ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти бакалавр

на тему: **«Розроблення програмного забезпечення для дослідження методів криптографії з відкритим ключем»**

Виконав: здобувач вищої освіти
за освітньо-професійною
програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні
системи та технології
ступеня вищої освіти бакалавр
групи 126ІСТ_бд_2020
Білокінь О.С.
Керівник: Одарущенко О. Б.
Рецензент: Брикун О.М.

Полтава – 2024 року

ВСТУП

Криптографія з відкритим ключем залишається актуальною в сучасному цифровому світі [1]. Крім того, криптографія відкритого ключа відіграє ключову роль у боротьбі з кіберзлочинністю та шпигунством. Вона забезпечує захист конфіденційності даних, що особливо важливо в контексті розширення Інтернету речей (IoT) та зростання кількості онлайн – транзакцій.

Протягом останніх двадцяти років, еволюція криптографічних ключів відбувалася на фоні стрімкого розвитку інформаційних технологій та зростання загроз кібербезпеці. Початок цього періоду відзначився публікаціями у періодичних виданнях та наукових роботах, які розкривали нові методи та алгоритми криптографії, а також їх застосування у сучасних системах інформаційної безпеки. Захист даних в Інтернеті, особливо через протоколи TLS/SSL [2], вимагав постійного оновлення криптографічних ключів для забезпечення безпеки передачі інформації між вузлами мережі. Починаючи зі слабких ключів та алгоритмів, які швидко ставали уразливими перед сучасними атаками, до впровадження більш потужних та надійних криптографічних алгоритмів, таких як RSA, ECC та інші.

Також спостерігається постійний розвиток алгоритмів та підходів до шифрування, щоб протистояти новим загрозам та підвищити рівень безпеки [3]. Цифрові підписи стали важливим інструментом підтвердження автентичності даних та документів у цифровому середовищі. Вони використовують криптографічні ключі для створення та перевірки підписів, що дозволяє впевнитися в їх цілісності та автентичності.

Криптовалюти стали предметом все більшого інтересу, а їх безпека базується на сильних криптографічних ключах та алгоритмах. Механізми дистрибуції ключів, такі як протокол Діффі – Геллмана, відіграють важливу роль у забезпеченні безпеки комунікацій шляхом створення спільних секретних ключів між вузлами мережі без необхідності обміну ними через відкриті канали зв'язку. Їх розвиток і вдосконалення продовжуються,

пристосовуючись до нових технологічних викликів та загроз кібербезпеці, забезпечуючи надійний захист даних у цифровому світі.

Мета роботи – провести аналіз різноманітних підходів до криптографії з відкритим ключем, а також у формуванні набору функціональних вимог та архітектури для створення програмного забезпечення, що забезпечує безпечну комунікацію. Завершальним етапом є розробка та реалізація програмного забезпечення з використанням обраного підходу до криптографії з відкритим ключем відповідно до сформованої архітектури та функціональних вимог.

Завдання роботи – провести аналіз різноманітних підходів до криптографії з відкритим ключем, а також формування набору функціональних вимог та архітектури для створення програмного забезпечення.

Об'єкт дослідження кваліфікаційної роботи – методи і інструментальні засоби розроблення криптографічних відкритих ключів.

Предмет дослідження кваліфікаційної роботи – процеси розробки криптографічного відкритого ключа.

Методи дослідження. Моделювання алгоритмів (створення комп'ютерних моделей для симуляції криптографічних алгоритмів та оцінки їх ефективності). Симуляція атак (розробка та тестування атак на криптографічні алгоритми для оцінки їх стійкості).

Інформаційна база, що використовувалася: публікації в періодичних виданнях, вебресурси, міжнародні стандарти з криптографії стандарти.

Практична значущість: дана робота полягає у створенні та впровадженні криптографічного ключа для дослідження (PKI). Це дозволяє забезпечити надійний механізм перевірки права власності на відкритий ключ, що є важливим для захисту інформації.

Робота складається зі вступу, трьох розділів, висновків та списку літератури. Обсяг роботи становить 50 сторінок, 2 таблиць, 24 рисунків.

РОЗДІЛ 1
АНАЛІЗ ЗАСОБІВ ТА ТЕХНОЛОГІЙ КРИПТОГРАФІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ

1.1 Короткий огляд криптографії з відкритим ключем

Криптографія з відкритим ключем вважається основою безпечного зв'язку в сучасному світі. Проте, однією з основних проблем є перевірка права власності на відкритий ключ, який надсилається однією стороною іншій. PKI вирішує цю проблему [4], забезпечуючи надійний механізм перевірки власності ключа. PKI застосовувалася, для захисту Інтернету та здатності подолати атаки Man – in – the – middle. Крім того, вони можуть зрозуміти значення кореня довіри у PKI та можливі наслідки порушення цієї довіри (рис.1.1).

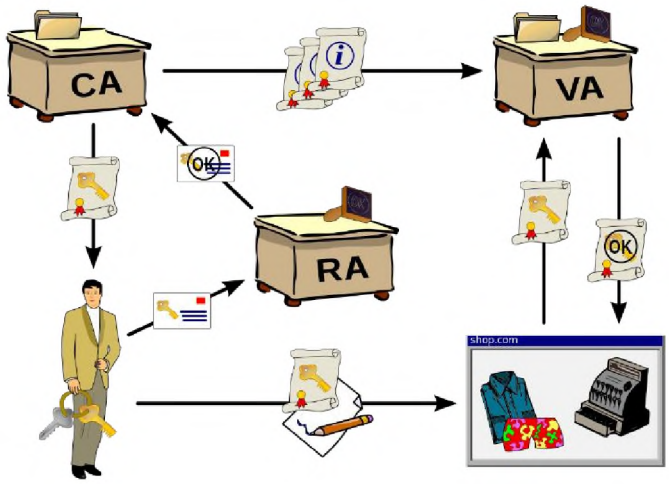

Рисунок 1.1 – Схема інфраструктури відритих ключів

Першою криптографією з відкритим ключем, яка набула широкого визнання, є система RSA (рис.1.2), запропонована в 1977 році Роном Рівестом, Аді Шаміром та Леонардом Адлеманом. RSA використовує два ключі - відкритий та закритий – для шифрування та розшифрування повідомлень відповідно. Вона ґрунтується на складності факторизації великих простих

чисел та забезпечує ідентифікацію особи через відкриті ключі, що дозволяє перевірити автентичність отримувача. RSA широко використовується для захисту зв'язку в Інтернеті, електронної пошти та інших цифрових середовищах.

Перше застосування криптографії з відкритим ключем відбулося у 1970 – х роках і було пов'язане з роботою Вітфілда Діффі та Мартіна Геллмана [5]. Вони опублікували статтю "Нові напрямки в сучасній криптографії" у журналі "IEEE Transactions on Information Theory" у 1976 році. Ця робота запропонувала новий підхід до обміну ключами в криптографії, відомий як протокол Діффі – Геллмана. Протокол Діффі – Геллмана дозволяє двом сторонам безпечно обмінюватися секретними ключами через ненадійний канал зв'язку. Основна ідея полягає в тому, що кожна сторона генерує публічний і приватний ключі.

Рисунок 1.2 – Вигляд першого криптографічного алгоритму

Публічні ключі можуть бути розповсюджені відкрито, тоді як приватні ключі залишаються секретними. За допомогою цих ключів сторони обмінюються інформацією, яка дозволяє обчислити спільний секретний ключ, який може бути використаний для подальшого шифрування повідомлень. Цей протокол відкрив нові можливості для безпечного обміну інформацією через відкриті мережі, такі як Інтернет. Він став основою для багатьох сучасних

криптографічних протоколів і алгоритмів, таких як TLS [6], який використовується для захисту комунікацій в мережі Інтернет.

Першими компаніями, які почали використовувати алгоритми криптографії з відкритим ключем, є наукові установи та урядові організації, які займалися дослідженнями в цій області. Найбільш відомим прикладом є Массачусетський Технологічний Інститут (MIT), де Рональд Рівест, Аді Шамір і Леонард Адлеман розробили алгоритм RSA (названий за першими літерами їхніх прізвищ) у 1977 році.

RSA став першим алгоритмом, який був придатний як для шифрування, так і для цифрового підпису. Цей алгоритм відразу ж викликав значний інтерес як у наукових, так і військових колах, оскільки він вирішував проблему обміну секретними ключами через ненадійні канали зв'язку. Крім того, урядові агентства, такі як центр урядового зв'язку Великої Британії (GCHQ), також проводили дослідження у цій області. Наприклад, Кліффорд Кокс і Малькольм Вільямсон з GCHQ розробили подібні алгоритми ще до розробки RSA, але їх робота була тримана в секреті.

Протягом 1980-х та 1990-х років криптографія з відкритим ключем пережила значний прогрес та активний розвиток [7]. Однією з найважливіших подій цього періоду стала широка популяризація алгоритму RSA. RSA став першим алгоритмом, придатним як для шифрування, так і для цифрового підпису, і його застосування охоплювало різноманітні криптографічні потреби.

Паралельно з цим розвивалися стандартизовані криптографічні протоколи, такі як SSL та TLS, що забезпечували безпечні з'єднання в мережі Інтернет, зокрема, шифрування комунікацій між веб–серверами та клієнтами. Крім того, інші алгоритми криптографії з відкритим ключем, такі як алгоритм Ель – Гамаля, також стали популярними для шифрування та генерації цифрових підписів. У цей період уряди багатьох країн почали активно використовувати криптографію з відкритим ключем для захисту своїх комунікацій та інформації. Це стало особливо актуальним у зв'язку з

поширенням комп'ютерних та інформаційних систем у сфері урядового управління. Паралельно з практичним використанням криптографії з відкритим ключем, протягом цього періоду було проведено значну кількість досліджень, спрямованих на вдосконалення та розширення концепцій криптографії з відкритим ключем. Нові алгоритми, протоколи та методи були розроблені та випробувані.

Протягом 1990 – 2000 років криптографія з відкритим ключем (PKC) пережила значний прогрес та зазнала широкого використання в різних сферах.

По – перше, у цей період було помітною тенденція зростання інтересу до криптографії з відкритим ключем з боку приватних компаній та наукових установ. Це відобразилося у збільшенні дослідницьких робіт, спрямованих на розробку нових методів шифрування та підвищення стійкості існуючих алгоритмів. Такий інтерес спонукало зростання усвідомлення важливості кібербезпеки та необхідності захисту конфіденційної інформації в цифровій епохі.

По – друге, важливим етапом була розробка нових криптографічних алгоритмів та протоколів, спрямованих на підвищення безпеки та ефективності. Прийняття алгоритму шифрування AES визначило цей напрямок. AES став відомим як надійний та ефективний метод шифрування, що широко використовувався в банківському секторі [8], електронній комерції та інших галузях, де конфіденційність даних була критично важливою. На рисунку1.3 можна бачити симетричний алгоритм блочної форми.

Крім того, урядові структури та міжнародні організації активно працювали над створенням стандартів безпеки та захисту інформації, що призвело до широкого впровадження криптографії з відкритим ключем в публічний та приватний сектори.

Криптографія з відкритим ключем відіграє критичну роль у ЦС. Ці центри відповідають за видачу, валідацію та управління цифровими сертифікатами, які підтверджують електронні ідентифікатори, такі як публічні ключі.

