Fragment of issued certificate

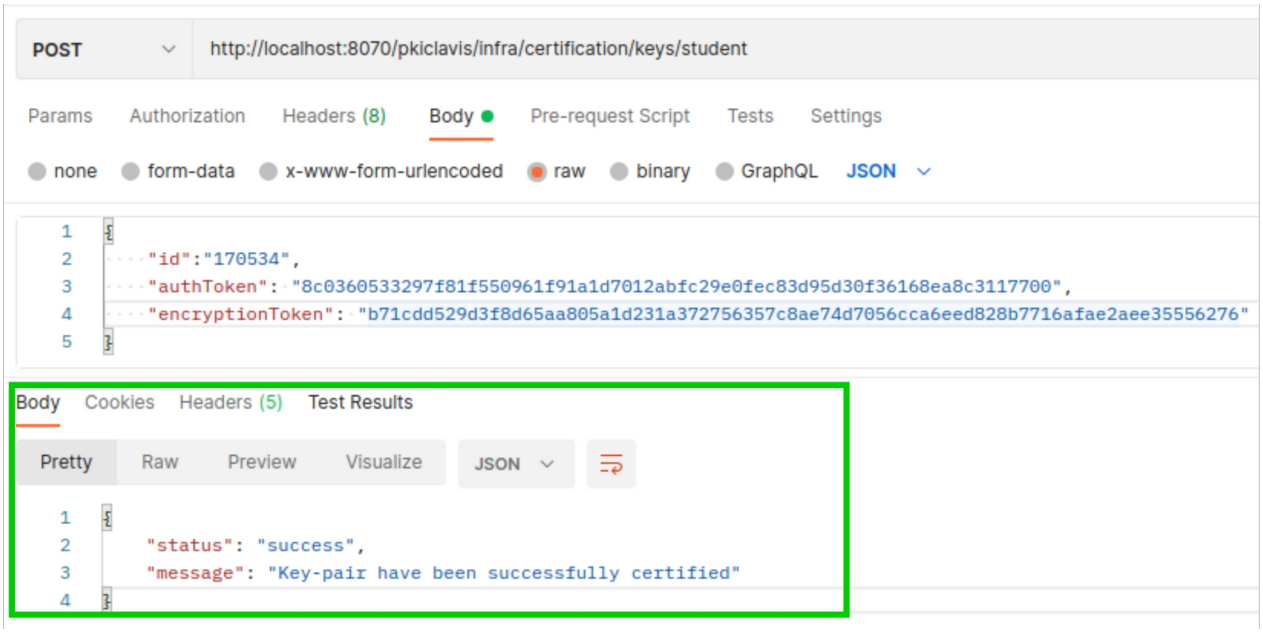

Figure 5.12 – Successful result of request for certified key-pair

5.2 Starting point of modernization of the studying process

The PKI Clavis system can be considered as experimental version of PKI environment that can be fully developed and integrated into university processes.

On the current stage of implementation, the developed Clavis Infra and Clavis Exa services can be used as basis for future PKI development. The functionality of Clavis Exa must be completed and Clavis Infra services can be extended, including separate authentication services, Registration Authority, services responsible for signing electronic documents with digital signature etc.

Utility libraries – Crypto Core and Core Web are the tools for simplified and faster development of PKI as it exactly provides low-level functions related to the common cryptographic operations in PKI and ready-to-use CMP objects mappers and formatters.

Moreover, Clavis Root, Clavis exa and Clavis Infra services together with simplified key and certificate repository interfaces can be useful for students to practice in PKI technology. The core libraries can be also extended to support, for example, OCSP or Certificate Transparency protocols in build-in services, that will allow student to easy construct custom PKI configuration and test it with monitoring of log output and repository content.

5.3 Summary

In this section we show the flow of Certificate issuance in PKI Clavis in action. We implemented the PKI client – Clavis Infra service and issuing Certificate Authority service – Clavis Exa. Firstly we configured Clavis Root – root offline CA, generated key-pair and created the self-signed root certificate. After we set up the Clavis Exa – generate key-pair and signed the certificate with root CA.

After initial configuration we provided the issuance flow in details. Firstly Clavis Infra receives the request from external client (for example Lider service) to create a certified key-pair for the student. Clavis Infra generates the key-pair and formats the Certification Request to the Clavis Exa. The last one receives Certification Request, verifies the incoming data and issues a certificate, saves this certificate as ‘unconfirmed’ and sends a response back to the Clavis Infra. Clavis Infra receives a response with Certificate, verifies the data and sends the Confirmation request to the CA. Clavis Exa receives confirmation, checks the status of Certificate in this request (because it can also be 'rejected') and publishes the issued certificate to the repository as 'confirmed'. Meanwhile Clavis Infra sends response back to the external client with 'success' status. With this the process is ended.

