МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Український державний університет науки і технологій

Кафедра «Електронні обчислювальні машини»

«ДО ЗАХИСТУ»

Завідувач кафедри

_____ Жуковицький І. В.
(підпис) (ПІБ)

«21» 12 2021 р.

ДИПЛОМНА РОБОТА

на здобуття освітнього ступеня «магістр»

Галузь знань 12 Інформаційні технології
(шифр) (назва)

Спеціальність 123 Комп'ютерна інженерія
(код) (повна назва)

Тема Програмно-апаратний комплекс надання електронних довірчих послуг

Theme Software and hardware system for providing electronic trust services

Керівник дипломного проекту зав. кафедри _____ Жуковицький І. В.
(посада) (підпис) (ПІБ)

Консультант розділу з БЖД доцент _____ Саблін О. І.
(посада) (підпис) (ПІБ)

Нормоконтролер доцент _____ Шаповалов В. О.
(посада) (підпис) (ПІБ)

Студент групи _____ _____ Колядін М. О.
(посада) (підпис) (ПІБ)

Student Koliadin Mykyta
(family name)

Дніпро
2021

Довідка

про відсутність плагіату у випускній кваліфікаційній роботі

Міністерство освіти і науки України

Український державний університет науки і технологій

Кафедра Електронні обчислювальні машини

ДОВІДКА

За результатами перевірки випускної кваліфікаційної роботи здобувача вищої освіти Колядіна Микиті Олександровича

(прізвище, ім'я, по батькові)

на тему: Програмно-апаратний комплекс ведення електронних довірчих послуг

в роботі не виявлено порушень академічної доброчесності.

Керівник ВКР _____

Український державний університет науки і технологій

Факультет КТС кафедра ЕОМ
Спеціальність Комп'ютерна інженерія

«ЗАТВЕРДЖУЮ»
Завідувач кафедри

_____
(підпис)
«___» _____ 20 р.

ЗАВДАННЯ

до дипломної роботи на здобуття освітнього ступеня _____
(освітнього ступеня)

студента групи КС2021 Колядіна Микити Олександровича
(номер групи) (ПІБ)

1 Тема дипломної роботи Програмно-апаратний комплекс надання електронних довірчих послуг

затверджена наказом по університету від «___» _____ 20 р. № _____.

2 Термін подання студентом закінченої роботи _____

3 Вихідні дані до дипломної роботи _____
1. Матеріали науково-дослідної та переддипломної практик.
2. Нормативно-законодавча база України та міжнародні стандарти у сфері інформаційної безпеки та кібербезпеки, наукові публікації щодо надання електронних довірчих послуг.
3.Технічна документація до мережного криптомодуля "Гряда-301".
4. Опис технології SmartID.

4 Зміст пояснювальної записки (перелік питань до розробки) _____
Вступ Розділ 1 - Аналіз понятійної бази електронного цифрового підпису Розділ 2 - Розробка комплексу надання електронних довірчих послуг із використанням хмарного сховища Розділ 3 - Дослідження економічної доцільності розробки комплексу Розділ 4 - Охорона праці та безпека в надзвичайних ситуаціях Висновки

5 Перелік креслень (демонстраційного матеріалу) _____

6 Розділи та консультанти

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та безпека в надзвичайних ситуаціях	Саблін О. І.		

КАЛЕНДАРНИЙ ПЛАН

Назва розділу	Термін виконання	Обсяг розділу, %
Вступ		5
Аналіз понятійної бази електронного цифрового підпису		20
Розробка комплексу надання електронних довірчих послуг із використанням хмарного сховища		45
Дослідження економічної доцільності розробки комплексу		15
Охорона праці та безпека в надзвичайних ситуаціях		10
Висновки		5

Дата видачі завдання: «___» _____ 20___ р.

Керівник дипломної роботи _____ _____
(підпис) (ПІБ)

Завдання прийняв до виконання _____ _____
(підпис) (ПІБ)

РЕФЕРАТ

Пояснювальна записка: 124 с., 11 рис., 4 табл., 2 додатків, 40 джерел.

Об’єкт досліджень: процес надання електронних довірчих послуг.

Метою дипломної роботи є вдосконалення процедури використання кваліфікованого електронного підпису при наданні електронних довірчих послуг.

Предмет досліджень: способи зберігання кваліфікованого електронного підпису.

В першому розділі дипломної роботи проведено аналіз технології надання електронного підпису, можливих варіантів його використання і зберігання. Показана актуальність проблеми зберігання електронного підпису на підставі прийняття нових законів та поставлена задача для її вирішення.

В другому розділі дипломної роботи проаналізовано функції кваліфікованих надавачів довірчих послуг на прикладі діючого центру сертифікації, проведено аналіз технологічної схеми та розроблені алгоритми роботи створеного комплексу надання електронних довірчих послуг. Розроблено програмне забезпечення для взаємодії із зовнішнім інтерфейсом комплексу.

В третьому розділі наведено дослідження економічної доцільності впровадження комплексу в технологічну структуру кваліфікованих надавачів електронних довірчих послуг.

Практична цінність роботи полягає у вдосконаленні процесу використання та зберігання кваліфікованого електронного підпису.

ЕЛЕКТРОННИЙ ПІДПИС, ЕЛЕКТРОННІ ДОВІРЧІ ПОСЛУГИ, ХМАРНЕ СХОВИЩЕ КЛЮЧІВ, ЦИФРОВІ СЕРТИФІКАТИ, ЦЕНТР СЕРТИФІКАЦІЇ КЛЮЧІВ

ABSTRACT

Explanatory note: 124 p., 11 fig., 4 tab, 2 applications, 40 sources.

Object of research: the process of providing electronic trust services.

Purpose of degree work: improve the use of qualified electronic signatures in the provision of electronic trust services.

Subject of research: methods for storing a qualified electronic signature.

In the first chapter, an analysis is made of the technology for providing electronic signatures, possible options for its use and storage. The urgency of the problem of storing electronic signatures on the basis of the adoption of new laws is shown and the task is set for its solution.

In the second chapter, the functions of qualified trust service providers are analyzed using the example of an existing certification center, an analysis of the technological scheme is carried out, and work algorithms for the created complex of electronic trust services are developed. Created software for interaction with the external interface of the complex.

The third chapter provides an research of economic justification for the feasibility of introducing the complex into the technological structure of qualified suppliers of electronic trust services.

The practical value of the work is to improve the use and storage of a qualified electronic signature.

ELECTRONIC SIGNATURE, ELECTRONIC TRUST SERVICES, CLOUD KEY STORAGE, DIGITAL CERTIFICATES, KEY CERTIFICATION CENTER

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

API	–	application programming interface;
CMP	–	certificate management protocol;
OCSP	–	online certificate status protocol;
RA	–	registration authority;
TSP	–	time stamp protocol;
ЕЦП	–	електронний цифровий підпис;
КЕП	–	кваліфікований електронний підпис;
КМУ	–	кабінет міністрів України;
НД	–	нормативний документ;
НКІ	–	носій ключової інформації;
ПК	–	персональний комп’ютер;
ПТК	–	програмно-технічний комплекс;
ЦЗО	–	центральний засвідчувальний орган;
ЦСК	–	центр сертифікації ключів;

ВСТУП

Через активний розвиток інформаційних технологій все більше і більше сфер життя інтегровані чи повністю перенесені в онлайн-середовище. Онлайн-системи дозволяють виконувати нові функції та роблять зручнішою взаємодію з усталеними структурами. Вже зараз існує безліч технологій, які дозволяють безпечно проводити фінансові онлайн-транзакції, відправляти або отримувати конфіденційні файли. Завдяки цьому такі відповідальні справи, як підпис документів або укладання контрактів можуть та мають бути інтегровані в онлайн застосунки. Адаптація процесу підпису в мережу зможе забезпечити в більш зручний процес документообігу. Зараз в законодавчих системах багатьох країн, включаючи Україну, використовується термін ЕЦП (електронний цифровий підпис).

На даний момент в Україні ЕЦП має юридичну силу та вважається рівноправним підпису або печатці. Треба зазначити, що сам принцип роботи ЕЦП дозволяє однозначно визначати, чи був конкретний документ підписаний конкретною людиною. Завдяки ЕЦП можно значно скоротити час руху документів під час оформлення звітів і обміну документацією між людьми.

Безпека ЕЦП базується на проходжені спеціальної експертизи та сертифікації засобів, які використовуються при роботі з ЕЦП, в Держаній службі спеціального зв’язку та захисту інформації України, що гарантує неможливість злому і підробки ЕЦП.

З моменту прийняття Закону України "Про електронні довірчі послуги" тільки кваліфікований електронний підпис прирівнюється за юридичною силою до власноручного підпису. Згідно Закону виникає проблема в необхідності зберігання підпису виключно в засобі кваліфікованого електронного підпису чи печатки, що негативно впливає на роботу надавачів електронних довірчих послуг. У роботі були поставлені такі задачі:

- Провести аналіз чинного законодавства України, дослідити стан процесу використання електронного цифрового підпису та обґрунтувати необхідність створення комплексу;

- Проаналізувати технологічну структуру та призначення центрів сертифікації ключів;
- Розробити алгоритм роботи комплексу та обґрунтувати його застосування;
- Розробити програмний засіб для реалізації взаємодії з комплексом;
- Дослідити економічну доцільність на розробку та впровадження комплексу в технологічну структуру кваліфікованих надавачів електронних довірчих послуг.

У розділі 4 подані основні правила безпеки при роботі за комп'ютером, та дії при виникненні надзвичайної ситуації і правила надання домедичної допомоги.

Метою дипломної роботи є вдосконалення процедури використання кваліфікованого електронного підпису.

Об'єкт досліджень: процес надання електронних довірчих послуг.

Предмет досліджень: способи зберігання кваліфікованого електронного підпису.

Наукова новизна: розробка та програмна реалізація алгоритмів підпису та генерації із використанням апаратного криптомодуля.

Практична цінність: вдосконалення процесу використання та зберігання кваліфікованого електронного підпису.

РОЗДІЛ 1
АНАЛІЗ ПОНЯТІЙНОЇ БАЗИ ЕЛЕКТРОННОГО ЦИФРОВОГО ПІДПИСУ

1.1 Призначення ЕЦП. Загальна характеристика

Останнім часом, у зв’язку з бурхливим розвитком комп’ютерної техніки і комп’ютерних мереж загального доступу, виникла можливість перенесення діяльності суб’єктів господарювання та державних органів управління в кіберпростір.

Поняття кіберпростіру чітко визначенно в законі України “Про основні засади забезпечення кібербезпеки”. Тож згідно до [1] кіберпростір це середовище (віртуальний простір), яке надає можливості для здійснення комунікацій та/або реалізації суспільних відносин, утворене в результаті функціонування сумісних (з’єднаних) комунікаційних систем та забезпечення електронних комунікацій з використанням мережі Інтернет та/або інших глобальних мереж передачі даних.

Інформаційні технології, надаючи можливості для вирішення одних соціальних проблем, викликають загострення інших або породжують нові, раніше невідомі проблеми, становляться джерелом нових ризиків. Якщо не достатньо прийнятих заходів щодо нейтралізації негативних факторів, дія від впровадження інформаційних технологій у соціально-економічних системах може бути багато в чому негативною. Тому забезпечення безпеки інформаційних систем та аналіз ризиків потребує постійної уваги.

Розвиток комп’ютерної техніки та її широке використання державними органами, комерційними та приватними суб’єктами підприємницької діяльності привели до виникнення і розповсюдження “комп’ютерних злочинів”. Це викликає занепокоєння не лише на підприємствах, де є комп’ютерна техніка, а й в органах підтримки правопорядку, серед користувачів мережі Інтернет, які використовують нові види інформаційного обслуговування. Електронну комерцію можна розглядати як одну із сучасних форм організації і здійснення

господарської, переважно банківської і торговельної, діяльності. Електронна комерція має(надає) ряд переваг в економічному відношенні.

Відмінною рисою даної форми діяльності є використання загальнодоступних інформаційних систем та комп'ютерних мереж, об'єднаних Інтернетом.

Важливим стримуючим фактором розвитку електронної комерції є відсутність необхідних умов для оформлення угод які проведяться за допомогою Інтернету. Ця ситуація робить актуальною проблему використання і оформлення електронних документів (документів в електронній формі).

Опираючись на закон України "Про електронні довірчі послуги" [2] для виконання електронним документом функцій, що на нього покладаються, використовується електронний цифровий підпис (ЕЦП). ЕЦП за допомогою спеціального програмного забезпечення підтверджує достовірність інформації документу, його реквізитів і факту підписання конкретного документу конкретною особою.

Електронний підпис - електронні дані, які додаються підписувачем до інших електронних даних або логічно з ними пов'язуються і використовуються ним як підпис [2].

Електронний цифровий підпис - вид електронного підпису, отриманого за результатом криптографічного перетворення набору електронних даних, який додається до цього набору або логічно з ним поєднується і дає змогу підтвердити його цілісність та ідентифікувати підписувача. Електронний цифровий підпис накладається за допомогою особистого ключа та перевіряється за допомогою відкритого ключа [2].

Цифровий підпис дозволяє здійснити:

1) Аутентифікацію особи - автора електронного документа.

2) Контроль цілісності переданого документа: при будь-якій випадковій або навмисній зміні документа підпис стане недійсним, тому що він обчислений на підставі вихідного стану документа і відповідає лише йому.

3) Захист від модифікації документа через наявність можливості контролю цілісності. Гарантія виявлення підробки при контролі цілісності робить підробку недоцільною у більшості випадків.

4) Неможливість відмови від авторства. ЕЦП створюється із використанням закритого ключа, який повинен бути відомим тільки підписанту, що не дає змоги відмовитися від свого підпису під документом.

5) Доказове підтвердження авторства документа. Створити коректний підпис можливо, лише знаючи закритий (особистий) ключ, а він повинен бути відомим лише власнику, через що власник пари ключів може довести своє авторство під документом. В документі можуть та мають бути підписані такі поля як “мітка часу”, “автор”, “внесені зміни” і т.п.

1.2 Область застосування ЕЦП

Завдяки своїм властивостям електронний підпис використовують в таких основних цілях сферах електронної економіки і електронного документообігу:

- Використання в банківських платіжних системах.
- Електронна комерція (торгівля).
- Електронна реєстрація угод по об'єктах нерухомості.
- Митне декларування товарів і послуг (митні декларації).
- Контролюючі функції виконання державного бюджету (якщо мова йде про країну) і виконання кошторисних призначень та лімітів бюджетних зобов'язань (в даному випадку якщо розмова йде про галузь або про конкретний бюджетній установі).
- В електронних системах звернення громадян до органів влади, в тому числі і з економічних питань.
- Формування обов'язкової податкової (фіскальної), бюджетної, статистичної та іншої звітності перед державними установами і позабюджетними фондами.

- Організація юридично легітимного внутрішньокорпоративного, внутрішньогалузевого або національного електронного документообігу.
- Застосування ЕЦП в різних розрахункових і трейдингових системах, а також Forex.
- Управління акціонерним капіталом.

1.3 Аналіз видів шифрування

Алгоритми шифрування здебільшого поділяються на асиметричні (з відкритим або публічним ключем) та симетричні (з приватним, секретним ключем або одноключові). Симетричний алгоритми шифрування базуються на тому принципі, що і відправники, і одержувачі користуються однаковим ключем. Таємний (особистий) ключ повинен триматися в секреті та передаватись таким чіном, щоб запобігти його перехоплення. В тому разі, якщо таємний ключ захищений то при розшифруванні повідомлення автоматично відбудеться автентифікація відправника. Лише він є власником ключа, за яким можливо зашифрувати інформацію. Лише отримувач має ключ, за допомогою якого можна дешифрувати повідомлення. Власниками та єдиними користувачами симетричного ключа є лише відправник та одержувач то при спробі несанкціонованого доступа до ключа та спробі його використання буде скомпрометована взаємодія виключно цих двох користувачів. Згідно до аналізу дипломної роботи “Система захисту даних на базі методу шифрування” [3] маємо, що безпека цього типу системи шифрування залежить від збережності забезпечення захищеності ключа, що використовується в алгоритмі шифрування, а не від зберігання в секреті алгоритму. Зазвичай симетричний варіант ЕЦП передбачає присутність в системі третьої сторони – "арбітра", який виступає довіреною стороною обох користувачів. Аутентифікація документу визначається самим фактом зашифрування його секретним ключем і передачею

документу арбітру. Використовується рідко через відсутність ефективних алгоритмів.

Проаналізувавши дипломну роботу “Система захисту даних на базі методу шифрування” [3] ми можемо вилучити переваги симетричного методу генерування електронного цифрового підпису:

- криптостійкість симетричних схем має високі показники за рахунок стійкості блочних шифрів, що використовуються. Їх надійність також достатньо досліджена;
- якщо стійкість шифру недостатня, його легко замінити іншим.

Недоліки симетричних методів для створення ЕЦП [3]:

- необхідно підписувати кожен з бітів інформації, що значно збільшує довжину підпису (він часто буває на порядок довшим за довжину документу);
- створені для підпису ключі можна використати лише один раз, тому що після підпису розсекречується половина секретного ключа.

Будь-які криптосистеми, які використовують закритий ключ, залежать від ступеня секретності даних. Власник може зберігати ключ на своєму власному комп’ютері захистивши його паролем. Але такий варіант має свої недоліки [3]:

- підписувати документи можна тільки на комп'ютері власника ЕЦП;
- збереження даних ЕЦП безпосередньо залежить від захищеності комп'ютера користувача.

Проаналізувавши дипломну роботу “Система захисту даних на базі методу шифрування” [3] маємо, що алгоритми шифрування, які називають асиметричними (або шифрування з публічним ключем), на відміну від симетричних схем, використовують два ключі - секретний ключ (secret key или private key) і відкритий ключ (public key), які створені таким чином, що їх послідовне застосування до інформаційного об’єкту не змінює цей об’єкт. Знаючи лише один ключ, неможливо відновити інший, незважаючи на те, що вони згенеровані разом. При зашифруванні використовується тільки відкритий ключ, для дешифрування використовується лише секретний ключ. Розшифровка коду безсекретного ключа – надмірно складна задача. Зокрема, задача

розрахунку секретного ключа по відомому відкритому ключу є практично нерозв'язною [3].

Основною перевагою шифрування з використанням публічного ключа є простий механізм передачі ключів. В мережі передається лише відкритий ключ. Це дозволяє використовувати звичайний канал і виключає потребу у використанні спеціального каналу безпеки для пересилання ключа [3]. Опираючись на доповідь про "Електронний цифровий підпис" [4] маємо, що на відміну від асиметричних алгоритмів шифрування, в яких зашифрування проводиться за допомогою відкритого ключа, а розшифрування - за допомогою закритого, в схемах цифрового підпису підписування проводиться із застосуванням закритого ключа, а перевірка - із застосуванням відкритого.

В асиметричної моделі побудови ЕЦП є свої недоліки [3]:

- асиметричні схеми ЕЦП базуються, як і асиметричне шифрування, на обрахунково складних задачах (задача дискретного логарифмування або задача факторизації);
- для покращення криптостійкості необхідно робити довжину ключів більшою.

Лише криптосистеми з відкритими ключами широко застосовуються для генерування ЕЦП. Цифрові підписи, базовані на асиметричному шифруванні, дозволяють одержувачу зробити перевірку повідомлення на автентичність джерела інформації (автора інформації), та перевірити, чи змінилася інформація, під час передачі її адресату. Тож виходить, що цифрові підписи є також засобом аутентифікації та контролю цілісності даних.

1.4 Аналіз алгоритмів формування ЕЦП

Основним документом, який регулює процедуру створення та перевірки ЕЦП, є ДСТУ 4145-2002 "Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка" [5]. Розпоширеності набувають системи ЕЦП, які дають можливість як проводити верифікацію підпису так і відновлювати

початкове повідомлення з цифрового підпису. Усі схеми ЕЦП можна розділити на два великих класи: звичайні цифрові підписи (з доповненням) та електроні цифрові підписи з відновленням повідомлення. В ЕЦП з відновленням частина або повне повідомлення можуть бути відновленими з цифрового підпису, тобто для перевірки цифрового підпису необхідно знати тільки цифровий підпис та, можливо, сертифікат відкритого ключа. В ЕЦП з додаванням – цифровий підпис приєднується до повідомлення та зберігається і передається з ним, а для перевірки ЕЦП потрібно обов'язково мати сертифікат відкритого ключа. Це означає, що документ, який підписується, не буде передаватися відкритим каналом зв'язку бо буде складовою частиною підпису та перевіряючий його абонент отримає доступ тільки в разі підтвердження авторства та цілісності. Маємо, що подібна структура ЕЦП забезпечує одну додаткову послугу безпеки — конфіденційність. ДСТУ 4145-2002 не має здатності до відновлення повідомлення. Опираючись на доповідь про "Сучасні шляхи удосконалення процедури формування та верифікації електронно-цифрового підпису" [6] стандарт ДСТУ 4145-2002 установлює механізм цифрового підписування, оснований на властивостях груп точок еліптичних кривих над полями GF(2m), та правила застосування цього механізму до повідомлень, що пересилаються каналами зв'язку та/або обробляються у комп'ютеризованих системах загального призначення. До недоліків розглянутого алгоритму можна віднести те, що механізм формування підпису та верифікації в самому стандарті досить складно описаний, тобто можуть виникнути труднощі з його розумінням та реалізацією [6]. Крім того, в даному алгоритмі майже немає можливості розпаралелювання дій (тобто все повинно виконуватися лише послідовно).

Але переваг даний стандарт має більше. Перш за все, завдяки тому, що стандарт прийнятий на державному рівні, він має юридичну силу в Україні и є повністю законним. Також, стандарт є гнучким стосовно вибору параметрів безпеки (для нього можна легко обирати будь-яку функцію гешування, крім тієї, що зазначена в самому стандарті). Це дозволяє використовувати і підлаштовувати його під будь-яке апаратне, апаратно-програмне та програмне середовище. Також треба відмітити те, що існують доступні бібліотеки на мовах

C/C++, Java та інш., у яких реалізовані основні криптопримітиви стандарту. Також найбільшими перевагами є невеликий обсяг відкритого ключа (а це забезпечує досить швидку передачу та процес верифікації). Стандарт заснований на математичній проблемі знаходження дискретного логарифму в групі точок еліптичної кривої (ПДЛЕК). Відомо, що алгоритми, які базуються на цій основі, мають більш більшу криптостійкість при малих параметрах, що не можна сказати про алгоритми, які спираються на проблему факторизації чисел (ПФЧ) та проблему дискретного логарифмування в кінцевому полі (ПДЛ).

Згідно [6] детальніше це зображено в таблиці 1.1, у якій наведені значення для різних алгоритмів.

Таблиця 1.1 - Порівняння характеристик алгоритмів

Алгоритм	Проблема в основі	Відкритий ключ, біт
ДСТУ 4145-2002	ПДЛЕК	163-768
DSA	ПДЛ	1024-3072
ECDSA	ПДЛЕК	112-570
RSA	ПФЧ	512-4096

Схема ЕЦП за ДСТУ 4145-2002 належить до схем з доповненням повідомлення, проте існують і схеми з відновленням. Згідно до [6] підпис з відновленням повідомлення, порівняно з підписом з доповненням, надає додаткову послугу безпеки — конфіденційність. Також для невеликих обсягів повідомлення можливо зробити таємною всю інформацію, що передається, у самому підписі. Відновити повідомлення можливо у разі відтворення передпідпису, що у загальному випадку можливо тільки при перевірці ЕЦП. Таким чином, можна стверджувати, що повідомлення можна відновити лише за наявності відкритого ключа. Структурно схеми с відновленням повідомлення відрізняються від схем з доповненням тим, що вони не гешують повністю повідомлення (проте в різних алгоритмах частково можуть використовуватися

геш-функції), а замість них користуються функціями маскування та знаходження збитковостей повідомлення. Такий алгоритм має свої переваги:

− користувач, який верифікує підписане повідомлення, отримає до нього доступ тильки в разі підтвердження дійсності підпису;

− крім забезпечення цілісності повідомлення, алгоритм може ще й забезпечувати його конфіденційність;

− ЕЦП з відновленням може забезпечувати менший обсяг підпису при невеликих повідомленнях;

− обов'язкове використання функцій формування збитковості повідомлення (етап створення передпідпису), що надає гнучкий функціонал (наприклад, обираючи тип збитковості, можна визначити, чи буде відновлена частина повідомлення або повністю усе повідомлення), а також підвищує криптостікість [7].

В книзі "Особливості ЕЦП з відновленням повідомлення" [7] сказано, що до алгоритмів, заснованих на даній схемі, можна віднести RSA, NR (схема Німберга-Руппеля) та її модифікація на еліптичних кривих ECNR, а також Zhang, ECMR, ECAO, ECPV, ECKNR. Алгоритм RSA має переваги в тому, що він є найбільш поширеним, має високі показники швидкості під час підписання/ верифікації, а також є універсальним, тому що й придатний ще для шифрування/дешифрування. NR Схеми на відміну від RSA ефективні для роботи з невеликими повідомлення, бо в такому випадку буде відновлюватися все повідомлення. Проте дані алгоритми мають перевагу в криптостійкості та ступені складності математичних проблем, на яких вони засновані. Використання функцій знаходження збитковостей може гарантувати менші обсяги підпису на коротких повідомленнях. Окрім цього, завдяки тому, що схема ECNR діє з використання еліптичних кривих, це дає змогу використовувати параметри та ключі менших обсягів. При детальному розгляданні можна побачити, що обидві схеми, ДСТУ 4145-2002 та ECNR, мають майже однакову структуру. Тому алгоритм ECNR ідеально підходить для удосконалення алгоритму ДСТУ 4145 [6].

1.4.1 Керування ключами ЕЦП

Відкритий ключ

Згідно до [4] важливою проблемою всієї криптографії з відкритим ключем, в тому числі і систем ЕЦП, є управління відкритими ключами. Так як відкритий ключ доступний будь-якому користувачеві, то необхідний механізм перевірки того, що цей ключ належить саме своєму власникові. Необхідно забезпечити доступ будь-якого користувача до справжнього відкритого ключа будь-якого іншого користувача, захистити ці ключі від підміни зловмисником, а також організувати відгук ключа у разі його компрометації. Завдання захисту ключів від підміни вирішується за допомогою сертифікатів. Сертифікат дозволяє засвідчити укладені в ньому дані про власника і його відкритий ключ підписом довіреної особи.

В законі україни “Про електронні довірчі послуги” [2] сказано, що існують системи сертифікатів двох типів: централізовані і децентралізовані. У децентралізованих системах шляхом перехресного підписування сертифікатів знайомих і довірених людей кожним користувачем бформується мережа довіри. У централізованих системах сертифікатів використовуються центри сертифікації, підтримувані довіреними організаціями. Центр сертифікації формує закритий ключ і власний сертифікат, формує сертифікати кінцевих користувачів і засвідчує їх автентичність своїм цифровим підписом. Також центр проводить скасування минулих і скомпрометованих сертифікатів і веде бази виданих та відкликаних сертифікатів. Звернувшись в сертифікаційний центр, можна отримати власний сертифікат відкритого ключа, сертифікат для іншого користувача і дізнатися, які ключі відкликані [4].

Закритий ключ

Згідно до [4] закритий ключ є найбільш вразливим компонентом всієї криптосистеми цифрового підпису. Зловмисник, який вкрав закритий ключ користувача, може створити дійсний цифровий підпис будь-якого електронного документа від імені цього користувача. Тому особливу увагу потрібно приділяти

способу зберігання закритого ключа. Користувач може зберігати закритий ключ на своєму персональному комп'ютері, захистивши його за допомогою пароля. Однак такий спосіб зберігання має ряд недоліків, зокрема, захищеність ключа повністю залежить від захищеності комп'ютера, і користувач може підписувати документи лише на цьому комп'ютері. Для безпечного зберігання закритого ключа використовують наступні варіанти зберігання:

- Токен (USB-пристрій)
- Смарт-карта

Крадіжка або втрата одного з таких пристроїв зберігання може бути легко помічена користувачем, після чого відповідний сертифікат повинен бути негайно відкликаний. Для того, щоб використовувати смарт-карту, користувачеві необхідно не тільки її мати, але й ввести PIN-код, тобто, виходить двофакторна аутентифікації [4].