Однією з ключових потреб криптографії з відкритим ключем у ЦС є можливість забезпечення безпеки взаємодії між різними суб'єктами в інтернет–середовищі. Це досягається за допомогою цифрових підписів та шифрування інформації. Наприклад, при видачі сертифіката ЦС може підписати цифровий документ своїм приватним ключем, що гарантує його автентичність. Крім того, асиметрична криптографія дозволяє безпечно обмінюватися секретною інформацією, шифруючи її публічним ключем отримувача, який може розшифрувати її за допомогою власного приватного ключа.

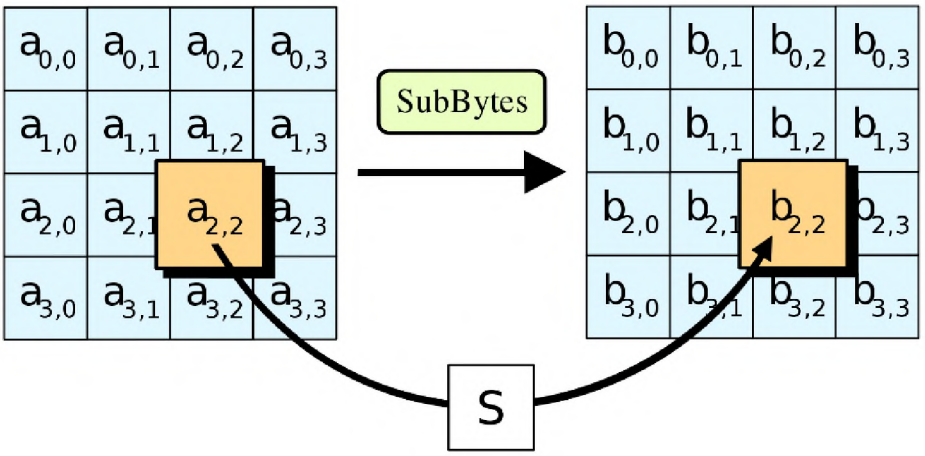

Рисунок 1.3 – Симетричний алгоритм блочної форми

Також криптографія з відкритим ключем дозволяє здійснювати безпечний обмін публічними ключами між різними суб'єктами. Наприклад, ЦС може використовувати механізми, такі як цифрові сертифікати та протоколи обміну ключами, для забезпечення валідності та безпеки публічних ключів, що використовуються в їхніх операціях. У Windows асиметрична криптографія широко використовується для різних цілей, включаючи забезпечення безпеки та ідентифікації в різних системах та програмах. Одним із важливих застосувань є використання асиметричної криптографії для захисту цифрових підписів та шифрування даних. Більш детально про цю тему буде написано в розділі 1.2.

Також у Windows криптографія з відкритим ключем широко використовується для різних цілей, включаючи забезпечення безпеки та ідентифікації в різних системах та програмах. Одним із важливих застосувань є використання асиметричної криптографії для захисту цифрових підписів та шифрування даних докладно про це буде описано в розділі 1.3.

1.2 Використання PKI в центрах сертифікації

Центри сертифікації почали існувати з появою PKI, що розвинулися в кінці 20 – го століття. Одним із перших прикладів PKI був RSA Security, який в 1986 році впровадив інфраструктуру для створення та управління цифровими сертифікатами. Проте системи сертифікації розвивалися далі, а основним поглядом стало використання ЦС у великих мережах, таких як Інтернет.

Перші ЦС зазвичай були приватними компаніями [9], що спеціалізувалися на кібербезпеці. Однак з часом почали виникати й громадські ЦС, які надавали цифрові сертифікати для відкритого використання. Один з перших громадських ЦС, що став відомим, це VeriSign, заснований у 1995 році. Він швидко став ключовим гравцем у сфері видачі сертифікатів для безпеки та аутентифікації в Інтернеті.

З поширенням Інтернету та розвитком електронних торгівлі та інших онлайн – послуг, попит на цифрові сертифікати і ЦС зростав. Відтак, ЦС почали впроваджуватися широкою мережею, співпрацюючи з браузерами та іншими програмами для забезпечення безпеки та аутентифікації в Інтернеті.

У сучасному періоді часу ЦС відіграють критичну роль у забезпеченні безпеки та аутентифікації в Інтернеті. Вони видають цифрові сертифікати, які підтверджують ідентичність власників вебсайтів, серверів електронної пошти, програм тощо. Ці сертифікати забезпечують шифрування даних і перевірку автентичності в мережах, де важлива конфіденційність та безпека.

Одним із провідних гравців у галузі ЦС є компанія Sectigo, раніше відома як Comodo CA. Sectigo є одним з найбільших світових постачальників цифрових сертифікатів і послуг з кібербезпеки. Вона видає широкий спектр сертифікатів, включаючи SSL / TLS для безпеки вебсайтів, кодові підписи для програмного забезпечення, електронні підписи для електронних документів та багато інших.

Сертифікати, видаються компанією Sectigo, використовуються мільйонами вебсайтів та додатків по всьому світу для захисту конфіденційності та безпеки користувачів. Крім того, Sectigo також надає інші послуги кібербезпеки, такі як виявлення загроз, моніторинг безпеки та розробка безпечних архітектур. Вона є важливим гравцем у сфері кібербезпеки та продовжує розвиватися, щоб забезпечити надійність та безпеку в Інтернеті.

Завдяки роботі центрів сертифікації, користувачі можуть бути впевнені в тому, що їхні дані залишаються конфіденційними та захищеними від небажаних атак. Компанії, такі як Sectigo, є ключовими учасниками у цій сфері, надаючи широкий спектр цифрових сертифікатів та послуг з кібербезпеки для забезпечення безпеки та надійності в Інтернеті. Завдяки розвитку технологій та постійному вдосконаленню стандартів безпеки, центри сертифікації продовжують відігравати важливу роль у забезпеченні безпеки та довіри в цифровому середовищі.

1.3 Використання PKI в Windows

Початок використання PKI в ОС Windows є важливим кроком для забезпечення безпеки та захисту конфіденційної інформації. PKI – це система, яка використовує публічні та приватні ключі для забезпечення безпеки у мережевому середовищі. Основними складовими PKI є сертифікати, публічні та приватні ключі, і ЦС.

У Windows PKI надає інструменти для створення, керування та використання сертифікатів. Це може бути виконано через такі компоненти, як Certificate Services, який може встановлюватися на серверах Windows Server, а також через вбудовані інструменти у самій ОС.

Certificate Services дозволяє створювати і керувати сертифікатами у вашій мережі. Вони можуть використовуватися для різних цілей, таких як шифрування, цифровий підпис та аутентифікація. Крім того, вони можуть бути використані для створення SSL/TLS – з'єднань, що забезпечує безпеку під час передачі даних через Інтернет.

Керування сертифікатами у Windows зазвичай відбувається через MMC або за допомогою спеціальних інструментів [10], таких як Certificate Manager. Ці інструменти дозволяють переглядати, видаляти, експортувати та імпортувати сертифікати, а також керувати довіреними кореневими сертифікатами.

Використання PKI в Windows дозволяє забезпечити безпеку у мережевому середовищі шляхом застосування сучасних методів шифрування, аутентифікації та цифрового підпису. Це є важливою складовою в захисті конфіденційної інформації та забезпеченні цілісності даних у сучасному цифровому світі.

Продовженням впровадження PKI є використання Microsoft Cloud PKI, яке включає в себе використання хмарних послуг Microsoft для управління сертифікатами та ключами. Це надає більш гнучкі можливості для компаній у впровадженні та керуванні інфраструктурою ключів та сертифікатів.

Microsoft Cloud PKI дозволяє підприємствам зосередитися на своїй основній діяльності, використовуючи послуги хмарного сертифікаційного владаря. Це зменшує необхідність у власній інфраструктурі, що може бути складною для керування та підтримки. Замість цього, компанії можуть покластися на безпеку та надійність послуг Microsoft у керуванні сертифікатами та ключами.

За допомогою Microsoft Cloud PKI, компанії можуть використовувати широкий спектр функцій, таких як автоматизоване видалення сертифікатів, моніторинг використання та статистики, а також резервне копіювання та відновлення ключів. Це дозволяє підприємствам забезпечити високий рівень безпеки та надійності в управлінні своєю PKI, не тратячи при цьому значних зусиль на налаштування та підтримку власної інфраструктури.

Також Microsoft Cloud PKI може інтегруватися з іншими хмарними послугами Microsoft, такими як Azure Active Directory, що дозволяє використовувати одні й ті ж ідентифікатори та автентифікаційні методи для доступу до різних сервісів та ресурсів.

В цілому, використання Microsoft Cloud PKI відкриває нові можливості для підприємств у забезпеченні безпеки своєї інфраструктури PKI, зменшуючи складність управління та підтримки, і підвищуючи надійність та безпеку.

1.4 Використання PKI в macOS

З появою цифрового світу і зростанням важливості безпеки інформації, виникла потреба у створенні надійних систем шифрування та аутентифікації. Одним із засобів, що забезпечує ці потреби, є PKI – інфраструктура з відкритим ключем. PKI використовується для створення, управління та розповсюдження цифрових сертифікатів, які використовуються для забезпечення безпеки у зв'язку та ідентифікації користувачів.

Вперше використання PKI в macOS відзначається в версії операційної системи Mac OS X 10.3 Panther. В цій версії Apple вперше включила в свою операційну систему інструменти для роботи з цифровими сертифікатами та інфраструктурою ключів. Це був важливий крок у напрямку забезпечення безпеки інформації для користувачів Mac.

Запровадження PKI в macOS сприяло збільшенню безпеки і захисту приватності користувачів [11]. Одним із ключових аспектів PKI в macOS є

можливість створення та керування ключами шифрування та підпису. Кожен користувач може створювати свої власні ключі та отримувати цифрові сертифікати від відомих центрів сертифікації.

Завдяки використанню PKI, macOS здатна забезпечити безпеку комунікаційних каналів, таких як електронна пошта, веббраузери та інші мережеві сервіси. Діяти як посередник у процесі перевірки автентичності вебсайтів, забезпечуючи, що користувачі взаємодіють саме з тими сервісами, які вони очікують.

Більшість сучасних версій macOS, включаючи macOS Catalina та macOS Big Sur, продовжують підтримувати та розвивати можливості PKI. Apple продовжує вдосконалювати інструменти для керування цифровими сертифікатами, забезпечуючи користувачам надійну систему безпеки та захисту їхньої приватності.

У світі, де зростає кількість загроз кібербезпеці, використання PKI в macOS залишається одним із ключових засобів захисту інформації та забезпечення безпеки користувачів. Це підтверджує важливість і актуальність розвитку та вдосконалення цієї технології в майбутньому.

Одним з ключових аспектів використання PKI в macOS є можливість генерації та управління ключами шифрування та підпису [12]. macOS надає користувачам можливість генерувати власні ключі шифрування та підпису, а також отримувати цифрові сертифікати від надійних центрів сертифікації. Це забезпечує високий рівень безпеки для інформаційної взаємодії користувачів у мережі.

PKI в macOS також використовується для аутентифікації користувачів та служб в мережі. Це дозволяє переконатися, що тільки вірні користувачі та сервіси мають доступ до конфіденційної інформації та ресурсів. Завдяки використанню цифрових сертифікатів та ключів, мережева взаємодія в macOS стає безпечною та надійною.