Finally, we described the options of usage of the current PKI Clavis version in the education process and also further development of the PKI system in university.

SUMMARY

In this diploma project we performed a detailed research in the PKI environment: components, functions, management protocols, transport basic protocols, main requirements The Certificate Authority component has been described with its functionality, architecture types, communication processes with other PKI components. We have also learned certificate and key-pair lifecycles in order to have a better vision of PKI algorithms of work.

Integrating the PKI technology into the university system, we investigated the legislative framework and law basis for Certificate Authority in Ukraine. With fulfillment of all requirements, the digital signatures that are verified by Certificate Authority can be equal to the real signatures on paper and notarially valid.

Before to create an architecture of the university's PKI system, we looked at popular ready-to-use PKI solutions and analyzed if one of them is applicable in our case and can be a good choice to integrate. As a result, we highlight the main advantages and disadvantages of these products and consider to build our own PKI system with configuration efficient

Based on performed research, we designed our own PKI system called PKI Clavis. We analyzed which architecture and protocols are efficient in our case. We provided a resolution for a certificate revocation issue that is frequently in our scope.

Finally, we implement the initial version of the PKI system, including root CA, issuing CA, PKI client, provided key repository and certificate / CRL repository interfaces. Besides, we created two Java libraries that have prepared methods to build a PKI flow more easily, from small 'ready-to-use' blocks. The PKI Clavis is ready to work as the first basic version of PKI as a sample of a future integrated PKI system in university, can be used for educational purposes as a detailed example of PKI flows in action and can be easily developed with usage of previously mentioned prepared libraries.

RESOURCES

1. John R. Vacca, Public key infrastructure : building trusted applications and Web services / John R. Vacca – Auerbach Publications, 2019. – 404 p.

2. Long-Term Archive Service Requirements [Electronic resource] – Access mode: https://datatracker.ietf.org/doc/rfc4810/

3. Johannes A. Buchmann, Introduction to Public Key Infrastructures / Johannes A. Buchmann, Evangelos Karatsiolis, Alexander Wiesmaier – Springer-Verlag Berlin Heidelberg, 2013. – 194 p.

4. Suranjan Choudhury, Public Key Infrastructure Implementation and Design / Suranjan Choudhury, Kartik Bhatnagar, Wasim Haque – Hungry Minds, 2002. – 316p.

5. Recommendation for Key Management Part 3: Application-Specific Key Management Guidance [Electronic resource] – Access mode: http://dx.doi.org/10.6028/NIST.SP.800-57pt3r1

6. Gorbatov V. S., Osnovi tehnologii PKI / Gorbatov V. S., Polyanskaya O. Y. - M .: Goryachaya liniya-Telecom, 2004. - 248 p.

7. C. Ellison, Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure / C. Ellison, B. Schneier – Computer Security Journal, v 16, n 1, 2000. – pp. 1-7.

8. Recommendation for Key Management: Part 1 – General [Electronic resource] – Access mode: https://doi.org/10.6028/NIST.SP.800-57pt1r5

9. Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP) [Electronic resource] – Access mode: https://www.rfc-editor.org/rfc/rfc4210

10. DNS Certification Authority Authorization (CAA) Resource Record [Electronic resource] – Access mode: https://www.rfc-editor.org/rfc/rfc8659

11. Compliance FAQs: Federal Information Processing Standards (FIPS) [Electronic resource] – Access mode: https://www.nist.gov/standardsgov/compliance-faqs-federal-information-processing-standards-fips

12. Build Your Own Certificate Authority (CA) [Electronic resource] – Access mode: https://developer.hashicorp.com/vault/tutorials/secrets-management/pki-engine

13. Elaine Barker, Recommendation for Key Management: Part 1 – General / Elaine Barker NIST – Special Publication 800-57 Part 1 Revision 5, 2020. – 170 p.

14. EJBCA Introduction [Electronic resource] – Access mode: https://doc.primekey.com/ejbca/ejbca-introduction

15. PKCS #7: Cryptographic Message Syntax [Electronic resource] – Access mode: https://www.ietf.org/rfc/rfc2315

16. PKCS #10: Certification Request Syntax Specification[Electronic resource] – Access mode: https://www.rfc-editor.org/rfc/rfc2986

17. Klaus Schmeh, Cryptography and Public Key Infrastructure on the Internet / Klaus Schmeh – John Wiley & Sons Ltd, 2003. – 472 p.