Після цього підписується документ або його хеш передається в карту, її процесор здійснює підписування хешу і передає підпис назад. У процесі формування підпису таким способом не відбувається копіювання закритого ключа, тому весь час існує тільки єдина копія ключа. Крім того, зробити копіювання інформації зі смарт-карти складніше, ніж з інших пристроїв зберігання [4].

Також з моменту прийняття закону про Електронні довірчі послуги розроблений новий спосіб зберігання та використання ЕЦП – MobileID.

З презентації “Mobile ID: відповіді на актуальні запитання” [8] Mobile ID передбачає використання спеціалізованих SIM-карт для зберігання приватних ключів і створення електронного підпису за допомогою криптомодулю і захищеного програмного аплета. За допомогою Mobile ID прямо з мобільного телефону можна:

– підтверджувати особу під час ідентифікації на електронних ресурсах в інтернеті – включаючи портали надання державних (адміністративних) електронних послуг;

– підписувати документи в електронному вигляді;

– віддалено підтверджувати операції, які зазвичай вимагають особистої присутності з оригіналами документів. Та використання вищенаведених апаратних засобів має низку недоліків:

- Висока ціна носіїв.
- Необхідність завжди мати пристрій при собі (у разі утрати пристрою КЕП потрібно перевипускати).
- Один захищений носій може зберігати лише один секретний ключ.
- Електронний підпис неможливо використовувати на веб-ресурсах без установки на комп’ютер необхідних бібліотек (драйверів) для роботи з носієм.

Тому можливим рішенням, направленим на усунення наведених недоліків, є створення спеціалізованого комплексу із використанням “хмарного” сховища для ключової інформації.

1.4.2 Застосування хеш-функцій в контексті ЕЦП

Документи можуть мати різну довжину. Саме тому в випадку застосування ЕЦП дуже зручно використовувати не документ в початковому вигляді, а лише його хеш. Хеш – це результат роботи хеш-функцій або функцій згортання [3]. Лише криптографічні хеш-функції використовуються для забезпечення виявлення змін документів, які виникли під час перевірки підпису.

Хеш-функція не є частиною алгоритму ЕЦП, тому може бути використана будь-яка або не використовуватись взагалі. Хеш-функція призначена для стиснення послідовності вхідних даних довільної довжини в бітовий рядок попередньо визначеного розміру. Це зазвичай кілька десятків або сотень біт [3]. На рисунку 1.1 зображена асиметрична схема ЕЦП із використянням хеш-функції.

Рисунок 1.1 – Асиметрична схема ЕЦП із використанням хеш-функції

Аналіз, який використовує хеш-функцію, часто використовується для контроля цілісності критичних важливих програм, важливих даних, системних файлів. Моніторинг цілісності можна проводити на регулярній основі або за необхідності. Спочатку потрібно визначити, цілісність яких файлів необхідно відстежувати. Для кожного файла його значення обчислюється за допомогою спеціального алгоритму, який зберігає результат. Через деякий час проводиться обрахунок значення його хеша зі збереженням результату. Якщо значення відрізняються, інформація, що міститься у файлі, була змінена. Хеш-функції повинні володіти наступними характеристиками [3]:

- повинні мати можливість виконувати перетворення даних довільної довжини у дані фіксованої довжини;

- повинні мати відкритий алгоритм, щоб мати можливість вивчити його криптографічну стійкість;

- вона повинна бути односторонньою, тобто не повинно бути математичного способу визначення вихідних даних з результату;

- ймовірність зіткнень, тобто ймовірність того, що значення хеш-функцій двох різних документів (незалежно від їх довжини) будуть рівні, має бути незначною.

- вона не повинна вимагати великих обчислювальних ресурсів;

- найменша зміна вхідних даних повинна істотно змінити результат.

Використовування хеш-функції в алгоритмі створення ЕЦП забезпечує деякі переваги, зокрема [4]:

1) Обчислювальну складність. Зазвичай хеш цифрового документа робиться у багато разів меншого обсягу, ніж обсяг вихідного документа, і алгоритми обчислення хешу є більш швидкими, ніж алгоритми ЕЦП. Тому формувати хеш документа і підписувати його виходить набагато швидше, ніж підписувати сам документ.

2) Сумісність. Більшість алгоритмів оперує з рядками біт даних, але деякі використовують інші уявлення. Хеш-функцію можна використовувати для перетворення довільного вхідного тексту у відповідний формат.

3) Цілісність. Без використання хеш-функції великий електронний документ у деяких схемах потрібно розділяти на досить малі блоки для застосування ЕЦП. При верифікації неможливо визначити, чи всі блоки отримані в правильному порядку.

Опираючись на результати дипломної роботи “Система захисту даних на базі методу шифрування” [3] маємо, що за рахунок того, що хеш-функція значно зменшує обсяг документу, обрахунки на виході стають швидшими. Тому створювати хеш документу і тільки потім підписувати його – це оптимальне рішення в рамках поставленої задачі. Хеш можна використовувати для перетворення будь-якого вхідного тексту у потрібний формат. Якщо не використовувати функції стиснення, то документи великого розміру інколи розділяти на менші блоки задля застосування ЕЦП. Загалом, односпрямованість не розуміється як неможливість, а скоріше як великий рівень складності при

поверненні повідомлення від свого хешу, оскільки в даний час не існує хеш-функції з доведеною односпрямованістю.

Найбільшу популярність мають такі хеш-функції: MD4, MD5, RIPEMD-128 (128 біт), RIPEMD-160, SHA (160 біт). В українському стандарті цифрового підпису використовується розроблена вітчизняними криптографами хеш-функція (256 біт) стандарту ГОСТ 34.311-95.

1.4.3 Використання цифрових сертифікатів

Щоб забезпечити можливість практичного використання методів криптографії з відкритим ключем, необхідно реалізувати певний механізм, який би забезпечив однозначну прив'язку користувачів до їх відкритих ключів.

В якості такого механізму використовуються цифрові сертифікати (digital certificates). У загальному випадку цифровий сертифікат являє собою підписану авторитетною організацією - центром сертифікації (certificate authority) - сукупність відкритого ключа та ідентифікаційної інформації його власника.

В законі України “Про електронні довірчі послуги” [2] сказано, що сертифікат відкритого ключа - електронний документ, який засвідчує належність відкритого ключа фізичній або юридичній особі, підтверджує її ідентифікаційні дані та/або надає можливість здійснити автентифікацію веб-сайту.

Найпоширеніший формат сертифікатів - сертифікати X.509 версій 1, 2 і 3. Промисловий стандарт сертифікатів, визначений в RFC-2459 – це X.509.

У процесі обчислення цифрового підпису сертифіката хешируются значення всіх полів і розширень, таким чином після випуску сертифіката його поля і розширення вже не можуть бути змінені, інакше цифровий підпис стане недійсним. Структура сертифіката представлена на рисунку 1.2.

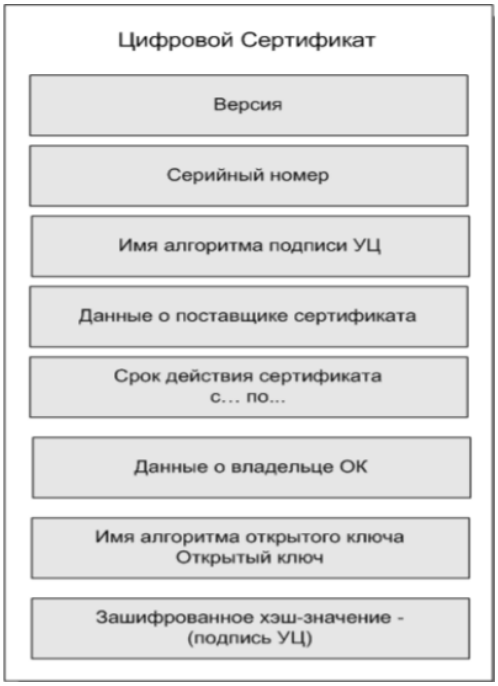

Рисунок 1.2 – Структура сертифіката X.509

Згідно закону “Про електронні довірчі послуги” [2] кваліфіковані сертифікати відкритих ключів обов’язково повинні містити:

1) позначку, що сертифікат відкритого ключа виданий як кваліфікований сертифікат відкритого ключа;

2) позначку, що сертифікат відкритого ключа виданий в Україні;

3) ідентифікаційні дані, які однозначно визначають кваліфікованого надавача електронних довірчих послуг, засвідчувальний центр або центральний засвідчувальний орган, які видали кваліфікований сертифікат відкритого ключа.

4) ідентифікаційні дані, які однозначно визначають користувача електронних довірчих послуг

5) місцезнаходження юридичної особи, якій видано кваліфікований сертифікат відкритого ключа;

6) значення відкритого ключа, який відповідає особистому ключу;

7) відомості про початок та закінчення строку дії кваліфікованого сертифіката відкритого ключа;

8) серійний номер кваліфікованого сертифіката відкритого ключа, унікальний для суб’єкта, який видав сертифікат;

9) кваліфікований електронний підпис або кваліфіковану електронну печатку, створені суб’єктом, який видав сертифікат;

10) відомості щодо розміщення у вільному доступі кваліфікованих сертифікатів відкритих ключів суб’єкта, який видав сертифікат;

11) відомості щодо розміщення інформації, необхідної для отримання кваліфікованої електронної довірчої послуги формування, перевірки та підтвердження чинності кваліфікованих сертифікатів відкритих ключів;

12) відомості про те, що особистий ключ зберігається в засобі кваліфікованого електронного підпису чи печатки (для кваліфікованого сертифіката електронного підпису чи печатки);

13) відомості про обмеження використання кваліфікованого електронного підпису чи печатки (для кваліфікованого сертифіката електронного підпису чи печатки);

1.5 Нормативно-правове забезпечення ЕЦП

Електронну комерцію можна розглядати як одну із сучасних форм організації і здійснення господарської, переважно банківської і торговельної, діяльності. Відмінною рисою даної форми діяльності є використання загальнодоступних інформаційних систем та комп’ютерних мереж, об’єднаних Інтернетом. Електронна комерція має ряд переваг в економічному відношенні. Дослідження правового забезпечення розвитку даної форми спрямовано на удосконалення законодавства і практики його застосування таким чином, щоб вони сприяли цьому розвиткові.

Відносини, пов'язані з використанням електронних цифрових підписів, регулювалися с 2003р. Законом України “Про електронний цифровий підпис” [9]. Відповідно до статті 5 електронний документ – це документ, інформація в якому зафіксована у вигляді електронних даних, включаючи обов’язкові реквізити документа. Оригіналом електронного документа вважається електронний примірник документа з обов'язковими реквізитами, у тому числі з електронним цифровим підписом автора, яким і завершується створення електронного документа. Використання інших видів електронних підписів в електронному документообігу здійснюється суб'єктами електронного

документообігу на договірних засадах відповідно до вимог чинного законодавства, зокрема, Закону “Про електронну комерцію”.

З правової точки зору електронний цифровий підпис – це різновид електронного підпису, який за юридичним статусом прирівнюється до власноручного в разі виконання технічних умов, визначених Законом. Електронний цифровий підпис призначений для забезпечення діяльності фізичних та юридичних осіб, яка здійснюється із використанням електронних документів. Він використовується фізичними та юридичними особами – суб'єктами електронного документообігу для ідентифікації підписувача та підтвердження цілісності даних в електронній формі. Стаття 1 Закону України “Про електронний цифровий підпис” [9] визначає послуги електронного цифрового підпису як надання у користування засобів електронного цифрового підпису, допомога при генерації відкритих та особистих ключів, обслуговування сертифікатів ключів (формування, розповсюдження, скасування, зберігання, блокування та поновлення), надання інформації щодо чинних, скасованих і блокованих сертифікатів ключів, послуги фіксування часу, консультації та інші послуги, визначені цим Законом. Аналіз наведеного визначення послуги дозволяє виокремити такі ознаки послуги:

- послугою є вчинення певних дій чи діяльності;
- надається відповідно до завдання замовника;
- послуга споживається у процесі її надання;
- послуга є благом, вартість якого підлягає оцінюванню.

Такі послуги надаються Акредитованими центрами сертифікації ключів (далі – АЦСК), які видають та обслуговують сертифікати електронних цифрових підписів для юридичних та фізичних осіб для здійснення електронного документообігу. Треба зазначити, що прийняття вищезазначених законів мало велике позитивне значення й сприяло поширенню використання електронно-цифрового підпису, зокрема, для подання податкової звітності. Також у сфері приватно-правових відносин ЕЦП із прийняттям вищевказаних Законів і до тепер не став зручною та поширеною альтернативою паперових документів та власноручних підписів і печаток. Це пояснюється, складною

процедурою виготовлення цифрового підпису, та низьким рівнем довіри суспільства до держ-нституцій, особливо представлених в електронному вигляді.

Здебільшого суб'єкти суспільних відносин довіряють традиційним засобам об'єктивації інформації. Однак для повномасштабної реалізації можливостей інформаційних технологій усталені моделі поведінки суб'єктів суспільного обороту потребують трансформації. Важливою передумовою прийняття Закону України "Про електронні довірчі послуги" є Регламент (ЄС) № 910/2014 Європейського Парламенту і Ради ЄС від 23 липня 2014 року про електронну ідентифікацію і довірчі послуги для електронних трансакцій у рамках внутрішнього ринку. Опираючись на статтю "До питання про поняття довірчих електронних послуг та їх договірного забезпечення" [10], ми можемо зробити висновок, що положення цього Регламенту спрямовані на підвищення довіри до електронних операцій, здійснюваних на внутрішньому ринку, що забезпечують загальну основу для безпечної електронної взаємодії між громадянами, підприємствами і державними органами, що, у свою чергу, сприяє підвищенню ефективності державних і приватних організацій, які надають онлайн-послуги, електронного бізнесу і електронної комерції в Європейському Союзі. Ці правові акти є орієнтиром для подальшого розвитку українського законодавства у сфері здійснення електронних трансакцій.

Відповідно до закону "Про електронні довірчі послуги" [2], електронною довірчою є послуга, яка надається для забезпечення електронної взаємодії двох або більше суб'єктів, які довіряють надавачу електронних довірчих послуг щодо надання останньої.

У вище зазначенному законі визначається склад електронних довірчих послуг, що надаються зареєстрованими надавачами електронних довірчих послуг, а саме:

- створення, перевірка та підтвердження удосконаленого електронного підпису чи печатки;
- формування, перевірка і підтвердження дії сертифікату електронного підпису або друку;

- формування, перевірка і підтвердження дії сертифікату шифрування;
- формування, перевірка і підтвердження дії сертифікату автентифікації веб-сайту;
- формування, перевірка і підтвердження електронної позначки часу;
- реєстрована електронна доставка;
- зберігання удосконалених електронних підписів, печаток, електронних позначок часу та сертифікатів, пов'язаних із цими послугами.

Кожна послуга, яка входить до складу електронних довірчих послуг, може надаватися як окремо, так і в сукупності зареєстрованим постачальником електронних довірчих послуг і кваліфікованим надавачем електронних довірчих послуг. Електронні довірчі послуги надаються, як правило, на договірних засадах надавачами електронних довірчих послуг (ст. 16 Закону), тому доцільно розглянути, яким чином відбувається забезпечення даних послуг договірним регулюванням. Сьогодні специфіка ЕЦП послуг дозволяє дійти висновку про публічну природу договору, що опосередковує їх надання [10]. Інакше кажучи, за таким договором сторона АЦСК бере на себе обов’язок надати електронні довірчі послуги всім, хто до неї звернеться на правомірних засадах. Закон “Про електронні довірчі послуги” [2] чітко визначив перелік кваліфікованих надавачів електронних довірчих послуг та інформації про послуги, що ними надаються (довірчий список). Тобто довірчим списком чітко визначений перелік суб’єктів господарювання, уповноважених державою на здійснення обслуговування користувачів електронних підписів.

Положеннями Регламенту (ЄС) (ст. 30 та 39) передбачено сертифікацію засобів кваліфікованого електронного підпису та печатки (засоби КЕП) на відповідність вимогам стандартів щодо оцінки безпеки, перелік яких визначений ІМПЛЕМЕНТАЦІЙНИМ РІШЕННЯМ КОМІСІЇ (ЄС) 2016/650 від 25.04.2016 (Рішення).

Так, Рішенням передбачено перевірку засобів КЕП, які здійснюють генерацію і зберігання ключа, на відповідність профілю захисту для засобів

створення захищеного підпису з генерацією ключів, визначеного стандартом EN 419211-2:2013 (прийнятий в Україні як ДСТУ EN 419211-2:2016).

Згідно положень стандарту EN 419211-2:2013 засіб для створення безпечного підпису - це комбінація апаратного та програмного забезпечення, налаштованого для безпечного створення, використання та управління підписом та захисту ключових даних протягом усього їх життєвого циклу (п. 4.3.2). Означений засіб створює електронний підпис, за допомогою особистого ключа, що зберігається в засобі та знаходиться під одноосібним контролем підписувача (п. 6.3.2-6.3.3).

З метою імплементації означених положень законодавства ЄС вимогами Закону передбачено внесення до кваліфікованого сертифіката відкритого ключа відомостей про те, що особистий ключ підписувача зберігається в засобі КЕП. Та підтвердження правового статусу кваліфікованого електронного підпису чи печатки у разі, якщо за допомогою перевірки кваліфікованого сертифіката відкритого ключа отримано підтвердження того, що особистий ключ, який належить підписувачу чи створювачу електронної печатки, зберігається в засобі КЕП [2].

Постановою Кабінету Міністрів України 07.11.2018 № 992 “Про затвердження вимог у сфері електронних довірчих послуг та Порядку перевірки дотримання вимог законодавства у сфері електронних довірчих послуг” (Постанова 992) [11] встановлено перелік стандартів, що визначають вимоги до засобів КЕП, серед яких є, зокрема, ДСТУ EN 419211-2:2016.

Таким чином, всі функції, які здійснюються засобами КЕП, реалізуються згідно відповідних профілів захисту, наведених в Рішенні та визначених Постановою 992.

Пунктом 2 Постанови 992 [11] встановлено, що для надання кваліфікованих електронних довірчих послуг можуть використовуватись надійні засоби електронного цифрового підпису, які отримали позитивні експертні висновки за результатами державної експертизи у сфері криптографічного захисту інформації.

1.6 Висновки. Постановка задачі

У роботі були поставлені такі задачі:

- Провести аналіз чинного законодавства України, дослідити стан процесу використання електронного цифрового підпису та обґрунтувати необхідність створення комплексу;

- Проаналізувати технологічну структуру та призначення центрів сертифікації ключів;

- Розробити алгоритм роботи комплексу та обґрунтувати його застосування;

- Розробити програмний засіб для реалізації взаємодії з комплексом;

- Дослідити економічну доцільність на розробку та впровадження комплексу в технологічну структуру кваліфікованих надавачів електронних довірчих послуг.

Проаналізувавши юридичний стан питання щодо використання кваліфікованого електронного підпису, установлено, що згідно чинного законодавства процес генерації та зберігання КЕП повинен відбуватися із використанням засобу кваліфікованого електронного підпису чи печатки, який згідно положенню стандарту EN 419211-2:2013, визначеного у постанові КМУ 07.11.2018 №992 - це комбінація апаратного та програмного забезпечення, налаштованого для безпечного створення, використання та управління підписом та захисту ключових даних протягом усього їх життєвого циклу. Це означає, що всі кваліфіковані надавачі КЕП мають надавати електронні довірчі послуги лише з використанням захищених носіїв ключової інфомації (токени, смарт-карти).

Такий варіант зберігання КЕП має низку недоліків, можливим шляхом усунення яких може буде створення програмно-технічного комплексу із використанням хмарного сховища на базі існуючого АЦСК.

РОЗДІЛ 2
РОЗРОБКА КОМПЛЕКСУ НАДАННЯ ЕЛЕКТРОННИХ ДОВІРЧИХ ПОСЛУГ ІЗ ВИКОРИСТАННЯМ ХМАРНОГО СХОВИЩА

2.1 Кваліфіковані надавачі електронних довірчих послуг (АЦСК)

Згідно до закону України “Про електронні довірчі послуги” [2] центром сертифікації ключів може бути юридична особа незалежно від форми власності або фізична особа, яка є суб’єктом підприємницької діяльності, що надає послуги електронного цифрового підпису та засвідчила свій відкритий ключ у центральному засвідчувальному органі або засвідчувальному центрі.

Центр сертифікації ключів, акредитований в установленому порядку, є акредитованим центром сертифікації ключів.

Акредитований центр сертифікації ключів має право:

- надавати послуги електронного цифрового підпису та обслуговувати виключно посилені сертифікати ключів;
- отримувати та перевіряти інформацію, необхідну для реєстрації підписувача і формування посиленого сертифіката ключа, безпосередньо у юридичної або фізичної особи чи її представника.

Акредитований центр сертифікації ключів має виконувати усі зобов’язання та вимоги, встановлені законодавством для центру сертифікації ключів, та додатково зобов’язаний використовувати для надання послуг електронного цифрового підпису надійні засоби електронного цифрового підпису.

З моменту прийняття Закону України “Про електронні довірчі послуги” поняття “Акредитований центр сертифікації ключів” змінилося на поняття “кваліфікований надавач електронних довірчих послуг”.

Кваліфікований надавач електронних довірчих послуг - юридична особа незалежно від організаційно-правової форми та форми власності, фізична особа - підприємець, яка надає одну або більше електронних довірчих послуг,

діяльність якої відповідає вимогам Закону та відомості про яку внесені до Довірчого списку [2].

Електронна довірча послуга - послуга, яка надається для забезпечення електронної взаємодії двох або більше суб'єктів, які довіряють надавачу електронних довірчих послуг щодо надання такої послуги [2].

Електронні довірчі послуги надаються, як правило, на договірних засадах надавачами електронних довірчих послуг. Кожна послуга, що входить до складу електронних довірчих послуг, може надаватися як окремо, так і в сукупності [2].

Згідно до закону України "Про електронні довірчі послуги" [2] кваліфіковані надавачі електронних довірчих послуг зобов'язані забезпечити:

- захист персональних даних користувачів електронних довірчих послуг відповідно до вимог законодавства;
- функціонування програмно-технічного комплексу, що ними використовується, та захист інформації, що в ньому обробляється, відповідно до вимог законодавства;
- створення та функціонування свого веб-сайту;
- впровадження, підтримання в актуальному стані та публікацію на своєму веб-сайті реєстру чинних, блокованих та скасованих сертифікатів відкритих ключів;
- можливість цілодобового доступу до реєстру чинних, блокованих та скасованих сертифікатів відкритих ключів та до інформації про статус сертифікатів відкритих ключів через телекомунікаційні мережі загального користування;
- цілодобовий прийом та перевірку заяв підписувачів та створювачів електронних печаток про скасування, блокування та поновлення їхніх сертифікатів відкритих ключів;
- скасування, блокування та поновлення сертифікатів відкритих ключів відповідно до вимог цього Закону;
- встановлення під час формування сертифіката відкритого ключа належності відкритого ключа та відповідного йому особистого ключа підписувачу чи створювачу електронної печатки;

- внесення ідентифікаційних даних підписувача чи створювача електронної печатки до відповідного сертифіката відкритого ключа;
- інформування контролюючого органу про порушення конфіденційності та/або цілісності інформації, що впливають на надання електронних довірчих послуг або стосуються персональних даних користувачів електронних довірчих послуг, не пізніше 24 годин з моменту, коли їм стало відомо про таке порушення;
- інформування користувачів електронних довірчих послуг про порушення конфіденційності та/або цілісності інформації, що впливають на надання їм електронних довірчих послуг або стосуються їхніх персональних даних, не пізніше двох годин з моменту, коли їм стало відомо про такі порушення;
- унеможливлення використання особистого ключа у разі його компрометації;
- постійне зберігання всіх виданих кваліфікованих сертифікатів відкритих ключів;

Суб’єктами відносин у сфері електронних довірчих послуг є:

- користувачі електронних довірчих послуг;
- надавачі електронних довірчих послуг;
- органи з оцінки відповідності;
- засвідчувальний центр;
- центральний засвідчувальний орган;
- контролюючий орган.

Згідно до закону України “Про електронні довірчі послуги” [2] державне регулювання у сферах електронних довірчих послуг та електронної ідентифікації здійснюють:

1) Кабінет Міністрів України;
2) головний орган у системі центральних органів виконавчої влади, що забезпечує формування та реалізує державну політику у сфері електронних довірчих послуг;

3) спеціально уповноважений центральний орган виконавчої влади з питань організації спеціального зв’язку та захисту інформації;
4) Національний банк України (у банківській системі України).

2.2 Центральний засвідчувальний орган

Повноваження центрального засвідчувального органу визначено у Статті 7 Закону України “Про електронні довірчі послуги” [2]. До повноважень центрального засвідчувального органу по відношенню до кваліфікованих надавачів електронних довірчих послуг належить надання таких послуг:

- адміністративна послуга внесення юридичних осіб та фізичних осіб - підприємців, які мають намір надавати електронні довірчі послуги, до Довірчого списку;
- надання кваліфікованих електронних довірчих послуг надавачам електронних довірчих послуг з використанням самопідписаного сертифіката електронної печатки центрального засвідчувального органу, що призначений для надання таких послуг;
- надання послуги постачання передачі сигналів точного часу, синхронізованого з Державним еталоном одиниць часу і частоти.

Технічне та технологічне забезпечення виконання функцій центрального засвідчувального органу здійснюється адміністратором інформаційно-телекомунікаційної системи центрального засвідчувального органу - державним підприємством, яке належить до сфери управління головного органу у системі центральних органів виконавчої влади, що забезпечує формування та реалізує державну політику у сфері електронних довірчих послуг (державним підприємством “Національні інформаційні системи”) [2].

Центральний засвідчувальний орган впроваджує, підтримує в актуальному стані та публікує на своєму офіційному веб-сайті Довірчий список, в якому міститься інформація про кваліфікованих надавачів електронних довірчих послуг разом з інформацією про кваліфіковані електронні довірчі послуги, які вони надають [2].

Довірчий список впроваджується, підтримується в актуальному стані та публікується в безпечному режимі з обов’язковим додаванням електронної печатки центрального засвідчувального органу у вигляді, придатному для автоматичної обробки [2].

В законі “Про електронні довірчі послуги” [2] сформовано довірчий список, в якому містяться відомості про кваліфікованих надавачів електронних довірчих послуг та їх кваліфіковані електронні довірчі послуги, надання яких передбачає використання алгоритмів електронного підпису, визначених ДСТУ 4145-2002 “Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірки”, зображений нижче:

- Міністерство юстиції України.
- Національний банк України.
- Військова частина 2428.
- Генеральна прокуратура України.
- Генеральний штаб Збройних Сил України.
- Державна казначейська служба України.
- Державне підприємство “Оператор ринку”.
- Державне підприємство "Національні інформаційні системи".
- Державне підприємство “Український інститут інтелектуальної власності”.
- Державне підприємство “Українські спеціальні системи”.
- Інформаційно-довідковий департамент ДПС.
- Міністерство внутрішніх справ України.
- Акціонерне товариство “Державний ощадний банк України”.
- АКЦІОНЕРНЕ ТОВАРИСТВО КОМЕРЦІЙНИЙ БАНК “ПРИВАТБАНК”.
- Публічне акціонерне товариство “Національний депозитарій України”.
- АКЦІОНЕРНЕ ТОВАРИСТВО “УКРСИББАНК”.
- Товариство з обмеженою відповідальністю “Алтерсайн”.
- ТОВ “Арт-мастер”.

- ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ “ІНТЕР-МЕТЛ”.
- Товариство з обмеженою відповідальністю “КЛЮЧОВІ СИСТЕМИ”.
- ТОВ “Центр сертифікації ключів “Україна””.
- Філія “Головний інформаційно-обчислювальний центр” акціонерного товариства “Українська залізниця”.
- АКЦІОНЕРНЕ ТОВАРИСТВО “АЛЬФА-БАНК”.

2.3 Структура і функціонування ЦСК

Далі наведена детальна інформація про програмно-технічний комплекс акредитованого центру сертифікації ключів, його призначення та функціонування окремих складових частин. Наведена схема необхідна для функціонування інфраструктури відкритих ключів.

Структура та склад комплексу

Інституту інформаційних технологій [12] наводить схему комплекса ЦСК. До складу комплексу входять такі технічні засоби:

- робочі станції (РС) обслуговуючого персоналу (адміністратора безпеки, системного адміністратора та адміністратора реєстрації);
- центральні сервери (сервери ЦСК);
- внутрішнє комунікаційне обладнання локальної обчислювальної мережі (ЛОМ);
- сервери взаємодії;
- міжмережний екран (МЕ) та система виявлення втручання (IDS);
- комунікаційне обладнання для підключення до зовнішніх комунікаційних мереж (ЗКМ);
- РС генерації ключів користувачів (ізольована);

На рисунку 2.1 зображено структуру програмно-технічного комплексу центру сертифікації ключів.

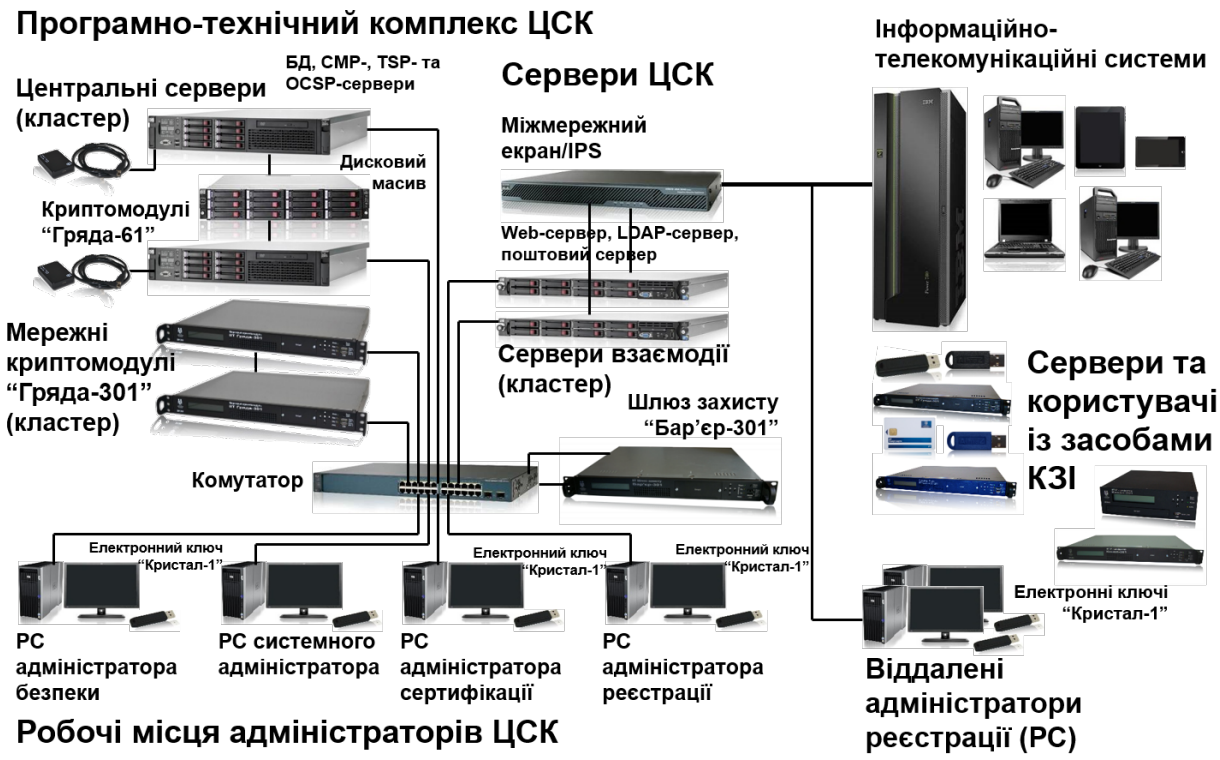

Рисунок 2.1 – Структура центру сертифікації ключів

РС адміністратора безпеки, адміністратора сертифікації, системного адміністратора, адміністратора реєстрації, центральні сервери та сервери взаємодії мають взаємодіяти через внутрішню комунікаційну мережу на основі кабельної мережі та комутаторів і утворювати ЛОМ [12].

Функції комплексу

Комплекс забезпечує реалізацію регламентних процедур та механізмів роботи ЦСК, пов'язаних з [12]:

1) обслуговуванням сертифікатів відкритих ключів (далі – сертифікатів) користувачів, що включає:
   1) реєстрацію користувачів;
   2) сертифікацію відкритих ключів користувачів;
   3) розповсюдження сертифікатів;
   4) управління статусом сертифікатів;
   5) розповсюдження інформації про статус сертифікатів;
2) надання послуг фіксування часу;

3) надання користувачам засобів ЕЦП та шифрування даних, а також засобів генерації та управління ключами.

Комплекс забезпечує виконання наступних функцій, пов'язаних з обслуговуванням ЦСК сертифікатів користувачів [12]:

1) реєстрацію користувачів, що включає:
    1) введення реєстраційних даних користувачів до реєстру користувачів;
    2) зберігання реєстру користувачів та забезпечення доступу до реєстраційних даних;
    3) резервне копіювання та архівування реєстру користувачів;
    4) зміну реєстраційних даних користувачів у реєстрі;
    5) видалення реєстраційних даних користувачів з реєстру;
2) сертифікацію відкритих ключів користувачів, що включає:
    1) приймання та реєстрацію запитів користувачів на формування сертифікатів;
    2) зберігання запитів, отриманих від користувачів, у базі даних запитів;
    3) архівування бази даних запитів;
    4) формування сертифікатів користувачів;
    5) внесення сформованих сертифікатів у реєстр сертифікатів;
    6) зберігання реєстру сертифікатів;
    7) архівування реєстру сертифікатів;
3) розповсюдження сертифікатів відкритих ключів користувачів, що включає:
    1) публікацію реєстру сертифікатів на інформаційному ресурсі ЦСК (у LDAP-каталозі та на web-сторінці);
    2) забезпечення доступу користувачів до реєстру сертифікатів на інформаційному ресурсі ЦСК;
4) управління статусом сертифікатів відкритих ключів користувачів та розповсюдження інформації про статус сертифікатів, що включає:

1) приймання та реєстрацію запитів користувачів на скасування, блокування чи поновлення сертифікатів;
2) зберігання запитів, отриманих від користувачів, у базі даних запитів;
3) архівування бази даних запитів;
4) скасування, блокування або поновлення сертифікатів на основі запитів;
5) внесення інформації про поточний статус сертифіката до реєстру сертифікатів;
6) формування списків відкликаних сертифікатів користувачів;
7) публікацію списків відкликаних сертифікатів на інформаційному ресурсі ЦСК;
8) забезпечення доступу користувачів до списків відкликаних сертифікатів на інформаційному ресурсі ЦСК;
9) забезпечення доступу користувачів до інформації про статус сертифікатів (та самих сертифікатів) з використанням протоколу визначення статусу сертифіката (OCSP).

До складу комплексу входять засоби користувачів у складі [12]:

1) засобів генерації особистих і відкритих ключів користувачів, які призначені для:
   1) генерації особистого та відкритого ключів користувача;
   2) формування та передачу запиту на формування сертифіката користувача до ЦСК;
   3) отримання, перевірку, зберігання та використання сформованого сертифікату;
   4) формування та передачу запитів на блокування, скасування та поновлення сертифіката користувача до ЦСК.
2) засобів ЕЦП та шифрування даних користувачів, які призначені для:
   1) введення та використання особистих ключів користувача;
   2) пошуку та інтерактивної перевірки статусу сертифікатів у ЦСК за протоколом OCSP (через OCSP-сервер ЦСК);

3) пошуку сертифікатів у LDAP-каталозі ЦСК (на LDAP-сервері ЦСК);
4) отримання позначок часу у ЦСК (через TSP-сервер ЦСК);
5) реалізації механізмів формування та перевірянні ЕЦП, а також шифрування даних користувача у системах електронної пошти, електронного документообігу тощо (шляхом використання у вигляді бібліотек та ін.).

2.4 Інфраструктура відритих ключів

Інфраструктура відкритих ключів (*Public key infrastructure*, PKI) — інтегрований комплекс методів та засобів (набір служб), призначених забезпечити впровадження та експлуатацію криптографічних систем із відкритими ключами.

PKI є комплексом апаратних і програмних засобів, політик та процедур. Продукти і послуги, що входять в інфраструктуру відкритих ключів, пропонуються на ринку цілою низкою компаній. Більшість компаній, які починають використовувати технологію PKI, з двох шляхів вибирають один: вони або створюють власний орган видачі сертифікатів, або надають право це зробити іншим компаніям, що спеціалізуються на послугах PKI, наприклад, VeriSign, DigitalSignatureTrust та ін.

Універсальне застосування сертифікатів забезпечує стандарт Міжнародного Союзу по телекомунікаціях X.509, який є базовим і підтримується цілим рядом протоколів безпеки. У їх числі-стандарти шифрування з відкритими ключами (PKCS), протокол зв'язку SSL і безпечний протокол передачі гіпертекстових повідомлень (SecureHTTP).

Інфраструктура відкритих ключів побудована на криптосистемах з відкритим ключем. Ця технологія має властивості, які відіграють важливу роль у захисті даних в розподілених системах [13].

PKI служить не лише для створення цифрових сертифікатів, але і для зберігання великої кількості сертифікатів і ключів, забезпечення резервування і

відновлення ключів, взаємної сертифікації, ведення списків анульованих сертифікатів і автоматичного відновлення ключів та сертифікатів після закінчення терміну їхньої дії [13].

Основними складовими інфраструктури відкритих ключів є центри сертифікації, центри реєстрації (засвідчення), репозиторії та архіви сертифікатів [13].

Центр сертифікації ключів (*Certificate Authority*, CA) — юридична особа незалежно від форми власності або фізична особа, яка є суб'єктом підприємницької діяльності, що надає послуги щодо сертифікації відкритих ключів. Центр сертифікації складається з апаратного, програмного забезпечення та обслуги. Центри сертифікації мають два важливих атрибути: назву та відритий ключ. Центри сертифікації виконують чотири основні завдання [13]:

- видача сертифікатів (тобто, центр створює та підписує електронний цифровий сертифікат);
- оброблення статусу сертифікатів та підтримання списків анульованих сертифікатів (CRL);
- поширення поточного списку дійсних та анульованих сертифікатів, аби користувачі мали можливість перевірити стан сертифіакту;
- підтримання архівних даних про сертифікати та їхній стан.

Центри сертифікації інколи можуть делегувати відповідальність за деякі з цих задач до інших складових інфраструктури [13].

Центр сертифікації ключів разом із об'єктами, яким видано сертифікати цього центру, утворюють домен (ЦСК-домен) [13].

Центр реєстрації (*Registration Authority*, RA) — об'єкт, що відповідає за ідентифікацію та аутентифікацію суб'єктів сертифікатів, але не підписує і не випускає сертифікати (тобто, RA делегуються деякі задачі від імені центру сертифікації) [13].

Архів — це база даних, яка зберігає та захищає інформацію для розв'язання різних суперечок, які можуть виникнути у майбутньому. Основним завданням архіву є безпечне зберігання інформації, необхідної для встановлення вірності “старих” електронних підписів [13].

В основному виділяють 5 видів архітектур PKI, це(а саме):

1) проста PKI (одиночний ЦС);

2) ієрархічна PKI;

3) мережева PKI;

4) крос-сертифіковані корпоративні PKI;

Проста PKI – найпростіша з архітектур. Це архітектура одиночного ЦСК. У даному випадку всі користувачі довіряють одному ЦСК і переписуються між собою. У цій архітектурі, якщо зловмисник видасть себе за ЦСК, необхідно перевипустити всі виписані сертифікати і продовжити нормальну роботу.

Ієрархічна структура – це архітектура PKI, що найбільш часто зустрічається. У даному випадку на чолі всієї структури знаходиться один головний ЦСК, якому всі довіряють і йому підпорядковуються. Крім цього головного ЦСК, в структурі присутні ще центри сертифікації, які підпорядковуються вищестоящому. Окремий приклад ієрархічної PKI - корпоративна PKI. Наприклад, коли існує одна велика фірма, у якої в підпорядкуванні безліч філій по всій країні. В ієрархічній PKI, навіть якщо зловмисник видав себе за будь – який ЦСК, мережа продовжує працювати без нього, коли він відновлює нормальну працездатність – він знову включається в структуру. Дана структура використовується в Українських центрах сертифікації, на чолі якої знаходиться ЦЗО.

Мережева архітектура PKI – це модель встановлення довірчих відносин між окремими ізольованими та ієрархічними ЦСК — доменами без довірчого посередника. Довірчі відносини встановлюються через механізм кросс-сертифікації між Головними ЦСК в кожному з доменів [13]. До цієї архітектуру PKI легко додається новий ЦСК. На відміну від ієрархії, побудова шляху сертифікації від сертифіката держателя до точки довіри не детермінована (не жорстко визначена). Це ускладнює встановлення шляху сертифікації[13].

Архітектура крос-сертифікованої корпоративної PKI. Даний вид архітектури можна розглядати як змішаний вид ієрархічної й мережної архітектури. Наприклад, існує кілька організацій, у кожної з яких організована якась своя PKI, але вони хочуть спілкуватися між собою, в результаті чого

виникає їх загальна міжорганізаційна PKI. В архітектурі крос-сертифікованої корпоративної PKI найскладніша система ланцюжка сертифікації. Архітектура мостового ЦСК розроблялася для того, щоб прибрати недоліки складного процесу сертифікації в крос-сертифікованої корпоративної PKI. У даному випадку всі компанії довіряють не якоїсь однієї або двом фірмам, а одному ЦСК, який є практично їх головним ЦСК, але він не є основним пунктом довіри, а виступає в ролі посередника між іншими ЦСК.

Важливу роль у функціонуванні інфраструктрури відкритих ключів відіграють ряд протоколів, які дозволяють отримувати додаткову інформацію про цифрові сертифікати OCSP и TSP та протокол керування сертифікатами CMP.

Протокол OCSP (Online Certificate Status Protocol) - протокол отримання статусу сертифіката в реальному часі. Застосовується для надання користувачам УЦ актуальної інформації про статуси сертифікатів ключів підпису. За протоколом OCSP можна отримати інформацію про зміну статусу цифрового сертифікату в реальному часі. Протокол OCSP необхідний при роботі з важливою інформацією, наприклад, при обміні цінними паперами великого гідності або масштабних фондових продажів.

Протокол OCSP працює за принципом “запит-відповідь”. OCSP-клієнт генерує OCSP-запит і відправляє його на сервер. OCSP-сервер отримує цей запит, перевіряє статус сертифіката, генерує OCSP-відповідь і відсилає його клієнту.

Розглянемо докладніше структуру OCSP запитів і відповідей.

OCSP-запит складається з номера версії протоколу, типу запиту на обслуговування і одного або декількох ідентифікаторів сертифікатів. Ідентифікатор сертифікату включає хеш-коди відкритого ключа видавця сертифіката, а також серійний номер сертифіката. У запиті іноді можуть бути присутніми необов'язкові доповнення.

OCSP-відповідь складається із ідентифікатора сертифіката, статусу сертифіката (“нормальний”, “анульований” або “невідомий”) і терміну дії відповіді, пов'язаного з ідентифікатором кожного зазначеного в вихідному

запиті сертифіката. Якщо сертифікат має статус анульованого, то відображається час анулювання та може бути вказана причина анулювання (необов'язково). Термін дії задається інтервалом від поточного оновлення до наступного оновлення. Відповідь може містити необов'язкові доповнення, а також код помилки, якщо обробка запиту не була завершена коректно.

2.5 Використання хмарного сховища

Проаналізувавші статтю “Хмарні сховища даних та їх характеристики” [14] можно зробити висновок, що останнім часом у галузі інформаційно-комунікаційних технологій спостерігається бурхливий розвиток хмарних технологій. Відповідно до цього виникають численні хмарні сервіси, що все частіше застосовуються у різних сферах людської діяльності. Одним із найпоширеніших подібних сервісів являються хмарні сховища даних.

Хмарне сховище даних – це модель он-лайн сховища, в якому дані зберігаються на численних розподілених у мережі серверах, що надаються в користування клієнтам, в основному, третьою стороною [14].

Таким чином, замість розміщення файлів на носіях зовнішньої пам’яті (або на вінчестерах комп’ютерів) інструменти і результати роботи поступово переносяться та розміщуються у хмарному сховищі даних або у “хмарі”. За таких умов дані доступні з багатьох комп’ютерів [14].

Розглянемо переваги та недоліки застосування хмарних сховищ даних у процесі діяльності користувачів [14].

Серед переваг використання хмарних сховищ даних можна виокремити такі [14]:

- доступ до даних здійснюється з будь-якого місця та в будь-який час за наявності під’єднання до глобальної мережі Інтернет;
- користувач сплачує тільки за те місце у сховищі, яке фактично використовує;
- всі процедури із збереження цілісності даних забезпечуються провайдером хмарного центру.

До недоліків використання хмарних сховищ даних належать [14]:

- небезпека у процесі зберігання та пересилання даних, особливо конфіденційних, приватних;
- загальна продуктивність при роботі з даними в “хмарі” може бути нижчою, ніж при роботі з локальними копіями даних;
- необхідна наявність стабільного та швидкісного під’єднання до мережі Інтернет.

В комплексі надання електронних довірчих послуг можливе застосування апаратного криптомодуля “Гряда” в якості хмарного сховища ключів.

2.6 Обґрунтування структури комплексу хмарного КЕП

На підставі поставленої задачі було проаналізовано структуру АЦСК, де буде впроваджуватися комплекс, створенно архітектуру взаємодії складових частин, вибрані апаратні та програмні засоби, необхідні для побудови і забезпечення безперервного функціонування комплексу. Спираючись на інфраструктуру центру сертифікації був розроблений та впроваджений комплекс для надання електронних довірчих послуг із використання хмарного сховища ключової інформації. Структуру комплексу зображено на рисунку 2.2.

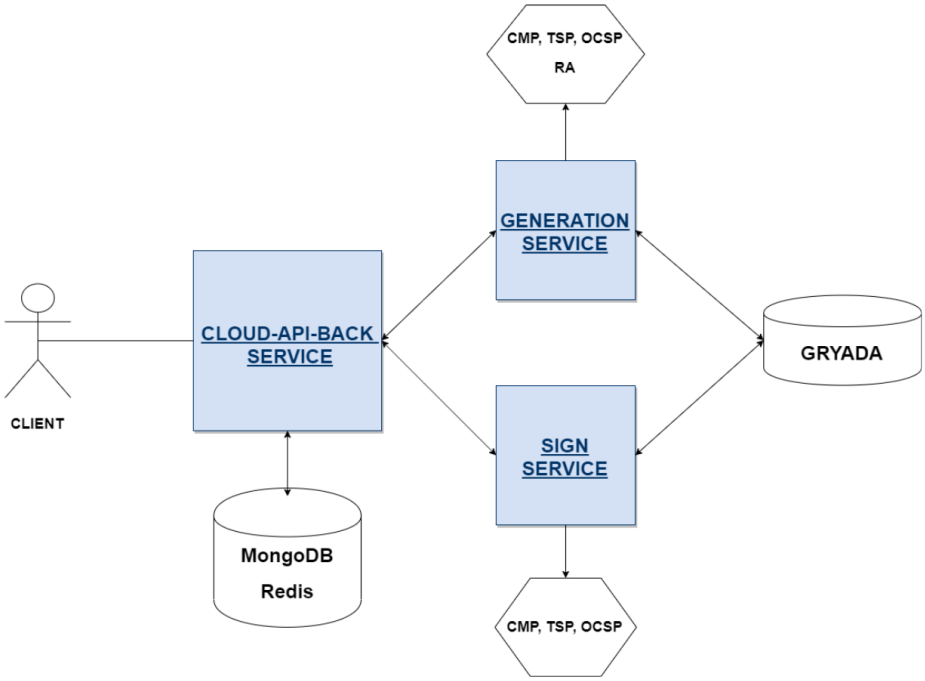

Рисунок 2.2 – Структура комплексу хмарного КЕП

CLOUD-API-BACK SERVICE – застосунок, що надає API для зовнішніх ресурсів;

SIGN SERVICE – сервіс підпису;

GENERATION SERVICE – сервіс генерації ключів;

GRYADA – апаратний криптомодуль;

Даний комплекс реалізований із використанням мікросервісної архітектури. Такий підхід до створення застосунків полягає у відмові від єдиної, монолітної структури та використанні сукупності невеликих, самодостатніх, незалежних сервісів, що спілкуються між собою за допомогою HTTPS протоколу.

Переваги мікросервісної архітектури над монолітною:

- Високий рівень незалежності: незалежна розробка, незалежне розгортання.
- Незалежне масштабування.
- Невелика кодова база зменшує кількість конфліктів та дозволяє швидко залучувати нових розробників.
- Простота заміни однієї реалізації сервісу іншою.
- Простота додавання нового функціоналу в систему.
- Ефективне використання ресурсів.
- Еластичність: вихід з ладу одного сервісу зазвичай не призводить до виходу з ладу всієї системи

Кожен сервіс працює в окремому docker- контейнері, що знаходятся під контролем системи управління контейнерами – Kubernetes.

Логіка сервісів та внутрішні обчислювальні методи реалізовані за допомогою мови програмування Java 8 версії.

Для зберігання інформації використовуються NoSQL бази даних MongoDB та Redis.

В якості хмарного сховища ключів застосовуються високопродуктивні мережні криптомодулі Гряда-301, які поєднані у відмовостійкий кластер. Зовнішній вигляд криптомодуля зобрадено на рисунку 2.3:

Рисунок 2.3 – Мережний криптомодуль Гряда-301

Проаналізувавши повний набір можливостей мережного кріптомодуля згідно [15] можно зробити висновок, що засіб виконує наступні функції:

- автентифікацію ЕОМ при доступі до модуля;
- генерацію особистих та відкритих ключів для алгоритму ЕЦП;
- генерацію особистих та відкритих ключів для протоколу розподілу ключів;
- генерацію ключів для алгоритму шифрування та генерацію випадкових послідовностей на основі апаратного генератора;
- зберігання особистих ключів у внутрішній пам'яті та захист їх від НСД;
- формування і перевірку ЕЦП;
- обчислення геш-функції;
- розподіл ключових даних на основі асиметричного протоколу розподілу;
- зберігання довільних даних у внутрішній пам’яті та захист їх від НСД;

- резервне копіювання ключів на зовнішні носії (електронні ключі) та відновлення ключів з носіїв;
- контроль цілісності і працездатності вбудованого програмного забезпечення та ін.

Конструкція та технічні характеристики засобу

Мережний криптомодуль виконаний у вигляді окремого мережного вузла. Конструктивно мережний криптомодуль являє собою подовжену системну платформу у металевому корпусі висотою 1U та призначену для встановлення в 19-ти дюймову стійку. Засіб має 2 мережних інтерфейси Ethernet 10/100/1000 та 2 мережних інтерфейси SFP+ (1000/10000). Інтерфейси можуть бути налаштовані в режимі агрегації [15].

Мережний криптомодуль реалізує наступні криптографічні алгоритми та протоколи [15]:

- шифрування за ДСТУ ГОСТ 28147:2009 (режим простої заміни, режим гамування та режим вироблення імітовставки);
- ЕЦП за ДСТУ 4145-2002 (всі довжини ключів передбачені стандартом - поля 163, 167, 173, 179, 191, 233, 257, 307, 367, 431), RSA за PKCS#1 (IETF RFC 3447) та ECDSA за ДСТУ ISO/IEC 14888-3:2014;
- гешування за ГОСТ 34.311-95;
- протоколи розподілу ключів за ДСТУ ISO/IEC 15946-3

Генерація ключових даних здійснюється згідно методики генерації ключових даних, яка погоджена з Адміністрацією Держспецзв'язку (ЄААД.468244.020 Д1.05) [15].

Швидкість формування ЕЦП за ДСТУ 4145-2002, поле 257: час формування ЕЦП - 0,14 мс, кількість формувань ЕЦП - 6900 формувань/с. Швидкість формування спільного секрету за ДСТУ ISO/IEC 15946-3 (Діффі-Гелмана в гр.т. еліптичної кривої), поле 431: час формування спільного секрету - 1,15 мс, кількість формувань спільного секрету (державного): 875 формувань/с [15].

Швидкість формування ЕЦП за RSA, 2048 біт (SHA-256): час формування ЕЦП - 4,5 мс, кількість формувань ЕЦП - 225 формувань/с. Швидкість формування ЕЦП за ECDSA, NIST P-256 (secp256r1), 256 біт: час формування ЕЦП - 0,32 мс, кількість формувань ЕЦП - 3100 формувань/с [15].

Швидкодія протоколу розподілу ключів за RSA: час розшифрування даних - 4,5 мс, кількість розшифрувань даних - 220 розшифрувань/с. Швидкодія протоколу розподілу ключів за ECDH: час формування спільного секрету - 5,12 мс, кількість формувань спільного секрету - 196 формувань/с [15].

Кількість ключів – 4 194 304 (зберігання наборів ключів ЕЦП та протоколів розподілу) [15].

Апаратна реалізація мережного криптомодуля забезпечує захищеність виконання усіх криптографічних перетворень усередині модуля та унеможливлює доступ до особистих ключів з боку апаратного-програмного середовища [15].

Особисті ключі генеруються, зберігаються та використовуються тільки у середині мережного криптомодуля, та жодним способом не потрапляють за його межі. Зберігання особистих ключів та інших ключових даних здійснюється у постійному запам’ятовуючому пристрої мережного криптомодуля (його внутрішнього криптомодуля) [15].

2.7 Розробка алгоритму формування (генерації) КЕП

Для генерації ключів та формування сертифікатів розроблено наступний алгоритм роботи сервісу генерації GENERATION-SERVICE, який зображено на рисунку 2.4:

Рисунок 2.4 – Алгоритм роботи сервісу генерації GENERATION-SERVICE

Після отримання та обробки даних, необхідних для генерації ключів, формується спеціальний обьект для формування заявок на генерацію сертифікатів, та за допомогою криптобібліотеки взаємодіє з криптомодулем. В криптомодулі генеруються ключі та заявки на сертифікат підпису та шифрування. Ключі генеруються за стандартом ДСТУ 4145-2002. Отримані заявки відправляються на сервіс регістрації (RA) та в результаті позитивного відгуку, завантажуються сформовані сертифікати користувача за CMP протоколом. Наступною дією є перевірка статусу отриманого сертифіка за OCSP протоколом та фомування договору про надання електронних довірчих послуг між надавачем та клієнтом. Договір підписується технічним ключем центру сертифікації та ключами клієнта. Після чого формується відповідь і відправляється на сервіс зовнішньої взаємодії.

Для роботи сервісу підписання данних SIGN-SERVICE реалізован алгоритм. Схема алгоритму зображено на рисунку 2.5:

Рисунок 2.5 – Алгоритм роботи сервісу підпису SIGN-SERVICE

Після отримання даних, які представляють собою вже обрахований хеш документу, завантажується сертифікат підпису користувача за CMP протоколом та перевіряється його статус за OCSP. в результаті успішної перевірки та наявності необхідного ключа в криптомодулі дані підписуються і формується відповідь. Для коректного створення КЕП необхідно використовувати функцію гешування ГОСТ 34.31195. Обьеднання оригіналу документу та підпису має виконати користувач.