Додатково, PKI в macOS використовується для забезпечення безпеки комунікаційних каналів, таких як електронна пошта та веббраузери. Під час

взаємодії з вебсайтами або відправки електронних листів, macOS перевіряє дійсність цифрових сертифікатів, що забезпечує користувачам впевненість у безпеці їхніх даних та комунікацій.

У світі, де кіберзлочинність та загрози кібербезпеці постійно зростають, використання PKI в macOS залишається надійним засобом захисту інформації та забезпечення безпеки користувачів. Зусилля Apple по постійному вдосконаленню цієї технології свідчать про його зобов'язання забезпечувати найвищий рівень безпеки для своїх користувачів у цифровому середовищі.

Отже, використання PKI в macOS продовжує бути важливим елементом для забезпечення безпеки та конфіденційності в цифровому світі, і його розвиток та вдосконалення залишаються пріоритетними завданнями для Apple у майбутньому.

1.5 Огляд технологій для створення криптографії з відкритим ключем

Криптографія, наука про захист інформації від несанкціонованого доступу, відображає в собі багатовікову історію. Протягом віків, люди шукали методи захисту конфіденційності, цілісності та доступності даних. Однак, із розвитком комп'ютерів і мереж, виникла потреба у більш сучасних і надійних методах криптографічного захисту, що призвело до створення технологій для розробки криптографії з відкритим ключем.

Однією з найважливіших і перших технологій у цій області була система RSA, названа за ініціалами її винахідників Рівеста, Шамира та Адлемана. RSA стала першою широко використовуваною системою криптографії з відкритим ключем [13]. Ця технологія базується на складності факторизації великих простих чисел і відображає принципи асиметричного шифрування.

Ще однією ключовою технологією у цій області є система шифрування Ель – Гамаля. Цей метод також використовує асиметричне шифрування,

базуючись на складності обчислення дискретного логарифму. Хоча RSA і Ель – Гамаль використовують різні математичні принципи, обидва вони забезпечують ефективний та надійний захист інформації.

Початкові технології для розробки криптографії з відкритим ключем були важливим кроком у забезпеченні безпеки електронних комунікацій. Вони дозволили вирішити проблеми, пов'язані з обміном ключами для шифрування та розшифрування даних, шляхом використання публічного та приватного ключів. Це відкрило нові можливості для безпечного обміну інформацією у віртуальному просторі.

Технології криптографії з відкритим ключем також знайшли широке застосування в інших галузях, включаючи електронну комерцію , банківські операції та цифрові підписи [14-16]. Вони стали невід'ємною частиною сучасного цифрового світу, забезпечуючи конфіденційність та безпеку даних мільйонам користувачів по всьому світу.

Наприкінці XX століття і початку XXI століття, зростання обсягів даних та розвиток обчислювальної техніки спонукали до подальших досліджень у галузі криптографії. Це призвело до створення нових методів та алгоритмів, які поєднували в собі ефективність, надійність та стійкість до криптоаналізу.

Ранні технології для розробки криптографії з відкритим ключем відіграли ключову роль у становленні сучасної кібербезпеки та забезпеченні конфіденційності даних. Завдяки їм відкрилася нова ера в цих сферах, де кожен змішаний крок перестав стати викликом. Вони створили міцний фундамент, на якому базуються подальші інновації, дозволяючи сучасним системам ефективно захищати інформацію та забезпечувати приватність в цифровому середовищі.

У світі швидкозмінюваних технологій та постійного розвитку кіберзагроз, пошук нових технологій для розробки криптографії з відкритим ключем є постійним завданням для дослідників у цій галузі [17]. Виробники криптографічних рішень постійно шукають нові методи та алгоритми, які були б ефективними, надійними та стійкими до атак. Нижче розглянемо деякі

можливі заміни та пропозиції щодо технологій для розробки криптографії з відкритим ключем.

Перш за все, однією з можливих замін для сучасних технологій криптографії з відкритим ключем може бути розвиток і вдосконалення систем на основі квантових обчислювачів. Квантові комп'ютери відкривають нові можливості у сфері криптографії, оскільки вони здатні розгадувати класичні криптографічні алгоритми швидше, ніж сучасні комп'ютери. Тому розробники повинні активно працювати над створенням квантово – стійких алгоритмів, які залишаться надійними навіть у світлі потенційних загроз квантової обчислювальної технології.

Далі, іншою можливою заміною є розвиток та застосування нових криптографічних протоколів та примітивів, які використовуються для побудови систем безпеки [18]. Наприклад, протоколи нульового знання, які вже згадувалися раніше, можуть бути використані для підтвердження ідентичності без розкриття конфіденційних даних.

Крім того, важливо продовжувати дослідження та розвиток криптографічних систем, які базуються на квантових ключах. Квантова криптографія використовує принципи квантової механіки для забезпечення безпеки комунікацій шляхом використання квантових ключів. Це може бути перспективним напрямом для майбутніх криптографічних систем, оскільки вона пропонує теоретично стійке рішення для захисту даних від квантових атак.

У підсумку, розробники криптографічних систем повинні постійно досліджувати та експериментувати з новими технологіями, алгоритмами та протоколами для забезпечення ефективного та надійного захисту даних у сучасному цифровому світі.

Існує ряд програм, які можуть бути використані для цих цілей, розглянемо деякі з них.

Перш за все, однією з найпоширеніших програм для генерації криптографічних відкритих ключів є OpenSSL. OpenSSL є відкритою

програмою, яка надає набір криптографічних функцій та інструментів для захисту даних [19]. Вона підтримує різноманітні криптографічні алгоритми та протоколи, включаючи RSA, ECC, і DH, і забезпечує можливість генерації та керування криптографічними ключами.

Далі, програма GnuPG (GNU Privacy Guard) є іншим популярним вибором для розробки та керування криптографічними ключами. GnuPG надає інтерфейс командного рядка для генерації, імпорту, експорту та керування ключами [20]. Вона підтримує різноманітні алгоритми шифрування та підпису, включаючи RSA, DSA та ECC, і може бути використана для захисту електронної пошти, файлів та комунікацій.

Також варто зазначити програму PuTTYgen [21], яка є частиною пакету програм PuTTY для роботи з мережевими протоколами. PuTTYgen дозволяє генерувати ключі SSH для безпечного з'єднання з віддаленими серверами. Вона підтримує різні типи ключів, включаючи RSA, DSA та ECDSA, і може забезпечити безпеку під час мережевого зв'язку.

Нарешті, програма Kleopatra, яка є частиною пакету програм Gpg4win для операційних систем Windows, також заслуговує на увагу [22]. Kleopatra надає графічний інтерфейс користувача для керування криптографічними ключами, включаючи їх генерацію, імпорт, експорт та дешифрацію. Вона спрощує процес роботи з криптографічними ключами та підписами, забезпечуючи зручний інтерфейс для користувачів.

Усі ці програми представляють лише деякі можливі варіанти для розробки та керування криптографічними відкритими ключами та їх дешифрацією. Залежно від потреб та конкретних вимог проєкту, може бути обрана програма, яка найкраще відповідає вимогам щодо безпеки, функціональності та зручності користування.

У підсумку, технології для розробки PKI важливу роль у забезпеченні безпеки та захисту даних в цифровому світі. Починаючи з ранніх методів, таких як RSA та Ель – Гамаль, і завершуючи новітніми інноваціями, такими як

квантова криптографія та системи на основі кривих еліптичних кривих, ці технології надають ефективний та надійний захист інформації.

Сучасні програми та інструменти для розробки криптографії з відкритим ключем, такі як OpenSSL, GnuPG, PuTTYgen та Kleopatra, надають зручні та потужні засоби для генерації, керування та використання криптографічних ключів.

З розвитком технологій, таких як квантові обчислювачі та блокчейн, спостерігається поява нових можливостей та викликів у галузі криптографії [23]. Незважаючи на це, постійне дослідження та інновації дозволяють залишатися на крок попереду у забезпеченні безпеки даних у цифровому віці.

РОЗДІЛ 2
АРХІТЕКТУРА ТА ФУНКЦІОНАЛ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО РОЗРОБЛЮЄТЬСЯ

2.1 Вимоги до функціоналу криптографічних ключів

Необхідно розробити криптографічний відкритий ключ, який можна використовувати в Ubuntu з використанням Docker та docker – compose [24]. Цей відкритий ключ буде використовуватися для шифрування та розшифрування даних, а також для цифрового підпису та перевірки цілісності даних.

Відкритий ключ повинен відповідати широкому спектру криптографічних стандартів, включаючи RSA, ECC та інші відомі асиметричні алгоритми шифрування [25]. Довжина ключа має бути достатньою для забезпечення належного рівня безпеки, наприклад, 2048 біт для RSA або 256 біт для ECC. Він повинен бути легко інтегрованим з Docker та docker – compose, що дозволить використовувати його в контейнеризованих додатках.

Необхідно забезпечити можливість генерації ключових пар (відкритого та приватного ключів) за допомогою зручного інтерфейсу командного рядка або графічного інтерфейсу користувача. Відкритий ключ повинен надавати можливість експортувати та імпортувати ключі у різних форматах, таких як PEM, DER або інших поширених форматах [26]. Він має підтримувати шифрування та розшифрування даних різних типів, включаючи текстові файли, бінарні дані та потоки даних, забезпечуючи можливість обробки великих обсягів даних.

Відкритий ключ повинен забезпечувати генерацію та перевірку цифрових підписів для забезпечення цілісності та автентичності даних [27]. Він має бути масштабованим та ефективним, щоб забезпечити швидку

обробку великих обсягів даних, використовуючи паралельну обробку та оптимізацію використання ресурсів.

Для забезпечення безпечного та зручного використання відкритого ключа, необхідно розробити наступний функціонал:

Генерація ключових пар. Користувач повинен мати можливість генерувати ключові пари (відкритий та приватний ключі) за допомогою зручного інтерфейсу командного рядка або графічного інтерфейсу користувача. Під час генерації користувач повинен мати можливість вибрати алгоритм шифрування (RSA, ECC тощо), довжину ключа та інші відповідні параметри.

Експорт та імпорт ключів. Користувач повинен мати можливість експортувати та імпортувати ключі у різних форматах, таких як PEM, DER або інших поширених форматах. Це забезпечить сумісність з різними додатками та системами, а також можливість резервного копіювання та відновлення ключів.

Шифрування та розшифрування даних. Користувач повинен мати можливість шифрувати та розшифровувати дані різних типів, включаючи текстові файли, бінарні дані та потоки даних. Шифрування та розшифрування повинні забезпечувати високий рівень безпеки та використовувати відкритий ключ для шифрування, а приватний ключ – для розшифрування.

Генерація та перевірка цифрових підписів. Користувач повинен мати можливість генерувати цифрові підписи за допомогою приватного ключа та перевіряти цілісність і автентичність даних за допомогою відкритого ключа та цифрового підпису. Це забезпечить захист від несанкціонованих змін даних та підтвердить їх походження.

Інтеграція з Docker та docker – compose. Відкритий ключ повинен бути легко інтегрованим з Docker та docker – compose, що дозволить використовувати його в контейнеризованих додатках. Це може включати створення Docker – образів з відкритим ключем, налаштування контейнерів

для використання відкритого ключа та інтеграцію з додатками, що працюють у контейнерах.