18. Stattya 8. Centr sertyfikaciyi klyuchiv [Electronic resource] – Access mode: https://protocol.ua/ua/pro_elektronniy_tsifroviy_pidpis_stattya_8/

19. Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework [Electronic resource] – Access mode: https://www.rfc-editor.org/rfc/rfc2527

20. Farid F. Elwailly, QuasiModo: Efficient Certificate Validation and Revocation. In: Bao, F., Deng, R., Zhou, J. (eds) Public Key Cryptography – PKC 2004. PKC 2004. Lecture Notes in Computer Science / Farid F. Elwailly, Craig Gentry, Zulfikar Ramzan – Springer, Berlin, Heidelberg, 2004. – 334p.

21. Kapil Raina, PKI Security Solutions for the Enterprise: Solving HIPAA, E-Paper Act, and Other Compliance Issues / Kapil Raina – Wiley Publishing, Inc., 2003. – 334p.

22. What is a Public Key Infrastructure Example (PKI Architecture) [Electronic resource] – Access mode: https://cloudinfrastructureservices.co.uk/what-is-a-public-key-infrastructure-example-pki-architecture/

23. Andrew Nash, PKI: Implementing and Managing E-Security / William Duane, Celia Joseph, Derek Brink – McGraw-Hill/OsborneMedia, 2001. – 545 p.

24. Exploring PKI weaknesses and how to combat them [Electronic resource] – Access mode: https://www.redhat.com/sysadmin/pki-protection

25. Internet X.509 Public Key Infrastructure Certificate and CRL Profile [Electronic resource] – Access mode: https://datatracker.ietf.org/doc/html/rfc2459

26. Managed Root Certificate Authority (CA) [Electronic resource] – Access mode: https://www.entrust.com/-/media/documentation/brochures/pk19-1009-001-offline-root-service_v6.pdf

Appendix A

Sample of Root Certificate Authority (Clavis Root) configuration file

```
[ ca ]
default_ca = CLAVIS_ROOT

[ CLAVIS_ROOT ]
prompt            = no
dir               = /home/kiberohrannik/pkiclavis/
certs             = $dir/certs
crl_dir           = $dir/crl
new_certs_dir     = $dir/certificates
database          = $dir/index.txt
serial            = $dir/serial
RANDFILE          = $dir/private/.rand
private_key       = $dir/root_ca_keys.key
certificate       = $dir/certs/clavis_root.crt
crlnumber         = $dir/crlnum
crl               = $dir/crl/clavis_root_crl.pem
default_crl_days  = 30
preserve          = no
policy            = policy
default_days      = 1095

[ policy ]
commonName              = supplied
stateOrProvinceName     = supplied
countryName             = supplied
emailAddress            = supplied
organizationName        = supplied
organizationalUnitName  = supplied

[ req ]
default_bits        = 4096
distinguished_name  = req_distinguished_name

string_mask         = utf8only
default_md          = sha256
x509_extensions     = v3_ca

[ req_distinguished_name ]
countryName                     = UA
stateOrProvinceName             = DN
localityName                    = Dnipro
organizationName                   = Ukrainian State University of Science and
Technologies
organizationalUnitName          = CA Server
commonName                      = Clavis Root
emailAddress                    = university_email@example.com

[ v3_ca ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature
```

Appendix B

Sample of Issung Certificate Authority (Clavis Exa) configuration file

```
[ ca ]
default_ca = CLAVIS_ROOT

[ CLAVIS_EXA ]
prompt            = no
dir               = /home/kiberohrannik/pkiclavis/
certs             = $dir/certs_exa
crl_dir           = $dir/crl
new_certs_dir     = $dir/certificates_exa
database          = $dir/index_exa.txt
serial            = $dir/serial_exa
RANDFILE          = $dir/private/.rand
private_key       = $dir/exa_ca_keys.pem
certificate       = $dir/certs_exa/clavis_exa0.crt
crlnumber         = $dir/crlnum
crl               = $dir/crl/clavis_exa_crl.pem
default_crl_days  = 30
preserve          = no
policy            = policy
default_days      = 730

[ policy ]
commonName              = supplied
stateOrProvinceName     = supplied
countryName             = supplied
emailAddress            = supplied
organizationName        = supplied
organizationalUnitName  = supplied

[ req ]
default_bits        = 4096
distinguished_name  = req_distinguished_name

string_mask         = utf8only
default_md          = sha256
x509_extensions     = v3_exa_ca

[ req_distinguished_name ]
countryName             = UA
stateOrProvinceName     = DN
localityName            = Dnipro
organizationName        = Ukrainian State University of Science and Technologies
organizationalUnitName  = CA Server
commonName              = Clavis Exa
emailAddress            = university_email@example.com

[ v3_exa_ca ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature
```