2.8 Розробка програмного засобу для реалізації взаємодії із зовнішнім програмним інтерфейсом комплексу

Для демонстрації можливостей роботи комплексу був розроблений програмний засіб для взаємодії із зовнішнім програмним інтерфейсом. Необхідні функціонал для взаємодії надає сервіс CLOUD-API-BACK;

Програмне забезпечення представляє собою клієнт-серверний додаток. Цей додаток забезпечує функціонування як клієнтської частини так і бізнес-логіки. Модулі працюють окремо один від одного. Щоб скористатися программою потрібна ліше актуальна версія браузера. Саме браузер виступає в ролі клієнта. На рисунку 2.6 представлена взаємодія з зовнішнім програмним інтерфейсом комплексу SmartID із використанням мобільного додатку Private24 Mobile.

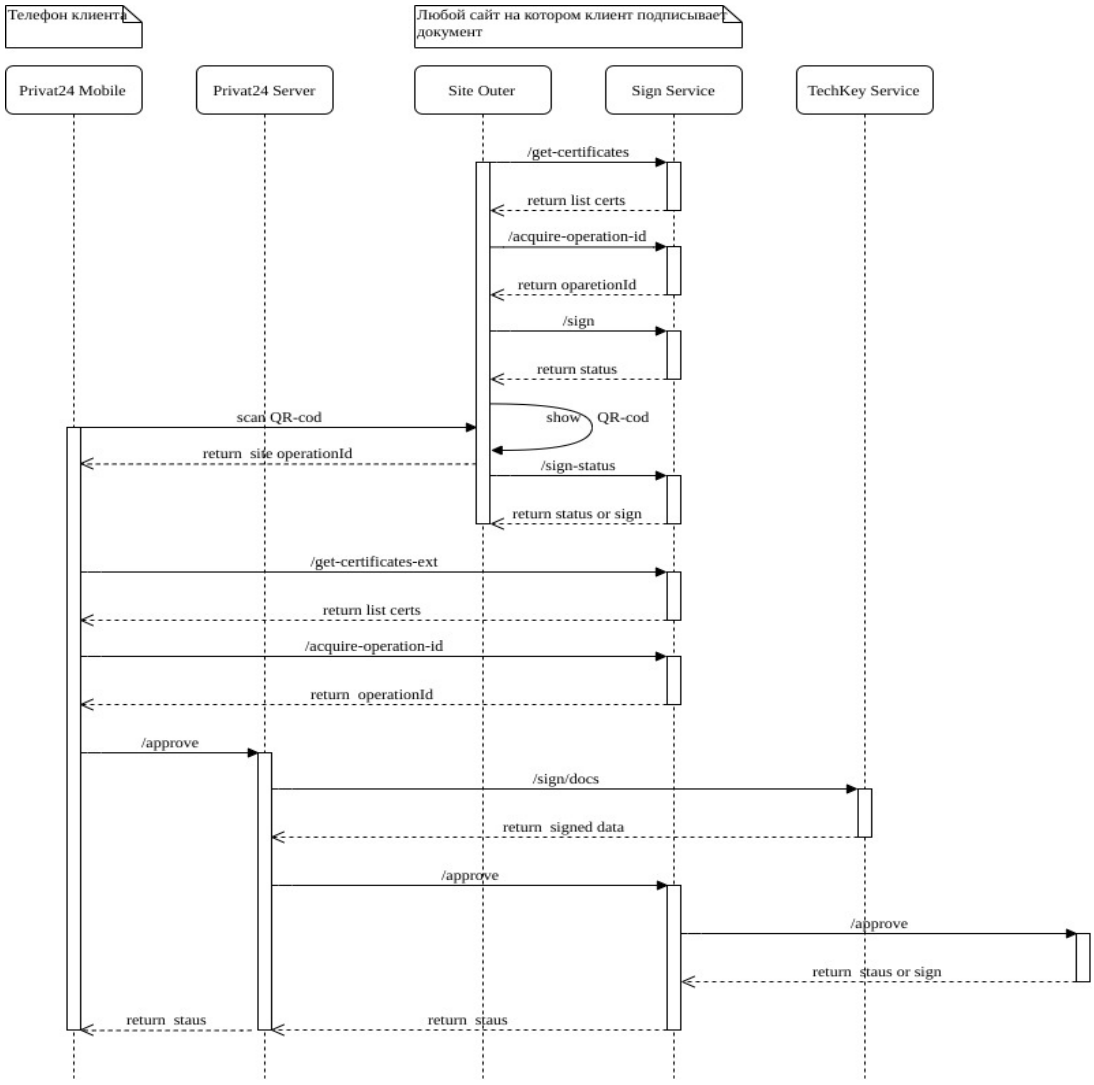

Рисунок 2.6 – Структура програмної взаємодії

Реалізований програмний засіб складається з декількох частин:

1) Інтерфейс користувача (front-end)
2) Серверна частина (back-end)

Інтерфейс користувача реалізований за допомогою JavaScript бібліотеки ReactJS. Вона призначена для створення користувацьких інтерфейсів, вирішувати проблеми часткового оновлення вмісту веб-сторінки, які виникають при розробці односторінкових застосунків. Бібліотека React дозволяє розробникам створювати дуже великі веб-застосунки, які використовують дані, що змінюються з часом, без перезавантаження сторінки.

Інтерфейс користувача дозволяє генерувати сертифікати та підписувати документи. Зовнішній вигляд інтерфейсу користувача зображено на рисунку 2.7

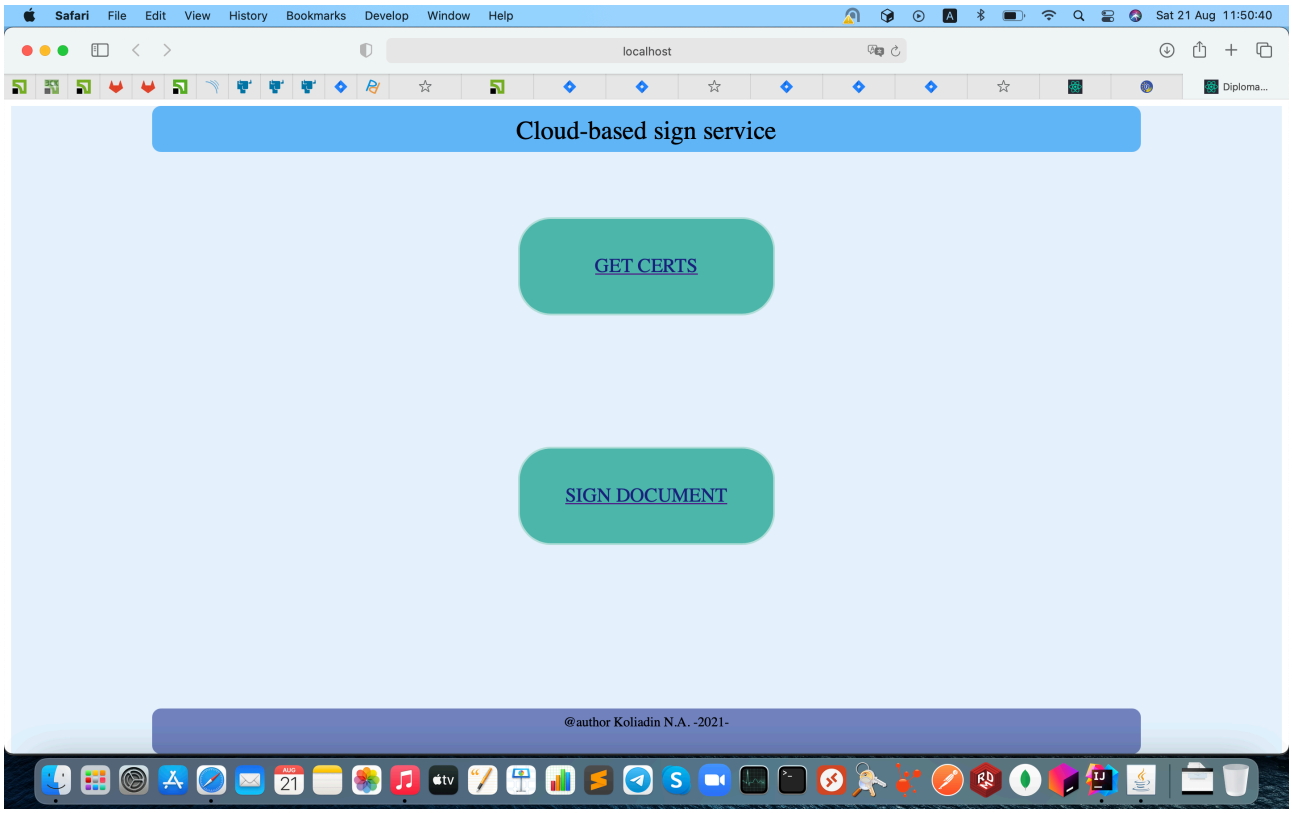

Рисунок 2.7 – Зовнішній вигляд інтерфейсу

Серверна частина реалізована за допомою мови програмування JAVA 8 версії. Вона повністью реалізує взаємодію з програмним інтерфейсом комплексу. Засіб створений із використанням клієнт-серверної архітектури REST. Серверною частини є веб-сервер Grizzly та Jersey Framework. Для збірки

проекту використано фреймворк Maven. Середовищем розробки обрано IntelliJ IDEA. Код застосунків знаходиться у відкритому доступі на git-репозиторії:

1) Інтерфейс користувача - https://github.com/nikitakoliadin/diploma.project.front та в додатку А
2) Серверна частина - https://github.com/nikitakoliadin/diploma.project.back та в додатку Б

2.9 Висновки

У другому розділі дипломної роботи проаналізовані підходи для реалізації алгоритму КЕП. Проаналізована робота кваліфікованих надавачів електронних довірчих послуг. Розроблено алгоритм функціонування комплексу.

Виконана поставлена задача з реалізації комплексу з використанням хмарного сховища та створений програмний засіб для використання можливостей комплексу.

РОЗДІЛ 3
ДОСЛІДЖЕННЯ ЕКОНОМІЧНОЇ ДОЦІЛЬНОСТІ РОЗРОБКИ КОМПЛЕКСУ

3.1 Вступ

Метою даного розділу є дослідження економічної доцільності створення і забезпечення роботи комплексу надання електронних довірчих послуг для кваліфікованих надавачів.

Для визначення ефективності необхідно буде розрахувати:

1) капітальні витрати на придбання апаратури, розробку програмного забезпечення, налагодження та підтримку складових частин комплексу;
2) річні експлуатаційні витрати на впровадження та підтримку комплексу;
3) річний економічний ефект від впровадження комплексу;
4) показники економічної ефективності в результаті інтеграції запропонованого комплексу в існуючу структуру центру сертифікації.

3.2 Розрахунок фіксованих (капітальних) витрат

Капітальні інвестиції – це кошти, призначені для створення та придбання основних фондів і нематеріальних активів, що підлягають амортизації [16].

До фіксованих витрат, що повинні бути здійснені в рамках реалізації комплексу , необхідно включити:

- вартість створення документації (розробка алгоритмів, політик функціонування складових комплексу);
- вартість апаратного та ліцензійного програмного забезпечення, необхідного для інтеграції комплексу;
- вартість створення програмного забезпечення без урахування витрат на підтримку та обслуговування;

- витрати на інтеграцію нового програмного та апаратного забезпечення із середовищем, що вже функціонує.
- витрати на первісні закупівлі апаратного забезпечення.

3.2.1 Визначення витрат на створення програмної частини комплексу

Для розрахунку загальної вартості реалізації комплексу необхідно враховувати такі показники:

- визначення трудомісткості розробки та опрацювання ПЗ;
- розрахунок витрат на створення програмного продукту;
- оцінку швидкодії та надійності роботи розробленого продукту.

3.2.1.1 Визначення трудомісткості розробки та опрацювання програмного продукту

Трудомісткість розробки можна розрахувати за формулою (3.1):

$$t = t_{\text{ТЗ}} + t_{в} + t_{а} + t_{\text{пр}} + t_{\text{опр}} + t_{д}, \text{ годин, (3.1)}$$

де $t_{\text{ТЗ}}$ – тривалість складання технічного завдання на розробку ПЗ;

$t_{в}$ – тривалість вивчення ТЗ, літературних джерел;

$t_{а}$ – тривалість розробки блок-схеми алгоритму;

$t_{\text{пр}}$ – тривалість програмування за готовою блок-схемою;

$t_{\text{опр}}$ – тривалість опрацювання програми на ПК;

$t_{д}$ – тривалість підготовки технічної документації на ПЗ.

Складові трудомісткості визначаються на підставі умовної кількості операторів у програмному продукті. Умовну кількість операторів системи виявлення можна визначити за формулою(3.2):

$$Q = q \times c \times \left(1 + p\right), \text{ штук, (3.2)}$$

де q – очікувана кількість операторів;

c – коефіцієнт складності програмного забезпечення;

p – коефіцієнт корекції програми у процесі опрацювання.

Коефіцієнт складності програмного забезпечення, що розробляється, відносно типового завдання складатиме:

$$c = 2,7.$$

Можлива корекція функціонування програмної частини комплексу чи алгоритму є вірогідною через можливе зростання кількості користувачів програмного забезпечення. Тому можлива корекція складатиме:

$$p = 0,08.$$

Очікувана кількість операторів програми складає:

$$q = 70.$$

Враховуючи параметри, що приведені вище, умовна кількість операторів програмного забезпечення, що розробляється, складатиме:

$$Q = 70 \times 2,7 \times (1 + 0,08) = 205 \text{ (штук)}.$$

Приймаючи середню кваліфікацію спеціаліста та специфіку роботи, складові показника трудомісткості розробки та опрацювання реалізованого методу будуть такі:

- Час складання технічного завдання на розробку програмних рішень:

$$tmз = 20 \text{ години};$$

- Тривалість опрацювання технічного завдання визначається за формулою (3.3):

$$tв = \frac{Q \times B}{(75\ldots85) \times k}, годин, (3.3)$$

де В – коефіцієнт збільшення тривалості роботи над розробкою через недостатнього опису завдання;

k – коефіцієнт, що залежить від стажу розробника.

Приймаючи відповідні значення для даних коефіцієнтів, маємо такі результати:

$$B = 1{,}4;$$

$$k = 1{,}0;$$

$$tв = \frac{205 \times 1{,}4}{75} = 4(години).$$

- Тривалість розробки проекту (алгоритму), за яким працюватиме метод, розраховується за формулою (3.4):

$$ta = \frac{Q}{(20\ldots25) \times k}, годин. (3.4)$$

Тривалість розробки алгоритму складатиме:

$$ta = \frac{205}{20} = 10(годин).$$

- Тривалість процесу імплементації програмного рішення розраховується за формулою (3.4) і складатиме:

$$tпр = \frac{205}{20} = 10(годин).$$

- Час опрацювання реалізованого методу на ПК можна розрахувати за наступною формулою (3.5):

$$tопр = \frac{1{,}5 \times Q}{(4\ldots5) \times k}, годин. \quad (3.5)$$

Час опрацювання реалізованого методу на ПК складатиме:

$$tопр = \frac{1{,}5 \times 205}{4} = 80(годин).$$

- Час розробки експлуатаційної документації для даного програмного забезпечення можна розрахувати за формулою (3.6):

$$tд = \frac{Q}{(15\ldots20) \times k} + \frac{Q}{(15\ldots20)} \times 0{,}75, годин. \quad (3.6)$$

Час розробки експлуатаційної документації складатиме:

$$tд = \frac{205}{15} + \frac{205}{15} \times 0{,}75 = 21(годин).$$

Загальна трудомісткість створення програмного забезпечення, складатиме:

$$t = 20 + 4 + 10 + 10 + 80 + 21 = 145(годин).$$

3.2.1.2 Розрахунок витрат на розробку програмного продукту

Згідно до дисертації [16] маємо, що витрати на розробку програмного продукту Кпз складаються із витрат на заробітну плату розробника програмного забезпечення Ззп та вартості витрат машинного часу, який необхідний для опрацювання програми на ПК Змч:

$$К_{пз} = (З_{зп} + З_{мч}) \times N,$$

де N – кількість фахівців,.

За результатами аналізу ринку послуг програмного забезпечення середньогодинна заробітна платня розробника складатиме:

$$Зпр = 250 \text{ грн/год.}$$

Витрати на заробітну плату виконавця складатимуть:

$$Ззп = 145 \times 250 = 36250(грн).$$

Вартість машинного часу для впровадження комплексу визначається за формулою (3.7):

$$З_{мч} = t_{опр} \cdot C_{мч} + t_{д}, \text{ грн, } (3.7)$$

де $t_{опр}$ – трудомісткість налогодження всіх необхідних операцій на ПК, годин (80 год);

$t_д$ – трудомісткість підготовки документації на Пк, годин (40 год);

$C_{мч}$ – це вартість однієї години машинного часу ПК, грн./година.

Вартість однієї години машинного часу ПК визначається за формулою (3.8):

$$C_{мч} = P \cdot C_e + \frac{Ф_{зал} \cdot H_a}{F_p} + \frac{K_{лпз} \cdot H_{апз}}{F_p}, \quad (3.8)$$

де Р – встановлена потужність ПК, 0.5 кВт;

C_e – тариф на електричну енергію, 1.68 грн/кВт·година;

$Ф_{зал}$ – залишкова вартість ПК на початок року, 5000 грн.;

H_a – річна норма амортизації на програмний комплекс, 0.1 частки одиниці;

$H_{апз}$ – річна норма амортизації на ліцензійне ПЗ, 0.1 частка одиниці.;

$K_{лпз}$ – вартість ліцензійного програмного забезпечення, до вартості входить ціна за використання криптобібліотеки (3000 грн) та ліцензія на середовище розробки (2000грн);

F_p – річний фонд робочого часу (40-годинного робочого тижня F_p = 1920 год).

$$\text{Смч} = 0.5 \times 1{,}68 + \frac{5000 \times 0.1}{1920} + \frac{3000 \times 0.1}{1920}$$

$$\text{Смч} = 1.5 \text{ грн/год}$$

$$Змч = 70 \times 1.5 \times 21 = 2205(грн).$$

$$К_{пз} = (36250 + 2205) \times 2 = 77000(грн)$$

Таким чином, визначити остаточні капітальні витрати на проектування та впровадження методу виявлення атак можна за формулою (3.9) [16]:

$$К = К_{пр} + К_{зпз} + К_{пз} + К_{аз} + К_{н}, \quad (3.9)$$

де $К_{пр}$ – вартість розробки проекту інформаційної безпеки та залучення для цього зовнішніх консультантів, тис. Грн;

$К_{зпз}$ – вартість закупівель ліцензійного основного й додаткового програмного забезпечення (ПЗ), 5000 тис. грн;

$К_{пз}$ – вартість створення основного й додаткового програмного забезпечення, 77000 тис. грн;

$К_{аз}$ – вартість закупівлі апаратного криптомодуля “Гряда-301” та моноблоку DELL , 400 000 тис. грн;

$К_{н}$ – витрати на встановлення обладнання та налагодження системи, 5000 тис. грн.

$$K = 77000 + 5000 + 400\,000 + 5000 = 487\,250 \text{ тис. грн} \quad (3.10)$$

3.3. Розрахунок поточних (експлуатаційних) витрат

Експлуатаційні витрати – це поточні витрати на експлуатацію та обслуговування об'єкта проектування за визначений період (наприклад, рік), що виражені у грошовій формі [16].

Витрати на підтримку функціонування (щороку):

$$С_1=(З_м \cdot \times N) \times \cdot m= (20\,000 \cdot \times 2) \times 12= 480\,000, \text{ грн.} \quad (3.11)$$

де m - кількість місяців на рік,

N – кількість фахівців,

$З_м$ – середня заробітна плата фахівця

Річні поточні (експлуатаційні) витрати на функціонування системи інформаційної безпеки складають [16]:

$$С = С_а + С_з + С_{ел} + С_{тос}, \text{ грн,} \quad (3.12)$$

де Са - річний фонд амортизаційних відрахувань (Са) визначається у відсотках від суми капітальних інвестицій за видами основних фондів і нематеріальних активів (ПЗ) [16]:

$$С_а = \frac{К_{пз}}{2} + \frac{К_{аз}}{2} = \frac{400000}{2} + \frac{77000}{2} = 238\ 500, \text{ грн, (3.13)}$$

$С_з$ - річний фонд заробітної плати інженерно-технічного персоналу, що обслуговує систему інформаційної безпеки, складає:

$$С_з = (З_м + 22\%) \cdot N \cdot m = (20\ 000 + 4\ 400) \cdot 2 \cdot 12 = 565\ 600 \text{ грн, (3.14)}$$

До річного фонду заробітної плати додається єдиний внесок (22%) на загальнообов'язкове державне соціальне страхування – консолідований страховий внесок, збір якого здійснюється відповідно до класів професійного ризику виробництва, до яких віднесено платників єдиного внеску, з урахуванням видів їх економічної діяльності [16].

Розмір єдиного внеску на загальнообов'язкове державне соціальне страхування визначається на підставі встановленого чинним законодавством відсотка від суми основної та додаткової заробітної плати [16].

$С_е$ - Вартість електроенергії, що споживається апаратурою системою інформаційної безпеки протягом року, визначається за формулою:

$$С_{ел} = Р \cdot F_р \cdot Ц_е, \text{ грн, (3.15)}$$

де Р – встановлена потужність апаратури інформаційної безпеки, 0.5 кВт;

Fр – річний фонд робочого часу системи інформаційної безпеки (за 40-годинного робочого тижня Fр = 1920 год);

Це – тариф на електроенергію, грн/кВт·годин, 1.68 грн/кВт·година.

$$С_{ел} = 0{,}5 \cdot 1920 \cdot 1{,}68 = 1\ 612 \text{ грн.}$$

$С_{тос}$ - Витрати на технічне й організаційне адміністрування та сервіс системи інформаційної безпеки складає 1,5% від вартості капітальних витрат (7200 грн).

Отже, річні поточні (експлуатаційні) витрати складають:

$$С =238500 + 565\ 600 + 1\ 612 + 7200 = 812\ 900 \text{ грн.}$$

3.4 ОЦІНКА МОЖЛИВОГО ЗБИТКУ ВІД АТАКИ (ЗЛОМУ) НА ВУЗОЛ АБО СЕГМЕНТ КОРПОРАТИВНОЇ МЕРЕЖІ

3.4.1 Оцінка величини збитку

Кінцевим результатом впровадження й проведення заходів щодо забезпечення інформаційної безпеки є величина відвернених втрат, що розраховується, виходячи з імовірності виникнення інциденту інформаційної безпеки й можливих економічних втрат від нього. По суті, ця величина відображає ту частину прибутку, що могла бути втрачена [16].

Для розрахунку вартості такого збитку можна застосувати наступну спрощену модель оцінки [16].

Необхідні *вихідні дані* для розрахунку:

- $t_п$ – час простою вузла корпоративної мережі, у годинах;
- $t_в$ – час, необхідний для відновлення системи, у годинах;
- $t_{ві}$ – час відновлення інформації, у годинах;
- $З_0$ – заробітна платня обслуговуючого персоналу;
- $З_с$ – заробітна платня співробітників атакованого вузла;
- $Ч_0$ – чисельність обслуговуючого персоналу;
- $Ч_с$ – чисельність співробітників атакованого вузла;
- О – обсяг продажів атакованого вузла, у грн.;
- П – вартість доопрацювання, модифікації програмного забезпечення чи апаратного устаткування;
- І – число атакованих вузлів;

- N – середнє число атак на рік.

Втрати від простою атакованого вузла можна визначити за формулою (3.16):

$$U = Пп + Пв + V, \quad (3.16)$$

де $П_п$ – оплачувані втрати робочого часу при простоях системи, у грн.;

$П_в$ – вартість відновлення системи (переустановлення, зміна налаштувань), у грн.;

V – втрати від зниження обсягу продажів за час простою системи, у грн..

Для розрахунку витрат на оплату робочого часу при простоях системи використовується формула (3.17):

$$Пп = \frac{\sum Зс}{F} \times tn, \quad (3.17)$$

де F – місячний фонд робочого часу, що становить 176 год..

При кількості працівників атакованого вузла, що становить 7 осіб, та середній заробітній платні у 25000 грн., маємо число витрат на оплату робочого часу при простоях системи складатимуть:

$$Пп = \frac{175000 \times 7}{176} = 6960(грн.).$$

Для розрахунку вартості відновлення системи використовується формула (3.18):

$$Пв = Пві + Ппв + Пзч, \quad (3.18)$$

де $П_{ві}$ – витрати на повторне уведення інформації у систему, у грн.;

$П_{пв}$ – витрати на відновлення вузла системи, у грн.;

$П_{зч}$ – вартість доопрацювання та зміни в коді системи.

Витрати на повторне уведення інформації у систему розраховуються за формулою (3.19):

$$Пві = \frac{\sum Зс}{F} \times тві, (3.19)$$

Таким чином, маємо наступний показник витрат на відновлення інформації у системі:

$$Пві = \frac{175000 \times 6}{176} = 5965(грн.).$$

Витрати на відновлення вузла системи визначаються за формулою (3.20):

$$Ппв = \frac{\sum Зо}{F} \times тв, (3.20)$$

$$Ппв = \frac{100000 \times 3}{176} = 3409(грн.).$$

Вартість відновлення системи становить:

$$П_в = 6960 + 3409 + 5965 = 16360 \text{ (грн.)}.$$

Для розрахунку втрат від зниження обсягу продажів під час простою використовується формула (3.21):

$$V = \frac{О}{Fг} \times (тп + тв + тві), (3.21)$$

де $F_г$ - річний фонд робочого часу роботи організації, $F_г = 2080$ год..

Для розрахунку обсягу продажів атакованого вузла, використовується формула (3.22):

$$О = П_{оп} \times d \times П_{кіл} \text{ ,(3.22)}$$

де $П_{оп}$ – вартість проведення транзакції, 50 000 грн;

$П_{кіл}$ - кількість транзакцій оброблених у момент часу, 80 од. ;

d – сума яку отримує банк за кожну операцію з КЕП, 10%;

$$V = \frac{50000*0,1*150}{2080} \times \left(7 + 6 + 3\right) = 6490\left(грн.\right)$$

Число упущеної вигоді внаслідок здійснення атаки становить:

$$U = 6960 + 16360 + 6490 = 29840 \text{ (грн.).}$$

Таким чином, загальний збиток від атаки на вузол або сегмент корпоративної мережі організації складе (3.23):

$$B = \sum_{I} \sum_{N} U \text{(3.23)}$$

$$В = 5 \times 40 \times 29840 = 5\ 968\ 000 \text{ грн.}$$

3.4.2 Загальний ефект від впровадження комплексу хмарного КЕП

Загальний ефект від впровадження комплексу визначається з урахуванням ризиків порушення інформаційної безпеки і становить:

$$E = B \cdot R - C, \text{ (3.24)}$$

де В – загальний збиток від атаки на вузол або сегмент корпоративної мережі, тис. грн;

R – очікувана імовірність атаки на вузол або сегмент корпоративної мережі, частки одиниці;

С – щорічні витрати на експлуатацію комплексу, тис. грн.

Таким чином, загальний ефект складатиме:

$$E = 5\,968\,000 \times 0{,}4 - 612000 = 1\,776\,000(грн.).$$

3.5. Визначення та аналіз показників економічної ефективності системи інформаційної безпеки

Оцінка економічної ефективності впровадження рекомендацій, здійснюється на основі визначення та аналізу наступних показників [16]:

a) коефіцієнт повернення інвестицій (ROI). У сфері інформаційної безпеки йому відповідає показник ROSI (Return on Investment for Security);
b) термін окупності капітальних інвестицій То.

Ключовою перевагою показника ТСО є те, що він дозволяє зробити висновки про доцільність реалізації проекту в області інформаційної безпеки на підставі оцінки одних тільки витрат [16].