Відкритий ключ повинен відповідати найвищим стандартам безпеки та бути захищеним від відомих вразливостей і атак [28]. Він повинен регулярно оновлюватися та тестуватися на наявність вразливостей і потенційних загроз безпеці. Крім того, необхідно ретельно перевіряти його на сумісність з різними операційними системами, додатками та середовищами. Для забезпечення стабільної роботи та сумісності з широким спектром систем і застосунків слід проводити всебічне тестування відкритого ключа в різних конфігураціях та умовах.

Ці функціональні вимоги забезпечать створення безпечного, надійного та зручного у використанні криптографічного відкритого ключа, який можна інтегрувати з Docker та docker – compose для забезпечення шифрування, розшифрування та цифрового підпису даних у контейнеризованих додатках. Застосування таких високих стандартів безпеки та зручності використання забезпечить довіру користувачів до відкритого ключа та його широке застосування в різноманітних сценаріях.

2.2 Призначення криптографічних відкритих ключів в локальній мережі

Пропонована архітектура криптографічної системи з відкритим ключем, що розгортатиметься в Ubuntu з використанням Docker та Docker Compose, складатиметься з двох основних компонентів – Сервера Ключів та Клієнтського Застосунку. Ці компоненти будуть розгорнуті як окремі Docker – контейнери та взаємодіятимуть через локальну мережу.

Криптографічна система захисту даних, що розглядається, складається з двох основних компонентів: сервера Ключів та Клієнтського Застосунку. Ця система забезпечує безпечний обмін даними між клієнтами в локальній мережі

за допомогою асиметричної криптографії з відкритим ключем. Сервер Ключів відповідає за генерацію та розповсюдження криптографічних ключів, тоді як Клієнтський Застосунок використовує ці ключі для шифрування та розшифрування даних під час комунікації з іншими клієнтами. Розглянемо детальніше функції кожного з компонентів:

1. Сервер Ключів (Key Server):

– відповідатиме за генерацію пар криптографічних ключів (відкритого та приватного);

– зберігатиме та розповсюджуватиме відкриті ключі через захищений канал;

– забезпечуватиме безпечне зберігання та доступ до приватних ключів;

– надаватиме API для взаємодії з Клієнтським Застосунком.

2. Клієнтський Застосунок (Client Application):

– взаємодіятиме з Сервером Ключів через його API;

– отримуватиме відкриті ключі для шифрування даних;

– надсилатиме зашифровані дані іншим клієнтам через локальну мережу;

– розшифровуватиме отримані зашифровані дані, використовуючи відповідний приватний ключ від Сервера Ключів.

Обидва компоненти будуть розгорнуті як Docker – контейнери, що забезпечить ізоляцію, портативність та полегшить керування системою. Docker Compose буде використовуватися для оркестрації та взаємодії між контейнерами.

Приклад використання цієї архітектури в розділі 3 продемонструє налаштування та розгортання системи в Ubuntu, включаючи створення Docker – контейнерів, налаштування мережі та взаємодію між Сервером Ключів та Клієнтським Застосунком. Це дозволить наочно побачити, як працює система криптографії з відкритим ключем у локальному середовищі.

Така архітектура забезпечить модульність, масштабованість та гнучкість системи. Вона дозволить легко додавати нові функції, такі як управління

сертифікатами, авторизацію клієнтів або додаткові механізми безпеки. Крім того, використання Docker та Docker Compose полегшить розгортання та підтримку системи в різних середовищах.

2.3 Вимоги до створення криптографічних ключів

Для створення криптографічних ключів на Ubuntu з використанням Docker та docker – compose, будуть використані наступний підход:

1. Створення Docker – образу. Процес починається зі створення Docker – образу, який буде містити все необхідне програмне забезпечення та бібліотеки для генерації криптографічних ключів. Для цього можна використати базовий образ Ubuntu та встановити необхідні пакети, такі як OpenSSL або GnuPG (GNU Privacy Guard). Ці пакети надають інструменти та бібліотеки для створення різних типів криптографічних ключів, таких як RSA, ECC, EdDSA та інших.

2. Налаштування контейнера. Після створення Docker – образу, необхідно налаштувати контейнер для генерації ключів. Для цього можна використати docker – compose. У файлі docker – compose.yml визначив сервіс для генерації ключів та налаштовуєте необхідні параметри, такі як обсяг для зберігання згенерованих ключів, змінні середовища для конфігурації процесу генерації ключів та інші налаштування, що можуть знадобитися.

3. Генерація ключів. Всередині контейнера використовував відповідні команди для генерації криптографічних ключів. Наприклад, для генерації RSA – ключа можна використати команду openssl genrsa, для генерації пари ключів ECC – openssl ecparam та openssl genpkey. Автоматизувати процес генерації ключів за допомогою скриптів або запускати команди вручну.

4. Зберігання ключів. Згенеровані ключі необхідно зберігати в безпечному місці. Для цього ви можете використовувати обсяг, який був налаштований у файлі docker – compose.yml. Таким чином, ключі будуть

зберігатися на хост – машині, а не всередині контейнера. Це забезпечує безпеку та можливість використовувати ключі в інших проєктах або додатках.

5. Використання ключів. Після генерації ключів їх можна використовувати для різних цілей, наприклад, для шифрування/дешифрування даних, цифрових підписів або обміну ключами. Для цього ви можете створити додатковий сервіс у docker – compose.yml, який буде використовувати згенеровані ключі або ж інтегрувати їх у ваші існуючі додатки.

Порівняльна функцій, можливостей та недоліків різних типів криптографічних ключів продемонстровано в таблиці 2.1. Кожен тип криптографічного ключа має свої переваги та обмеження, що впливають на його застосування в різних сценаріях. Вибір конкретного типу ключа залежить від вимог безпеки, продуктивності та сумісності з існуючими системами.

Таблиця 2.1 – Порівняння функцій, можливостей та недоліків.

Тип ключа	Функції та можливості	Недоліки
RSA	Широко використовується для шифрування, цифрових підписів та обміну ключами – Висока стійкість до криптографічних атак – Відкриті стандарти.	Великий розмір ключів (зазвичай 2048 біт і більше) – Повільніший, ніж деякі інші алгоритми.
ECC	Менші розміри ключів, ніж RSA, при тому ж рівні безпеки – Високошвидкісні операції – Низьке енергоспоживання.	Складніші математичні обчислення – Менш поширені, ніж RSA.
EdDSA	Швидкі та безпечні цифрові підписи – Простота реалізації – Стійкість до атак на основі квантових обчислень.	Призначені лише для цифрових підписів – Не такі поширені, як RSA або ECC.
X25519	Безпечний обмін ключами – Високошвидкісні операції – Стійкість до атак на основі квантових обчислень	Використовується лише для обміну ключами – Не такий поширений, як RSA.

Після аналізу функцій, можливостей та недоліків різних типів криптографічних ключів, наведених у таблиці, найкращим вибором для цього проєкту є використання RSA – ключів.

RSA – ключі є широко використовуваними та добре зарекомендували себе в різноманітних криптографічних операціях, таких як шифрування даних, цифрові підписи та обмін ключами [29].

Беручи до уваги вимоги до безпеки та універсальності, а також широку підтримку RSA в різноманітних криптографічних бібліотеках та інструментах, використання RSA – ключів є найкращим рішенням для створення криптографічних ключів на Ubuntu з використанням Docker та docker – compose в рамках цього проєкту.

РОЗДІЛ 3
РОЗРОБЛЕННЯ ПОСЛІДОВНОСТІ СТВОРЕННЯ ЦЕНТРУ СЕРТИФІКАЦІЇ

3.1 Вибір засобів розробки криптографічних ключів

У розділі 2 було проведено ґрунтовний аналіз та порівняння різних типів криптографічних ключів, таких як RSA. Після ретельного дослідження їхніх переваг, недоліків, міцності та ефективності використання в різних сценаріях, RSA – ключі виявилися найкращим вибором для створення ключів у цій системі генерації.

Для створення криптографічних ключів та управління ними буде використовуватися Docker та docker – compose. Ці інструменти забезпечать генерацію міцних ключів, підписання та шифрування даних, а також управління сертифікатами та ключами. Розробка системи генерації криптографічних ключів буде здійснюватися в середовищі Ubuntu, що забезпечить портативність, ізольованість та легку масштабованість нашого рішення.

Використання Ubuntu як базової операційної системи забезпечить стабільність, надійність та безпеку. Портативність рішення буде забезпечена завдяки контейнерам Docker, що дозволить легко переносити та розгортати систему в різних середовищах, таких як локальні машини, хмарні провайдери або сервери в центрах обробки даних. Ізольованість а точніше можливість роботи в офлайні режимі контейнерів Docker та забезпечить конфіденційності а також взаємодію лише з машинами на котрих встановлено додатки на тій же машині.

Тож на базі Ubuntu буде створено комплексну систему для забезпечення безпеки даних. Ключовим компонентом є окремий сервер для генерування та управління криптографічними ключами, заснований на інфраструктурі

відкритих ключів (PKI). Також передбачено потужне сховище даних з функціями швидкого доступу, резервного копіювання та відновлення для зберігання великих обсягів конфіденційної інформації. Для зручного доступу користувачів та керування налаштуваннями розроблено захищений вебінтерфейс з багаторівневою автентифікацією. Всі компоненти розгорнуті на окремих віртуальних машинах або Docker – контейнерах з системою моніторингу для забезпечення гнучкості, масштабованості, ефективної обробки даних та безперервного доступу.

Визначившись із криптографічними алгоритмами та бібліотеками, які будуть використані при розробці, наступним кроком повинен бути вибір програмних засобів розробки та відладки системи керування криптографічними ключами.

Хоча OpenSSL в цілому надає все необхідне для повного циклу криптографічної обробки, буде використовуватися лише для генерації та управління ключами, а безпосередньо написання коду програми буду виконувати через термінал котрий використовується в Ubuntu [30].

Також, знадобиться Docker (зрештою, контейнери дозволяють ізоляцію і безпечне виконання криптографічних операцій).

Для коректної роботи криптографічних операцій потрібні не найновіші версії OpenSSL, в моєму випадку найбільш стабільною виявилася версія 1.1.1.

Останній у списку додаткового програмного забезпечення – Docker Compose, за допомогою якого буде керуватися вся система контейнеризації криптографічних сервісів.

Після скачування і встановлення всього перерахованого вище, додатково потрібно створити дві змінні середовища в Ubuntu: OPENSSL_HOME, яка буде містити шлях до встановленої бібліотеки OpenSSL, а також DOCKER_HOME, який міститиме шлях до Docker.

Для роботи з Docker та Docker Compose буде використовуватися термінал Ubuntu (але загалом і будь – який інший термінал Linux теж підійде).

3.2 Створення сертифікатів та їх функціонал

Отримання статусу центра сертифікації на Ubuntu дозволяє створювати та управляти цифровими сертифікатами. ЦС є авторитетним джерелом, що засвідчує справжність сертифікатів, використовуваних для забезпечення безпечного обміну даними в Інтернеті. Після налаштування центра сертифікації можна -розпочати процес генерування та випуску сертифікатів для вебсерверів.

Кожен сертифікат унікально ідентифікує вебсайт, забезпечуючи шифрування даних, автентифікацію та конфіденційність під час з'єднання. Сертифікати виконують важливу роль у захисті конфіденційної інформації, такої як облікові дані користувачів, дані платіжних карток та інші чутливі відомості, що передаються через Інтернет.

Крім вебсерверів, сертифікати також використовуються для авторизації користувачів, електронного підпису документів та перевірки цілісності даних. Належне управління центром сертифікації та дотримання політик безпеки, таких як захист приватних ключів, своєчасне оновлення сертифікатів та процедури відкликання, є критично важливими для підтримки довіри та забезпечення безпеки в цифровому середовищі.

Початком практичної частини буде отримання статусу центру сертифікації, що дасть змогу працювати з сертифікатами та забезпечувати їх легітимність. Це дозволить виконувати криптографічні операції, такі як шифрування та підписання документів, відповідно до стандартів безпеки.

Спершу вирішив налаштувати файли та встановити Docker та Docker Compose для роботи над майбутнім проєктом. Це дозволить легко створювати, запускати та керувати контейнерами для різних сервісів проєкту. Усе – таки використання цих інструментів спростить розгортання та масштабування додатку (рис. 3.1).

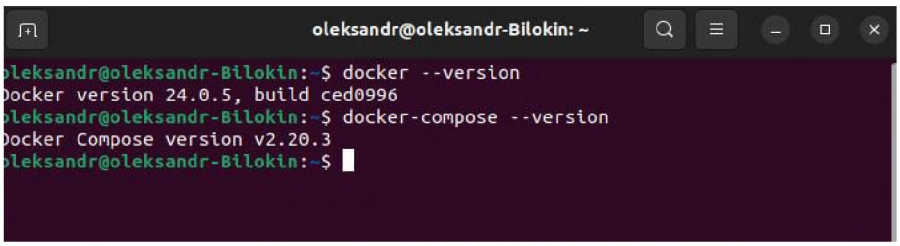

Рисунок 3.1 – Версії встановлених інструментів

Змінив файл /etc/hosts, щоб змінити локальні DNS записи. Це дозволяє перенаправляти доменні імена на інші IP – адреси з комп'ютера. Таким чином, може тестуватися вебсайт або програми без зміни глобальних DNS записів (рис. 3.2).

Рисунок 3.2 – Зовнішній вигляд зміненого файлу

Налаштовую копію файлу openssl.cnf і переміщую її до робочої папки. Цей файл містить конфігурацію для. Після налаштування та переміщення, зможу використовувати модифіковану копію файлу для подальших дій (рис. 3.3).

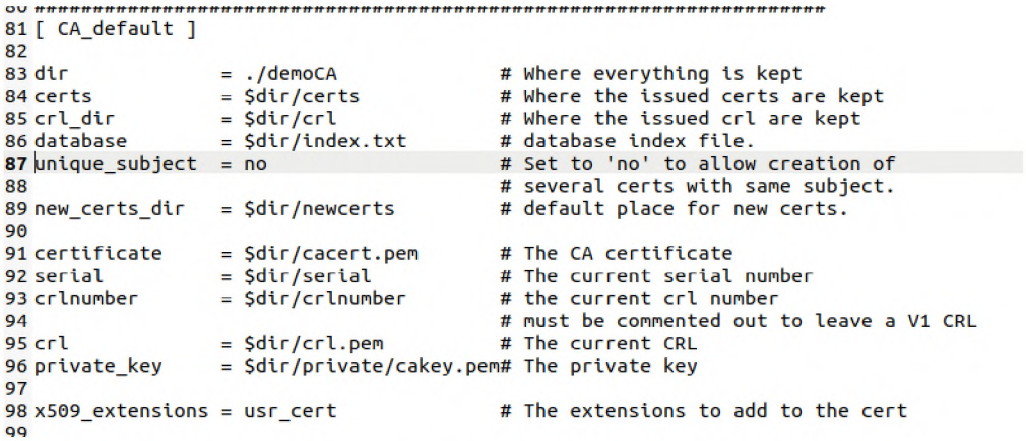

Рисунок 3.3 – Зовнішній вигляд зміненого файлу

Створюю файл index.txt та файл seril, який містить число 1000 за допомогою команди cat та переміщую їх в створену папку demoCA (рис. 3.4).

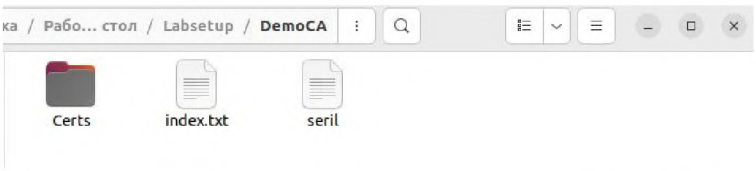

Рисунок 3.4 – Зовнішній вигляд створених файлів

Першим кроком створюю простий текстовий файл index.txt. Після цього створюю інший файл з назвою seril, який містить лише одне число – 1000. Для створення файлу seril використовую команду cat в терміналі в ньому увів PEM pass 1234 для зручності роботи з проєктом (рис. 3.5).

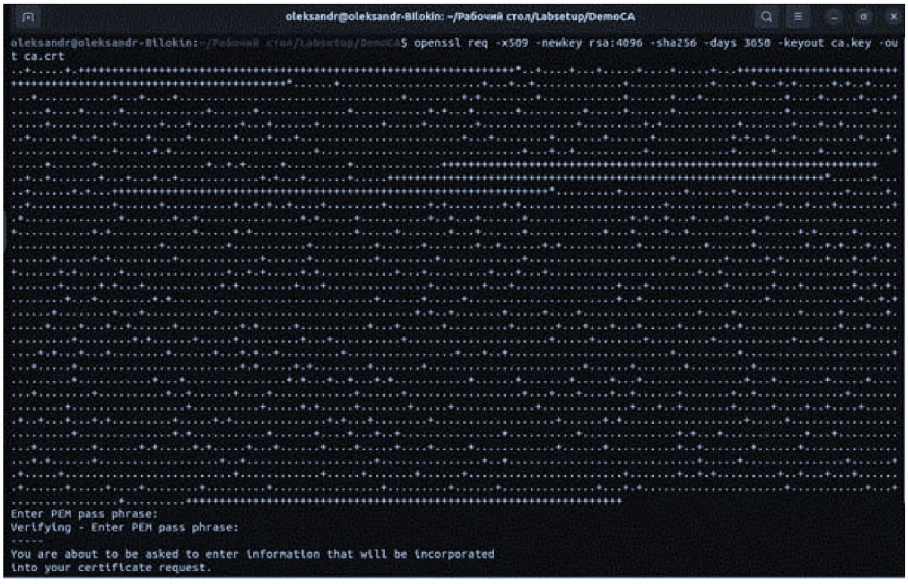

Рисунок 3.5 – Зовнішній вигляд створеного файлу

Виконавши команду 'openssl x509 -in ca.crt -text -noout', отримав текстове представлення вмісту сертифіката x509, що зберігається у файлі ca.crt. Параметр '-text' вказує OpenSSL вивести деталі сертифіката у зручному для читання текстовому форматі. А параметр '-noout' інструктує не виводити закодований бінарний вміст сертифіката, лише його текстове представлення (рис. 3.6).

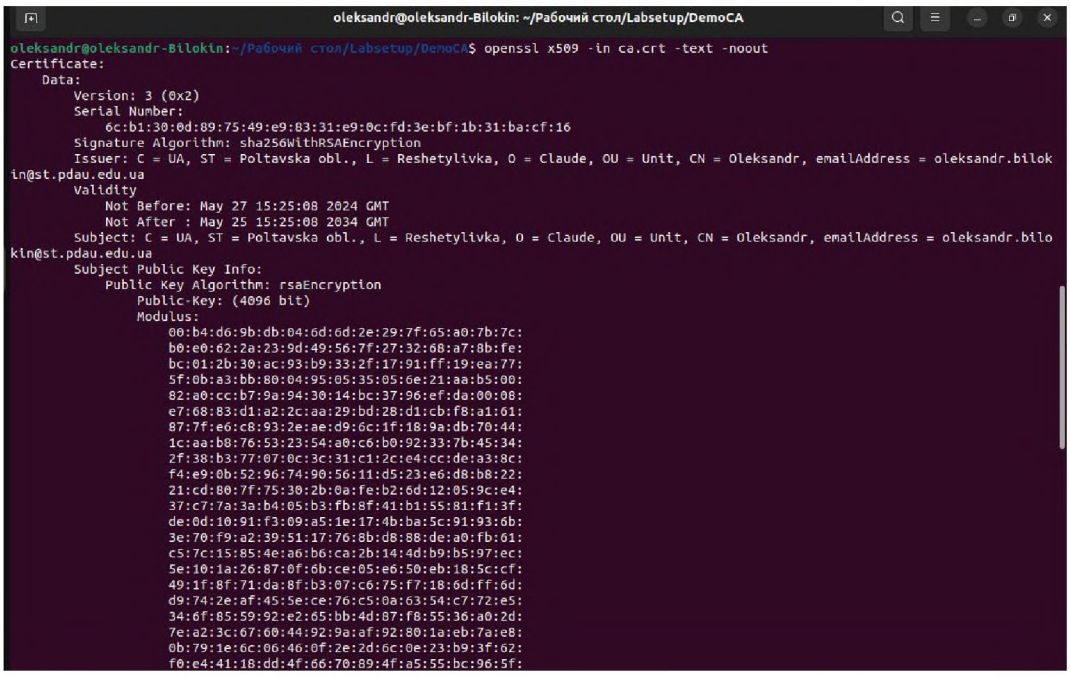

Рисунок 3.6. – Зовнішній вигляд сертифіката

Тепер почну зі створення сертефікату для вебсервера. Створю запит на підписання сертифіката (CSR) за допомогою команди: openssl req -newkey rsa:2048 -sha256 -keyout server.key -out server.csr -subj "/CN=www.oleksandr2024.com/O=Claude /C=UA" -passout pass:1234 (рис.3.7).

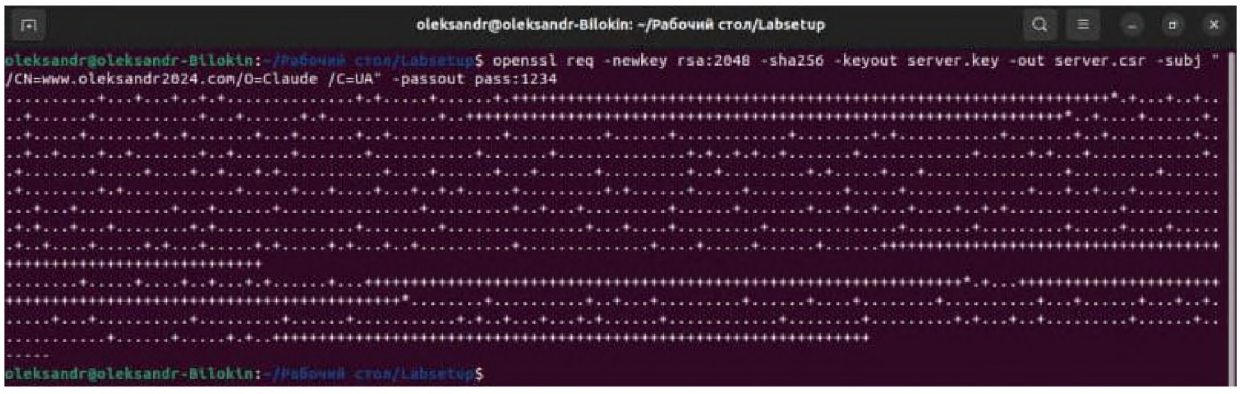

Рисунок 3.7 – Зовнішній вигляд результату введеної команди

Ця команда генерує новий криптографічний ключ RSA з довжиною 2048 біт, використовуючи алгоритм хешування SHA-256. Згенерований ключ буде збережено у файлі server.key, а запит на підписання сертифіката – у файлі server.csr (рис. 3.8).