У цьому випадку необхідно порівняти сукупну вартість володіння, розраховану для двох варіантів проектного рішення щодо створення або удосконалення системи інформаційної безпеки, і вибрати варіант із найменшою з них [16].

Коефіцієнт повернення інвестицій ROSI показує, скільки гривень додаткового прибутку приносить одна гривня капітальних інвестицій на впровадження системи інформаційної безпеки [16].

Щодо до інформаційної безпеки говорять не про прибуток, а про запобігання можливих втрат від атаки на сегмент або вузол корпоративної мережі, а отже [16]:

$$\text{ROSI} = \frac{E}{K} \quad (3.25)$$

де Е – загальний ефект від впровадження системи інформаційної безпеки , тис. грн;

К – капітальні інвестиції за варіантами, що забезпечили цей ефект, тис. Грн.

$$ROSI = \frac{1\ 776\ 000}{487250} = 3{,}65$$

Економічно проект можна визначити як ефективний, а його впровадження є доцільним, якщо виконується наступна умова (3.26):

$$ROSI > \frac{Nдеп - Nінф}{100}, \quad (3.26)$$

де $N_{деп}$ – річна депозитна ставка або прибутковість альтернативного варіанту вкладення коштів, у %;

$N_{інф}$ – річний рівень інфляції, у %.

Приймаючи значення рівня інфляції поточного року рівним 4,2% та середнє значення депозитної ставки рівною 16%, маємо відповідний результат нерівності:

$$ROSI > 0{,}122\,.$$

$$3{,}65 > 0{,}122$$

Для розрахунку терміну окупності капітальних вкладень використовується наступна формула (3.27):

$$\frac{1}{ROSI}, років. \quad (3.27)$$

Термін окупності капітальних інвестицій То показує, за скільки років капітальні інвестиції окупляться за рахунок загального ефекту від впровадження системи інформаційної безпеки [16]:

$$T_o = \frac{K}{E} = \frac{1}{\text{ROSI}} = 0.27 \text{ років} \approx 98 \text{ днів}.$$

3.6 Висновки

В результаті розрахунку витрат на впровадження комплексу для підвищення рівня інформаційної безпеки в організаціях, які надають електронні довірчі послуги було визначено, що розмір капітальних витрат складатиме 487 250 грн, а щорічні експлуатаційних витрати 812 900 грн.

Коефіцієнт повернення інвестицій ROSI показав, що 3,6 грн додаткового прибутку приносить одна гривня капітальних інвестицій витрачених на комплекс.

Загальний ефект від впровадження комплексу визначається з урахуванням ризиків порушення інформаційної безпеки та становить 1468000 грн.

Було доведено, що застосування комплексу в організаціях зменшить можливий рівень збитків у розмірі до 5 968 000 грн на рік та окупиться лише за 98 дні.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Вимоги безпеки праці під час виконання робіт на робочому місці

Питання охорони праці - дуже важливе та його необхідно вирішувати протягом всіх етапів трудового процесу незалежно до галузі професійної діяльності. Основні положеня, що стосуються реалізації конституційного права громадянина що до охорони її життя та здоров’я у процесі трудової діяльності міститься у законі України "Про охорону праці" згідно з Постановою Верховної Ради України No 345-VI від 2 вересня 2008 року [17].

Відповідно до нормативно правових актів з охорони праці «Правила безпечної експлуатації електроустановок споживачів», що затверджено: наказ Держнаглядохоронпраці України №4 від 9 січня 1998 року [18], та НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час робот з екранними пристроями», затверджені наказом Міністерства соціальної політики України "Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями" № 207 від 14 лютого 2018 року [19], вимоги безпеки під час роботи з електронно обчислювальною машиною з відео-дисплейними терміналами (далі - ВДТ) і периферійними пристроями (далі - ПП) наступні:

- Щодня перед початком роботи оператор ЕОМ повинен перевірити своє робоче місце на наявність ознак пошкодження обладнання;
- Очищати перед початком роботи щодень екран від пилу та інших забруднень;
- Перед початком роботи оператор ЕОМ повинен перевірити правильність організації робочого місця;
- Обладнання, принесене у холодну пору року з вулиці в робоче приміщення, можна підключати до електричної мережі тільки після того, як температура обладнання зрівняється з температурою повітря відповідного робочого приміщення;

- Перед початком роботи оператор ЕОМ повинен перевірити правильність підключення обладнання ЕОМ до електромережі;

Забороняється:

- відключати захисні пристрої, самочинно проводити зміни у конструкції та складі ЕОМ або їх технічне налагодження;
- виконувати на робочому місці, ремонт та налагодження ЕОМ;
- зберігання біля ЕОМ дискет, паперу, інших носіїв інформації, запасних блоків, деталей тощо;
- працювати з ВДТ, у яких під час роботи з'являються нестабільне зображення на екрані, нехарактерні сигнали тощо;
- доторкання до задньої панелі системного блоку при включеному живленні;
- допускання попадання вологи на поверхню системного блоку;
- приймання напоїв та їжі на робочому місці;
- вимикання живлення під час виконання активного завдання.

Про виявлення несправності обладнання або інших факторів, які створюють загрозу для життя або здоров’я працівників, необхідно негайно інформувати свого безпосереднього керівника.

Правильна оцінка небезпечних та шкідливих виробничих факторів значно впливає на забезпечення безпечних умов праці. Нервово-емоційна напруга, надмірне розумове та фізичне навантаження, фактори виробничого середовища а також різне сполучення цих причин можуть впливати на зміни в організмі людини.

У приміщенні на програміста можуть негативно впливати наступні психофізіологічні та фізичні фактори:

- підвищена або знижена температура;
- підвищена або знижена вологість;
- недостатня освітленість;
- підвищений рівень шуму;
- підвищена іонізація повітря;
- підвищений рівень електромагнітних випромінювань;

- нервово-психічні перевантаження.

Одним з найважливіших факторів, що впливають на здоров’я людини є організація робочого місця. Так у нормативно правовому акті з охорони праці НПАОП 22.1-1.01-96 «Правила охорони праці для видавництв і редакцій., що затверджено наказом Державного комітету України по нагляду за охороною праці № 122 від 18 липня 1996 року [20] йдеться мова про об’єм виробничих приміщень для програмістів. Відповідно [20] об’єм виробничих приміщень на одного працівника повинна складати 20 м3, а площа приміщень - не менше 6 м2 з урахуванням максимального числа працівників в одну зміну.

Відповідно до санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99., що затверджено Постановою Головного державного санітарного лікаря України №42 від 1 грудня 1999 року [21] робота програміста за енерговитратами відноситься до категорії легких робіт, зокрема до категорій Іа та Іб.

Мікрокліматичні умови виробничих приміщень характеризуються наступними такими показниками, як: швидкість руху повітря, температура повітря, інтенсивність теплового (інфрачервоного) опромінення, відносна вологість повітря, температура поверхні.

Таблиця 4.1 – Оптимальні мікрокліматичні умови виробничого приміщення, для категорій Іа та Іб легких робіт

Період року	Категорія робіт	Температура повітря	Відносна вологість	Швидкість руху, м/сек.
Холодний період року	Легка Іа	22 - 24	60 - 40	0,1
	Легка Іб	21 - 23	60 - 40	0,1
Теплий період року	Легка Іа	23 - 25	60 - 40	0,1
	Легка Іб	22 - 24	60 - 40	0,2

У таблиці 4.1 наведено оптимальні мікрокліматичні умови виробничого приміщення, для категорій Іа та Іб легких робіт, відповідно до [19].

У таблиці 4.2 наведено допустимі мікрокліматичні умови виробничого приміщення, для категорій Іа та Іб легких робіт, відповідно до [18].

Таблиця 4.2 – Допустимі мікрокліматичні умови виробничого приміщення, для категорій Іа та Іб легких робіт

Період року	Категорія робіт	Температура, град.С				Відносна вологість (%)	Швидкість руху, м/сек.
		Верхня межа		Нижча межа			
		На постійних робочих місцях	На непостійних робочих місцях	На постійних робочих місцях	На непостійних робочих місцях		
Холодний період року	Легка Іа	25	26	21	18	75	не більше 0,1
	Легка Іб	24	25	21	18	75	не більше 0,2
Теплий період року	Легка Іа	28	30	22	20	55 - при 28 град.С	0,2 - 0,1
	Легка Іб	28	30	21	19	60 - при 27 град.С	0,3 - 0,1

Рівні звукового тиску в октавних смугах частот, рівні звуку та еквівалентні рівні звуку на робочих місцях, обладнаних ПЕОМ і ВДТ ЕОМ, мають відповідати вимогам що наведені у таблиці 4.3 – Допустимі рівні звуку, еквівалентні рівні звуку і рівні звукового тиску в октавних смугах частот для програміста відповідно до [18].

Таблиця 4.3 – Допустимі рівні звуку, еквівалентні рівні звуку і рівні звукового тиску в октавних смугах частот для програміста

Вид трудової діяльності	Рівні звукового тиску в дБ в октанових смугах із середньогеометричними частотами, Гц									
	31,5	63	125	250	500	1000	2000	4000	8000	Рівні звуку, еквівалентні рівні звуку, дБА/дБАекв.
Програмісти ЕОМ	86	71	61	54	49	45	42	40	38	50

Устаткування, що становить джерело шуму відповідно до [21], слід розташовувати поза приміщенням для роботи ЕОМ, а також для забезпечення допустимих рівнів шуму на робочих місцях слід застосувати засоби звукопоглинання.

Відповідно до державних будівельних норм ДБН В.2.5-28:2018 «Природне і штучне освітлення», що затверджено наказом Мінрегіону №264 від 3 жовтня 2018 року [22] нормативним параметром природного освітлення є коефіцієнт природного освітлення (КПО). Коефіцієнт природного освітлення встановлюється в залежності від розряду виконуваних зорових робіт. Робота програміста відноситься до робіт середньої точності (IV розряд зорових робіт, мінімальний розмір об'єкту розрізнення складає 0,5-1,0мм), для яких при використанні бокового освітлення КПО=1,5%. Для IV розряда зорових робіт мінімальна освітленість складає 300-500 лк.

Розрахунок штучного освітлення для кімнати площею 27 м2, ширина якої складає 4,5 м, довжина – 6 м, висота – 3 м за методом коефіцієнта використання світлового потоку наведено далі.

Для визначення потрібної кількості світильників, які повинні забезпечити нормований рівень освітленості, необхідно визначити світловий потік, що падає на робочу поверхню за формулою (4.1):

$$F = \frac{ESKZ}{n}, \quad (4.1)$$

де F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк E = 300 Лк;

S – площа освітлюваного приміщення;

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (K = 1,5);

Z – відношення середньої освітленості до мінімальної (Z =1,1);

n – коефіцієнт використання світлового потоку, (залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються ($\rho_{ст.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{ст}$ = 50% і $\rho_{стелі}$ = 50%).

Обчислимо індекс приміщення за формулою (4.2)

$$i = \frac{S}{H(A+B)}, \quad (4.2)$$

де S – площа приміщення, S = 27 м^2;

H – розрахункова висота підвісу, H = 3 м;

A – ширина приміщення, A = 4,5 м;

B – довжина приміщення, B = 6 м.

Підставивши значення отримаємо: i = 0,85. Знаючи індекс приміщення, знаходимо n = 0,33. Підставимо всі значення у формулу для визначення світлового потоку F.

$$F = \frac{300 * 27 * 1.1 * 1.5}{0.33} = 40500 \text{ Лм}$$

Для освітлення використані люмінісцентні лампи типу ЛБ 40-1, світловий потік яких F=4600 Лм. Розрахуємо необхідну кількість ламп у всітильниках за формулою (4.3)

$$N = \frac{F}{Fl}, \quad (4.3)$$

де N – кількість ламп, що визначається;

F – світловий потік;

Fl – світловий потік ламп.

$$N = \frac{40500}{4600} = 8.8 = 9$$

В приміщені кожен світильник комплектується двома лампами, тобто необхідно використовувати 4 світильника з 2 працюючими лампами. Схема розташування світильників наведено на рисунку 4.1.

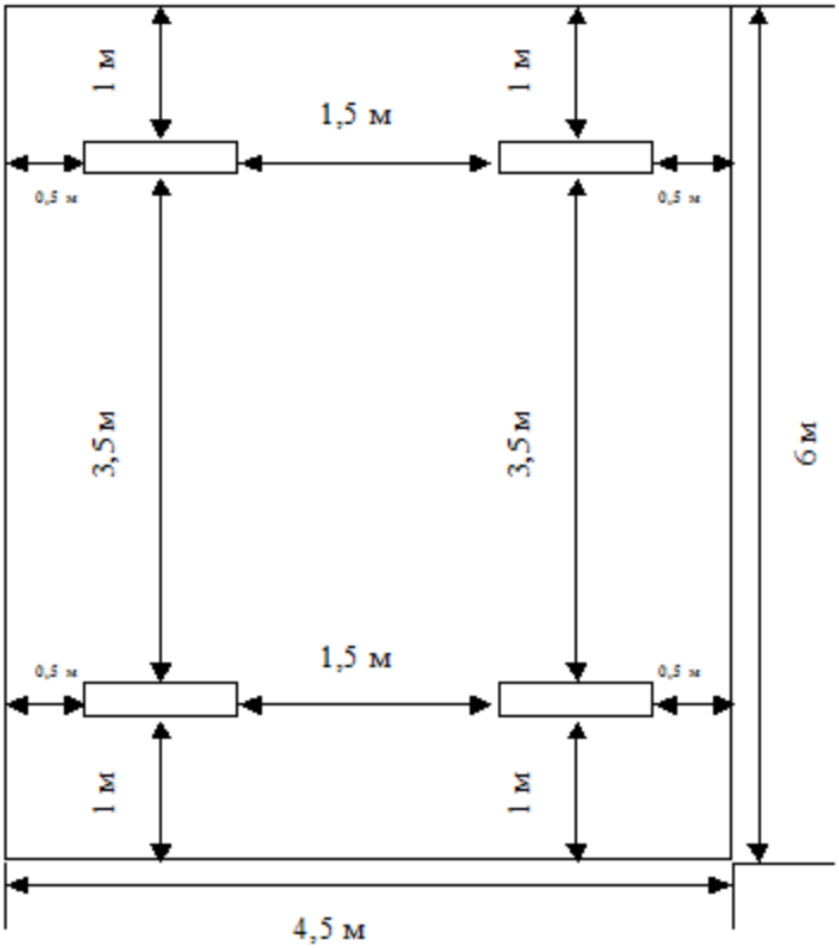

Рисунок 4.1 – Схема розташування світильників

Таким чином не задовольняється достатні умови штучного освітлення. Рекомендується додати ще один світильник.

4.2 Дій працівників в аварійних ситуаціях

У випадку Аварійної ситуації програміст зобов’язаний:

- при попаданні людини під електричну напругу негайно вимкнути електричне живлення, до прибуття лікаря надати долікарську медичну допомогу;
- при виявленні будь-яких неполадок в роботі персонального комп'ютера програміст повинен припинити роботу, вимкнути комп'ютер і повідомити про це безпосереднього керівника для організації ремонту;

- при нещасному випадку, отруєнні, раптовому захворюванні необхідно негайно надати першу допомогу потерпілому, викликати лікаря або допомогти доставити потерпілого до лікаря, а потім повідомити керівника про те, що трапилося;
- при будь-яких випадках порушень роботи технічного обладнання негайно викликати представника технічної служби;
- при загорянні обладнання негайно відключити його від електромережі, ужити заходів щодо ліквідації вогню за допомогою вуглекислотного або порошкового вогнегасник;
- у випадку виникнення різі в очах, різкого погіршення зору, виникнення головного болю, больових почуттів у пальцях та кистях рук, посилення серцебиття – негайно припинити роботу з використанням ЕОМ, повідомити про те, що сталося, свого безпосереднього керівника й звернутися до медичної установи.

План евакуації наведено на рисунку 4.2

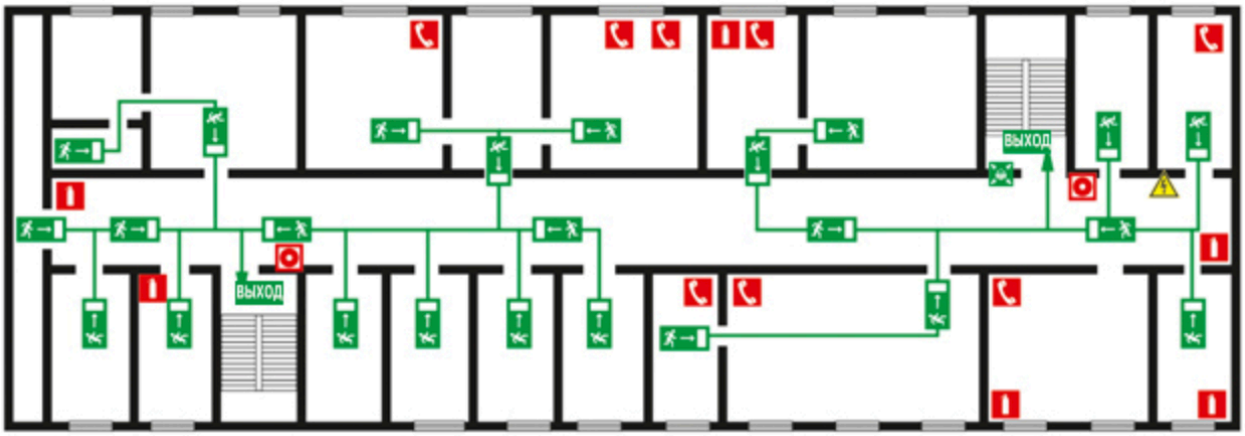

Рисунок 4.2 – План евакуації

Згідно з [23] загальними правилами надання до медичної допомоги э наступними:

- Перш за все оглянути місце пригоди і впевнитись в особистій безпеці і безпеці постраждалого;
- Провести первинний огляд постраждалого;

- Викликати швидку медичну допомогу;
- Провести вторинний огляд постраждалого, з метою виявлення інших проблем (пошкоджень), які потребують надання домедичної допомоги (розпочніть із загального огляду всього тіла, починаючи з голови).

Надання першої домедичної допомоги постраждалим при ураженні електричним струмом:

1) Як можливо швидко відокремити потерпілого від джерела струму.
2) Викликайте швидку, якщо це необхідно.
3) Покладіть та/або зігрійте людину.
4) Закрийте опіки - якщо у потерпілого є опіки, їх треба накрити стерильною марлею (якщо є під рукою) або чистою гладкою тканиною. Звичайно, тільки в тому випадку, якщо стан людини дозволяє зняти або розрізати одяг на обпалених місцях.
5) Якщо з'являються ознаки шоку - блювання, слабкість, сильна блідість, - трохи підніміть ноги, підклавши під ступні валик з речей.
6) Якщо потерпілий погано дихає або не дихає зовсім, негайно починайте робити штучне дихання рот в рот.
7) Якщо у людини немає пульсу і відсутній серцебиття, крім штучного дихання, необхідний непрямий масаж серця.

Надання першої домедичної допомоги постраждалим при пожежі:

1) Якщо горить одяг, його слід скинути або погасити полум'я, щільно накривши людини ковдрою або будь-яким шматком тканини. Обпалені ділянки одягу акуратно розрізати і скидати по частинах, у запобігання подальшої травматизації шкіри.
2) Якщо закрита рана необхідно охолоджувати водою уражену ділянку протягом 10 хвилин.
3) На поверхню рани слід накласти стерильну пов’язку.
4) Забезпечити потерпілому спокій.
5) Дати випити велику кількість рідини (чай, вода і тому подібне).
6) Негайно викликати бригаду невідкладної допомоги.

7) При можливості знеболити потерпілого, дати прийняти таблетку анальгіну.

4.3 Висновки

У четвертому розділі подані основні правила безпеки при роботі за комп’ютером, та дії при виникненні надзвичайної ситуації і правила надання медичної допомоги.

ВИСНОВКИ

Найвагомішим результатом виконання дипломної роботи на здобуття освітнього ступеня магіст є розроблений програмно-апаратний комплекс надання електронних довірчих послуг з використанням хмарного сховища ключів. Даний комплекс може мати широке практичне застосування, що було доведено в відповідних розділах роботи.

Шляхом розробки останнього, була досягнута головна мета роботи, а саме: вдосконалення процедури використання кваліфікованого електронного підпису.

В процесі реалізації, поставлених у дипломній роботі задач, були отримані наступні наукові та практичні результати:

1. Запропонований варіант приведення процедури використання кваліфікованого електронного підпису до чинного законодавства;
2. Отримане обґрунтування необхідності розробки комплексу, що стало результатом аналізу чинного законодавства України у сфері електронних довірчих послуг;
3. Досліджена технологічна структура центрів сертифікації ключів, на прикладі Акредитованого центру сертифікації ключів АТ КБ “ПРИВАТБАНК” та знайдений шлях інтеграції комплексу в єдину систему;
4. Спроектована та обґрунтована доцільність використання апаратної частини комплексу з використанням розробок від Інституту інформаційних технологій;
5. Розроблений алгоритм функціонування складових частин комплексу і обґрунтовано його використання;
6. Розроблений програмний засіб для взаємодії із зовнішнім інтерфейсом комплексу, що дасть змогу генерувати та накладати КЕП;
7. Досліджена економічна доцільність шляхом розрахунку витрат на розробку та впровадження комплексу в технологічну структуру кваліфікованих надавачів електронних довірчих послуг.

Результати дипломної магістерської роботи доповідались на XIV та XV Міжнародній науково-практичній конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті», що відбулись в Українському державному університеті науки і технологій в 2020 та 2021 роках відповідно.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про основні засади забезпечення кібербезпеки: Закон України від 08.07.2018 № 2469-VIII // Відомості Верховної Ради (ВВР), 2017, № 45, ст.403. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2163-19#Text.
2. Про електронні довірчі послуги: Закон України від 05.10.2017 № 2155-VIII // Відомості Верховної Ради. – 2017. – № 45. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2155-19#Text.
3. Дипломна робота “Система захисту даних на базі методу шифрування RSA” [Електронний ресурс]. - 2019. - Режим доступу до ресурсу: https://ela.kpi.ua/bitstream/123456789/28777/1/Yarmoshevych_bakalavr.pdf.
4. Доповідь “Електронний цифровий підпис” [Електронний ресурс]. - 2015. - Режим доступу до ресурсу: https://referatwork.ru/refs/source/ref-103718.html.
5. ДСТУ 4145-2002. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка. — К. : Держстандарт України, 2003.
6. Сучасні шляхи удосконалення процедури формування та верифікації електронно-цифрового підпису [Електронний ресурс] // Національний авіаційний університет. – 2018. – Режим доступу до ресурсу: http://jrnl.nau.edu.ua/index.php/SBT/article/view/12370.
7. Шевчук А.А. Особливості ЕЦП з відновленням повідомлення / А.А.Шевчук // Прикладная радиоэлектроника. – 2010. – Т.9, №3. – С.489-492.
8. Mobile ID: відповіді на актуальні запитання [Електронний ресурс]. - 2019. - Режим доступу до ресурсу: https://dealssign.com/blog/mobile-id-vidpovidi-na-aktualni-zapitannya/.
9. Закон України «Про електронний цифровий підпис» № 852-IV від 22.05.03: зі змінами внесеними згідно із Законом України № 222-VIII

від 02.03.2015 [Електронний ресурс]. — Режим доступу: http://zakon3.rada.gov.ua/laws/show/851-15 2. ДСТУ 4145-2002.

10. До питання про поняття довірчих електронних послуг та їх договірного забезпечення [Електронний ресурс]. — Режим доступу: http://pgp-journal.kiev.ua/archive/2018/5/15.pdf.
11. Постанова Кабінету Міністрів України від 07.11.2018 № 992 "Про затвердження вимог у сфері електронних довірчих послуг та Порядку перевірки дотримання вимог законодавства у сфері електронних довірчих послуг".
12. Центр сертифікації ключів [Електронний ресурс]. — Режим доступу: https://iit.com.ua/index.php?page=itemdetails&p=3&gtype=1&type=1&id=37.
13. Інфраструктура відкритих ключів [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Інфраструктура_відкритих_ключів.
14. Хмарні сховища даних та їх характеристики [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://informatika.udpu.edu.ua/?page_id=1896.
15. Мережний криптомодуль "Гряда-301" (високопродуктивний пристрій) [Електронний ресурс]. – Режим доступу до ресурсу: https://iit.com.ua/index.php?page=itemdetails&p=3&gtype=1&type=1&id=99.
16. Дисертація "Трансформація фінансових механізмів державного управління пенсійною системою в Україні" [Електронний ресурс]. – Режим доступу до ресурсу: http://dspace.wunu.edu.ua/bitstream/316497/5357/1/Толубяк%20Докторська.pdf.
17. Закон України «Про охорону праці» згідно з Постановою Верховної Ради України № 345-VI від 2 вересня 2008 року.
18. НПАОП 40.1–1.21–98 «Правила безпечної експлуатації електроустановок споживачів». Затверджено: наказ Держнаглядохоронпраці України № 4 від 9 січня 1998 року.

19. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час робот з екранними пристроями». Затверджено: наказ Міністерства соціальної політики України «Про затвердження вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» № 207 від 14 лютого 2018 року.
20. ДСанПІН 3.3.2.007-98 з візуальними диспейними терміналами електронно-обчислювальних машин». Затверджено Постановою Головного державного санітарного лікаря України № 7 від 10 грудня 1998 року.
21. ДБН В.2.5-28:2018 «Природне і штучне освітлення». Затверджено наказом Мінрегіону № 264 від 3 жовтня 2018 року.
22. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень». Затверджено Постановою Головного державного санітарного лікаря України № 42 від 1 грудня 1999 року.
23. Ненько С. К., Полівода Л. А. Надання першої медичної допомоги при надзвичайних ситуаціях, Херсон: «Навчально-методичний центр цивільного захисту та безпеки життєдіяльності Херсонської області», 2014, 28 с.
24. ISO/IEC 9796-3:2006: Information technology — Security techniques — Digital signature schemes giving message recovery — Part 3: Discrete logarithm based mechanisms, 2006. — URL:http://www.iso.org, doi.org/ 10.3403/30117202.
25. Хеш функція [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: https://wiki.tntu.edu.ua/%D0%A5%D0%B5%D1%88_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F#.D0.A5.D0.B5.D1.88_.D1.84.D1.83.D0.BD.D0.BA.D1.86.D1.96.D1.8F.
26. Чмора А., Современная прикладная криптография., М.: Гелиос АРВ, 2001.

27. До питання про поняття довірчих електронних послуг та їх довірного забезпечення. // ГОСПОДАРСЬКЕ ПРАВО І ПРОЦЕС. – 2018. – №5. – С. 73–76.
28. Про електронні документи та електронний документообіг : Закон України від 22.05.2003 № 851-IV // Відомості Верховної Ради України. – 2003. – № 36. – Ст. 275.
29. ДСТУ EN 419211-1:2016. Профілі захисту для пристроїв створення безпечного підпису. Частина 1.
30. Саломаа А. Криптография с открытым ключом. Пер. с англ. – М.: Мир, 1995. – 318 с.
31. Смарт Н. Криптография. Москва: Техносфера, 2005. – 528 с.
32. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка. — К. : Держстандарт України, 2003.
33. Болтьонков В. О., Єнікєєв Р. І. Практичне дослідження сучасних систем електронного цифрового підпису / В. О. Болтьонков, Р. І. Єнікєєв // Інформатика та математичні методи в моделюванні. — 2014. — Т. 4, №3. — С. 201–209.
34. Горбатов В.С., Полянская О.Ю. Г67 Основы технологии PKI. - М.: Горячая линия - Телеком, 2004. - 248 с.: ил. ISBN 5-93517-154-6.
35. Внедрение инфраструктуры открытых ключей и электронной цифровой подписи на современном предприятии / Меньшенин А.О. – Питер, 2007 – 234 стр.
36. Петров, А. А. Компьютерная безопасность. Криптографические методы защиты / А. А. Петров. – М. : ДМК, 2000. – 448 с. 13. Шаньгин, В.Ф.
37. Информационная безопасность компьютерных систем и сетей : учеб. пособие / В. Ф. Шаньгин. – М. : ИД «Форум»: Инфра-М, 2011. – 416 с.
38. Регламент АЦСК [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://acsk.privatbank.ua/arch/docs/Reglament_ACSK_PB.pdf.