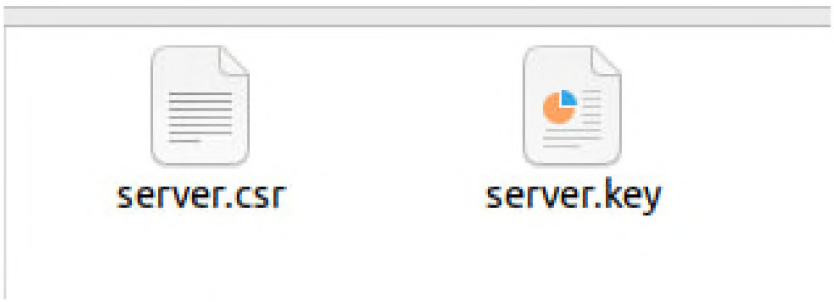

Рисунок 3.8 – Зовнішній вигляд створених файлів

Тепер переглядаю зміст файлу server.csr, який містить запит на підписання сертифіката, за допомогою команди "openssl req -in server.csr -text -noout" (рис. 3.9). Ця команда відображає деталі запиту на підписання сертифіката у текстовому форматі, не виводячи закодований запит.

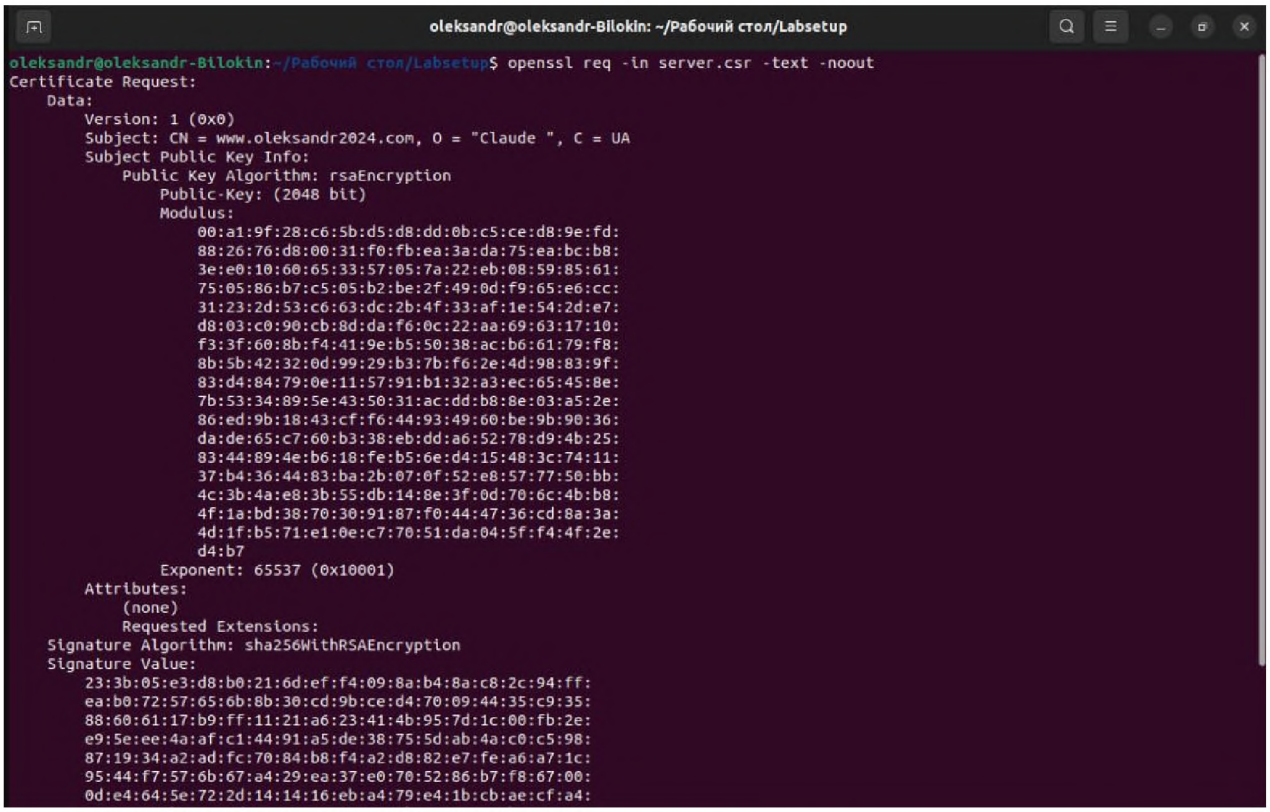

Рисунок 3.9 – Зовнішній вигляд файлу після введеня команди

Також переглядаю зміст файлу server.key, який містить згенерований приватний ключ, за допомогою команди "openssl rsa -in server.key -text -noout" (рис. 3.10).

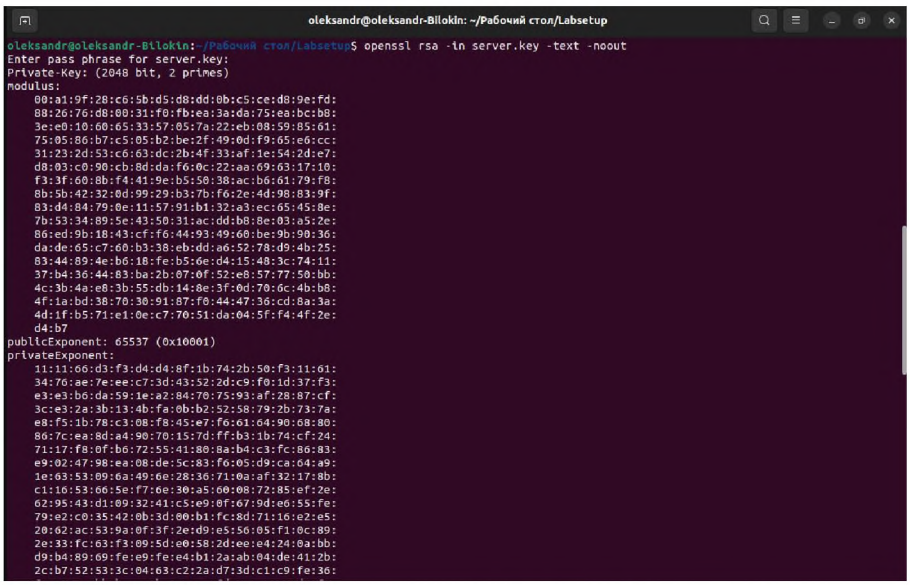

Рисунок 3.10 – Зовнішній вигляд файлу після введеня команди

3.3 Розгортання сертифіката на HTTPS – сайті на базі Apache

Безпечний HTTPS – сайт на базі Apache вимагає встановлення та налаштування SSL сертифіката [31]. Сертифікат являє собою цифровий файл, який визначає вашу кредитоспроможність і використовується для шифрування зв'язку між вебсервером та клієнтом (браузером). Існують різні варіанти отримання SSL сертифіката. Одним із найпопулярніших є придбання комерційного сертифіката у сторонньої організації, яка називається центром сертифікації (CA). Ці сертифікати коштують від кількох десятків до сотень доларів на рік, але їм довіряють усі основні браузери.

Після успішної конфігурації ваш вебсайт буде захищений і клієнти зможуть безпечно підключатися до нього через HTTPS протокол, який гарантує шифрування даних та перевірку автентичності вашого сервера.

Після отримання SSL сертифіката необхідно належним чином налаштувати його на вебсервері Apache. Важливо правильно інтегрувати сертифікат та встановити необхідні параметри для забезпечення безпечного з'єднання. Це включає налаштування підтримки SSL протоколу та використання надійних криптографічних алгоритмів шифрування.

Налаштовую Dockerfile для забезпечення оптимальної конфігурації та ефективного розгортання додатку в контейнерному середовищі. Цей процес дозволить створити надійну та масштабовану систему, яка легко адаптується до змін та вимог проєкту (рис. 3.11).

Рисунок 3.11 – Зовнішній вигляд зміненого файлу

Налаштовую конфігураційний файл для системи. Крім цього, додаю в папку image_www/certs необхідні файли сертифікатів. Серед них знаходяться файли ca.crt, server.key та server.crt (рис. 3.12).

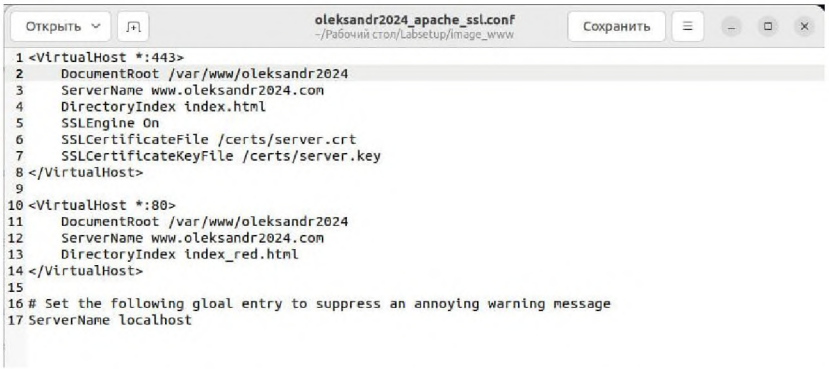

Рисунок 3.12 – Зовнішній вигляд зміненого файлу

Виконую збірку з оновленими файлами, щоб включити всі останні зміни. Після цього запускаю Docker контейнер для перевірки коректності роботи системи. Це дозволяє впевнитись, що всі зміни застосовані правильно і система функціонує без збоїв виконавши команду docker – compose up -d -build (рис. 3.13).

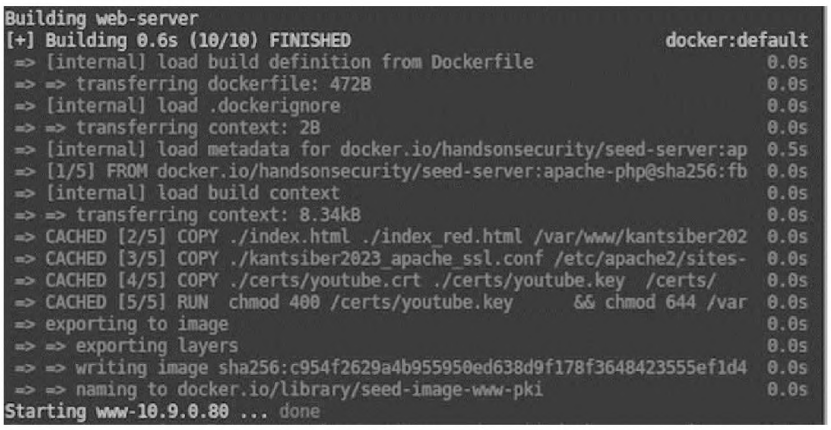

Рисунок 3.13 – Зовнішній вигляд виконаної команди

Ввів команду docker exec -it www-10.9.0.80 bash service apache2 start, щоб запустити інтерактивну оболонку bash у контейнері Docker. Після цього ввів команду service apache2 start, щоб запустити вебсервер Apache2. Під час запуску Apache2 система попросила мене ввести пароль для SSL/TLS ключів, які використовуються для сайту www.youtube.com:443 (рис. 3.14).