39. Довірчий список [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://czo.gov.ua/normative-documentation.
40. Комплекс центру сертифікації ключів [Електронний ресурс]. – 2018.– Режим доступу до ресурсу: https://www.iit.com.ua/index.php?page=itemdetails&p=3&gtype=1&type=1&id=32.

ДОДАТОК А. Програмна реалізація інтерфейсу користувача

diploma.project.front/public/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Diploma project front"
    />
    <link rel="apple-touch-icon" href="logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>Diploma project front</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

diploma.project.front/public/manifest.json

```
{
  "short_name": "Diploma project front",
  "name": "Diploma project front",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    },
    {
      "src": "logo192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "logo512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
```

```
 ],
 "start_url": ".",
 "display": "standalone",
 "theme_color": "#000000",
 "background_color": "#ffffff"
}
```

diploma.project.front/public/robots.txt

```
# https://www.robotstxt.org/robotstxt.html
User-agent: *
```

diploma.project.front/src/components/body/buttons/ButtonCerts.js

```
import React from 'react'
import s from "./ButtonCerts.module.css"
import {NavLink} from "react-router-dom";

const ButtonCerts = () => {
    return (
        <div>
            <NavLink to='/cert'>
                <button className={s.myButton} onClick={some}>
                    GET CERTS
                </button>
            </NavLink>
        </div>
    )
};
let some = () => {
    alert("Certs generated success!!")
};
export default ButtonCerts;
```

diploma.project.front/src/components/body/buttons/ButtonCerts.module.css

```
.myButton {
    background-color:#4db6ac;
    display: table-column;
    border-radius: 37px;
    border:2px solid #b2dfdb;
    cursor:pointer;
    color:#1a237e;
    font-family: Roboto, serif;
    font-size:3vh;
    margin: 10vh;
    width: 20vw;
    height: 15vh;
}
```

diploma.project.front/src/components/body/buttons/ButtonSign.js

```
import React from 'react'
import s from "./ButtonSign.module.css"
import {NavLink} from "react-router-dom";

const ButtonSign = () => {
    return (
        <NavLink to='/file'>
            <button className={s.myButton}>
                SIGN DOCUMENT
```

```
            </button>
        </NavLink>
    )
};

export default ButtonSign;
```

diploma.project.front/src/components/body/buttons/ButtonSign.module.css

```
.myButton {
    background-color:#4db6ac;
    display: table-column;
    border-radius: 37px;
    border:2px solid #b2dfdb;
    cursor:pointer;
    color:#1a237e;
    font-family: Roboto, serif;
    font-size:3vh;
    margin: 10vh;
    width: 20vw;
    height: 15vh;
}
```

diploma.project.front/src/components/body/buttons/SendHttp.js

```
import React from 'react'
import s from "./SendHttp.module.css";
import axios from 'axios';
import WritePass from "../modal/FormPassword";

function download(filename, text) {
    let element = document.createElement('a');
    element.setAttribute('href', 'data:text/plain;charset=utf-8,' + encodeURIComponent(text));
    element.setAttribute('download', filename);

    element.style.display = 'none';
    document.body.appendChild(element);

    element.click();

    document.body.removeChild(element);
};

class ButtonSendRequest extends React.Component {
    constructor(props) {
        super(props);
    }

    handleClick = () => {
        const data = JSON.stringify(this.props);
        console.log(data);
        axios.post("/back/cloud/make/sign", data, {
            headers: {
                'Content-Type': 'application/json;charset=UTF-8'
            }
        })
            .then(res => {
                let response = res.data.sign;
                console.log(response)
                download("test.sig", response);

            })
            .catch((err) => {
                console.log("AXIOS ERROR: ", err);
            })
    };

    render() {
        return (
            <div>
                <button className={s.myButton} onClick={this.handleClick}>
```

```
                SEND
              </button>
              <WritePass/>
            </div>
        );
    }
}

export default ButtonSendRequest;
```

diploma.project.front/src/components/body/buttons/SendHttp.module.css

```
.myButton {
    display: block;
    border-radius: 5px;
    border:2px solid #b2dfdb;
    border-image: none;
    width: 50vw;
    margin-left: auto;
    margin-right: auto;
    height: 5vh;
    text-align: center;
}
```

diploma.project.front/src/components/body/modal/FormPassword.js

```
import React from 'react';
import Button from '@material-ui/core/Button';
import TextField from '@material-ui/core/TextField';
import Dialog from '@material-ui/core/Dialog';
import DialogActions from '@material-ui/core/DialogActions';
import DialogContent from '@material-ui/core/DialogContent';
import DialogContentText from '@material-ui/core/DialogContentText';
import DialogTitle from '@material-ui/core/DialogTitle';

export default function WritePass(props) {
    const [open, setOpen] = React.useState(false);

    const handleClickOpen = () => {
        setOpen(true);
    };

    const handleClose = () => {
        setOpen(false);
    };

    return (
        <div>
            {/*<button onClick={handleClickOpen}>*/}
                {/*Open form dialog*/}
            {/*</button>*/}
            <Dialog open={open} onClose={handleClose}>
                <DialogTitle >Subscribe</DialogTitle>
                <DialogContent>
                    <DialogContentText>
                        To subscribe to this website, please enter your email address here. We will send updates
                        occasionally.
                    </DialogContentText>
                    <TextField
                        autoFocus
                        margin="dense"
                        id="name"
                        label="Email Address"
                        type="email"
                        fullWidth
                    />
                </DialogContent>
                <DialogActions>
                    <Button onClick={handleClose} >
                        Cancel
                    </Button>
```

```
                <Button onClick={handleClose} >
                    Send
                </Button>
            </DialogActions>
        </Dialog>
    </div>
  );
}
```

diploma.project.front/src/components/body/upload/FileZoneUpLoad.js

```
import React from 'react';
import Dropzone from 'react-dropzone'
import classes from './FileZoneUpLoad.module.css'
import ButtonSendRequest from "../buttons/SendHttp";

class FileZoneUpLoad extends React.Component {

    constructor(props) {
        super(props);
        this.state = {
            selectedFiles: null,
            files: []
        }
    }
    onDropHandler(files) {
        let file = files[0];
        const reader = new FileReader();
        reader.readAsDataURL(file);
        reader.onload = () => {
            let base64Data = reader.result;
            let dataToSend = base64Data.slice(base64Data.indexOf(",") + 1);
            this.setState({
                selectedFiles: dataToSend,
                files: files
            });
            // console.log(this.state.selectedFiles);
            console.log();
        };
    }

    render() {
        const files = this.state.files.map(file => (
            <p className={classes.files} key={file.name}>
                {file.name} - {file.size} bytes
            </p>
        ));
        return (
            <div>
                <Dropzone onDrop={acceptedFiles => this.onDropHandler(acceptedFiles)}>
                    {({getRootProps, getInputProps}) => (
                        <section>
                            <div className={classes.content} {...getRootProps()}>
                                <input {...getInputProps()} />
                                <p >Drop some files here, or click to select files</p>
                                <ul>{files}</ul>
                            </div>
                        </section>
                    )}
                </Dropzone>
                <ButtonSendRequest reqData={this.state.selectedFiles}/>
            </div>
        )
    }
}

export default FileZoneUpLoad;
```

diploma.project.front/src/components/body/upload/FileZoneUpLoad.module.css

```
.content {
    background: #e3f2fd;
    border-radius: 5px;
    border: 2px dashed rgb(0, 135, 247);
    border-image: none;
    width: 60vw;
    /*max-width: 500px;*/
    margin-left: auto;
    margin-right: auto;
    margin-bottom: 5vh;
    height: 20vh;
    text-align: center;
    padding-top: 20px;
}
.files {

}
```

diploma.project.front/src/components/body/upload/FilesDemo.js

```
// import React from 'react'
// import Files from 'react-files'
// import axios from 'axios'
// import Blob from 'blob'
// import FormData from 'form-data'
//
// class FileZoneUpLoad extends React.Component {
//     constructor (props) {
//         super(props)
//         this.state = {
//             files: []
//         }
//     }
//
//     onFilesChange = (files) => {
//         this.setState({
//             files
//         }, () => {
//             console.log(this.state.files)
//         })
//     }
//
//     onFilesError = (error, file) => {
//         console.log('error code ' + error.code + ': ' + error.message)
//     }
//
//     filesRemoveOne = (file) => {
//         this.refs.files.removeFile(file)
//     }
//
//     filesRemoveAll = () => {
//         this.refs.files.removeFiles()
//     }
//
//     filesUpload = () => {
//         const formData = new FormData()
//         Object.keys(this.state.files).forEach((key) => {
//             const file = this.state.files[key]
//             formData.append(key, new Blob([file], { type: file.type }), file.name || 'file')
//         })
//
//         axios.post(`/files`, formData)
//             .then(response => window.alert(`${this.state.files.length} files uploaded succesfully!`))
//             .catch(err => window.alert('Error uploading files :('))
//     }
//
//     render () {
//         return (
//             <div>
//                 <h1>Example 1 - List</h1>
//                 <Files
//                     ref='files'
//                     className='files-dropzone-list'
```

```
//                 style={{ height: '100px' }}
//                 onChange={this.onFilesChange}
//                 onError={this.onFilesError}
//                 multiple
//                 maxFiles={10}
//                 maxFileSize={10000000}
//                 minFileSize={0}
//                 clickable
//             >
//                 Drop files here or click to upload
//             </Files>
//             <button onClick={this.filesRemoveAll}>Remove All Files</button>
//             <button onClick={this.filesUpload}>Upload</button>
//             {
//                 this.state.files.length > 0
//                     ? <div className='files-list'>
//                         <ul>{this.state.files.map((file) =>
//                             <li className='files-list-item' key={file.id}>
//                                 <div className='files-list-item-preview'>
//                                     {file.preview.type === 'image'
//                                         ? <img className='files-list-item-preview-image' src={file.preview.url} />
//                                         : <div className='files-list-item-preview-extension'>{file.extension}</div>}
//                                 </div>
//                                 <div className='files-list-item-content'>
//                                     <div className='files-list-item-content-item files-list-item-content-item-1'>{file.name}</div>
//                                     <div className='files-list-item-content-item files-list-item-content-item-2'>{file.sizeReadable}</div>
//                                 </div>
//                                 <div
//                                     id={file.id}
//                                     className='files-list-item-remove'
//                                     onClick={this.filesRemoveOne.bind(this, file)} // eslint-disable-line
//                                 />
//                             </li>
//                         )}</ul>
//                     </div>
//                     : null
//             }
//         </div>
//     )
//   }
// }
// export default FileZoneUpLoad;
```

diploma.project.front/src/components/body/Body.js

```
import React from 'react';
import classes from './Body.module.css'
import ButtonCerts from "./buttons/ButtonCerts";
import ButtonSign from "./buttons/ButtonSign";
import {Link} from "react-router-dom";

const Body = (props) => {
  return (
    <div className={classes.content}>

        <ButtonCerts/>
        <ButtonSign/>
    </div>
  )
};
export default Body
```

diploma.project.front/src/components/body/Body.module.css

```
.content {
  grid-area: c;
  background-color: #e3f2fd;
}
```

diploma.project.front/src/components/footer/Footer.js

```
import React from 'react';
import classes from "./Footer.module.css";
const Footer = (props) => {
    return (
        <div className={classes.footer}>
            @author Koliadin N.A.
            -2021-
        </div>
    )
};
export default Footer;
```

diploma.project.front/src/components/footer/Footer.module.css

```
.footer {
    background-color: #7083c0;
    grid-area: f;
    border-radius: 10px;
    font-family: Roboto, serif;
    font-size: 2vh;
    padding: 1vh;
}
```

diploma.project.front/src/components/header/Header.js

```
import React from 'react';
import s from "./Header.module.css";


const Header = (props) => {
    return (
        <div className={s.header}>
            Cloud-based sign service
        </div>
    )
};

export default Header;
```

diploma.project.front/src/components/header/Header.module.css

```
.header {
    background-color: #64b5f6;
    grid-area: h;
    border-radius: 10px;
    text-align: center;
    font-family: Roboto, serif;
    font-size: 4vh;
    padding: 1.5vh;
}
```

diploma.project.front/src/routs/RoutePath.js

```
import React from 'react'
import {BrowserRouter as Router, Route} from 'react-router-dom'
import App from "../App";
```

```
import FileZoneUpLoad from "../components/body/upload/FileZoneUpLoad";

const routing = (
   <Router>

      <Route exact path="/" component={App}/>
      <Route path="/file" render={() => <FileZoneUpLoad/>}/>
      <Route path="/cert"  component={App}/>
   </Router>
);
export default routing;
```

diploma.project.front/src/App.css

```
.App-wrapper {
  font-family: Roboto, serif;
  display: grid;
  background-color: #e3f2fd;
  text-align: center;
  height: 99vh;
  grid-template-rows: 7vh 1fr 7vh;
  grid-template-columns: 1fr 7fr 1fr;
  grid-template-areas:
    ". h ."
    "c c c"
    ". f .";
}

/*.Content-body {*/
  /*background-color: #8b4c55;*/
/*}*/
/*.App-header {*/
  /*background-color: #282c34;*/
  /*min-height: 100vh;*/
  /*display: flex;*/
  /*flex-direction: column;*/
  /*align-items: center;*/
  /*justify-content: center;*/
  /*font-size: calc(10px + 2vmin);*/
  /*color: white;*/
/*}*/
```

diploma.project.front/src/App.js

```
import React from 'react';
import './App.css';
import Header from "./components/header/Header";
import Footer from "./components/footer/Footer";
import Body from "./components/body/Body";

function App() {
   return (
      <div className='App-wrapper'>
         <Header />
         <Body />
         <Footer />
      </div>
   );
}

export default App;
```

diploma.project.front/src/App.test.js

```
import React from 'react';
import ReactDOM from 'react-dom';
```

```
import App from './App';

it('renders without crashing', () => {
  const div = document.createElement('div');
  ReactDOM.render(<App />, div);
  ReactDOM.unmountComponentAtNode(div);
});
```

diploma.project.front/src/index.css

```
/*body {*/
  /*margin: 0;*/
  /*font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",*/
    /*"Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",*/
    /*sans-serif;*/
  /*-webkit-font-smoothing: antialiased;*/
  /*-moz-osx-font-smoothing: grayscale;*/
/*}*/

/*code {*/
  /*font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",*/
    /*monospace;*/
/*}*/
```

diploma.project.front/src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';

import routing from "./routs/RoutePath";


ReactDOM.render( routing ,document.getElementById('root')
);
// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: https://bit.ly/CRA-PWA
```

diploma.project.front/src/logo.svg

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 170.96 159.93"><defs><style>.cls-1{fill:#09d3ac;}</style></defs><path class="cls-1"
d="M141.35,73.27c0-6.89-8.11-13-20.58-16.73,3-12.67,1.69-22.75-4.28-26.16a9.55,9.55,0,0,0-4.79-1.2c-5.57,0-12.61,3.89-19.72,10.62-7.11-6.68-1
4.13-10.55-19.69-10.55a9.46,9.46,0,0,0-4.86,1.22c-5.94,3.42-7.17,13.44-4.21,26.05-12.41,3.71-20.48,9.77-20.5,16.63s8.12,13,20.59,16.72c-3,12.68-
1.7,22.75,4.28,26.16a9.41,9.41,0,0,0,4.78,1.2c5.58,0,12.62-3.89,19.73-10.62,7.1,6.68,14.12,10.55,19.69,10.55a9.59,9.59,0,0,0,4.86-1.22c5.94-3.42,7.
16-13.44,4.21-26C133.27,86.18,141.34,80.12,141.35,73.27ZM96.56,42.06c8.19-7.33,13.31-8.12,15.13-8.12h0a4.71,4.71,0,0,1,2.42.58c2.86,1.63,4,7.
38,3.09,15a56.73,56.73,0,0,1-1.5,79,97.7,97.7,0,0,0-12.58-2.07,98.47,98.47,0,0,0-8.24-10.08C95.75,42.79,96.15,42.42,96.56,42.06ZM71.77,78.94c.7
8,1.5,1.61,3,2.47,4.51S76,86.49,77,88c-2.71-.39-5.34-.88-7.84-1.46C69.86,84,70.75,81.5,71.77,78.94ZM69,59.9c2.53-.59,5.18-1.08,7.93-1.46-1,1.48
-1.86,3-2.76,4.59s-1.69,3-2.46,4.52Q70.18,63.65,69,59.9Zm5.21,13.34q1.86-3.93,4.09-7.86c1.5-2.62,3.11-5.17,4.77-7.61,2.91-.22,5.91-.34,9-.33s6,.
12,8.89.35c1.66,2.42,3.25,5,4.75,7.55s2.88,5.22,4.12,7.83c-1.23,2.62-2.6,5.25-4.08,7.85s-3.11,5.18-4.77,7.62c-2.91.23-5.91.34-9,.34s-6-.13-8.89-.36
c-1.66-2.42-3.26-4.94-4.76-7.55S75.5,75.85,74.25,73.24Zm32.86-14.77c2.72.38,5.35.87,7.84,1.45-.74,2.47-1.62,5-2.64,7.55-.79-1.5-1.61-3-2.48-4.5
1S108,59.93,107.11,58.47Zm2.76,24.92q1.29-2.27,2.46-4.53c1,2.6,2,5.16,2.7,7.66-2.52.59-5.17,1.07-7.92,1.45Q108.52,85.75,109.87,83.39ZM92,46
.56c1.8,1.92,3.57,4,5.3,6.23-1.71-.07-3.46-.12-5.23-.12s-3.58,0-5.33.12C88.45,50.57,90.2,48.48,92,46.56Zm-22.18-12A4.82,4.82,0,0,1,72.29,34a13.
11,13.11,0,0,1,5.19,1.31,39.07,39.07,0,0,1,10,6.78l1.17,1.07a98.49,98.49,0,0,0-8.16,10,98.88,98.88,0,0,0-12.65,2.06c-.44-1.94-.8-3.84-1-5.67C65.8,4
2,67,36.24,69.81,34.6ZM64.53,85.26a58.75,58.75,0,0,1-5.54-2c-7.1-3-11.5-6.85-11.5-10.14S51.91,66,59,63.05a57.63,57.63,0,0,1,5.44-1.94A97.93,9
7.93,0,0,0,69,73.25,98.72,98.72,0,0,0,64.53,85.26Zm23,19.1c-8.19,7.33-13.31,8.11-15.14,8.11a4.69,4.69,0,0,1-2.42-.58c-2.86-1.63-4-7.38-3.09-15a5
6.07,56.07,0,0,1,1-5.78,99.51,99.51,0,0,0,12.58,2.06,97.17,97.17,0,0,0,8.24,10.08Zm4.57-4.51c-1.8-1.92-3.57-4-5.31-6.23,1.72-08,3.47.12,5.24.12s3.
58,0,5.33-.11C95.63,95.85,93.87,97.93,92.09,99.85Zm22.18,12a4.82,4.82,0,0,1-2.48.59c-1.82,0-7-.8-15.16-8.1l-1.17-1.07a98.44,98.44,0,0,0,8.15-10,
97,97,0,0,0,12.66-2.06c.44,1.94.79,3.84,1,5.67C118.27,104.42,117.12,110.18,114.27,111.81Zm10.8-28.44c-1.71.7-3.52,1.35-5.44,1.93a98.54,98.54,0
,0,0-4.57-12.14,98.1,98.1,0,0,0,4.49-12,58.75,58.75,0,0,1,5.54,2c7.09,3,11.5,6.85,11.49,10.14S132.17,80.42,125.07,83.37ZM92,82.39a9.18,9.18,0,1,
0-9.17-9.19A9.17,9.17,0,0,0,92,82.39ZM31,17.88V128.53H153.07V17.88ZM148.3,123.77H35.78V22.65H148.3Zm-85-33.9c-3,12.68-1.7,22.75,4.2
8,26.16a9.41,9.41,0,0,0,4.78,1.2c5.58,0,12.62-3.89,19.73-10.62,7.1,6.68,14.12,10.55,19.69,10.55a9.59,9.59,0,0,0,4.86-1.22c5.94-3.42,7.16-13.44,4.2
1-26,12.41-3.72,20.48-9.78,20.49-16.63s-8.11-13-20.58-16.73c3-12.67,1.69-22.75-4.28-26.16a9.55,9.55,0,0,0-4.79-1.2c-5.57,0-12.61,3.89-19.72,10.
62-7.11-6.68-14.13-10.55-19.69-10.55a9.46,9.46,0,0,0-4.86,1.22c-5.94,3.42-7.17,13.44-4.21,26.05-12.41,3.71-20.48,9.77-20.5,16.63S50.84,86.13,63.
31,89.87Zm24.21,14.49c-8.19,7.33-13.31,8.11-15.14,8.11a4.69,4.69,0,0,1-2.42-.58c-2.86-1.63-4-7.38-3.09-15a56.07,56.07,0,0,1,1-5.78,99.51,99.51,
0,0,0,12.58,2.06,97.17,97.17,0,0,0,8.24,10.08Zm24.79-36.89c-.79-1.5-1.61-3-2.48-4.51s-1.8-3-2.72-4.49c2.72.38,5.35.87,7.84,1.45C114.21,62.39,11
```

3.33,64.92,112.31,67.47Zm2.72,19c-2.52.59-5.17,1.07-7.92,1.45q1.41-2.22,2.76-4.58t2.46-4.53C113.37,81.46,114.28,84,115,86.52Zm-5.21-13.35c-1.23,2.62-2.6,5.25-4.08,7.85s-3.11,5.18-4.77,7.62c-2.91.23-5.91.34-9,.34s-6-.13-8.89-.36c-1.66-2.42-3.26-4.94-4.76-7.55s-2.87-5.22-4.12-7.83q1.86-3.93,4.09-7.86c1.5-2.62,3.11-5.17,4.77-7.61,2.91-.22,5.91-.34,9-.33s6,.12,8.89.35c1.66,2.42,3.25,5,4.75,7.55S108.58,70.56,109.82,73.17ZM77,88c-2.71-.39-5.34-.88-7.84-1.46.74-2.46,1.63-5,2.65-7.55.78,1.5,1.61,3,2.47,4.51S76,86.49,77,88ZM74.21,63c-.87,1.5-1.69,3-2.46,4.52Q70.18,63.65,69,59.9c2.53-.59,5.18-1.08,7.93-1.46C76,59.92,75.11,61.45,74.21,63ZM92.09,99.85c-1.8-1.92-3.57-4-5.31-6.23,1.72.08,3.47.12,5.24.12s3.58,0,5.33-.11C95.63,95.85,93.87,97.93,92.09,99.85Zm22.18,12a4.82,4.82,0,0,1-2.48.59c-1.82,0-7-.8-15.16-8.1l-1.17-1.07a98.44,98.44,0,0,0,8.15-10,97,97,0,0,0,12.66-2.06c.44,1.94.79,3.84,1,5.67C118.27,104.42,117.12,110.18,114.27,111.81Zm5.28-50.66a58.75,58.75,0,0,1,5.54,2c7.09,3,11.5,6.85,11.49,10.14s-4.41,7.16-11.51,10.11c-1.71.7-3.52,1.35-5.44,1.93a98.54,98.54,0,0,0-4.57-12.14A98.1,98.1,0,0,0,119.55,61.15Zm-23-19.09c8.19-7.33,13.31-8.12,15.13-8.12h0a4.71,4.71,0,0,1,2.42.58c2.86,1.63,4,7.38,3.09,15a56.73,56.73,0,0,1-1,5.79,97.7,97.7,0,0,0-12.58-2.07,98.47,98.47,0,0,0-8.24-10.08C95.75,42.79,96.15,42.42,96.56,42.06ZM92,46.56c1.8,1.92,3.57,4,5.3,6.23-1.71-.07-3.46-.12-5.23-.12s-3.58,0-5.33.12C88.45,50.57,90.2,48.48,92,46.56Zm-22.18-12A4.82,4.82,0,0,1,72.29,34a13.11,13.11,0,0,1,5.19,1.31,39.07,39.07,0,0,1,10,6.78l1.17,1.07a98.49,98.49,0,0,0-8.16,10,98.88,98.88,0,0,0-12.65,2.06c-.44-1.94-.8-3.84-1-5.67C65.8,42,67,36.24,69.81,34.6ZM59,63.05a57.63,57.63,0,0,1,5.44-1.94A97.93,97.93,0,0,0,69,73.25a98.72,98.72,0,0,0-4.49,12,58.75,58.75,0,0,1-5.54-2c-7.1-3-11.5-6.85-11.5-10.14S51.91,66,59,63.05Zm33,1a9.18,9.18,0,1,0,9.17,9.19A9.17,9.17,0,0,0,92,64Zm0,0a9.18,9.18,0,1,0,9.17,9.19A9.17,9.17,0,0,0,92,64Zm0,0a9.18,9.18,0,1,0,9.17,9.19A9.17,9.17,0,0,0,92,64Zm49.35,9.24c0-6.89-8.11-13-20.58-16.73,3-12.67,1.69-22.75-4.28-26.16a9.55,9.55,0,0,0-4.79-1.2c-5.57,0-12.61,3.89-19.72,10.62-7.11-6.68-14.13-10.55-19.69-10.55a9.46,9.46,0,0,0-4.86,1.22c-5.94,3.42-7.17,13.44-4.21,26.05-12.41,3.71-20.48,9.77-20.5,16.63s8.12,13,20.59,16.72c-3,12.68-1.7,22.75,4.28,26.16a9.41,9.41,0,0,0,4.78,1.2c5.58,0,12.62-3.89,19.73-10.62,7.1,6.68,14.12,10.55,19.69,10.55a9.59,9.59,0,0,0,4.86-1.22c5.94-3.42,7.16-13.44,4.21-26C133.27,86.18,141.34,80.12,141.35,73.27ZM96.56,42.06c8.19-7.33,13.31-8.12,15.13-8.12h0a4.71,4.71,0,0,1,2.42.58c2.86,1.63,4,7.38,3.09,15a56.73,56.73,0,0,1-1,5.79,97.7,97.7,0,0,0-12.58-2.07,98.47,98.47,0,0,0-8.24-10.08C95.75,42.79,96.15,42.42,96.56,42.06ZM71.77,78.94c.78,1.5,1.61,3,2.47,4.51S76,86.49,77,88c-2.71-.39-5.34-.88-7.84-1.46C69.86,84,70.75,81.5,71.77,78.94ZM69,59.9c2.53-.59,5.18-1.08,7.93-1.46-1,1.48-1.86,3-2.76,4.59s-1.69,3-2.46,4.52Q70.18,63.65,69,59.9Zm5.21,13.34q1.86-3.93,4.09-7.86c1.5-2.62,3.11-5.17,4.77-7.61,2.91-.22,5.91-.34,9-.33s6,.12,8.89.35c1.66,2.42,3.25,5,4.75,7.55s2.88,5.22,4.12,7.83c-1.23,2.62-2.6,5.25-4.08,7.85s-3.11,5.18-4.77,7.62c-2.91.23-5.91.34-9,.34s-6-.13-8.89-.36c-1.66-2.42-3.26-4.94-4.76-7.55S75.5,75.85,74.25,73.24Zm32.86-14.77c2.72.38,5.35.87,7.84,1.45-.74,2.47-1.62,5-2.64,7.55-.79-1.5-1.61-3-2.48-4.51S108,59.93,107.11,58.47Zm2.76,24.92q1.29-2.27,2.46-4.53c1,2.6,2,5.16,2.7,7.66-2.52.59-5.17,1.07-7.92,1.45Q108.52,85.75,109.87,83.39ZM92,46.56c1.8,1.92,3.57,4,5.3,6.23-1.71-.07-3.46-.12-5.23-.12s-3.58,0-5.33.12C88.45,50.57,90.2,48.48,92,46.56Zm-22.18-12A4.82,4.82,0,0,1,72.29,34a13.11,13.11,0,0,1,5.19,1.31,39.07,39.07,0,0,1,10,6.78l1.17,1.07a98.49,98.49,0,0,0-8.16,10,98.88,98.88,0,0,0-12.65,2.06c-.44-1.94-.8-3.84-1-5.67C65.8,42,67,36.24,69.81,34.6ZM64.53,85.26a58.75,58.75,0,0,1,5.54-2c7.09,3,11.5,6.85,11.49,10.14s-4.41,7.16-11.51,10.11c-1.71.7-3.52,1.35-5.44,1.93a98.54,98.54,0,0,0-4.57-12.14,98.1,98.1,0,0,0,4.49-12,58.75,58.75,0,0,1,5.54,2c7.09,3,11.5,6.85,11.49,10.14S132.17,80.42,125.07,83.37ZM92,82.39a9.18,9.18,0,1,0-9.17-9.19A9.17,9.17,0,0,0,92,82.39Zm-69.32,54.5V26.2L17.89,31V141.66H139.94l4.78-4.77Z"/></svg>