Рисунок 3.14 – Зовнішній вигляд виконаної команди

Після виконання команди було розгоруто сертифікат на HTTPS-сайті на базі Apache. Тепер сайт забезпечений безпекою та готовий до використання.

3.4 Атаки Man – In – The – Middle

Атаки MITM, або атаки "людина посередині", є однією з найпоширеніших та небезпечних форм кібератак. Під час такої атаки зловмисник перехоплює комунікацію між двома сторонами, часто залишаючись непоміченим, і може змінювати або підроблювати передані дані.

Основна мета MITM – атак полягає в крадіжці конфіденційної інформації, такої як паролі, номери кредитних карток, або інші приватні

дані [32]. Це досягається шляхом втручання в незахищене з'єднання або через використання шкідливого програмного забезпечення. Одним із поширених методів є використання підроблених Wi – Fi точок доступу, де користувач підключається до мережі, яка виглядає легітимною, але насправді контролюється зловмисником.

Для захисту від MITM – атак важливо використовувати шифрування, наприклад, за допомогою протоколів HTTPS або VPN. Також рекомендується уникати використання незахищених публічних мереж Wi – Fi без додаткових заходів безпеки, таких як VPN. Регулярне оновлення програмного забезпечення та використання антивірусних програм можуть допомогти виявляти та запобігати спробам MITM – атак.

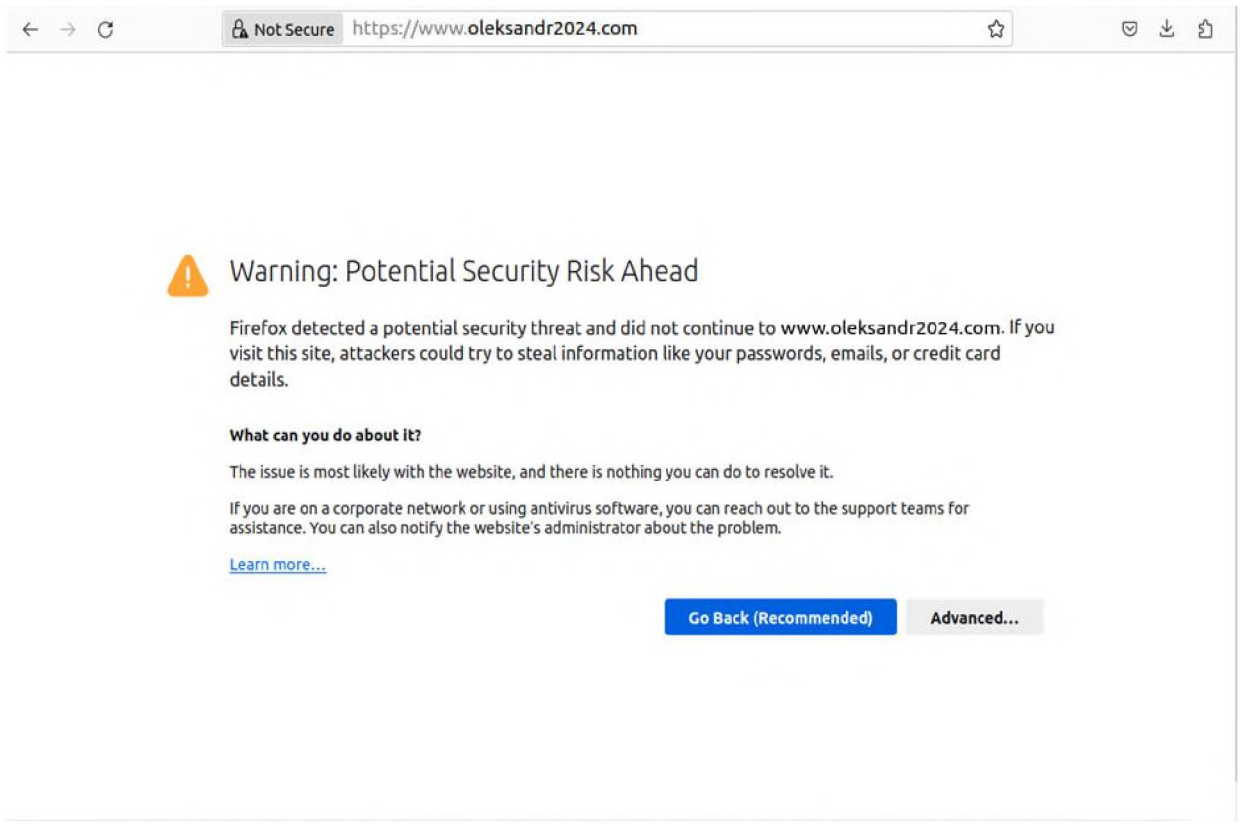

Рисунок 3.15 – Зовнішній вигляд сайту без сертифікату

У наступних двох підрозділах буде розглянуто атаки MITM як з скомпрометованим центром сертифікації, так і без нього. В першому випадку зловмисник отримає можливість видавати себе за легітимну сторону та

перехоплювати трафік, використовуючи підроблений сертифікат, виданий скомпрометованим центром сертифікації. У другому випадку, коли ЦС не був скомпрометований, атакуючий має використовувати інші методи, такі як підробка сертифікатів. Ці атаки дозволяють зловмиснику перехоплювати та модифікувати трафік між двома сторонами, що порушує конфіденційність та цілісність даних.

Розпочну з запуску атаки "люди посередині". Браузер блокує сайт як потенційно небезпечний оскільки не завантажив самопідписний сертифікат в браузер. Це стандартна процедура безпеки, щоб захистити користувача від потенційно шкідливих сайтів (рис. 3.15).

Додаю сертифікат ca.crt в браузер, щоб він більше не блокував сайт як потенційно небезпечний через відсутність самопідписного сертифіката (рис. 3.16).

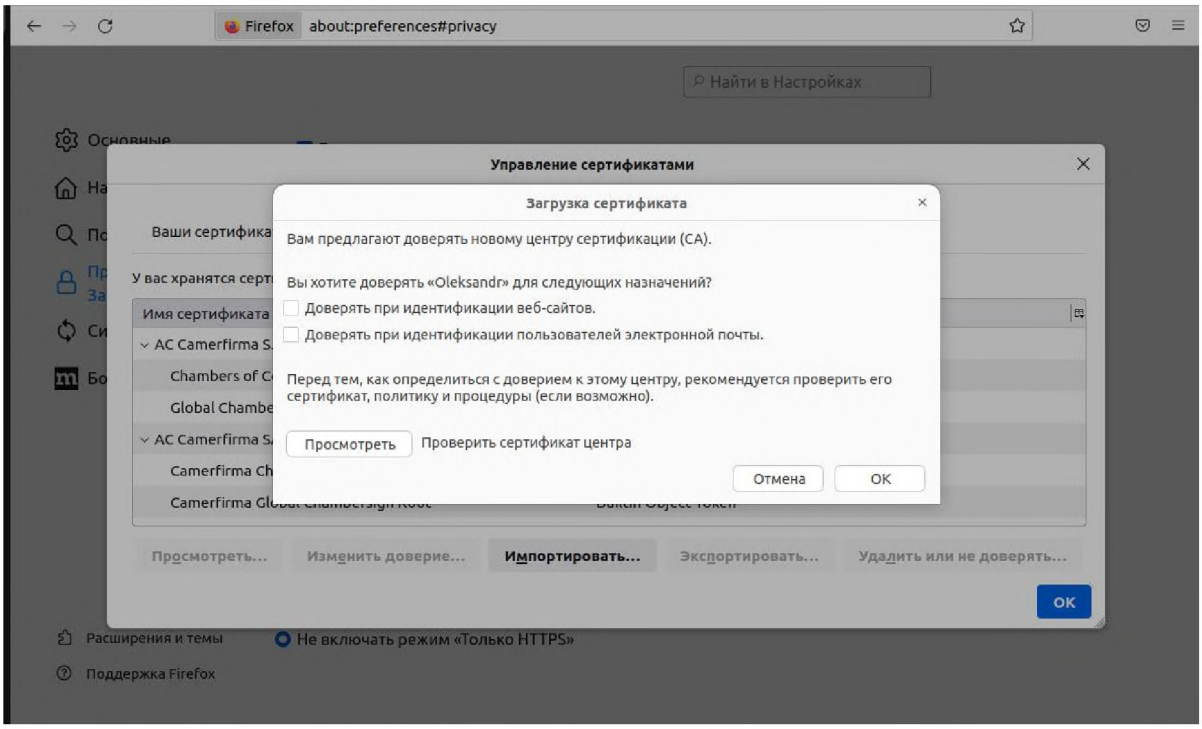

Рисунок 3.16 – Зовнішній вигляд додавання сертифікату

Тепер сайт працює через HTTPS, що є ознакою належної конфігурації сертифіката(рис. 3.17).

Рисунок 3.17 – Зовнішній вигляд сайту з сертифікатом

Тепер проведу атаку "людина посередині" з скомпрометованим ЦС.

Додаю до файлу hosts рядок 10.9.0.80 www.youtube.com для перенаправлення запитів на адресу www.youtube.com за адресою 10.9.0.80 (рис. 3.18).

Рисунок 3.18 – Зовнішній вигляд змін в файлі hosts

Виконав команду openssl ca –config openssl1.cnf –policy policy_anything –md sha256 –days 3650 –in youtube.csr –out youtube.crt –batch –cert ca.crt –keyfile ca.key (рис. 3.19), використав попередньо підписаний та згенерований запит на сертифікат youtube.csr і видачі кінцевого сертифіката youtube.crt, який буде дійсним протягом 3650 днів (10 років).

```
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4103 (0x1007)
        Validity
            Not Before: Sep 26 16:30:11 2023 GMT
            Not After : Sep 23 16:30:11 2033 GMT
        Subject:
            countryName               = UA
            organizationName          = Claude
            commonName                = www.youtube.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Key Identifier:
                7C:33:6D:2A:6A:75:0C:55:44:39:C2:03:BE:82:6C:D8:0D:06:90:D5
            X509v3 Authority Key Identifier:
                6C:23:AA:C8:9C:D0:09:10:D2:A8:AD:E4:DA:50:B0:89:F5:03:C4:18
Certificate is to be certified until Sep 23 16:30:11 2033 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
```

Рисунок 3.19 – Зовнішній вигляд підписання сертифікату

Редагував файли Dockerfile (рис. 3.20) та конфігураційні файли (рис. 3.21). Цей процес був необхідним кроком для забезпечення відповідності вимогам проєкту. Спочатку переглянув наявні налаштування і вніс необхідні зміни в Dockerfile, щоб оптимізувати процес побудови образу та зменшити його розмір. Потім оновив конфігураційні файли, враховуючи всі нові вимоги та специфікації проєкту.