diploma.project.front/src/serviceWorker.js

```
// This optional code is used to register a service worker.
// register() is not called by default.

// This lets the app load faster on subsequent visits in production, and gives
// it offline capabilities. However, it also means that developers (and users)
// will only see deployed updates on subsequent visits to a page, after all the
// existing tabs open on the page have been closed, since previously cached
// resources are updated in the background.

// To learn more about the benefits of this model and instructions on how to
// opt-in, read https://bit.ly/CRA-PWA

const isLocalhost = Boolean(
  window.location.hostname === 'localhost' ||
    // [::1] is the IPv6 localhost address.
    window.location.hostname === '[::1]' ||
    // 127.0.0.1/8 is considered localhost for IPv4.
    window.location.hostname.match(
      /^127(?:\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)){3}$/
    )
);

export function register(config) {
  if (process.env.NODE_ENV === 'production' && 'serviceWorker' in navigator) {
    // The URL constructor is available in all browsers that support SW.
    const publicUrl = new URL(process.env.PUBLIC_URL, window.location.href);
    if (publicUrl.origin !== window.location.origin) {
      // Our service worker won't work if PUBLIC_URL is on a different origin
      // from what our page is served on. This might happen if a CDN is used to
      // serve assets; see https://github.com/facebook/create-react-app/issues/2374
      return;
    }

    window.addEventListener('load', () => {
      const swUrl = `${process.env.PUBLIC_URL}/service-worker.js`;

      if (isLocalhost) {
        // This is running on localhost. Let's check if a service worker still exists or not.
        checkValidServiceWorker(swUrl, config);
```

```
      // Add some additional logging to localhost, pointing developers to the
      // service worker/PWA documentation.
      navigator.serviceWorker.ready.then(() => {
        console.log(
          'This web app is being served cache-first by a service ' +
          'worker. To learn more, visit https://bit.ly/CRA-PWA'
        );
      });
    } else {
      // Is not localhost. Just register service worker
      registerValidSW(swUrl, config);
    }
  });
 }
}

function registerValidSW(swUrl, config) {
 navigator.serviceWorker
   .register(swUrl)
   .then(registration => {
    registration.onupdatefound = () => {
      const installingWorker = registration.installing;
      if (installingWorker == null) {
        return;
      }
      installingWorker.onstatechange = () => {
        if (installingWorker.state === 'installed') {
          if (navigator.serviceWorker.controller) {
            // At this point, the updated precached content has been fetched,
            // but the previous service worker will still serve the older
            // content until all client tabs are closed.
            console.log(
              'New content is available and will be used when all ' +
                'tabs for this page are closed. See https://bit.ly/CRA-PWA.'
            );

            // Execute callback
            if (config && config.onUpdate) {
              config.onUpdate(registration);
            }
          } else {
            // At this point, everything has been precached.
            // It's the perfect time to display аывыiiss
            // "Content is cached for offline use." message.
            console.log('Content is cached for offline use.');

            // Execute callback
            if (config && config.onSuccess) {
              config.onSuccess(registration);
            }
          }
        }
      };
    };
   })
   .catch(error => {
    console.error('Error during service worker registration:', error);
   });
}

function checkValidServiceWorker(swUrl, config) {
 // Check if the service worker can be found. If it can't reload the page.
 fetch(swUrl)
   .then(response => {
    // Ensure service worker exists, and that we really are getting a JS file.
    const contentType = response.headers.get('content-type');
    if (
      response.status === 404 ||
      (contentType != null && contentType.indexOf('javascript') === -1)
    ) {
      // No service worker found. Probably a different app. Reload the page.
      navigator.serviceWorker.ready.then(registration => {
        registration.unregister().then(() => {
          window.location.reload();
        });
      });
    } else {
      // Service worker found. Proceed as normal.
      registerValidSW(swUrl, config);
    }
   })
```

```
    .catch(() => {
      console.log(
        'No internet connection found. App is running in offline mode.'
      );
    });
}

export function unregister() {
  if ('serviceWorker' in navigator) {
    navigator.serviceWorker.ready.then(registration => {
      registration.unregister();
    });
  }
}
```

diploma.project.front/.gitignore

```
# See https://help.github.com/articles/ignoring-files/ for more about ignoring files.

# dependencies
/node_modules
/.pnp
.pnp.js

/.idea

# testing
/coverage

# production
/build

# misc
.DS_Store
.env.local
.env.development.local
.env.test.local
.env.production.local

npm-debug.log*
yarn-debug.log*
yarn-error.log*
```

diploma.project.front/README.md

```
This project was bootstrapped with [Create React App](https://github.com/facebook/create-react-app).

## Available Scripts

In the project directory, you can run:

### `npm start`

Runs the app in the development mode.<br />
Open [http://localhost:3000](http://localhost:3000) to view it in the browser.

The page will reload if you make edits.<br />
You will also see any lint errors in the console.

### `npm test`

Launches the test runner in the interactive watch mode.<br />
See the section about [running tests](https://facebook.github.io/create-react-app/docs/running-tests) for more information.

### `npm run build`

Builds the app for production to the `build` folder.<br />
It correctly bundles React in production mode and optimizes the build for the best performance.

The build is minified and the filenames include the hashes.<br />
Your app is ready to be deployed!
```

See the section about [deployment](https://facebook.github.io/create-react-app/docs/deployment) for more information.

`npm run eject`

Note: this is a one-way operation. Once you `eject`, you can’t go back!

If you aren’t satisfied with the build tool and configuration choices, you can `eject` at any time. This command will remove the single build dependency from your project.

Instead, it will copy all the configuration files and the transitive dependencies (Webpack, Babel, ESLint, etc) right into your project so you have full control over them. All of the commands except `eject` will still work, but they will point to the copied scripts so you can tweak them. At this point you’re on your own.

You don’t have to ever use `eject`. The curated feature set is suitable for small and middle deployments, and you shouldn’t feel obligated to use this feature. However we understand that this tool wouldn’t be useful if you couldn’t customize it when you are ready for it.

Learn More

You can learn more in the [Create React App documentation](https://facebook.github.io/create-react-app/docs/getting-started).

To learn React, check out the [React documentation](https://reactjs.org/).

Code Splitting

This section has moved here: https://facebook.github.io/create-react-app/docs/code-splitting

Analyzing the Bundle Size

This section has moved here: https://facebook.github.io/create-react-app/docs/analyzing-the-bundle-size

Making a Progressive Web App

This section has moved here: https://facebook.github.io/create-react-app/docs/making-a-progressive-web-app

Advanced Configuration

This section has moved here: https://facebook.github.io/create-react-app/docs/advanced-configuration

Deployment

This section has moved here: https://facebook.github.io/create-react-app/docs/deployment

`npm run build` fails to minify

This section has moved here: https://facebook.github.io/create-react-app/docs/troubleshooting#npm-run-build-fails-to-minify

diploma.project.front/package.json

```
{
 "name": "diploma.project.front",
 "version": "1.0.0",
 "private": true,
 "dependencies": {
  "@material-ui/core": "^4.6.1",
  "@material-ui/icons": "^4.5.1",
  "axios": "^0.21.1",
  "blob": "0.0.5",
  "form-data": "latest",
  "react": "^16.11.0",
  "react-dom": "^16.11.0",
  "react-dropzone": "^10.2.0",
  "react-files": "^2.4.8",
  "react-native": "^0.64.2",
  "react-native-elements": "^1.2.7",
  "react-native-vector-icons": "^6.6.0",
  "react-router-dom": "^5.1.2",
  "react-scripts": "3.2.0",
  "typeface-roboto": "0.0.75"
 },
 "scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
 },
```

```
  "eslintConfig": {
    "extends": "react-app"
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

ДОДАТОК Б. Програмна реалізація серверної частини

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/binder/ApplicationBinder.java

```
package com.qthegamep.diploma.project.back.binder;

import com.google.gson.Gson;
import com.qthegamep.diploma.project.back.service.*;
import org.glassfish.jersey.internal.inject.AbstractBinder;

import javax.inject.Singleton;

public class ApplicationBinder extends AbstractBinder {

    private SignService signService;
    private SenderHttpService senderHttpService;
    private Gson gson;
    private ApiBackOperationService apiBackOperationService;
    private GenerationService generationService;

            private ApplicationBinder(SignService signService, SenderHttpService senderHttpService, Gson gson, ApiBackOperationService apiBackOperationService, GenerationService generationService) {
        this.signService = signService;
        this.senderHttpService = senderHttpService;
        this.gson = gson;
        this.apiBackOperationService =apiBackOperationService;
        this.generationService = generationService;
    }

    public GenerationService getGenerationService() {
        return generationService;
    }
    public SignService getSignService() {
        return signService;
    }

    public SenderHttpService getSenderHttpService() {
        return senderHttpService;
    }

    public Gson getGsonObj() {
        return gson;
    }

    public ApiBackOperationService getApiBackOperationService(){
        return apiBackOperationService;
    }

    public static ApplicationBinderBuilder builder() {
        return new ApplicationBinderBuilder();
    }

    @Override
    protected void configure() {
        bindGenerationService();
        bindSenderService();
        bindGson();
        bindApiBachService();
        bindSignService();
    }
    private void bindGenerationService() {
        if (generationService == null) {
            bind(GenerationServiceImpl.class).to(GenerationService.class).in(Singleton.class);
        } else {
            bind(generationService).to(GenerationService.class).in(Singleton.class);
        }
    }
    private void bindGson() {
        if (gson == null) {
            gson = new Gson();
            bind(gson).to(Gson.class).in(Singleton.class);
```

```
        }
    }

    private void bindSignService() {
        if (signService == null) {
            bind(SignServiceImpl.class).to(SignService.class).in(Singleton.class);
        } else {
            bind(signService).to(SignService.class).in(Singleton.class);
        }
    }
    private void bindApiBachService() {
        if (apiBackOperationService == null) {
            bind(ApiBackOperationServiceImpl.class).to(ApiBackOperationService.class).in(Singleton.class);
        } else {
            bind(signService).to(ApiBackOperationService.class).in(Singleton.class);
        }
    }

    private void bindSenderService() {
        if (senderHttpService == null) {
            bind(SenderHttpServiceImpl.class).to(SenderHttpService.class).in(Singleton.class);
        } else {
            bind(senderHttpService).to(SenderHttpService.class).in(Singleton.class);
        }
    }

    public static class ApplicationBinderBuilder {
        private GenerationService generationService;
        private SignService signService;
        private SenderHttpService senderHttpService;
        private Gson gson;
        private ApiBackOperationService apiBackOperationService;

        public ApplicationBinderBuilder setSignService(SignService signService) {
            this.signService = signService;
            return this;
        }
        public ApplicationBinderBuilder setGenService(GenerationService generationService) {
            this.generationService = generationService;
            return this;
        }

        public ApplicationBinderBuilder setSenderHttpService(SenderHttpService senderHttpService) {
            this.senderHttpService = senderHttpService;
            return this;
        }
        public ApplicationBinderBuilder setGson(Gson gson) {
            this.gson = gson;
            return this;
        }
        public ApplicationBinderBuilder setApiBackService(ApiBackOperationService apiBackOperationService) {
            this.apiBackOperationService = apiBackOperationService;
            return this;
        }

        public ApplicationBinder build() {
            return new ApplicationBinder(signService, senderHttpService, gson, apiBackOperationService, generationService);
        }
    }
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/controllers/CertGenerateController.java

```
package com.qthegamep.diploma.project.back.controllers;

import com.qthegamep.diploma.project.back.dto.GenerationRequestDTO;
import com.qthegamep.diploma.project.back.dto.GenerationResponseDTO;
import com.qthegamep.diploma.project.back.service.GenerationService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import javax.inject.Inject;
import javax.ws.rs.Consumes;
```

```
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.HttpHeaders;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

@Path("/generate")
public class CertGenerateController {
    private static final Logger LOG = LoggerFactory.getLogger(SignController.class.getName());
    private GenerationService generationService;

    @Inject
    CertGenerateController(GenerationService generationService) {
        this.generationService = generationService;
    }

    @Context
    private HttpHeaders httpHeaders;

    @POST
    @Path("/cert")
    @Consumes({MediaType.APPLICATION_JSON})
    @Produces({MediaType.APPLICATION_JSON})
    public Response generateCert(GenerationRequestDTO requestDTO) {
        String requestId = httpHeaders.getHeaderString("requestId");
        LOG.info("Start generate cert. RequestId: {}", requestId);
        GenerationResponseDTO responseDTO = generationService.generate(requestDTO, requestId);
        return Response.status(javax.ws.rs.core.Response.Status.OK)
                .entity(responseDTO)
                .build();
    }
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/controllers/SignController.java

```
package com.qthegamep.diploma.project.back.controllers;

import com.qthegamep.diploma.project.back.dto.SignRequestDTO;
import com.qthegamep.diploma.project.back.dto.SignResponseDTO;
import com.qthegamep.diploma.project.back.service.SignService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import javax.inject.Inject;
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.HttpHeaders;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import java.time.LocalDateTime;

@Path("/make")
public class SignController {
    private static final Logger LOG = LoggerFactory.getLogger(SignController.class.getName());
    private SignService signService;

    @Inject
    public SignController(SignService signService) {
        this.signService = signService;
    }

    @Context
    private HttpHeaders httpHeaders;

    @POST
    @Path("/sign")
    @Consumes({MediaType.APPLICATION_JSON})
    @Produces({MediaType.APPLICATION_JSON})
```

```
    public Response catchRequestToSign(SignRequestDTO signRequestDTO) {
        LOG.info("Time getting request: {}", LocalDateTime.now());
        LOG.info("Request: {}", signRequestDTO.getData());
        String requestId = httpHeaders.getHeaderString("requestId");
        SignResponseDTO responseDTO = signService.sign(signRequestDTO, requestId);
        return Response.status(Response.Status.OK)
                .entity(responseDTO)
                .build();
    }
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/crypto/CryptoOperation.java

```
package com.qthegamep.diploma.project.back.crypto;

import com.iit.certificateAuthority.endUser.libraries.signJava.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;


public class CryptoOperation {
    private static final Logger LOG = LoggerFactory.getLogger(CryptoOperation.class);
            private static final String CRYPTO_END_USER_CHARSET = System.getProperty("crypto.enduser.charset",
StandardCharsets.UTF_8.displayName());
    private static final boolean CRYPTO_END_USER_UI_MODE = Boolean.parseBoolean(System.getProperty("crypto.enduser.ui.mode", "false"));
            private static final String CRYPTO_END_USER_LIBRARY_PATH = System.getProperty("crypto.enduser.library.path", "/opt/iit/
diploma.project.back/EUSign-x64-1.3.148");
                        private static final boolean CRYPTO_END_USER_OFFLINE_MODE =
Boolean.parseBoolean(System.getProperty("crypto.enduser.offline.mode", "false"));
        private static final boolean CRYPTO_END_USER_USE_PROXY = Boolean.parseBoolean(System.getProperty("crypto.enduser.use.proxy",
"false"));
                    private static final boolean CRYPTO_END_USER_PROXY_ANONYMOUS =
Boolean.parseBoolean(System.getProperty("crypto.enduser.proxy.anonymous", "true"));
    private static final String CRYPTO_END_USER_PROXY_ADDRESS = System.getProperty("crypto.enduser.proxy.address", "");
    private static final String CRYPTO_END_USER_PROXY_PORT = System.getProperty("crypto.enduser.proxy.port", "");
    private static final String CRYPTO_END_USER_PROXY_USER = System.getProperty("crypto.enduser.proxy.user", "");
    private static final String CRYPTO_END_USER_PROXY_PASSWORD = System.getProperty("crypto.enduser.proxy.password", "");
                private static final boolean CRYPTO_END_USER_PROXY_SAVE_PASSWORD =
Boolean.parseBoolean(System.getProperty("crypto.enduser.proxy.save.password", "false"));
    private static final String CRYPTO_END_USER_FILE_STORE_PATH = System.getProperty("crypto.enduser.file.store.path", "/opt/certs/");
                  private static final boolean CRYPTO_END_USER_SAVE_LOADED_CERTS =
Boolean.parseBoolean(System.getProperty("crypto.enduser.save.loaded.certs", "true"));
    private static final boolean CRYPTO_END_USER_CMP_USE = Boolean.parseBoolean(System.getProperty("crypto.enduser.cmp.use", "true"));
            private static final String CRYPTO_END_USER_CMP_ADDRESS = System.getProperty("crypto.enduser.cmp.address", "http://
acsk.privatbank.ua/services/cmp/");
    private static final String CRYPTO_END_USER_CMP_PORT = System.getProperty("crypto.enduser.cmp.port", "80");
    private static final String CRYPTO_END_USER_CMP_COMMON_NAME = System.getProperty("crypto.enduser.cmp.common.name", "");
                    private static final boolean CRYPTO_END_USER_TSP_GET_STAMPS =
Boolean.parseBoolean(System.getProperty("crypto.enduser.tsp.get.stamps", "true"));
     private static final String CRYPTO_END_USER_TSP_ADDRESS = System.getProperty("crypto.enduser.tsp.address", "http://acsk.privatbank.ua/
services/tsp/");
    private static final String CRYPTO_END_USER_TSP_PORT = System.getProperty("crypto.enduser.tsp.port", "80");
    private static final boolean CRYPTO_END_USER_OCSP_USE = Boolean.parseBoolean(System.getProperty("crypto.enduser.ocsp.use", "true"));
            private static final String CRYPTO_END_USER_OCSP_ADDRESS = System.getProperty("crypto.enduser.ocsp.address", "http://
acsk.privatbank.ua/services/ocsp/");
    private static final String CRYPTO_END_USER_OCSP_PORT = System.getProperty("crypto.enduser.ocsp.port", "80");
    private static EndUser endUser;

    public static EndUser getEndUser() {
        return endUser;
    }

    static {
        try {
            initEndUser();
            initEndUserMode();
            initEndUserProxy();
            initFileStore();
```

```
            initCMP();
            initTSP();
            initOCSP();
            initLDAP();
            String libraryPath = EndUserResourceExtractor.GetInstallPath();
            LOG.info("EndUser library path: {}", libraryPath);
            String storePath = endUser.GetFileStoreSettings().GetPath();
            LOG.info("EndUser store path: {}", storePath);
        } catch (Exception e) {
            LOG.error("[EUSign] Initialization ERROR:", e);
        }
    }

    private CryptoOperation() {
    }

    private static void initEndUser() throws Exception {
        endUser = new EndUser();
        endUser.SetUIMode(CRYPTO_END_USER_UI_MODE);
        endUser.SetLanguage(EndUser.EU_RU_LANG);
        endUser.SetCharset(CRYPTO_END_USER_CHARSET);
        LOG.info("EndUser path: {}", CRYPTO_END_USER_LIBRARY_PATH);
        Path path = Paths.get(CRYPTO_END_USER_LIBRARY_PATH);
        if (!Files.exists(Paths.get(path.toString()))){
            Files.createDirectories(path);
        }
        EndUserResourceExtractor.SetPath(CRYPTO_END_USER_LIBRARY_PATH);
        EndUserResourceExtractor.ExtractResources();
        endUser.Initialize();
        LOG.info("EndUser initialized: {}", endUser.IsInitialized());
    }

    private static void initEndUserMode() throws Exception {
        LOG.info("EndUser offline mode: {}", CRYPTO_END_USER_OFFLINE_MODE);
        EndUserModeSettings modeSettings = new EndUserModeSettings(CRYPTO_END_USER_OFFLINE_MODE);
        modeSettings.SetOfflineMode(CRYPTO_END_USER_OFFLINE_MODE);
        endUser.SetModeSettings(modeSettings);
    }

    private static void initEndUserProxy() throws Exception {
        EndUserProxySettings proxySettings = endUser.CreateProxySettings();
        proxySettings.SetUseProxy(CRYPTO_END_USER_USE_PROXY);
        proxySettings.SetAnonymous(CRYPTO_END_USER_PROXY_ANONYMOUS);
        proxySettings.SetAddress(CRYPTO_END_USER_PROXY_ADDRESS);
        proxySettings.SetPort(CRYPTO_END_USER_PROXY_PORT);
        proxySettings.SetUser(CRYPTO_END_USER_PROXY_USER);
        proxySettings.SetPassword(CRYPTO_END_USER_PROXY_PASSWORD);
        proxySettings.SetSavePassword(CRYPTO_END_USER_PROXY_SAVE_PASSWORD);
        endUser.SetProxySettings(proxySettings);
        LOG.info("EndUser proxy was initialized");
    }

    private static void initFileStore() throws Exception {
        EndUserFileStoreSettings fileStoreSettings = endUser.CreateFileStoreSettings();
        Path path = Paths.get(CRYPTO_END_USER_FILE_STORE_PATH);
        if (!Files.exists(Paths.get(path.toString()))){
            Files.createDirectories(path);
        }
        fileStoreSettings.SetPath(CRYPTO_END_USER_FILE_STORE_PATH);
        fileStoreSettings.SetSaveLoadedCerts(CRYPTO_END_USER_SAVE_LOADED_CERTS);
        endUser.SetFileStoreSettings(fileStoreSettings);
        LOG.info("EndUser file store was initialized");
    }

    private static void initCMP() throws Exception {
        EndUserCMPSettings cmpSettings = endUser.CreateCMPSettings();
        cmpSettings.SetUseCMP(CRYPTO_END_USER_CMP_USE);
        cmpSettings.SetAddress(CRYPTO_END_USER_CMP_ADDRESS);
        cmpSettings.SetPort(CRYPTO_END_USER_CMP_PORT);
        cmpSettings.SetCommonName(CRYPTO_END_USER_CMP_COMMON_NAME);
        endUser.SetCMPSettings(cmpSettings);
        LOG.info("EndUser CMP was initialized. CMP address: {}", CRYPTO_END_USER_CMP_ADDRESS);
    }

    private static void initTSP() throws Exception {
        EndUserTSPSettings tspSettings = endUser.CreateTSPSettings();
        tspSettings.SetGetStamps(CRYPTO_END_USER_TSP_GET_STAMPS);
        tspSettings.SetAddress(CRYPTO_END_USER_TSP_ADDRESS);
        tspSettings.SetPort(CRYPTO_END_USER_TSP_PORT);
        endUser.SetTSPSettings(tspSettings);
        LOG.info("EndUser TSP was initialized. TSP address: {}", CRYPTO_END_USER_TSP_ADDRESS);
```

```
    }

    private static void initOCSP() throws Exception {
        EndUserOCSPSettings ocspSettings = endUser.CreateOCSPSettings();
        ocspSettings.SetUseOCSP(CRYPTO_END_USER_OCSP_USE);
        ocspSettings.SetAddress(CRYPTO_END_USER_OCSP_ADDRESS);
        ocspSettings.SetPort(CRYPTO_END_USER_OCSP_PORT);
        endUser.SetOCSPSettings(ocspSettings);
        LOG.info("EndUser OCSP was initialized. OCSP address: {}", CRYPTO_END_USER_OCSP_ADDRESS);
    }

    private static void initLDAP() throws Exception {
        EndUserLDAPSettings lDAPSettings = endUser.CreateLDAPSettings();
        endUser.SetLDAPSettings(lDAPSettings);
    }
    public static EndUserSession genSessionEnc(byte[] cert) {
        EndUserSession clientSession = null;
        try {
            clientSession = endUser.ClientDynamicKeySessionCreate(900, cert);
        } catch (EndUserException e) {
            LOG.error("Error: {} : {}", e, e.GetErrorCode());
        } catch (Exception e) {
            LOG.error("Error Exception", e);
        }
        return clientSession;
    }

    public static byte[] decrResp(String encData, EndUserSession session) throws Exception {
        return endUser.SessionDecrypt(session, encData);
    }

    public static EndUserSignInfo getDataFromSign(String signData) throws Exception {
        return endUser.VerifyInternal(signData);
    }
    public static String sessionEnc(EndUserSession session, String data) throws Exception {
        return endUser.SessionEncrypt(session, data);
    }
    public static String getHashData(byte[] data) throws Exception {
        EndUserContext context = endUser.CtxCreate();
        EndUserHashContext hashContext = endUser.CtxHashBegin(context, EndUser.EU_CTX_HASH_ALGO_GOST34311, null);
        endUser.CtxHashContinue(hashContext, data);
        String hash = endUser.CtxHashEnd(hashContext);
        LOG.warn("[EUSign] Hash: {}", hash);
        return hash;
    }
    public static String appendSign(byte[] data, String sign) throws Exception {
        return endUser.AppendSign(data, sign);
    }

}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/dto/EncryptedRequestDTO.java

```
package com.qthegamep.diploma.project.back.dto;

import java.util.Objects;

public class EncryptedRequestDTO {

    private String encryptedData;
    private String authData;

    @Override
    public String toString() {
        return "EncryptedRequestDTO{" +
                "encryptedData='" + encryptedData + '\'' +
                ", authData='" + authData + '\'' +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
```

```
        if (o == null || getClass() != o.getClass()) return false;
        EncryptedRequestDTO that = (EncryptedRequestDTO) o;
        return Objects.equals(encryptedData, that.encryptedData) &&
                Objects.equals(authData, that.authData);
    }

    @Override
    public int hashCode() {
        return Objects.hash(encryptedData, authData);
    }

    public String getEncryptedData() {
        return encryptedData;
    }

    public void setEncryptedData(String encryptedData) {
        this.encryptedData = encryptedData;
    }

    public String getAuthData() {
        return authData;
    }

    public void setAuthData(String authData) {
        this.authData = authData;
    }

}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/dto/EncryptedResponseDTO.java

```
package com.qthegamep.diploma.project.back.dto;

import java.util.Objects;

public class EncryptedResponseDTO {
    private String encryptedData;

    public String getEncryptedData() {
        return encryptedData;
    }

    public void setEncryptedData(String encryptedData) {
        this.encryptedData = encryptedData;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        EncryptedResponseDTO that = (EncryptedResponseDTO) o;
        return Objects.equals(encryptedData, that.encryptedData);
    }

    @Override
    public int hashCode() {
        return Objects.hash(encryptedData);
    }

    @Override
    public String toString() {
        return "EncryptedResponseDTO{" +
                "encryptedData='" + encryptedData + '\'' +
                '}';
    }
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/dto/GenerationRequestDTO.java

```
package com.qthegamep.diploma.project.back.dto;

import java.util.Objects;

public class GenerationRequestDTO {
    private String alias;
    private String pass;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        GenerationRequestDTO that = (GenerationRequestDTO) o;
        return Objects.equals(alias, that.alias) &&
                Objects.equals(pass, that.pass);
    }