Открыть | Dockerfile ~/Рабочий стол/Labsetup | Сохранить

```
1 FROM handsonsecurity/seed-server:apache-php
2
3 ARG WWWDIR=/var/www/oleksandr2024
4
5 COPY ./index.html ./index_red.html $WWWDIR/
6 COPY ./oleksandr2024_apache_ssl.conf /etc/apache2/sites-available
7 COPY ./certs/youtube.crt ./certs/youtube.key  /certs/
8
9 RUN  chmod 400 /certs/server.key \
10      && chmod 644 $WWWDIR/index.html \
11      && chmod 644 $WWWDIR/index_red.html \
12      && a2ensite oleksandr2024_apache_ssl
13
14 CMD  tail -f /dev/null
15
```

Рисунок 3.20 – Зовнішній вигляд зміненого файлу Dockerfile

Открыть | oleksandr2024_apache_ssl.conf ~/Рабочий стол/11/image_www | Сохранить

```
1 <VirtualHost *:443>
2     DocumentRoot /var/www/oleksandr2024
3     ServerName www.youtube.com
4     DirectoryIndex index.html
5     SSLEngine On
6     SSLCertificateFile /certs/youtube.crt
7     SSLCertificateKeyFile /certs/youtube.key
8 </VirtualHost>
9
10 <VirtualHost *:80>
11     DocumentRoot /var/www/oleksandr2024
12     ServerName www.youtube.com
13     DirectoryIndex index_red.html
14 </VirtualHost>
15
16 # Set the following gloal entry to suppress an annoying warning message
17 ServerName localhost
```

Рисунок 3.21 – Зовнішній вигляд зміненого конфігураційного файлу

Після внесення всіх змін, створив новий образ контейнера, виконавши команду build (рис. 3.22). Під час процесу побудови перевірив, чи всі залежності були правильно встановлені, і впевнився, що новий образ працює без помилок. Після успішного створення образу провів його тестування, щоб підтвердити, що всі зміни були застосовані коректно і система працює згідно з очікуваннями.

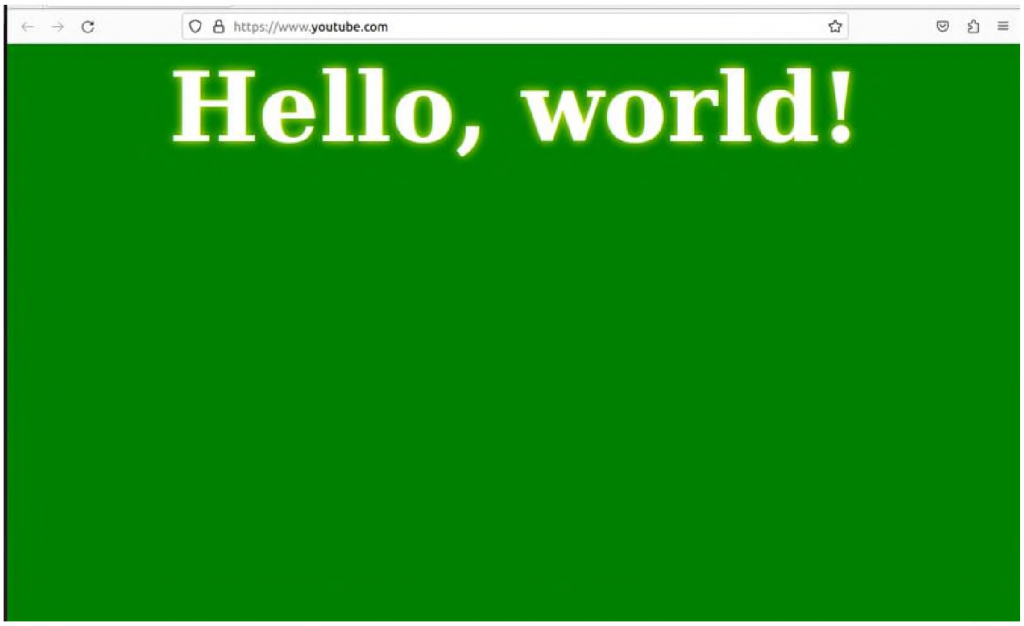

Рисунок 3.22 – Зовнішній вигляд сайту з скомпрометованим ЦС

3.5 Економічна ефективність розробки криптографічних ключів та сертифікатів

Основними цілями розробки спеціалізованого програмного забезпечення для дослідження методів криптографії з відкритим ключем є покращення якості навчання, підвищення рівня знань і навичок студентів у сфері захисту інформації, а також підготовка фахівців до реальних викликів в галузі кібербезпеки. Це допоможе навчальним закладам забезпечити студентам сучасну та практичну освіту, що відповідає вимогам ринку праці.

Основними завданнями проєкту є розробка інтерактивних модулів, симуляторів реальних загроз та системи тестування знань. Ці компоненти забезпечать інтерактивне та практичне навчання студентів, дозволяючи їм відпрацьовувати навички захисту інформації в умовах, максимально наближених до реальних.

Аналіз вимог до економічної цінності та аналіз ринку криптографічних ключів є ключовими етапами в розробці проєкту.

Дослідження потреб навчальних закладів і студентів показало, що більшість навчальних закладів потребують сучасних засобів навчання для викладання криптографії з відкритим ключем. Студенти також висловлюють зацікавленість у таких програмах, оскільки це підвищує їх конкурентоспроможність на ринку праці.

Аналіз існуючих рішень на ринку показав, що більшість з них не відповідають повною мірою сучасним вимогам або є занадто дорогими. Ця розробка має заповнити цю нішу, надавши доступне та ефективне рішення.

Таблиця 3.1 – Вартість розробки

Категорія витрат	Одиниця виміру	Кількість	Вартість за одиницю (грн)	Загальна вартість (грн)	Коментарі
Розробники	година	160	1 800	288 000	10 розробників по 16 годин на місяць
Тестувальники	година	160	1 080	172 800	10 тестувальників по 16 годин на місяць
Технічна підтримка, хостинг і ліцензування	місяць	1	72 000	72 000	Оплата послуг хостингу, оновлення ПЗ та підтримка користувачів
Навчання викладачів та персоналу	працівник	20	18 000	360 000	Вартість на одного працівника

Законодавчі вимоги та стандарти у сфері захисту інформації включають дотримання міжнародних та національних стандартів криптографії, таких як ISO/IEC 27001 та Закон України "Про захист інформації в інформаційно-телекомунікаційних системах".

Розробка ПЗ: Розраховано на 10 розробників і 10 тестувальників, кожен працює по 16 годин на місяць. Витрати на оплату праці розробників складають 1 800 грн за годину, що становить 288 000 грн на місяць. Витрати на оплату праці тестувальників складають 1 080 грн за годину, що становить 172 800 грн на місяць. Це все зображено на таблиці 3.1.

Технічна підтримка, хостинг і ліцензування: Загальні витрати становлять 72 000 грн на місяць.

Навчання викладачів та персоналу: Витрати на навчання складають 18 000 грн на одного працівника, що при 20 працівниках складає 360 000 грн.

Використання криптографічних ключів принесе значні економічні вигоди, зокрема зменшення витрат на навчання криптографії, підвищення кваліфікації викладачів та зменшення ризиків, пов'язаних з інформаційною безпекою. Це дозволить навчальним закладам зекономити кошти на традиційних методах навчання та підвищити рівень захисту інформації.

Оцінка впливу на якість навчання показує, що впровадження криптографії з відкритим ключем підвищить конкурентоспроможність навчального закладу, зробивши його привабливішим для студентів. Це може призвести до збільшення кількості студентів, зацікавлених у програмі, що в свою чергу підвищить доходи навчального закладу.

Розрахунок економічних показників.

Простий термін окупності визначається за формулою:

$$PBP = \frac{\text{Інвестиції}}{\text{Щорічнівигоди}}. \quad (3.1)$$

За даними, інвестиції складають 892 800 грн, а щорічні вигоди від зменшення витрат на навчання і підвищення кваліфікації складають приблизно 450 000 грн. Таким чином:

PBP = 892 800 / 450 000 ≈ 1,98 роки

Чиста приведена вартість (NVR) визначається шляхом дисконтування грошових потоків і віднімання початкових інвестицій.

$$NPV = \sum \frac{CF_t}{(1+r)^t} - C_0. \quad (3.2)$$

При ставці дисконту 10% і річних вигодах 450 000 грн:

NPV = 450 000 / $(1 + 0,1)^1$ + 450 000 / $(1 + 0,1)^2$ + 450 000 / $(1 + 0,1)^3$ – 892 800 ≈ 275 620 грн

IRR – це ставка дисконту, при якій NPV дорівнює нулю. Розрахунок проводиться методом ітерацій.

При IRR ≈ 15%:

NPV = 450 000 / $(1 + 0,15)^1$ + 450 000 / $(1 + 0,15)^2$ + 450 000 / $(1 + 0,15)^3$ – 892 800

ROI визначається шляхом ділення чистого прибутку на інвестиції та множення на 100 за формулою:

$$ROI = \frac{\text{Чистийприбуток}}{\text{Інвестиці}} \text{ x } 100 . \quad (3.3)$$

При чистому прибутку 275 620 грн і інвестиціях 892 800 грн:

ROI = 275 620 / 892 800 × 100 ≈ 30,87%

Можливі ризики включають зміни ринку, технічні проблеми та зміни в законодавстві. Щоб мінімізувати ці ризики, розроблені плани на випадок їх виникнення. Наприклад, для зменшення ризиків технічних проблем передбачено регулярні оновлення та технічну підтримку ПЗ.

Тестування було проведено на реальних групах студентів для оцінки ефективності та зручності використання ПЗ. Оцінка результатів тестування включала аналіз зворотного зв'язку від студентів та викладачів, виявлення недоліків та їх усунення.

Усі розрахунки, проведені в рамках даного дослідження, зведені в таблиці та формули. Основні економічні показники свідчать про високу ефективність проєкту.

Проєкт має високу економічну ефективність, про що свідчать позитивні значення NPV, короткий термін окупності та високий коефіцієнт

рентабельності інвестицій. Впровадження криптографії з відкритим ключем підвищить якість навчання та конкурентоспроможність навчального закладу.

На основі аналізу отриманих даних прийнято рішення про доцільність інвестування в розробку криптографічних ключів. Розробка та впровадження спеціалізованого програмного забезпечення для дослідження методів криптографії з відкритим ключем є економічно вигідною та сприятиме підвищенню рівня захисту інформації в навчальних закладах.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було проведено огляд історії появи та розвитку криптографії з відкритим ключем та принципів, на яких вона базується. Також, було виконано аналіз існуючих технологій розробки та застосування криптографічних алгоритмів з відкритим ключем, які є найбільш популярними та широко використовуваними на сьогоднішній день.

При підготовці в другому розділі, було зібрано та сформульовано вимоги до криптографічного алгоритму з відкритим ключем, який потрібно було реалізувати. Після отримання набору вимог було описано набір основних компонентів та процесів, які реалізують поставлені вимоги. Перед початком розробки було проведено аналіз функціоналу ключів, після яких вони будуть використані для реалізації алгоритму.

У третьому розділі було обрано алгоритм RSA для генерації ключів та середовище Ubuntu для розробки з використанням Docker. Було продемонстровано створення кореневого сертифіката центру сертифікації, генерування сертифіката для вебсервера та розгортання HTTPS-сайту на базі Apache з використанням згенерованих сертифікатів. Також було розглянуто атаку "людина посередині" з використанням підробленого сертифіката від скомпрометованого центру сертифікації для ілюстрації важливості забезпечення безпеки центру сертифікації.

Проведено розрахунок вартості виконаних робіт. Завдяки тому, що все програмне забезпечення, яке було використано в роботі, є безкоштовним, вдалося уникнути витрат і отримати прибуток від виконаного замовлення.