    @Override
    public int hashCode() {
        return Objects.hash(alias, pass);
    }

    @Override
    public String toString() {
        return "GenerationRequestDTO{" +
                "alias='" + alias + '\'' +
                ", pass='" + pass + '\'' +
                '}';
    }

    public String getAlias() {
        return alias;
    }

    public void setAlias(String alias) {
        this.alias = alias;
    }

    public String getPass() {
        return pass;
    }

    public void setPass(String pass) {
        this.pass = pass;
    }
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/dto/GenerationResponseDTO.java

```
package com.qthegamep.diploma.project.back.dto;

public class GenerationResponseDTO {
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/dto/SignRequestDTO.java

```
package com.qthegamep.diploma.project.back.dto;

import com.fasterxml.jackson.annotation.JsonProperty;

import java.util.Objects;

public class SignRequestDTO {
```

```
    @JsonProperty(value = "reqData")
    private String data;

    public String getData() {
        return data;
    }

    public void setData(String data) {
        this.data = data;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        SignRequestDTO that = (SignRequestDTO) o;
        return data.equals(that.data);
    }

    @Override
    public int hashCode() {
        return Objects.hash(data);
    }

    @Override
    public String toString() {
        return "SignRequestDTO{" +
                "data='" + data + '\'' +
                '}';
    }
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/dto/SignResponseDTO.java

```
package com.qthegamep.diploma.project.back.dto;

import java.util.Objects;

public class SignResponseDTO {

    @Override
    public String toString() {
        return "SignResponseDTO{" +
                "sign='" + sign + '\'' +
                '}';
    }

    public String getSign() {
        return sign;
    }

    public void setSign(String sign) {
        this.sign = sign;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        SignResponseDTO that = (SignResponseDTO) o;
        return sign.equals(that.sign);
    }

    @Override
    public int hashCode() {
        return Objects.hash(sign);
    }

    private String sign;

}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/dto/SignedDataDTO.java

```
package com.qthegamep.diploma.project.back.dto;

import java.util.Objects;

public class SignedDataDTO {

    private String signedData;

    public String getSignedData() {
        return signedData;
    }

    public void setSignedData(String signedData) {
        this.signedData = signedData;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        SignedDataDTO that = (SignedDataDTO) o;
        return Objects.equals(signedData, that.signedData);
    }

    @Override
    public int hashCode() {
        return Objects.hash(signedData);
    }

    @Override
    public String toString() {
        return "SignedDataDTO{" +
                "signedData='" + signedData + '\'' +
                '}';
    }
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/service/ApiBackOperationService.java

```
package com.qthegamep.diploma.project.back.service;

import com.iit.certificateAuthority.endUser.libraries.signJava.EndUserSession;
import com.qthegamep.diploma.project.back.dto.EncryptedRequestDTO;
import org.json.JSONException;

import java.util.List;

public interface ApiBackOperationService {
    List<EndUserSession> getSessions(String requestId) throws Exception;

    String getOperationId(EndUserSession sessionCloud, String requestId) throws Exception;

    EncryptedRequestDTO createEncryptedRequest(EndUserSession session, String dataToEnc);

    String createEncReqWithEndUserSession(String data, EndUserSession clientSession) throws Exception;

    String getDataFromSign(String signData, String value) throws Exception;

    String createJsonWithTechSign( byte[] data) throws JSONException;

    String generateKeys(EndUserSession sessionHsm, String requestId) throws Exception;
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/service/ApiBackOperationServiceImpl.java

```
package com.qthegamep.diploma.project.back.service;

import com.google.gson.Gson;
import com.iit.certificateAuthority.endUser.libraries.signJava.EndUserSession;
import com.iit.certificateAuthority.endUser.libraries.signJava.EndUserSignInfo;
import com.qthegamep.diploma.project.back.crypto.CryptoOperation;
import com.qthegamep.diploma.project.back.dto.EncryptedRequestDTO;
import com.qthegamep.diploma.project.back.dto.EncryptedResponseDTO;
import com.qthegamep.diploma.project.back.dto.SignedDataDTO;
import com.qthegamep.diploma.project.back.utils.Consts;
import org.json.JSONException;
import org.json.JSONObject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import javax.inject.Inject;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import java.nio.charset.StandardCharsets;
import java.util.*;

public class ApiBackOperationServiceImpl implements ApiBackOperationService {
    private static final Logger LOG = LoggerFactory.getLogger(ApiBackOperationServiceImpl.class.getName());

    @Inject
    private SenderHttpService senderHttpService;
    @Inject
    private Gson gson;

    @Override
    public List<EndUserSession> getSessions(String requestId) throws Exception {
        Response response = senderHttpService.getApiCerts(requestId);
        SignedDataDTO signRequestDTO = gson.fromJson(response.readEntity(String.class), SignedDataDTO.class);
        JSONObject jsonObject = getCertsFromSign(signRequestDTO);
        String cloudCert = jsonObject.getJSONArray("certificates").getJSONObject(0).getString("cert");
        LOG.info("CloudApiBackCert: {}", cloudCert);
        String hsmCert = jsonObject.getJSONArray("certificates").getJSONObject(1).getString("cert");
        LOG.info("GryadaCert: {}", hsmCert);
        byte[] certCloudByte = Base64.getDecoder().decode(cloudCert);
        byte[] certHsmByte = Base64.getDecoder().decode(hsmCert);
        EndUserSession sessionCloud = CryptoOperation.genSessionEnc(certCloudByte);
        EndUserSession sessionHsm = CryptoOperation.genSessionEnc(certHsmByte);
        List<EndUserSession> sessions = new ArrayList<>();
        sessions.add(sessionCloud);
        sessions.add(sessionHsm);
        return sessions;
    }

    public String createUserPassJson(String pass) throws JSONException {
        JSONObject object = new JSONObject();
        object.put("namedKeyPassword", pass);
        return object.toString();
    }

    public String generateKeys(EndUserSession sessionHsm, String requestId) throws Exception {
        String userPassJson = createUserPassJson(Consts.USER_PASS);
        String userPassJsonEnc = createEncReqWithEndUserSession(userPassJson, sessionHsm);
        String requestToGenKeysEnc = createReqToGenerateKey(userPassJsonEnc);
        String requestToGenKeysEncWithSign = createJsonWithTechSign(requestToGenKeysEnc.getBytes(StandardCharsets.UTF_8));
        Response resultGen = senderHttpService.generateNewCerts(requestId, requestToGenKeysEncWithSign);
        return resultGen.readEntity(String.class);

    }
    public String createJsonWithTechSign( byte[] data) throws JSONException {
        String signed = null;
//        String prominSession = Util.getProminSession();
        String prominSession = Consts.PROMIN;
```

```
                String req = "<?xml version=\"1.0\" encoding=\"UTF-8\"?><req><sid>#sid#</sid><busid>CLOUD</busid><bank>1</
bank><tickets><ticket>#data#</ticket></tickets></req>";
        String info = req.replaceFirst("#sid#", prominSession).replaceFirst("#data#", Base64.getEncoder().encodeToString(data));
        Client client = ClientBuilder.newClient();
        Response response = client.target(Consts.URL_ACSK_TECH_SIGN)
                .request(MediaType.APPLICATION_XML)
                .post(Entity.xml(info));
        String result = response.readEntity(String.class);
        LOG.info("SIGN_TECH_KEY: {}", result);
        if (result.contains("<signed>")) {
            signed = result.substring(result.indexOf("<signed>") + 8, result.indexOf("</signed>"));
        }
        JSONObject job = new JSONObject();
        job.put("signedData", signed);
        return job.toString();
    }

    public String getOperationId(EndUserSession sessionCloud, String requestId) throws Exception {
        String reqClientIdJson = createClientIdJson();
        String encReqClientId = CryptoOperation.sessionEnc(sessionCloud, reqClientIdJson);
        EncryptedRequestDTO encryptedRequestDTO = createEncryptedRequest(sessionCloud, encReqClientId);
        Response responseOperId = senderHttpService.getOperationId(requestId, gson.toJson(encryptedRequestDTO));
//        LOG.info("OperationId response: {}", responseOperId.readEntity(String.class));
        EncryptedResponseDTO responseDTO = gson.fromJson(responseOperId.readEntity(String.class), EncryptedResponseDTO.class);
        LOG.info("**Acquire-operation-id response: {}", responseDTO);
        String signData = new String(CryptoOperation.decrResp(responseDTO.getEncryptedData(), sessionCloud), StandardCharsets.UTF_8);
        String operationId = getDataFromSign(signData, "operationId");
        LOG.info("**Acquire-operation-id** {}", operationId);
        return operationId;
    }

    public String createReqToGenerateKey(String pass) {
        JSONObject obj = new JSONObject();
        obj.put("ekbId", Consts.EKB_ID);
        obj.put("nameKeyLabel", Consts.USER_ALIAS);
        obj.put("password", new JSONObject(pass));
        return obj.toString();
    }

    public EncryptedRequestDTO createEncryptedRequest(EndUserSession session, String dataToEnc) {
        EncryptedRequestDTO encryptedRequestDTO = new EncryptedRequestDTO();
        String authData = Base64.getEncoder().encodeToString(session.GetData());
        encryptedRequestDTO.setAuthData(authData);
        encryptedRequestDTO.setEncryptedData(dataToEnc);
        return encryptedRequestDTO;
    }
    private JSONObject getCertsFromSign(SignedDataDTO signedDataDTO) throws Exception {
        EndUserSignInfo endUserSignInfo = CryptoOperation.getDataFromSign(signedDataDTO.getSignedData());
        return new JSONObject(new String(endUserSignInfo.GetData(), StandardCharsets.UTF_8));
    }

    private String createClientIdJson() {
        Map<String, String> map = new HashMap<>();
        map.put("clientId", Consts.CLIENT_ID);
        return gson.toJson(map);
    }
    public String createEncReqWithEndUserSession(String data, EndUserSession clientSession) throws Exception {
        String encPass = CryptoOperation.sessionEnc(clientSession, data);
        String authData = Base64.getEncoder().encodeToString(clientSession.GetData());
        Map<String, String> map = new HashMap<>();
        map.put("authData", authData);
        map.put("encryptedData", encPass);
        return gson.toJson(map);
    }

    public String getDataFromSign(String signData, String value) throws Exception {
        JSONObject jsonObject = new JSONObject(signData);
        EndUserSignInfo endUserSignInfo = CryptoOperation.getDataFromSign(jsonObject.getString("signedData"));
        jsonObject = new JSONObject(new String(endUserSignInfo.GetData(), StandardCharsets.UTF_8));
        LOG.info("Data from signed: {} Value: {}", jsonObject, value);
        return jsonObject.getString(value);
    }
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/service/GenerationService.java

```
package com.qthegamep.diploma.project.back.service;

import com.qthegamep.diploma.project.back.dto.GenerationRequestDTO;
import com.qthegamep.diploma.project.back.dto.GenerationResponseDTO;

@FunctionalInterface
public interface GenerationService {
    GenerationResponseDTO generate(GenerationRequestDTO generationRequestDTO, String requestId);
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/service/GenerationServiceImpl.java

```
package com.qthegamep.diploma.project.back.service;

import com.iit.certificateAuthority.endUser.libraries.signJava.EndUserSession;
import com.qthegamep.diploma.project.back.dto.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import javax.inject.Inject;
import java.util.List;


public class GenerationServiceImpl implements GenerationService {

    private static final Logger LOG = LoggerFactory.getLogger(GenerationServiceImpl.class.getName());
    @Inject
    private ApiBackOperationService apiBackOperationService;
    @Override
    public GenerationResponseDTO generate(GenerationRequestDTO generationRequestDTO, String requestId) {
        try {
            List<EndUserSession> sessions = apiBackOperationService.getSessions(requestId);
            EndUserSession sessionHsm = sessions.get(1);
            EndUserSession sessionCloud = sessions.get(0);
//            String operationId = apiBackOperationService.getOperationId(sessionCloud, requestId);
            String response = apiBackOperationService.generateKeys(sessionHsm, requestId);
            LOG.info("Result generation: {} RequestId: {}", response, requestId);

        } catch (Exception e) {
            LOG.warn("GENERATION ERROR", e);

        }


        return null;
    }
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/service/SenderHttpService.java

```
package com.qthegamep.diploma.project.back.service;

import javax.ws.rs.core.Response;

public interface SenderHttpService {
    Response getApproveResponse(String requestId, String data);

    Response getStatusSignResponse(String requestId, String data);

    Response getApiCerts(String requestId);

    Response getSignResponse(String requestId, String data);

    Response getOperationId(String requestId, String data);
```

```
    Response generateNewCerts(String requestId, String data);
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/service/SenderHttpServiceImpl.java

```
package com.qthegamep.diploma.project.back.service;

import com.qthegamep.diploma.project.back.utils.Consts;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

public class SenderHttpServiceImpl implements SenderHttpService {

    private Client client = ClientBuilder.newClient();

    public Response getApproveResponse(String requestId, String data) {
        return client.target(Consts.URL_KUBER_APPROVE)
                .request(MediaType.APPLICATION_JSON)
                .header("requestId", requestId)
                .post(Entity.json(data));
    }

    public Response getStatusSignResponse(String requestId, String data) {
        return client.target(Consts.URL_KUBER_SIGN_STATUS)
                .request(MediaType.APPLICATION_JSON)
                .header("requestId", requestId)
                .post(Entity.json(data));
    }

    public Response getApiCerts(String requestId) {
        return client.target(Consts.URL_KUBER_GET_CERTS_EXT)
                .request()
                .header("requestId", requestId)
                .get();
    }

    public Response getSignResponse(String requestId, String data) {
        return client.target(Consts.URL_KUBER_SIGN)
                .request(MediaType.APPLICATION_JSON)
                .header("requestId", requestId)
                .post(Entity.json(data));
    }

    public Response getOperationId(String requestId, String data) {
        return client.target(Consts.URL_KUBER_GET_OPERATION_ID)
                .request(MediaType.APPLICATION_JSON)
                .header("requestId", requestId)
                .post(Entity.json(data));
    }

    public Response generateNewCerts(String requestId, String data) {
        return client.target(Consts.URL_KUBER_KEY_GENERATE)
                .request(MediaType.APPLICATION_JSON)
                .header("requestId", requestId)
                .post(Entity.json(data));
    }

}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/service/SignService.java

```
package com.qthegamep.diploma.project.back.service;

import com.qthegamep.diploma.project.back.dto.SignRequestDTO;
import com.qthegamep.diploma.project.back.dto.SignResponseDTO;

@FunctionalInterface
public interface SignService {
    SignResponseDTO sign(SignRequestDTO requestDTO, String requestId);
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/service/SignServiceImpl.java

```
package com.qthegamep.diploma.project.back.service;

import com.google.gson.Gson;
import com.iit.certificateAuthority.endUser.libraries.signJava.EndUserSession;
import com.iit.certificateAuthority.endUser.libraries.signJava.EndUserSignInfo;
import com.qthegamep.diploma.project.back.crypto.CryptoOperation;
import com.qthegamep.diploma.project.back.dto.*;
import com.qthegamep.diploma.project.back.utils.Consts;
import org.json.JSONException;
import org.json.JSONObject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import ua.privatbank.cryptonite.CryptoniteX;

import javax.inject.Inject;
import javax.ws.rs.core.Response;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.*;

public class SignServiceImpl implements SignService {
    private static final Logger LOG = LoggerFactory.getLogger(SignServiceImpl.class.getName());
    private SenderHttpService senderHttpService;
    private ApiBackOperationService apiBackOperationService;
    private Gson gson;

    @Inject
    public SignServiceImpl(SenderHttpService senderHttpService, ApiBackOperationService apiBackOperationService, Gson gson) {
        this.senderHttpService = senderHttpService;
        this.apiBackOperationService = apiBackOperationService;
        this.gson = gson;
    }

    @Override
    public SignResponseDTO sign(SignRequestDTO requestDTO, String requestId) {
        SignResponseDTO signResponseDTO = new SignResponseDTO();
        try {
            byte[] reqData = requestDTO.getData().getBytes(StandardCharsets.UTF_8);
            String hashToSign = CryptoOperation.getHashData(reqData);
            List<EndUserSession> sessions = apiBackOperationService.getSessions(requestId);
            EndUserSession sessionCloud = sessions.get(0);
            EndUserSession sessionHsm = sessions.get(1);

            String operationId = apiBackOperationService.getOperationId(sessionCloud, requestId);

            String signRequest = createSignRequest(operationId, sessionCloud, hashToSign);
            Response responseSign = senderHttpService.getSignResponse(requestId, signRequest);
            EncryptedResponseDTO encryptedResponseDTO = gson.fromJson(responseSign.readEntity(String.class), EncryptedResponseDTO.class);
                        String signData = new String(CryptoOperation.decrResp(encryptedResponseDTO.getEncryptedData(), sessionCloud),
StandardCharsets.UTF_8);
            String status = getDataFromSignLong(signData, "status");
            LOG.info("**SIGN** {}", status);

            String userPassJson = createUserPassJson(Consts.USER_PASS);
            String userPassJsonEnc = apiBackOperationService.createEncReqWithEndUserSession(userPassJson, sessionHsm);
            String requestDataToApprove = createGsonToApproveSign(userPassJsonEnc, Consts.CLIENT_ID, operationId);
                                                                String requestDataToApproveWithTechSign =
apiBackOperationService.createJsonWithTechSign(requestDataToApprove.getBytes(StandardCharsets.UTF_8));
            Response responseApprove = senderHttpService.getApproveResponse(requestId, requestDataToApproveWithTechSign);
```

```
        LOG.info("**APPROVE** {}", responseApprove.readEntity(String.class));

        String reqStatusSign = createJsonSignStatus(Consts.CLIENT_ID, operationId);
        String encryptRequestStatusSign = apiBackOperationService.createEncReqWithEndUserSession(reqStatusSign, sessionCloud);
        Response responseStatusSign = senderHttpService.getStatusSignResponse(requestId, encryptRequestStatusSign);
                EncryptedResponseDTO encryptedResponseStatusSign = gson.fromJson(responseStatusSign.readEntity(String.class),
EncryptedResponseDTO.class);
                signData = new String(CryptoOperation.decrResp(encryptedResponseStatusSign.getEncryptedData(), sessionCloud),
StandardCharsets.UTF_8);
        String signResponse = apiBackOperationService.getDataFromSign(signData, "signature");
        LOG.info("**SIGN_STATUS** {}", signResponse);
        byte[] signatureByte = Base64.getDecoder().decode(signResponse);
        byte[] cmsWithSign = CryptoniteX.cmsSetData(signatureByte, reqData);
        String resultSign = Base64.getEncoder().encodeToString(cmsWithSign);
        LOG.info("**CMS** {}", resultSign);
        Path pathToSave = Paths.get("src/main/resources/sign.sig");
//          Files.write(pathToSave, cmsWithSign);
//          try (OutputStream stream = new FileOutputStream("/result.sign")) {
//              stream.write(cmsWithSign);
//          }
        signResponseDTO.setSign(resultSign);


      } catch (Exception e) {
          LOG.warn("[SignService ERROR]", e);
      }
      return signResponseDTO;
   }

   public String createJsonSignStatus(String clientID, String operId) throws JSONException {
      JSONObject object = new JSONObject();
      object.put("clientId", clientID);
      object.put("operationId", operId);
      return object.toString();
   }

   private String createUserPassJson(String pass) throws JSONException {
      JSONObject object = new JSONObject();
      object.put("namedKeyPassword", pass);
      return object.toString();
   }

   public String createGsonToApproveSign(String encData, String clientId, String operationId) throws JSONException {
      JSONObject obj = new JSONObject();
      obj.put("nameKeyLabel", Consts.USER_ALIAS);
      obj.put("password", new JSONObject(encData));
      obj.put("clientId", clientId);
      obj.put("operationId", operationId);
      return obj.toString();
   }
   private String getDataFromSignLong(String signData, String value) throws Exception {
      JSONObject object = new JSONObject(signData);
      EndUserSignInfo endUserSignInfo = CryptoOperation.getDataFromSign(object.getString("signedData"));
      object = new JSONObject(new String(endUserSignInfo.GetData(), StandardCharsets.UTF_8));
      LOG.info("Data from sign: {} Value: {}", object, value);
      return String.valueOf(object.getLong(value));
   }

   private String createSignRequest(String operationId, EndUserSession session, String hash) throws Exception {
      JSONObject obj = new JSONObject();
      obj.put("clientId", Consts.CLIENT_ID);
      obj.put("operationId", operationId);
      obj.put("time", new Date().toString());
      obj.put("originatorDescription", "P24");
      obj.put("operationDescription", "QAZWSX123");
      obj.put("hash", hash);
      obj.put("signatureAlgorithmName", "DSTU4145");
      obj.put("signatureFormat", "PKCS7");
      String encAllReq = apiBackOperationService.createEncReqWithEndUserSession(obj.toString(), session);
      LOG.info("Sign request: {} Encrypted req: {}", obj.toString(), encAllReq);
      return encAllReq;
   }
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/utils/Consts.java

```
package com.qthegamep.diploma.project.back.utils;

public class Consts {

    public static final String URL_KUBER_KEY_GENERATE = "https://acsk.privatbank.ua/cloud/api/back/key";
    public static final String URL_KUBER_APPROVE = "https://acsk.privatbank.ua/cloud/api/back/approve";
    public static final String URL_KUBER_SIGN = "https://acsk.privatbank.ua/cloud/api/back/sign";
    public static final String URL_KUBER_SIGN_STATUS = "https://acsk.privatbank.ua/cloud/api/back/sign-status";
    public static final String URL_KUBER_GET_CERTS_EXT = "https://acsk.privatbank.ua/cloud/api/back/get-certificates-ext";
    public static final String URL_KUBER_GET_OPERATION_ID = "https://acsk.privatbank.ua/cloud/api/back/acquire-operation-id";
    public static final String URL_ACSK_TECH_SIGN = "https://acsk.privatbank.ua/cloud/tickets/sign/docs";


    public static final String USER_PASS = "";
    public static final String USER_ALIAS = "";
    public static final String CLIENT_ID = "";
    public static final String EKB_ID = "";
    public static final String PROMIN = "";
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/utils/Util.java

```
package com.qthegamep.diploma.project.back.utils;

import redis.clients.jedis.Jedis;

public class Util {
    public static String getProminSession() {
        Jedis jedis = new Jedis(System.getProperty("acsk.redis.host"), Integer.parseInt(System.getProperty("acsk.redis.port")));
        return jedis.get("current.session.ekb");
    }
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/Config.java

```
package com.qthegamep.diploma.project.back;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.Objects;
import java.util.Properties;

public class Config {
    private static final Logger LOG = LoggerFactory.getLogger(Config.class);

    private Config() {
    }

    public static void init() throws IOException, URISyntaxException {
        String propertiesFilePath = System.getProperty("config.properties");
        if (propertiesFilePath == null || propertiesFilePath.isEmpty()){
            URL res = Config.class.getResource("/config.properties");
            File file = Paths.get(Objects.requireNonNull(res).toURI()).toFile();
            propertiesFilePath = file.getAbsolutePath();
        }
        LOG.info("Loading properties... {}", propertiesFilePath);
```

```
        try (FileInputStream is = new FileInputStream(propertiesFilePath)) {
            Properties properties = new Properties();
            properties.load(is);
            properties.stringPropertyNames()
                    .forEach(key -> {
                        String value = properties.getProperty(key);
                        System.setProperty(key, value);
                        LOG.debug("Properties: {} : {}", key, value);
                    });
        } catch (Exception e) {
            LOG.error("Can't load config.properties. Exception: ", e);
        }
    }
}
```

diploma.project.back/src/main/java/com/qthegamep/diploma/project/back/Main.java

```
package com.qthegamep.diploma.project.back;

import com.qthegamep.diploma.project.back.binder.ApplicationBinder;
import com.qthegamep.diploma.project.back.crypto.CryptoOperation;
import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
import org.glassfish.jersey.server.ResourceConfig;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import ua.privatbank.cryptonite.CryptoniteX;

import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;

public class Main {
    private static final Logger LOG = LoggerFactory.getLogger(Main.class);

    public static void main(String[] args) throws IOException, InterruptedException, URISyntaxException {
        Config.init();
        CryptoOperation.getEndUser();
        CryptoniteX.init();
        LOG.info("CryptoniteX was initialized");
        String host = System.getProperty("host", "0.0.0.0");
        String port = System.getProperty("port", "8080");
        String appContext = System.getProperty("appContext", "");
        String appUrl = "http://" + host + ":" + port + appContext;

        startServer(appUrl);
        LOG.info("cloud-service-app started at {}", appUrl);
        Thread.currentThread().join();
    }

    private static void startServer(String appUrl) {
        final ResourceConfig rc = new ResourceConfig()
                .packages(Main.class.getPackage().getName())
                .register(ApplicationBinder.builder().build());
        GrizzlyHttpServerFactory.createHttpServer(URI.create(appUrl), rc);
    }
}
```

diploma.project.back/src/main/resources/config.properties

```
host=localhost
port=8081
appContext=/back/cloud
acsk.redis.host=10.1.206.33
acsk.redis.port=6379
fileStorePath=/opt/certs
acsk.ocsp.service.url=http://acsk.privatbank.ua/services/ocsp/
enduser.cmp.url=http://acsk.privatbank.ua/services/cmp/
enduser.tsp.url=http://acsk.privatbank.ua/services/tsp/
```

```
enduser.ocsp.url=http://acsk.privatbank.ua/services/ocsp/
libExtractionPath=/opt/iit/diploma.project.back/EUSign-x64-1.3.148
```

diploma.project.back/src/main/resources/logback.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<configuration>

   <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
      <!-- encoders are assigned the type
           ch.qos.logback.classic.encoder.PatternLayoutEncoder by default -->
      <encoder>
         <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>
      </encoder>
   </appender>

   <root level="INFO">
      <appender-ref ref="STDOUT" />
   </root>
</configuration>
```

diploma.project.back/.gitignore

```
*.iml
.idea/
target/
```

diploma.project.back/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>

   <groupId>com.qthegamep</groupId>
   <artifactId>diploma.project.back</artifactId>
   <version>1.0.0</version>
   <build>
      <plugins>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>2.5.1</version>
            <configuration>
               <source>8</source>
               <target>8</target>
            </configuration>
         </plugin>
      </plugins>
   </build>

   <dependencies>
      <dependency>
         <groupId>org.glassfish.jersey.containers</groupId>
         <artifactId>jersey-container-grizzly2-servlet</artifactId>
         <version>2.28</version>
      </dependency>
      <!-- https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
      <dependency>
         <groupId>org.glassfish.jersey.inject</groupId>
         <artifactId>jersey-hk2</artifactId>
         <version>2.28</version>
      </dependency>
      <!-- https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-media-json-jackson -->
```

```
        <dependency>
            <groupId>org.glassfish.jersey.media</groupId>
            <artifactId>jersey-media-json-jackson</artifactId>
            <version>2.28</version>
        </dependency>
        <dependency>
            <groupId>org.json</groupId>
            <artifactId>json</artifactId>
            <version>20160810</version>
        </dependency>
        <dependency>
            <groupId>cryptolib</groupId>
            <artifactId>EndUser13</artifactId>
            <version>0.0.9</version>
            <type>jar</type>
        </dependency>
        <dependency>
            <groupId>ch.qos.logback</groupId>
            <artifactId>logback-classic</artifactId>
            <version>1.2.3</version>
        </dependency>
        <dependency>
            <groupId>com.iit</groupId>
            <artifactId>EUSignJava</artifactId>
            <version>20210408</version>
            <type>jar</type>
        </dependency>
        <dependency>
            <groupId>cryptolib</groupId>
            <artifactId>cryptonite-x</artifactId>
            <version>210428</version>
        </dependency>
        <dependency>
            <groupId>redis.clients</groupId>
            <artifactId>jedis</artifactId>
            <version>2.9.0</version>
        </dependency>
        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>2.8.1</version>
        </dependency>
    </dependencies>
</project>
```